PROYECTO FIN DE CARRERA

ESTUDIO, INTEGRACIÓN Y CONFIGURACIÓN DE UN IDS EN UN ENTORNO REAL

Escuela Superior de Ingenieros
Universidad de Sevilla

Gema de Toro Sánchez
Tutor: Juan Antonio Ternero Muñiz

Ingeniería de Telecomunicación
Julio 2004

ÍNDICE

1 INTRODUCCIÓN

1.1 MOTIVACIÓN

La informática y las redes de telecomunicación se han convertido en herramientas indispensables para realizar las tareas críticas de la vida profesional. Son fundamentales en la enseñanza, la investigación, y la gerencia en general de las empresas. Por tanto, debemos asegurar su buen funcionamiento si queremos garantizar nuestra libertad de comunicación. Este buen funcionamiento conlleva, entre otros factores, la protección de los sistemas y datos nominativos, la fiabilidad de programas y materiales informáticos, el rendimiento y disponibilidad de los servicios, la protección de la información que intercambiamos y la protección del acceso a los sistemas.

Hasta el principio de los años 80, la centralización de los medios informáticos y la ausencia de comunicaciones con el exterior, permitían que, utilizando una buena administración centralizada, se garantizaran los objetivos anteriores. Pero los tiempos han cambiado y ahora los sistemas de información son distribuidos e interconectados entre ellos mediante redes. Si bien, esta arquitectura distribuida y la conexión a Internet ofrecen un caudal de nuevas y prometedoras posibilidades, introducen igualmente cada vez mayor número de riesgos. Además, las técnicas y metodologías de los ataques informáticos están evolucionando continuamente, al mismo tiempo que evoluciona la complejidad de las redes y los sistemas que las integran.

Cualquier ordenador conectado directa o indirectamente a Internet es una víctima potencial, independientemente del servicio que ofrezca o los datos que contenga. Incluso si un ordenador no contiene información interesante, puede ser utilizado como puente hacia otros sistemas que contengan datos más valiosos. Los atacantes se valen de todo tipo de información para entrar fraudulentamente en un sistema, desde información como el hardware, el software o el tipo de conexión utilizada hasta, por supuesto, informaciones concernientes a la propia seguridad, como contraseñas o algoritmos de encriptación. Por tanto, la seguridad en el campo de las tecnologías de la información y las telecomunicaciones tendría como objetivo principal proteger los sistemas informáticos del uso no autorizado. El impacto de este tipo de uso podría conllevar desde la imposibilidad de utilización de los servicios que se ofrecen, la sustracción de datos (en algunos casos de alta confidencialidad) o la alteración de los mismos hasta incluso la pérdida de credibilidad de una empresa frente a otras.

Hay que ser conscientes de los riesgos que estos ataques representan y actuar en consecuencia tomando las medidas necesarias. Estas medidas representan un coste financiero,

bastante inferior, sin embargo, al que nos podría acarrear la ausencia de seguridad en los sistemas de información.

No sólo las grandes empresas son víctimas de estos ataques informáticos. Según datos de Symantec referidos a diciembre de 2003, un 14% de las PYMES habían registrado un ataque severo a sus sistemas informáticos en los últimos seis meses. El colectivo de las PYMES con menos de 500 empleados sufre una media de 106 ciberataques al mes y registra, además, otros ataques más graves. Este número asciende a 132 para el caso de las grandes empresas, de las que más de un 40% habían registrado un incidente grave en los últimos seis meses.

Es fácil entrar en un sistema no protegido y es muy difícil encontrar a los responsables. Por eso un buen sistema de seguridad debería tener en cuenta las siguientes tres premisas: prevención, detección y reacción. La prevención se cubre actualmente con la utilización de filtros de tráfico como los *firewall*. Es en la detección donde, aún hoy en día queda bastante por avanzar. Es muy importante detectar cuando se está produciendo un ataque y cuanto más rápidamente mejor, puesto que permitiría una reacción pronta ante la amenaza. Es aquí, donde entrarían en juego los sistemas de detección de intrusiones (IDS) que permiten detectar ataques en tiempo real, al mismo tiempo que guarda información sobre ellos, facilitando a posteriori la identificación del pirata.

1.2 INTRODUCCIÓN A LOS IDS

Una intrusión es cualquier conjunto de acciones que pueda comprometer la integridad, la confidencialidad o la disponibilidad de una información o de un recurso informático. Los intrusos pueden aprovechar las vulnerabilidades en la arquitectura de los sistemas y el conocimiento interno de los sistemas operativos para superar el proceso normal de autentificación.

Un IDS (Intrusión Detection System) puede tratarse de un dispositivo de tipo *hardware*, *software* o de una combinación de ambos que alerta de la existencia de actividades sospechosas en el seno de un sistema o de una red de sistemas. La idea central de la detección de intrusiones es que una actividad intrusiva es un subconjunto de actividades anómalas, es decir, si alguien consigue entrar de manera ilegal en nuestro sistema, su conducta no será aquella de un usuario normal. Sin embargo, a veces, una actividad intrusiva es la suma de otras actividades individuales que no constituyen por sí mismas una intrusión. Por tanto, un sistema de detección de intrusiones puede equivocarse y generar:

- Falsos positivos: la actividad no representa una intrusión pero el IDS encuentra que el comportamiento es anómalo y alerta de la existencia de un ataque.

- Falsos negativos. la actividad es intrusiva, pero no se detecta porque el sistema cree que es una actividad normal.

Los IDS recogen y analizan la información proveniente de diferentes fuentes para determinar si existe un ataque o intrusión. Si el IDS tiene certeza de haber detectado un ataque, su función principal es alertar al administrador o al personal de seguridad. Es importante, por tanto, minimizar el número de falsos positivos para reducir el trabajo del IDS, y sobre todo, disminuir el número de falsos negativos puesto que podrían provocar una situación de falsa seguridad.

1.2.1 Tipos de IDS

Los IDS se clasifican en dos tipos fundamentalmente: los IDS de red o NIDS (Network IDS) y los IDS de sistema o HIDS (Host IDS). Los HIDS residen en un puesto o máquina y vigilan todas las tentativas de intrusión sobre dicha máquina. Los NIDS son más corrientes; se encargan de analizar el tráfico que transita por un segmento de la red y que va destinado a todos los puestos conectados a dicho segmento. Un tipo de IDS no es mejor que el otro, cada uno es apropiado a una situación específica.

1.2.1.1 NIDS

Los NIDS se sitúan sobre los segmentos estratégicos en la infraestructura de la red y vigilan el tráfico destinado a otras máquinas gracias al mecanismo de “sniffing” (captura todo el tráfico de la red).

1.2.1.2 HIDS

Los HIDS detectan los ataques perpetrados a nivel del sistema operativo, de una aplicación o del “kernel”. Tienen acceso a los ficheros de “logs” o de registro, a los mensajes de error, a los derechos de los diferentes servicios y aplicaciones y a todos los recursos disponibles sobre la máquina donde están instalados. Esto permite a los HIDS analizar las actividades que se producen con gran precisión, determinando que procesos y que usuarios están implicados en un ataque en particular.

1.2.1.3 Comparación

Los HIDS son más eficaces que los NIDS a la hora de determinar si un ataque se ha llevado a cabo con éxito y presentan la ventaja de generar un menor número de falsos positivos. Los HIDS tienen la capacidad de trabajar con sucesos locales a la máquina en la que se encuentran instalados, por lo que pueden detectar ataques que los NIDS no podrían ver, y, a diferencia de éstos, podrían trabajar en situaciones en las que el tráfico de red estuviera cifrado.

Por otra parte, un NIDS es más rentable que un HIDS ya que puede proteger una parte importante de la infraestructura de la red con un solo dispositivo. Un NIDS ofrece un nivel mayor de seguridad ya que está menos expuesto a las interrupciones que un HIDS. Esto es porque un NIDS no depende de la seguridad de un puesto en particular y, por tanto, es más complicado que sufra una destrucción de pruebas ya que éstas estarían almacenadas en una máquina diferente. Un NIDS debe instalarse en una sola máquina con una configuración reforzada que soporte solamente los servicios dedicados a la detección de intrusiones. Sin embargo, los HIDS pueden ser desactivados por ciertos ataques DoS, influyen en la máquina que los alberga y son incapaces de detectar ataques dirigidos a toda la red como los escaneos de puertos. Los NIDS podrían encontrar dificultades en cuanto al tratamiento de paquetes de red si ésta es grande o presenta una gran cantidad de tráfico.

Las razones anteriores, justificarían el empleo de un modelo híbrido que utilizara tanto NIDS como HIDS consiguiendo así una detección con mejores prestaciones. Los dos modelos de detección pueden representar componentes eficaces en una defensa sólida si son configurados y actualizados correctamente. Un NIDS posee ventajas que le permiten proteger bastante bien grandes partes de la infraestructura de una red. Un HIDS ofrece una protección adaptada a los sistemas cuya misión es esencial.

1.2.2 Métodos de detección de intrusiones

Existen principalmente dos técnicas de detección de intrusiones: por detección de firma o por detección de anomalías.

La detección por firma identifica los sucesos de seguridad que intentan utilizar el sistema de una manera no estándar. Las representaciones de las intrusiones conocidas son almacenadas en el IDS y se comparan con la actividad del sistema o de la red. Estas representaciones se llaman firmas. Cuando se descubre una intrusión en uno de los aspectos de utilización del sistema, el IDS emite una alerta. Las firmas deben ser creadas para reflejar exactamente las características de un ataque específico y no de ninguna otra actividad,

evitando así obtener un gran número de falsos positivos. Esto exige el conocimiento de un ataque a fin de generar la firma correspondiente.

La detección de anomalías identifica una actividad sospechosa midiendo una norma en un determinado periodo de tiempo y generando una alerta cuando el modelo de actividad se aleja de dicha norma. Un IDS por detección de anomalías recupera un conjunto de datos correspondientes a la actividad del sistema o de la red por parte de los usuarios habituales. Dicho conjunto se considera como "la actividad normal". Si un usuario se aleja de dicho modelo de actividad normal, se emitirá una alerta. Este método de detección es mejor a la hora de detectar ataques más sofisticados. Presenta la ventaja de no tener la necesidad de un conocimiento previo de los ataques, pero, al igual que los IDS por detección de firma, tienen tendencia a generar una tasa elevada de falsos positivos.

1.2.3 Tipos de respuesta

La respuesta de un IDS indica cual es la reacción de éste después de haber detectado un ataque. Existen dos tipos: las respuestas pasivas y las activas.

La respuesta pasiva consiste en enviar una notificación del ataque al responsable de la organización, al usuario del sistema o al CERT.

La respuesta activa es una acción automática que se lleva a cabo cuando cierto tipo de ataques son detectados. Por ejemplo, la toma de información adicional aumentando los niveles de sensibilidad de los detectores para obtener más indicios sobre un ataque en particular, o simplemente, el bloqueo de un ataque.

1.2.4 Otras herramientas y complementos a los IDS

1.2.4.1 Sistemas de evaluación y análisis de vulnerabilidades

Estas herramientas de análisis de vulnerabilidades determinan si una red o sistema es vulnerable o no a ataques conocidos y buscan los diferentes servicios y configuraciones con vulnerabilidades conocidas dentro de una red.

1.2.4.2 Herramientas de comprobación de integridad de ficheros

Las herramientas de comprobación de integridad de ficheros o "Files Integrity Checkers" son otro tipo de herramientas de seguridad que complementan los IDS. Utilizan resúmenes de mensajes o técnicas de cifrado para elaborar una síntesis del contenido de los

ficheros o elementos críticos del sistema y detectan las posibles modificaciones realizadas sobre ellos. Este tipo de detección es importante puesto que, con frecuencia, los atacantes modifican los sistemas de ficheros cuando tienen acceso a una máquina dejando puertas traseras o "backdoors" que después facilitarán su entrada de nuevo.

1.2.4.3 Honeypots

Son herramientas de seguridad diseñadas para ser atacadas y que permiten capturar silenciosamente todos los movimientos de un "hacker". Se utilizan para recuperar información sobre el atacante y también en la investigación de nuevos ataques. Los "honeypots" facilitan la incorporación de nuevas firmas para los IDS.

1.3 ESTANDARIZACIÓN

Uno de los problemas actuales de los sistemas de detección de intrusiones es la falta de interoperabilidad de éstos con otras herramientas de gestión de seguridad independientes. La mayoría de los fabricantes diseñan consolas propias a cada uno de los detectores de intrusiones, no permitiendo su integración en un entorno de red heterogéneo donde existen otros dispositivos de seguridad. Esto implica la elección de un sistema de detección de intrusiones de un fabricante en concreto y no del mejor IDS que se adapte a las características del entorno.

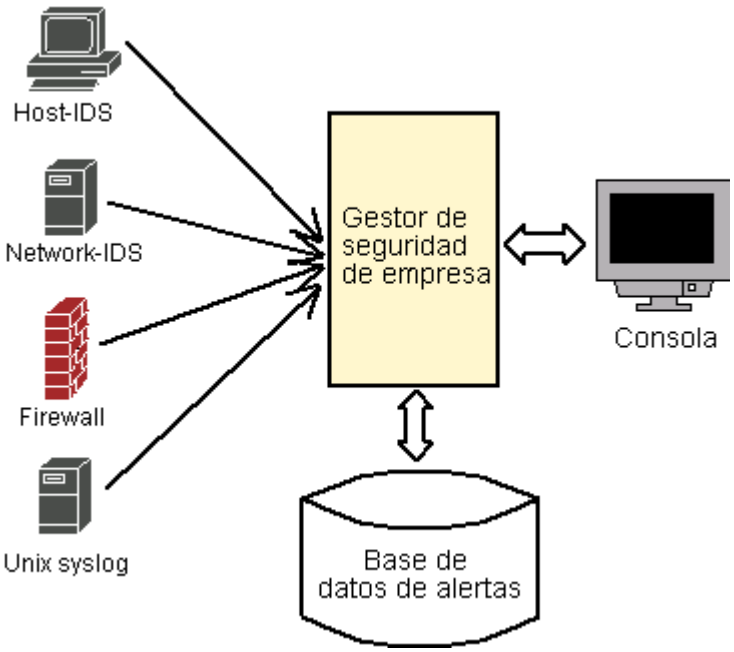


Figura 1.1.- Modelo de gestión de seguridad de una empresa

Esta problemática ha favorecido el desarrollo de diversos intentos de estandarización que permitan al gestor de seguridad de la empresa interpretar e integrar todos los dispositivos de seguridad. Hoy en día, existen distintas organizaciones trabajando en estándares que permitan la comunicación entre los IDS y también con las consolas de monitorización de seguridad.

1.3.1 CIDF

CIDF (Common Intrusion Detection Framework) fué un proyecto iniciado por DARPA (US Governement's Defense Advanced Research Projects). Su objetivo era el de crear interfaces de aplicaciones (API) y protocolos que permitieran la comunicación entre los diferentes IDSs con el objetivo de poder integrarlos dentro de otros sistemas. CIDF no fue un intento de estandarización en sí, sino una investigación motivada por el hecho de que un solo fabricante de IDS no puede cubrir todo el espectro de ataques existentes.

CIDF consiste en un modelo de alto nivel compuesto por una serie de componentes. Estos componentes representan cada una de las funcionalidades que debe realizar un IDS. Se definen estas funcionalidades y sus interconexiones dejando abierto al fabricante la implementación final de éstas. Los componentes principales son:

- Generadores de eventos (equipos E). Describe las funcionalidades de los sensores IDS. Recogen información de la red y envían alertas o informes a la consola de monitorización.

- Analizadores (equipos A). Se encargan de procesar la información obtenida de los sensores y realizar su análisis. Pueden también hacer recomendaciones al administrador, o incluso tomar alguna acción.

- Base de Datos (equipos D). Sería la base de datos donde se almacenaría toda la información relativa a las alertas detectadas por el IDS

- Generadores de respuesta (equipos R). Se encargaría de emitir una respuesta ante los eventos recogidos por los otros componentes (equipos E, A y D), o bien de proponer acciones al operador de la consola.

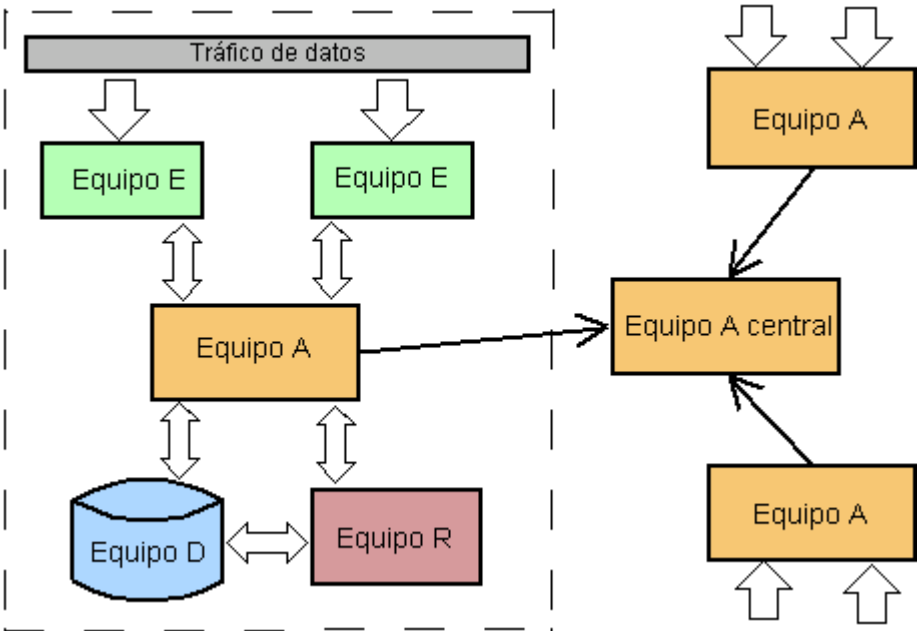


Figura 1.2.- Modelo CIDF

Además de definir protocolos e interfaces de aplicación para la comunicación entre sistemas de detección de intrusiones, CIDF ha desarrollado el lenguaje CISL (Common Intrusion Specification Language) que define un estándar de representación de datos de intrusiones. Este lenguaje surgió de la necesidad de comunicar los cuatro tipos de equipos CIDF. CISL permite expresar tres tipos de información:

- Información de Eventos en Bruto. Es la información bruta obtenida directamente de la red sin sufrir ningún tipo de modificación.

- Resultados de los Análisis. Son las descripciones de los ataques detectados.

- Prescripciones de Respuestas. Son unos conjuntos de acciones de respuesta definidas para ciertos eventos detectados.

Aunque no logró la aceptación como estándar, CIDF ha influenciado otras investigaciones. El desarrollo de CIDF se llevo a cabo entre 1997 y 1999 y muchas de sus ideas han sido tomadas por trabajos posteriores como IDMEF llevados a cabo por IDWG.

1.3.2 IDWG

IETF (Internet Engineering Task Force) es la organización que se encarga de desarrollar los nuevos estándares para Internet. Dentro de dicha organización se creó el grupo IDWG (Intrusion Detection Exchange Format Working Group) para el desarrollo de un

formato común para las alertas de los IDS. Dicho grupo se encarga de "definir formatos de datos y procedimientos de intercambio de datos para compartir información de detección de intrusiones y hacer compatibles las respuestas de los sistemas, y de gestionar los sistemas que puedan interactúan con ellos". Algunos de los objetivos de este grupo son:

- Redactar un documento que especifique los requisitos para la comunicación entre los diferentes IDS, y entre éstos y los sistemas de gestión.

- Desarrollar un lenguaje de intrusiones común con un formato de datos que permita cumplir los requisitos.

- Elaborar un documento que describa los mejores protocolos para la comunicación entre IDSs, y que indique qué relación tienen con los formatos de los datos.

IDWG ha propuesto IDMEF (basado en XML) como formato de los mensajes de intrusiones que permitirá la comunicación entre los distintos sistemas. Los mensajes IDMEF son independientes del protocolo de comunicación, aunque este grupo también ha desarrollado uno propio llamado IDXP.

IDWG ha atraído la participación de muchas empresas como Cisco, NAI, HP, Boeing, IBM, etc. Existen varias librerías que ayudan a construir el modelo de datos de IDMEF y, de hecho, Snort (uno de los IDS de software libre más conocido) utiliza una de estas librerías para generar una salida de alertas IDMEF XML.

1.3.3 IODEF

IODEF (The Incident Object Description and Exchange Format) es un formato para CSIRTs (Computer Security Incident Response Teams) que define un formato de datos y procedimientos de intercambio para compartir información de incidentes entre sus miembros y sus colaboradores. Esto permite proporcionar una base para el desarrollo e interoperabilidad entre herramientas y procedimientos para la notificación de incidentes.

IODEF está basado en el trabajo realizado por IDWG. Los mensajes IODEF son llamados "IncidentAlert" e utiliza los mensajes IDMEF para cubrir un mayor número de detalles.

1.3.4 CERT

Después del incidente sucedido a causa de un "gusano informático" en Noviembre de 1988, DARPA desarrollo el CERT(R)/CC (Computer Emergency Response Team Coordination Center) para proporcionar a la comunidad Internet una simple organización que pudiera coordinar las respuestas a los distintos incidentes de seguridad. La función del CERT es "trabajar junto a los usuarios de la comunidad Internet para facilitar la respuesta de los eventos de seguridad informática incluyendo los puestos Internet, emprender acciones para aumentar la conciencia de los usuarios ante los distintos incidentes de seguridad informática y conducir la investigación y la mejora de la seguridad de los sistemas existentes." El CERT se encarga actualmente de:

- Representar los distintos incidentes contra la seguridad informática.
- Publicar anuncios de seguridad y otras informaciones relacionadas.
- Investigar sobre seguridad de sistemas y redes.
- Responder a las peticiones de información.
- Desarrollar y mantener una base de datos de "conocimientos".
- Proporcionar otros servicios relacionados con la seguridad.

En la figura 1.3 podemos observar como aumenta cada año el número de incidentes comunicados al CERT.

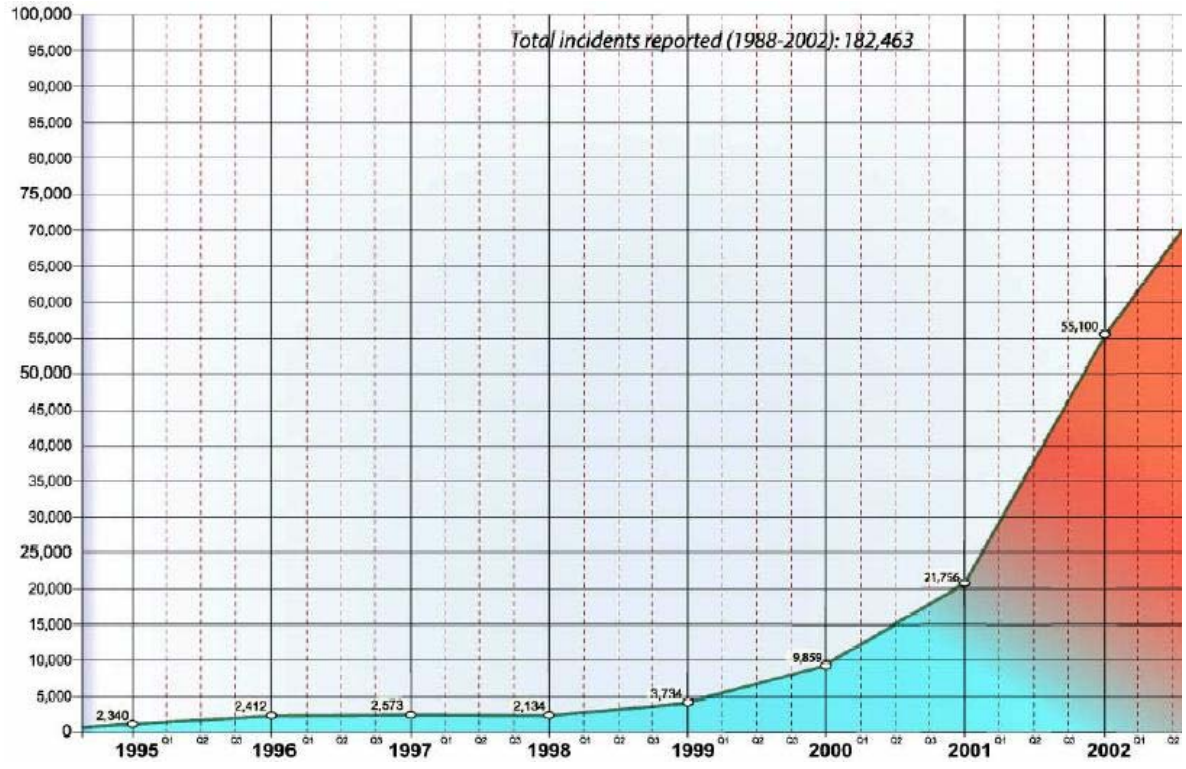


Figura 1.3.- Evolución del número total de incidentes comunicados al CERT entre 1995 y 2002.

1.4 INVESTIGACIÓN Y TENDENCIAS ACTUALES

Aunque los IDS son una herramienta importante en la seguridad de los sistemas informáticos, aún presentan un número importante de carencias. De entre ellas podemos destacar:

- Existen algunos ataques que pueden dejar indisponible un IDS.

- Necesidad continua de actualización. Se necesita un cierto tiempo desde que se descubre un nuevo ataque hasta que se crea una firma para dicho ataque.

- Falta de integración entre los distintos dispositivos de seguridad.

- Necesidad de ajustes continuos para evitar los falsos positivos/negativos. Esto requiere del administrador un alto nivel de dedicación al IDS.

Esto ha propiciado que en la actualidad, las investigaciones vayan dirigidas a intentar solucionar los problemas anteriores.

1.4.1 Inteligencia artificial: minería de datos

Para intentar resolver algunos de los problemas existentes, sobre todo aquellos relacionados con los falsos positivos/negativos, están empezando a aparecer herramientas que hacen uso de técnicas de inteligencia artificial. Las herramientas basadas en estas técnicas utilizan grandes cantidades de datos de entrenamiento para determinar que actividad es normal y cuál es anómala. Posteriormente aplicarán algoritmos a los datos que permitirán clasificarlos como normales o no.

Uno de los problemas principales es llevar a cabo el entrenamiento o proceso de aprendizaje de estas herramientas, puesto que en esta fase se necesitaría que los datos de entrenamiento reflejaran todas las actividades del comportamiento normal del sistema. Para ello, dichos datos deben estar libres de ataques, si no se generaría confusión en el sistema al considerar datos intrusivos como normales. Además debe evitarse que reflejen una actividad local (marcada por una infraestructura de red y sus componentes). Estas características no son fáciles de conseguir.

La "minería de datos" (data-mining) es considerada como una rama de la inteligencia artificial y es actualmente el centro de las investigaciones en cuanto a la detección de intrusiones. Esta técnica permite crear modelos estadísticos de forma automática a partir de los datos analizados (en cambio la estadística clásica utilizaba modelos ya conocidos) e identificar patrones de actividad que describan pautas de conductas (podemos decir que los ataques siguen ciertas pautas de conducta). Estos patrones se utilizarán después para encontrar signos de intrusiones.

La "minería de datos" es aplicada en el campo conocido como KDD (Knowledge Discovery in Databases) cuyo objetivo es "la extracción a partir de los datos, de información implícita, no trivial, previamente desconocida y potencialmente útil".

1.4.2 Correlación de datos

Otras de las tendencias de investigación es la correlación de datos. Se intenta realizar en el tiempo la correlación de los datos obtenidos de diversas fuentes de información para componer un nivel superior de abstracción. Esta técnica está basada en que, normalmente, en una organización coexisten diferentes medios de seguridad (*firewalls*, sistemas de comprobación de integridad de ficheros, analizadores de *logs*, etc...) que pueden aportar información sobre la actividad de un intruso. En el futuro, los IDS deberán ser capaces de relacionar todas estas informaciones obtenidas de diferentes medios para dar una idea global de las actividades que se llevan a cabo dentro de nuestra red.

1.4.3 IPS

Parece ser, que en un futuro próximo los IDS evolucionarán hacia los IPS (Intrusion Prevention Systems) o Sistemas de Prevención de Intrusiones. Un IPS se define como "un dispositivo (*hardware* o *software*) que tiene la capacidad de detectar ataques tanto conocidos como desconocidos y reaccionar ante ellos para impedir su éxito".

Un IPS puede verse como la evolución o combinación de los dos sistemas de seguridad más conocidos, los *firewall* y los IDS. Sus objetivos serían:

- Reaccionar automáticamente ante los incidentes, es decir, a medida que el sistema detecta nuevos ataques, va aplicando nuevos filtros para impedirlos.

- Reducir el número de falsos positivos y facilitar las tareas de administración.

1.5 IDSs COMERCIALES

Actualmente, existen en el mercado multitud de IDS comerciales. En este apartado vamos a realizar una pequeña introducción a los más conocidos. Los IDS comerciales presentan cada vez mejores prestaciones, pero continúan siendo bastantes costosos, lo que provoca que los IDS de distribución libre como Snort (que también tiene un buen rendimiento) sean los más utilizados.

Fabricante	Dispositivo IDS	Características	Precio
Cisco	IDS 4200 series sensors	Hardware. Tráfico hasta 80Mbit/s a 1 Gbit/s. Extensión para catalyst 6000 en forma de tarjeta (250Mbit/s).	De 17600 € a 22500 €
Anzen	Flight Jacket		16000 € por segmento
Easynet	Easynet Secure IDS	Proveedor del servicio. Administrador y supervisión 24h/24. Comunicación periódica de sucesos de seguridad. Gestión de incidentes.	15000 €
Enterasys	Dragon Sensor	Sondas HIDS y NIDS hardware o software. Hasta 1Gbit/s. Análisis de protocolos y firmas. Compatible con Snort.	9450 € (IDS + manager)
Exaprobe	Exaprotect	Sonda software o se provee el servicio. Basado en componentes de código libre (Snort, Nessus, …). Gestión de alertas de las diferentes sondas.	9500 € o 200 €/mes
Intrusion	SecureNet PRO	Hardware. Tráfico 10/100/1000 Mbit/s. Reconstrucción de sesiones en tiempo real. 1600 firmas y 35 protocolos.	De 2695 € a 39995 €
ISS	RealSecure Network sensor	Software. Tráfico de 10-1000 Mbit/s. 1350 firmas (gestión de firmas Snort), 97 protocolos,	De 12400 € (10/100Mb) a

		driver de captura de altas prestaciones.	35700 € (1Gb)
NFR	NID-310	Hardware o software. De 10-1000Mbit/s. Base de datos de firmas y reglas. Consola de administración centralizada.	12000 €
SourceFire	Network sensor	Software. De 22-1000Mbit/s. Basada en una versión mejorada de Snort (por el mismo autor de Snort). Alertas centralizadas por otro componente.	De 5995 € a 16995 € (con consola)
Symantec	Symantec ManHunt	Software. 10/100/1000Mbit/s. Detección y bloqueo de ataques. Detección por firmas y por anomalías. Gestión de firmas de Snort.	12000 €
Open Source	Snort	Software. Detección en protocolos IP, TCP, UDP e ICMP. Detección de Nmap, DoS, desbordamiento de buffer, etc. Reglas actualizadas regularmente.	gratis
Open Source	Prelude-IDS	Software. IDS híbrido (HIDS y NIDS). Centralización de sucesos gracias a un manager. Usa reglas de Snort.	gratis

Tabla 1.1- Comparativa entre diferentes IDSs.

1.5.1 Symantec ManHut

Symantec es capaz de integrar tanto una parte HIDS (Symantec Host IDS) como NIDS permitiendo la gestión de eventos de los IDS basados en red y en *host* desde una única consola. También realiza el análisis, la agregación y la correlación de eventos en tiempo real permitiendo una visión del estado de seguridad global del sistema.

Symantec ManHunt es un IDS basado en red que proporciona múltiples metodologías de detección para detectar ataques tanto conocidos como desconocidos. Puede analizar tráfico hasta 1 Gb/s. Es capaz de detectar protocolos anómalos, firmas, proporciona descripciones del estado del tráfico y análisis de flujo para identificar las intrusiones y ataques por denegación de servicio, con velocidades de hasta 2 *gigabits* por segundo, dependiendo de la configuración del sistema. Permite también el desarrollo de reglas y la definición de medidas o acciones.

Cuenta con una arquitectura basada en estándares pudiendo trabajar de manera conjunta con otros elementos de seguridad existentes, lo que permite tener una gestión centralizada de los distintos medios de seguridad.

1.5.2 ISS, RealSecurity Sensor

Esta gama de sensores puede trabajar con tráfico de 10 hasta 1Gb/s y son analizadores de tráfico de red. Proporcionan detección de intrusiones y capacidades de respuesta dentro de

una red de gestión de sucesos centralizada. Soporta todos los sistemas operativos más comunes. Detecta los ataques basados en protocolo y utiliza firmas para la detección. También proporciona detección de tráfico malicioso.

La instalación de RealSecure Network está centralizada administrativamente y mantenida a través del sistema de gestión "SiteProtector" que puede integrarse con otros productos de protección de la empresa ISS. Trabajando conjuntamente con la aplicación de seguridad X-Force y con X-Press Update (que asegura la actualización) se consigue una detección efectiva de ataques conocidos y desconocidos.

1.5.3 Flight Jacket de Anzen para NFR

Fligh Jacket de Anzen proporciona detección de intrusiones en tiempo real y puede ser personalizado para alertar al administrador determinados tipos de alertas. Es capaz de examinar el tráfico, filtrarlo, identificar ataques y otros sucesos anómalos, y monitorizar todo o un tipo especifico de tráfico en tiempo real, así como guardar las distintas actividades de un ataque. Existe otro componente que permite realizar la gestión centralizada de las sondas de manera remota.

Es uno de los productos IDS más flexible en cuanto a que incluye un potente lenguaje que permite a los usuarios definir virtualmente cualquier evento que puedan imaginar. Cuenta con un "sniffer" programable que facilita la adaptación del sensor a diferentes tipos de arquitecturas de red y con una interfaz para escritura de filtros sencilla de utilizar (con un lenguaje simple). Sin embargo, no permite la posibilidad de configurar acciones para reaccionar ante los ataques.

1.5.4 SecureNet PRO

SecureNet PRO es un sistema escalable de detección de intrusiones basado en red. Captura, analiza y reconstruye toda la actividad TCP/IP de la red en tiempo real. Puede tratar tráfico de hasta 1Gb/s dependiendo de la sonda. Es capaz de monitorizar, analizar y guardar cualquier transmisión de red relacionada con la detección de un ataque.

Este IDS cuenta con un lenguaje que permite personalizar las reglas de detección. Realiza detección por firmas y es capaz de decodificar numerosos protocolos. Toma acciones como la terminación de sesiones TCP o el envío de alertas vía mail. La gestión de los sensores puede ser gestionada de manera centralizada por una consola gráfica. Las comunicaciones entre todos los componentes del IDS serían cifradas. Cuenta también con una consola para visualización de eventos.

1.5.5 Enterasys, Dragon Sensor

Enterasys Dragon constituye un moderno dispositivo de seguridad de redes proporcionando sensores de alta velocidad para detectar y responder ante actividades sospechosas, análisis forense de datos para determinar el impacto de un ataque en la red y posibilidad de escalado y gestión de un gran número de sensores sin un impacto negativo en la red.

El IDS combina tres dispositivos, sondas NIDS, sondas HIDS y un servidor IDS para la gestión centralizada de alertas. Enterasys Dragon puede poner en marcha una variedad de respuestas activas contra los ataques detectados. Es capaz de combinar diferentes eventos provenientes de otros dispositivos de la red (hosts, firewalls, routers, switchs, otros IDS y aplicaciones) proporcionando una detección completa para pequeñas y grandes redes.

1.5.6 Cisco, IDS 4200 Series Sensors

Los sensores de la serie Cisco IDS 4200 trabajan junto a otros componentes para proteger la infraestructura de datos e información. Es un componente hardware y permite proteger con el mismo dispositivo varias subredes, ya que cuenta con varias interfaces de "sniffing" (es como si se tuvieran varios sensores en uno).

Los sensores de Cisco utilizan una combinación de innovadoras y sofisticadas técnicas de detección, incluyendo reconocimiento de patrones, análisis de protocolo, detección heurística y detección de anomalías, que proporciona detección de una gran variedad de ataques tanto conocidos como desconocidos. La tecnología de estos sensores permite a su vez la programación de firmas propias.

Cuando se detecta una actividad no autorizada, el sensor puede enviar alertas a una consola de gestión con los detalles de la actividad. Adicionalmente, un sistema de respuestas activas (Cisco IDS Active Response System) puede enviar órdenes a otros sistemas como *routers*, *firewalls*, *switchs*, etc. con el fin de controlar dichas actividades.

1.6 IDS DE CÓDIGO LIBRE

1.6.1 SNORT

Snort es un IDS o Sistema de detección de intrusiones basado en red (NIDS). Está constituido por un motor de detección de ataques y barrido de puertos que permite guardar, alertar y responder ante cualquier anomalía que hubiera sido definida anteriormente, como por

ejemplo, patrones de ataques y tentativas de ataque aprovechando las vulnerabilidades del sistema o los análisis de protocolo. Snort realiza todas estas tareas en tiempo real.

Snort está disponible bajo licencia GPL. Es gratuito y funciona tanto en plataformas Windows cómo UNIX/Linux. Dispone de numerosos filtros y patrones de ataques ya definidos, así como de actualizaciones constantes en caso de nuevos ataques o vulnerabilidades, que son continuamente publicados en los boletines de seguridad. Además, Snort utiliza un lenguaje de creación de reglas flexible, potente y simple.

1.6.1.1 Modos de funcionamiento

Snort puede trabajar en tres modos diferentes: *Sniffer*, *Packet Logger* y NIDS. En modo *Sniffer*, lee los paquetes de la red y los muestra en modo continuo en el monitor. Como *Packet Logger* almacena los paquetes en un disco. El modo NIDS es el más complejo y configurable. Permite el análisis del tráfico de la red para compararlo con un patrón o regla definida y desarrolla acciones dependiendo de los resultados de dicho análisis.

1.6.1.2 Arquitectura

Snort se puede dividir en cinco grandes módulos, todos ellos esenciales para la detección de intrusiones. Para capturar paquetes, Snort llama a una librería de captura de paquetes externa (libpcap). Una vez que los paquetes son capturados son transferidos a un decodificador de paquetes. Ésta representa la primera etapa de la arquitectura de Snort. El decodificador transforma los elementos de protocolos específicos en una estructura de datos interna. Estos datos serán tratados por los preprocesadores o módulos de detección. Ellos se encargarán de examinar y manipular los paquetes antes de transmitirlos al siguiente módulo, el motor de detección. Este motor realiza tests simples sobre cada paquete para detectar las intrusiones. El último componente son los módulos de salida, es decir, aquellos que transmiten las alertas para avisar de actividades sospechosas. La figura 1.4 muestra el flujo de datos en la arquitectura Snort.

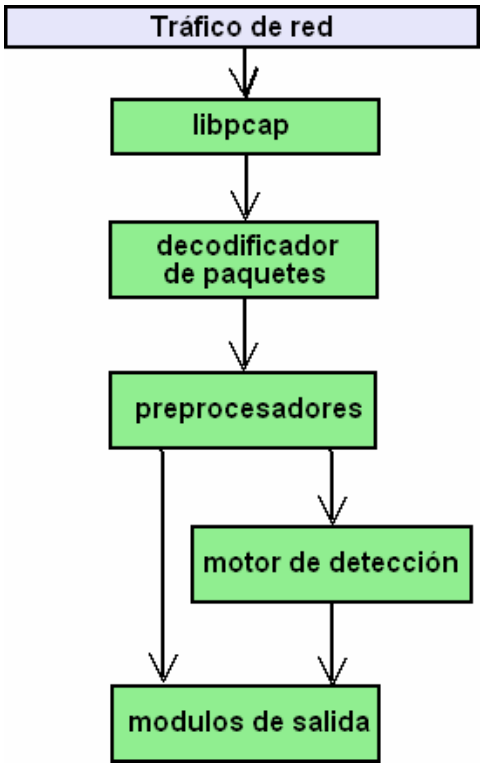


Figura 1.4.- Flujo de datos en la arquitectura Snort

Libpcap

Snort no dispone de una herramienta de captura de paquetes propia, por lo que se sirve de esta librería externa para realizar dicha tarea. Se ha elegido esta biblioteca por su independencia en cuanto a la plataforma. Se puede ejecutar en todos los diferentes tipos de procesadores y sistemas operativos. Gracias a esta adaptabilidad de *libpcap*, Snort es una aplicación independiente de la plataforma.

Libpcap es el responsable de extraer los paquetes directamente de la tarjeta de red en forma bruta, es decir, con toda la información original sin que sufra ninguna modificación por parte del sistema operativo. Snort necesita trabajar con los paquetes en forma bruta, por ejemplo, consulta las cabeceras de protocolo (que hubieran sido suprimidas por el sistema operativo) para detectar ciertos tipos de ataques.

La utilización de *libpcap* no es, sin embargo, el medio más eficaz de obtener paquetes en forma bruta, ya que no puede tratar más de un paquete a la vez. Esto constituye un cuello de botella cuando se quiere vigilar una red con un ancho de banda grande (1 Gbps). Posiblemente, en un futuro próximo, Snort utilice librerías específicas a cada sistema operativo.

Decodificador de paquetes

Una vez que los paquetes son recuperados, Snort debe decodificar los distintos elementos de los paquetes correspondientes a cada uno de los diferentes protocolos. Snort examina cada una de las capas de la torre de protocolo de red, comenzando por los protocolos de enlace de datos de bajo nivel y decodificando cada protocolo de las capas inmediatamente superiores en la torre de protocolo.

A medida que los datos atraviesan los diferentes decodificadores de protocolo, los paquetes decodificados se almacenan en una estructura de datos, que será posteriormente analizada por los preprocesadores y el motor de detección.

Preprocesadores

Los preprocesadores se pueden clasificar en dos categorías. Su misión es, o bien examinar paquetes en busca de actividades sospechosas, o bien, modificar los paquetes de forma que el motor de detección los pueda interpretar correctamente. Existe un cierto número de ataques que no pueden ser detectados por la búsqueda o comparación de firmas utilizando el motor de detección, lo que hace que los preprocesadores sean indispensables para descubrir ataques no identificables mediante un patrón o firma. Otros preprocesadores tienen por objetivo manipular los paquetes de manera que el motor de detección pueda buscar precisamente las firmas y hacen fracasar todo ataque que intente infiltrarse en el motor de detección de Snort manipulando las estructuras de protocolo.

Los paquetes examinados por los preprocesadores vuelven a ser examinados por el motor de detección, a fin de encontrar ataques que un preprocesador específico no haya podido detectar. Si esto no fuera así, los atacantes podrían utilizar esta cualidad para enmascarar sus ataques. Por ejemplo, un atacante puede enmascarar el código de un ataque de explotación distante en un paquete que hiciera saltar una alerta en un preprocesador. Si no se siguiera examinando el paquete, no se descubrirían las verdaderas intenciones del atacante y el ataque de explotación remota se llevaría a cabo.

Los distintos preprocesadores son configurables en el fichero de configuración de Snort (*snort.conf*) y pueden añadirse o eliminarse según las diferentes necesidades. Los principales preprocesadores son:

- Frag2. Permite detectar ataques basados en la fragmentación de paquetes.

- Stream4. Con este preprocesador se puede examinar el estado del flujo TCP. Es capaz de detectar irregularidades en el flujo TCP introducidas por los atacantes con la intención de enmascarar una tentativa de ataque de recuperación de información,

diferentes tipos de escaneos de puertos o incluso ataques “stealth” destinados a crear un ataque por denegación de servicio a Snort.

- Stream4_reassemble. Permite unir sesiones TCP.

- HTTP_decode. Encargado de detectar el tráfico HTTP anormal y de normalizarlo, de manera que el motor de detección lo pueda interpretar correctamente.

- RPC_decode. Funciona igual que el preprocesador anterior, pero esta vez para el protocolo RPC.

- BO. Este preprocesador detecta “Back Orifice”, que es un caballo de Troya bastante popular.

- Telnet_decode. Se encarga de la decodificación específica de protocolos Telnet y FTP.

- ARPspoof. Se encarga de detectar tráfico ARP malicioso.

- ASN1_decode. Concebido para detectar incoherencias en el protocolo ASN.1, que puedan indicar un comportamiento sospechoso. Este protocolo es utilizado por numerosos protocolos de nivel superior LDAP, SNMP, SSL y X.509.

- Fnord. Este preprocesador es concebido para detectar ataques utilizando código polimórfico.

- Conversation. Se usa junto al procesador Portscan2 y se encarga de gobernar las conversaciones para todo protocolo IP (tal que ICMP). Es útil para detectar el origen de las comunicaciones y en que instante comienzan.

- Portscan2. Detecta las tentativas de reconocimiento, emitiendo una alerta cuando constata que los paquetes se emiten a más de 4 puertos diferentes en menos de 3 segundos.

- SPADE. Este preprocesador utiliza métodos de detección por anomalías. Para detectar tráfico sospechoso, SPADE construye una tabla que describe los modelos de tráfico normal. En cuanto el tráfico se aleja de este esquema de comportamiento, hace saltar una alerta.

Motor de detección

El motor de detección es el principal componente de Snort. Se encarga principalmente de dos tareas: el análisis de reglas y la detección de firmas. El motor de detección crea las firmas de ataques analizando las reglas de Snort. Estas reglas son leídas línea por línea y son cargadas en una estructura de datos interna cada vez que se inicia la aplicación Snort.

El motor de detección aplica al tráfico el conjunto de reglas en el orden en el que éstas han sido cargadas en memoria. Este orden puede ser impuesto por nosotros modificando la prioridad de las reglas y organizándolas de manera apropiada. Las reglas se componen principalmente de dos partes: la cabecera y la opción. La cabecera de la regla contiene información concerniente a las condiciones de aplicación de la regla. Se trata del tipo de protocolo, el origen, el conjunto de direcciones IP destino y los puertos. Véase aquí, un ejemplo de regla para una explotación distante CRC32 OpenSSH:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22
```

La parte de opciones de la regla contiene la firma real, el nivel de prioridad y la documentación concerniente al ataque. Por ejemplo:

```
(msg: “EXPLOIT ssh CRC32 overflow /bin/sh”; flow:to_server,established; content:“/bin/sh”;
reference:bugtraq,2347; reference:cve,CVE-2001-0144; classtype:shellcode-detect; sid:1324;
rev:3;)
```

El motor de detección no trata la parte de cabecera y la de opciones de la misma manera. Éste construye un árbol de detección utilizando listas encadenadas como las que aparecen en la figura 1.5.

Por ejemplo, el motor de detección comienza por determinar si un paquete es o no TCP. En el caso de que fuera TCP, el paquete se transmite a la parte del árbol que contiene las reglas correspondientes al protocolo TCP. Snort, determina entonces si el paquete corresponde a una dirección origen dentro de una regla. En este caso, el motor carga el conjunto de reglas correspondientes. Se sigue este proceso hasta que se descubra una firma de ataque o bien hasta que el paquete sea considerado como inofensivo y se suprima de la estructura de datos. Snort comenzará a analizar otro paquete después de haber encontrado una firma correspondiente al anterior, lo que implica, que aunque otra firma hubiera podido ser encontrada para ese mismo paquete, el motor de búsqueda pasa al paquete siguiente. Por eso es muy importante organizar las reglas de manera que las firmas que representen mayor gravedad sean cargadas antes.

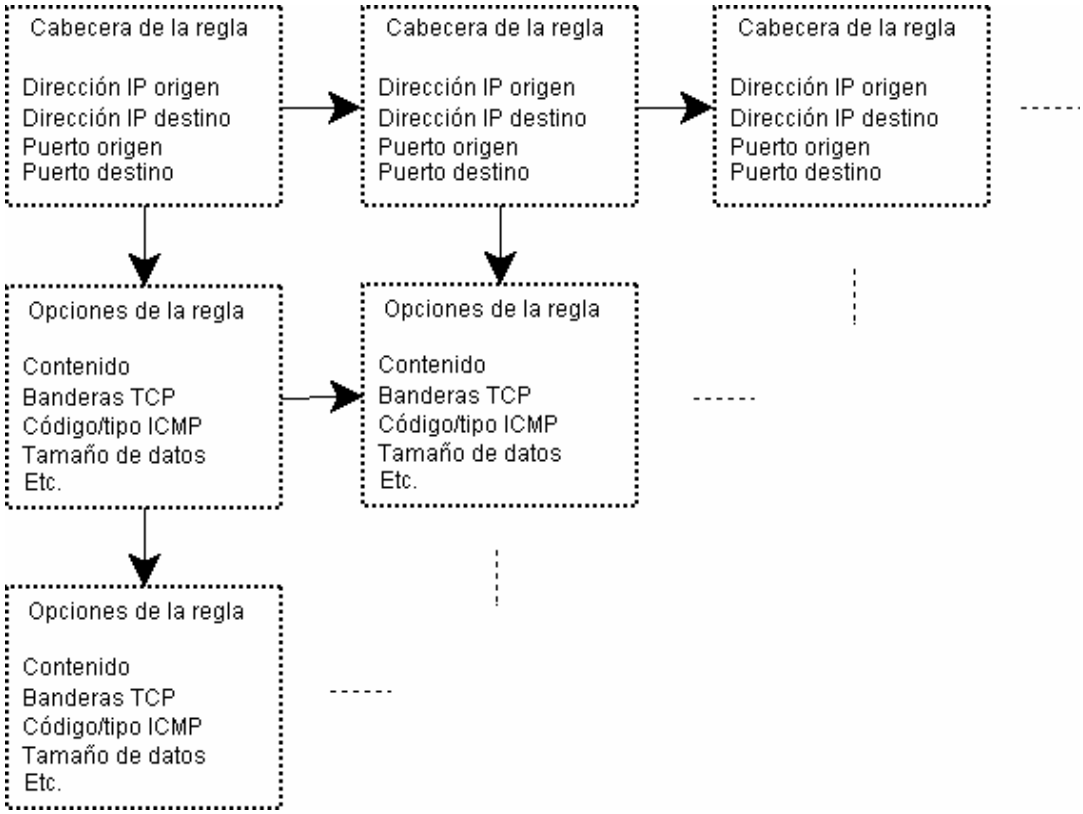


Figura 1.5.- Sistema de detección de firmas de Snort basado en árbol y listas encadenadas.

Módulos de salida

Snort presenta los datos de intrusión mediante los módulos de salida. La misión de estos módulos es guardar los datos referentes a las alertas en un otro fichero o recurso. Se pueden activar a la vez varios de estos módulos de salida. Existen numerosas aplicaciones externas, diseñadas para trabajar exclusivamente con Snort, para la administración de los datos de intrusión.

Los módulos de salida pueden representar un problema de rendimiento si, por ejemplo, el medio donde se guardan los datos es lento, como las bases de datos. Por este motivo, en entornos donde se utiliza un gran ancho de banda no se suelen utilizar. Normalmente, se recomienda utilizar el formato de salida *unified* y utilizar la aplicación Barnyard para guardar los datos en el recurso deseado.

Existen doce módulos de salida:

- alert_fast. Es el mecanismo de salida más simple y rápido que posee Snort. Guarda las alertas en un fichero, en una sola línea a la velocidad a la que el motor de búsqueda las transmite. Snort no escribe la información de los paquetes.

- alert_full. Crea un repertorio por cada IP que genera una alerta y almacena los paquetes decodificados. Incluye información sobre el paquete (cabeceras y datos).

- alert_smb. Transmite las alertas en entornos Windows abriendo una ventana por cada alerta. Es poco práctico.

- alert_unixsock. Este módulo instala un *socket* en los dominios Unix y le envía las alertas. Así, otros programas externos pueden escuchar y recibir alertas Snort.

- log_tcpdump. Guarda los paquetes en un fichero en formato *tcpdump*.

- CSV. Transmite la salida a un fichero cuyo delimitador es una coma. Esto permite que este tipo de ficheros sea fácilmente exportable a bases de datos u hojas de cálculo.

- XML. Permite guardar la salida en SNML. Este lenguaje se puede utilizar para reunir los datos de diferentes detectores y almacenarlos en una sola base de datos.

- alert_syslog. Escribe los informes de alertas en la función *syslog*. Esto permite mejorar las posibilidades de correlación de alertas con otros eventos para la identificación de problemas.

- database. Este módulo envía los datos de intrusión a una base de datos relacional. Podemos elegir entre MySQL, PostgreSQL, Oracle o entre las bases de datos compatibles con UnixODBC. El almacenamiento en una base de datos permite acceder a numerosos datos de intrusión, seleccionar alertas, realizar búsquedas u organizar los datos dependiendo de distintas variables como la prioridad. Además hay varias aplicaciones que utilizan los datos de intrusión de una base de datos para crear una interfaz gráfica de administración (ACID por ejemplo).

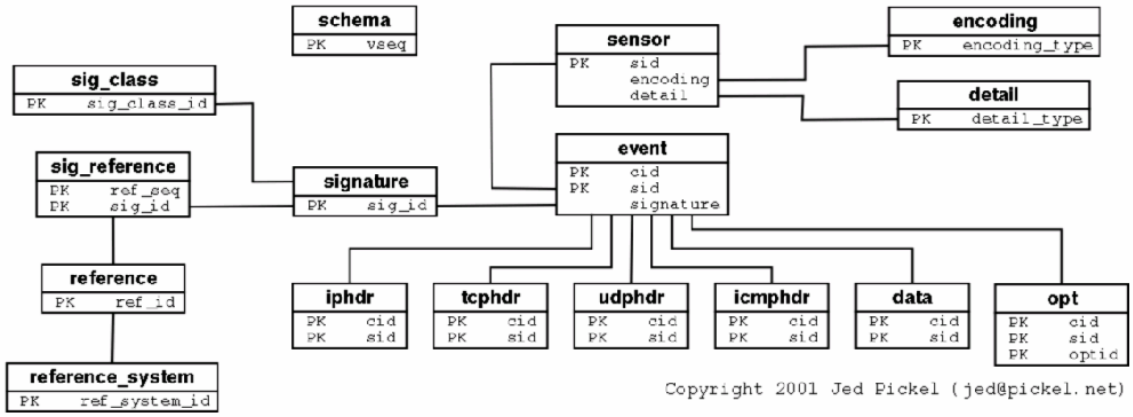


Figura 1.6.- Estructura de la base de datos de Snort.

- unified. Representa el método más rápido para transmitir los datos de intrusión de Snort, ya que guarda los datos en su propio formato binario. Su objetivo es extraer los datos de Snort lo más rápido posible y transmitirlos a una aplicación dedicada, en este caso, Barnyard. El módulo de salida crea dos archivos: uno de alertas y otro de paquetes. El primero contiene un resumen de las alertas y almacena sólo las direcciones IP, el protocolo, los puertos y el mensaje de identificación de la alerta, mientras el segundo contiene toda la información del paquete sospechoso.

1.6.2 PRELUDE-IDS

Prelude-IDS es un sistema de detección de intrusiones híbrido, es decir, realiza tanto las funciones de un IDS basado en red (NIDS) como las de uno basado en sistema (HIDS). Realiza la detección de intrusiones mediante el análisis del tráfico de la red utilizando firmas de ataques (basadas en las reglas de Snort) y realizando un análisis continuo de los ficheros de registros (ficheros de *logs*).

1.6.2.1 Arquitectura

Prelude-IDS presenta una arquitectura modular (permitiendo la integración de nuevas funcionalidades), distribuida (Prelude-IDS es una serie de componentes autónomos e interactivos, sensores y *managers*) y segura (utiliza soporte SSL para la autentificación y la encriptación de las comunicaciones). Los sensores (tanto de red como locales) efectúan solamente operaciones de vigilancia de tráfico y generación de alertas, mientras que los *managers* se ocupan de la gestión de los sensores y del registro de alertas.

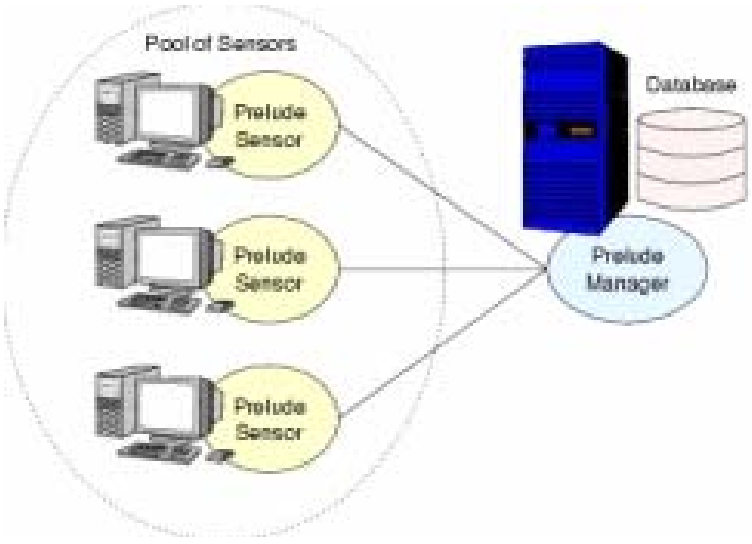


Figura 1.7.- Modelo de arquitectura Prelude-IDS

Básicamente, la arquitectura de Prelude-IDS se compone de cuatro módulos bien diferenciados: *libprelude*, sensores, *managers* y consolas.

Libprelude

La librería *libprelude* constituye la base de todos los módulos de Prelude-IDS a excepción de las consolas. Permite la comunicación entre los diferentes componentes utilizando el protocolo IDMEF. Gracias a este protocolo, *libprelude* es capaz de integrar otras aplicaciones (no sólo las propias a Prelude). Esta librería proporciona al resto de componentes las funciones siguientes:

- Gestión de la conexión entre los módulos (sensores y *managers*). Periódicamente se realiza un intento de conexión con el manager. Si no se puede establecer esta conexión, se elige otro manager. Si no responde ningún manager de los que el sensor está configurado para comunicarse, entonces las alertas son guardadas en un fichero para ser transmitidas más tarde.

- Gestión del modo de comunicación entre los componentes en cuanto a la encriptación y la autentificación de las comunicaciones.

- Actúa como interfaz, permitiendo la integración de diferentes módulos.

Es fundamental haber instalado esta librería para que el resto de componentes puedan funcionar correctamente.

Sensores

Los sensores son los responsables de detectar las actividades sospechosas en tiempo real tanto en cada puesto como en la red. La arquitectura modular de Prelude-IDS permite integrar varios tipos de sensores diferentes, entre ellos Snort o Nessus (sistema de evaluación y análisis de vulnerabilidades). Sin embargo, vamos a destacar aquellos que son propios de Prelude-IDS:

- Prelude-NIDS. Se encarga de realizar el análisis en tiempo real del tráfico en la red. Utiliza la librería *libprelude* y se compone de un motor de gestión de firmas compatible con las de Snort (pudiéndose añadir también otros tipos de reglas), de módulos especializados para el análisis de tráfico perteneciente a determinados protocolos (RPC, HTTP, etc) y de módulos especializados en la detección de ataques no basados en firmas como los escaneos de puertos. También tiene en cuenta situaciones como la desfragmentación IP y el ensamblado de tramas TCP, de manera que los sensores no sean vulnerables a los ataques de tipo “stealth”. Cuando se recibe

un paquete, el sensor decodifica las cabeceras del paquete y las almacena en una estructura interna de datos. Aquí, se realizan una serie de pruebas para comprobar si los paquetes son válidos o están malformados con la intención de inestabilizar el sistema. Si se detecta una anomalía, se descarta el paquete. Después de estas comprobaciones se comienza con la desfragmentación IP y la unión de tramas TCP. Después, los paquetes son transmitidos a los diferentes módulos:

  - Decodificadores HTTP, FTP, TELNET y RPC.
  - Módulo de detección SnortRules.
  - Módulo de detección de escaneos.
  - Módulo de detección de código polimórfico.
  - Módulo de detección ArpSpoof.
  - Módulo de detección de fragmentación de pila IP.
  - Módulo de ensamblado de flujo TCP.

- Prelude-LML. Este sensor se encarga de enviar las alertas detectadas localmente, sobre el sistema operativo en el que está instalado. No obstante, también puede monitorizar mensajes *logs* provenientes de otros sistemas de la red (si está configurado para aceptar mensajes *syslog* de la red) Esta detección es el resultado de la aplicación de un conjunto de reglas a determinados ficheros de *logs*. En el fichero de configuración se puede indicar que ficheros analizar y que reglas utilizar.

- Libsafe. Es una librería que sólo funciona en sistemas Linux y su función es proteger a un programa contra la explotación de vulnerabilidades como desbordamientos o cadenas con formato falso.

Managers

Prelude-manager recoge los mensajes que emiten los distintos sensores y los traduce en alertas. Es el responsable de la centralización y el registro de las alertas. Se encarga de seis funciones principales:

- Trabaja como servidor. Detecta la presencia de mensajes y los lee. Para interpretar dichos mensajes puede utilizar diferentes módulos de decodificación además de IDMEF.

- Organiza los mensajes siguiendo criterios de prioridad.

- Toma medidas o acciones. Puede tratar las alertas y decidir si llama a otros módulos capaces de reaccionar contra un ataque.

- Almacena las alertas. Prelude-manager se encarga de almacenar las alertas en los diferentes recursos posibles (en una base de datos MySQL, PostgreSQL o Oracle y en ficheros en formato texto o XML).

- Actúa como sistema de relevos. Puede enviar a su vez las alertas recibidas a otros *managers*.

Prelude-Frontend (consolas)

Prelude-Frontend es la interfaz que permite visualizar las alertas. Actualmente existen dos interfaces ligadas a Prelude, una desarrollada en PHP y otra desarrollada en Perl.

1.6.3 Comparación entre Snort y Prelude-IDS

Se puede decir que en cuanto al tipo de proyecto, tanto Snort como Prelude-IDS se parecen mucho. Ambos son herramientas de libre distribución. Los proyectos de los que han surgido están muy activos tanto en el desarrollo de código como en la actualización de las firmas de ataques.

Algunas ventajas de Snort sobre Prelude-IDS son su popularidad y su disponibilidad sobre numerosas plataformas. Prelude-IDS se restringe a las plataformas POSIX, mientras que Snort lo podemos encontrar también sobre Windows. A esto se une la importancia de la base de datos de firmas de ataques de Snort, que explica su gran popularidad. Prelude-IDS se aprovecha de esta característica y utiliza esa misma base de datos.

Una diferencia importante, en desventaja de Snort es que éste es un NIDS puro, mientras que Prelude-IDS integra las dos funcionalidades, tanto la de NIDS como de HIDS. Es decir, Prelude es un IDS híbrido. De aquí en adelante, sólo podremos comparar la parte que es comparable, es decir, la parte NIDS de Prelude y Snort.

1.6.3.1 Principio de funcionamiento

El modo de funcionamiento es similar en ambos casos. Las dos herramientas hacen análisis en busca de similitudes con un escenario previamente definido.

Prelude-IDS tiene la ventaja de ser muy modular en cuanto a su arquitectura cliente-servidor. Un manager puede gestionar varios sensores, y un sensor puede enviar sus alertas a varios *managers*. Para ello, sigue los estándares XML e IDMEF.

1.6.3.2 Las alertas

Tanto Snort como Prelude-NIDS se basan en la misma base de datos de firmas de ataque, por lo tanto, ambos generan más o menos el mismo número de alertas del mismo tipo. La diferencia está en las alertas generadas por los preprocesadores en el caso de Snort o por los módulos de decodificación en el caso de Prelude-NIDS. Según las pruebas aplicadas a condiciones reales que se están llevando a cabo, se podría decir que los sensores Prelude generan un número mayor de alertas (esto también puede significar una generación mayor de falsos positivos).

Ambos guardan bastante información sobre el paquete sospechoso, permitiendo conocer el origen y destino de un ataque, su tipo, su importancia, e incluso un enlace hacia un boletín oficial de alertas. Los dos archivan la carga útil de los datagramas sospechosos. Esto permite distinguir más fácilmente que alerta es un falso positivo y cual no.

1.6.3.3 Configuración

Los dos IDS son configurables. Prelude-IDS es más fácil de configurar que Snort, lo que se explica en el menor número de elementos configurables con los que cuenta. Es decir, con Snort podemos ajustar un mayor número de parámetros de manera que podemos ajustar más finamente el funcionamiento del IDS a nuestra red. También se debe a que Snort cuenta con un mayor número de módulos de detección o preprocesadores.

Además, existen herramientas gráficas que permiten configurar Snort, mientras que en el caso de Prelude esto no es posible por el momento

1.6.3.4 Los frontends

Las diferencias más significativas entre las dos herramientas están a este nivel:

- Número de *frontends* disponibles. Prelude no tiene realmente un *frontend* oficial. Los dos más conocidos son, un *frontend* PHP y un otro *frontend* Perl, llamado PIWI que es

el más utilizado y el que ofrece más prestaciones. Será éste último el que vamos a comparar con el *frontend* oficial de Snort, ACID. Snort, también cuenta con varias interfaces, así como numerosas herramientas de gestión gracias a su popularidad.

- Clasificación y agrupamiento de alertas. En ambos casos, el agrupamiento de alertas sigue los mismos criterios (misma dirección IP, mismo tipo de ataque, misma importancia de las alertas, etc.). Para filtrar alertas con más detalle, los dos *frontends* siguen lógicas diferentes. ACID permite seleccionar y buscar alertas siguiendo diferentes criterios. Para realizar esta misma tarea, PIWI permite realizar filtros, modificarlos y suprimirlos, teniendo la posibilidad de guardarlos.

- Gráficos. Ambos *frontends* permiten la visualización de estadísticas de alertas en modo gráfico. Aunque la monitorización de alertas sea equivalente en ambos casos, PIWI presenta una interfaz más amena.

- Mantenimiento de la base de datos. Las alertas pueden ser almacenadas, o bien en ficheros, o bien en una base de datos. Es interesante poder gestionar el contenido de la base de dato mediante la interfaz. ACID permite esta posibilidad, pudiendo borrar, copiar o mover de base de datos las alertas. Con PIWI, la única tarea que se puede realizar es el borrado de alertas sobre la base de datos. Sin embargo lo hace de manera menos eficiente que ACID. Esto implicará en algún momento la administración manual de la base de datos mediante peticiones SQL. Éste es el principal inconveniente que presenta la interfaz de Prelude.

1.6.3.5 Integración de otras herramientas

Uno de los puntos fuertes de Prelude-IDS es el de poder integrar funcionalidades de otras herramientas de seguridad. Por ejemplo, podemos utilizar como sensor Honeyd, Nessus e incluso Snort, que enviarían los resultados al manager. Éste integraría los datos en la misma base de datos.

El conjunto de reglas de Snort, puede ser, como hemos visto, integrado por Prelude-IDS. Por lo tanto, las numerosas firmas de ataques recocidas por Snort (y que son una de las causas de su popularidad) beneficia también a Prelude-IDS.

La integración de las vulnerabilidades detectadas por Nessus a los informes de alertas de Prelude puede resultar muy interesante. También se pueden integrar otras herramientas como Libsafe o Systrace.

1.6.3.6 Documentación y aplicaciones

Hay numerosa documentación sobre Snort y bastante completa. Sin embargo, en el caso de Prelude es más escasa y menos fiable.

Por otro lado, debido a la popularidad de Snort, podemos encontrar numerosas aplicaciones que han sido diseñadas para trabajar con este NIDS.

1.7 ATAQUES INFORMÁTICOS

Podemos definir un ataque como cualquier acción que represente una violación de la seguridad de un sistema. Los ataques pueden ser clasificados, a primera vista, en dos categorías:

- Ataques pasivos. Consisten en escuchar sin modificar los datos o el funcionamiento de la red. Son generalmente indetectables, pero existen formas de prevenirlos como por ejemplo el cifrado de la información.
- Ataques activos. Consisten en modificar los datos o mensajes, en introducirse en los equipos de red o en perturbar el funcionamiento de la misma. Además, no existe generalmente prevención posible ante estos ataques aunque normalmente son detectables (permitiendo a posteriori una respuesta adecuada).

Los ataques más importantes pueden clasificarse en cuatro grupos:

- Escaneos de puertos.
- Ataques contra la autentificación.
- Denegación de servicio (Denial of Service, DoS).
- Explotación de errores.
- Ingeniería social.

1.7.1 Escaneos de puertos

Un escáner es un programa que detecta automáticamente los puntos débiles en la seguridad de una máquina remota o local. Los escáneres interrogan los puertos TCP o UDP de un equipo objetivo y guardan su respuesta. De esta manera, pueden recoger informaciones útiles entre las que destacan:

- los servicios que se están ejecutando en ese momento preciso.
- los propietarios de dichos servicios.
- si son aceptadas las conexiones anónimas.
- si ciertos servicios de red requieren una autentificación.

Prácticamente todos los escáneres de puertos están basados en el "handshake", es decir, en el proceso que siguen dos equipos (cliente y servidor) para reconocerse antes de comenzar con el intercambio de datos. Este proceso se realiza con el protocolo TCP y se realiza generalmente en tres pasos o etapas:

1. El cliente demanda al servidor el inicio de una conexión enviándole un paquete TCP con el campo SYN activado (donde se indica también el número de puerto destino).

2. Si el servidor está escuchando, le envía un acuse de recepción mediante una paquete conteniendo SYN+ACK (significaría que acepta la conexión) y si no, le envía un paquete RST que rompe la conexión.

3. El cliente comprueba la respuesta del servidor y envía un ACK para aceptar la conexión.

Si se produce finalmente un intercambio de información, cuando éste acaba, se llevan a cabo otros tres pasos idénticos para cerrar la conexión sustituyendo el mensaje SYN por uno de FIN.

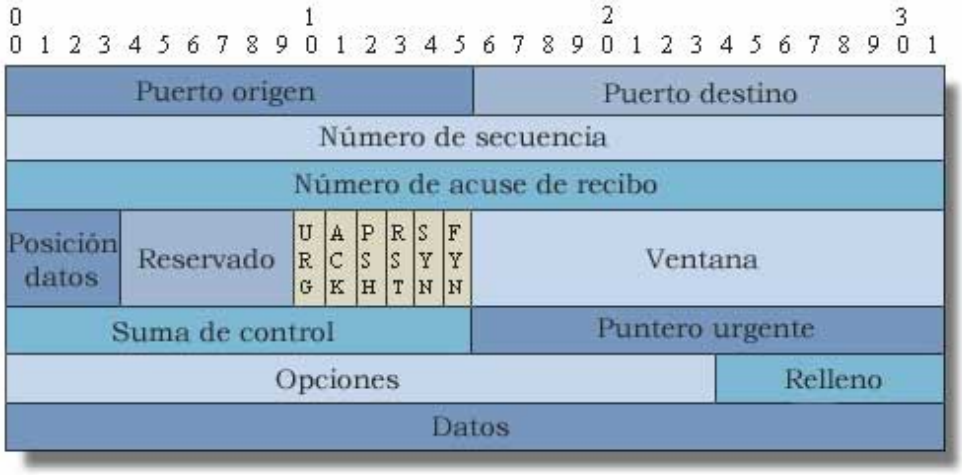


Figura 1.8.- Datagrama TCP.

Los tipos más importantes de escaneo pueden englobarse dentro de las cuatro técnicas siguientes:

- Escaneos "open". Se basan en el establecimiento de una conexión TCP completa mediante el "handshake". Este tipo de escaneo es conocido como TCP connect(). Consiste en enviar una petición de conexión *connect()* y si el puerto que se busca está abierto y escuchando entonces se recibirá un SYN/ACK y se enviará un ACK para establecer la conexión. Si, en cambio, se recibe una respuesta RST/ACK significará que el puerto está cerrado. Es una técnica sencilla y fiable que no requiere ningún privilegio especial en la máquina atacante.

- Escaneos UDP. Escanea los servicios que permiten tráfico UDP. Se envían paquetes UDP de 0 bytes a cada puerto del equipo remoto. Se supone el puerto abierto a no ser que se reciba un paquete ICMP de puerto no alcanzable.

- Escaneos "half-open". En este tipo de escaneos, el atacante finaliza la conexión antes de que se completen los tres pasos del "handshake", dificultando un poco su detección (no mucho, la mayoría de IDS son capaces de detectarlos). Dentro de esta técnica se encuentra el escaneo TCP SYN que consiste en enviar un paquete SYN a la máquina remota y esperar la contestación. Si se recibe un SYN/ACK, se anota el puerto como abierto y se envía un RST para romper la conexión.

- Escaneos "stealth". Es una técnica de escaneo que trata de eludir la acción de determinados sistemas de seguridad. Dentro de los escaneos "stealth" se engloban aquellas técnicas que cumplen alguna de las condiciones siguientes:

    - Eludir cortafuegos o listas de control de acceso.
    - No ser registradas por sistemas de detección de intrusos (ni NIDS ni HIDS).
    - Simular tráfico normal y real para no levantar sospechas ante un analizador de red.

  Las tipos de escaneo "stealth" más conocidos sólo se aplican a los sistemas operativos UNIX y son los siguientes:

    - TCP FIN. El atacante envía a su objetivo una trama con el bit FIN activo y éste responde con RST si el puerto está cerrado o no responde si está abierto.
    - TCP XMAS. Se trata de enviar a la máquina objetivo una trama con todos lo bits TCP (URG, ACK, PST, RST, SYN y FIN) activos. Si el puerto está abierto el sistema operativo eliminará la trama puesto que la considera una violación del "handshake", pero si está cerrado devolverá un RST al atacante.
    - TCP NULL. Es el método opuesto al tipo de escaneo anterior. Consiste en enviar tramas con todos los bits TCP reseteados. El resultado es similar al del caso anterior.

1.7.2 Ataques de autentificación o "spoofing"

Podemos definir "spoofing" como una técnica sofisticada que permite a una máquina sustituir el origen de una serie de datos, adoptando una identidad falsa de manera que se pueda engañar a los *firewalls* o filtros de red.

Existen diferentes tipos de ataque por "spoofing":

- "IP spoofing". El "spoofing IP" es una técnica que permite enviar a una máquina paquetes que parecen provenir de una dirección IP distinta de la del equipo origen. Se trata de un enmascaramiento de la dirección IP a nivel de paquetes enviados, es decir, una modificación de los paquetes con la intención de hacer creer al destinatario que los datos provienen de otra máquina.

- "DNS spoofing". Con el "DNS spoofing", el pirata compromete al servidor de nombres y modifica deliberadamente las tablas de correspondencias entre equipos y direcciones IP. Estas modificaciones afectarán a la resolución de nombres por parte del DNS. Por tanto, cuando un cliente envíe una petición al DNS, recibirá una dirección IP ficticia, aquella de una máquina bajo el control total del atacante.

- "Web spoofing". Consiste en hacer creer a un usuario que se encuentra sobre un sitio web, pero éste es en realidad una copia creada por un pirata. Por tanto, el acceso a esta página web estará gobernado por el atacante y podrá ver todas las acciones de la víctima: datos, contraseñas, números de tarjetas de crédito, etc.

- "Fake-mail". Es una especie de e-mail falso, enviado por una persona que falsifica su identidad (el pirata) a otra persona con el objetivo de obtener informaciones confidenciales.

1.7.3 Ataques por Denegación de Servicio (DoS)

El objetivo de este tipo de ataques es provocar que la víctima no pueda acceder o tener acceso a un recurso determinado. Por ejemplo, imposibilidad de conexión, de utilizar el correo electrónico o, a mayor nivel, impedir que un servidor ofrezca servicio. Es decir, los ataques de denegación de servicio consisten en dejar los servidores inactivos durante un tiempo con la intención de que no puedan utilizarse ni consultarse. Estos ataques pueden ser realizados a nivel de red enviando datagramas cuidadosamente preparados y malintencionados de manera que puedan causar un fallo en las conexiones de red. También pueden realizarse a nivel de aplicación utilizando órdenes específicas para tratar que un programa se encuentre muy ocupado y detenga su funcionamiento. Algunos ataques DoS conocidos son:

- "SYN Flood". Se basa en un « handshake » incompleto entre dos sistemas. Como hemos visto anteriormente para realizar una conexión TCP se sigue un proceso de tres pasos. Si no se realiza el último paso, la conexión queda en un estado "semiabierto". El cliente envía el paquete SYN para establecer la conexión pero luego no responde al paquete ACK del servidor causando que este último permanezca a la escucha durante un tiempo determinado hasta anular el mismo la llamada. Si se envían muchos "handshake" incompletos, el servidor puede quedarse bloqueado.

- "Land Attack". Es una consecuencia de un error en la pila TCP/IP de los sistemas Windows. El atacante envía a cualquier puerto abierto de un servidor un paquete maliciosamente construido con la IP y puerto origen iguales a la IP y puerto destino. La máquina terminará por se paralizar.

- "Nuke". Se denomina "nuke" a la caída de una conexión TCP/IP provocada por la actuación de un agente externo (normalmente un pirata). Existen multitud de ejemplos de "nukes". Los más conocidos son:

    - "Winnuke". Es un ataque muy común en los sistemas Windows, que provoca que los equipos que escuchan por los puertos UDP 137 y 139 (utilizados por el protocolo NetBios) queden fuera de servicio o disminuyan su rendimiento al enviarles paquetes o fragmentos UDP manipulados.

    - "ICMP nuke". El protocolo ICMP se encarga de comunicar cuando hay un fallo en el sistemas de envío de paquetes del protocolo TCP/IP. Si hay algún fallo, este protocolo avisará al TCP/IP y se cortará el envío. El "ICMP nuke" consiste en enviar paquetes ICMPs falsos al protocolo TCP/IP de la víctima.

    - "OOB nuke". Se trata de mandar cierto paquete de datos al puerto 139 de la máquina objetivo. Es un error existente en los protocolos de comunicación de determinadas versiones de Windows que permite que al recibirse un paquete con la bandera OOB al puerto 139 se produzca un fallo de protección general en el sistema que lo deja bloqueado.

- "Teardrop". Consiste en enviar paquetes TCP fragmentados. Cuando el ordenador víctima recibe dichos paquetes intentará reconstruirlos. Algunas implementaciones de colas IP no vuelven a recomponer correctamente los fragmentos ya que se superponen de tal manera que provocan que el sistema se bloquee.

- "Mail Bombing" y "Mail Spamming". "Mail Bombing" es el envió continuo y masivo de un mensaje idéntico a una misma dirección hasta saturar la capacidad del destinatario. "Mail Spamming", sin embargo, es un bombardeo publicitario que llega a provocar las mismas consecuencias.

1.7.4 Ataques por explotación de errores

En este tipo de ataques se incluyen aquellos que aprovechan agujeros de seguridad de los sistemas (vulnerabilidades) como "bugs" o "puertas traseras". Dichas vulnerabilidades son utilizadas para acceder a datos, obtener privilegios o hacer sabotaje. Cada día miles de vulnerabilidades son descubiertas en sistemas operativos, aplicaciones, protocolos de red, navegadores, etc. Los sistemas operativos abiertos (de tipo Unix y Linux) tienen "agujeros" más conocidos y controlados que aquellos existentes en los sistemas operativos cerrados (como Windows). Constantemente se encuentran en Internet avisos de descubrimientos de

nuevos problemas de seguridad y herramientas de "hacking" que los explotan ("exploits"), por lo que hoy en día se hace indispensable contar con herramientas o programas que sean capaces de parchear las vulnerabilidades encontradas.

1.7.5 Ingeniería social

La ingeniería social consiste en engañar y manipular a las personas para que voluntariamente realicen actos que normalmente no harían (como revelar contraseñas u otro tipo de información) y que van a perjudicar a la seguridad de sus sistemas informáticos. Si las intenciones de quien pone en práctica este tipo de actividad no son buenas, este método de ataque es uno de los métodos más sencillos (no requiere grandes conocimientos técnicos), menos peligroso para el atacante y uno de los más efectivos. Normalmente, el atacante se aprovecha del desconocimiento de unas mínimas medidas de seguridad por parte de las personas relacionadas con el sistema para poder engañarlas en beneficio propio. Este tipo de prácticas están cada vez más de moda.

1.8 METODOLOGÍA DE ATAQUE

Vamos a ver a continuación el comportamiento típico de un pirata informático a la hora de realizar un ataque. Normalmente seguirá los siguientes pasos:

1. Identificación del objetivo. Consiste en hacerse con las direcciones IP de las máquinas que se desean atacar. Existen dos posibilidades, decidirse por un sistema en concreto o encontrar uno al azar. En el primer caso las motivaciones suelen ser de tipo odio contra el objetivo, búsqueda de fama, etc. En el segundo caso las intenciones serían realizar prácticas de "hacking" o utilizar los sistemas que se desean atacar como puente hacía otros.
   Para realizar este paso del ataque se pueden usar herramientas como "whois" que proporcionan la dirección IP de una empresa a partir de su nombre o la base de datos "Arin" que proporciona información sobre dominios web.

2. Escaneo de puertos y recopilación de información. Consiste en rastrear los puertos de una máquina, de varias o de una red de ellas con el objetivo de recopilar información sobre la víctima. Los datos que se buscan son del tipo: direcciones IP, sistemas operativos y versión de éstos, puertos abiertos, servicios de red activos, vulnerabilidades asociadas o nivel de dificultad del ataque. La herramienta más popular para realizar escaneos de puertos es Nmap, que implementa un gran número de técnicas de exploración de puertos y presentan opciones que permiten conocer que sistema operativo utiliza la máquina objetivo. Este dato es muy importante cuando se quieren buscar vulnerabilidades en los sistemas. También se utiliza frecuentemente

Netcat, que presenta algunas opciones que dificultan la detección del escaneo de puertos.

3. Intento de entrada en el sistema aprovechando el análisis de información y la identificación de vulnerabilidades. En este punto el pirata ya cuenta con una cantidad importante de datos. Un "hacker" experto en protocolos, redes y seguridad, intentará buscar alguna fisura o vulnerabilidad en el sistema o aplicaciones del objetivo e intentará desarrollar una herramienta (un "exploit") para explotar dicha vulnerabilidad. Los "hacker" con menos experiencia ("lamer") utilizarán las vulnerabilidades descubiertas por el anterior y sus herramientas para intentar entrar en los sistemas. Una sencilla búsqueda en Internet proporciona una enorme cantidad de información sobre las vulnerabilidades en los sistemas y los "exploits".

4. Búsqueda de privilegios de "root" o administrador. Normalmente en una máquina existen servicios a los que los usuarios tienen acceso y que les permiten iniciar al menos una comunicación (telnet, ftp, http, pop3, snmp, etc). Sin embargo, conectarse como usuario anónimo no permite generalmente llevar a cabo operaciones en el sistema que pongan en peligro su seguridad. Por tanto, el atacante intenta escalar privilegios, es decir, pasar de este usuario anónimo si ningún tipo de prerrogativas a un usuario con mayor número de permisos, y así sucesivamente hasta llegar a conseguir todos los privilegios del administrador. Lograr este objetivo puede costar algunos pasos. No obstante, existen programas como *getadmin.exe* que ejecutado por un usuario sin demasiados privilegios puede llegar a obtener el acceso "root" o incluirse en el grupo de administradores. Otra técnica sería la utilización de troyanos que ejecutarán un código que añade a un usuario al grupo de administradores. Por último, está la opción de intentar adivinar la contraseña del "root". Para ello existen también distintos programas llamados "password crackers" o reventadores de contraseñas.

5. Realización del ataque sobre el objetivo. Una vez conseguidos los privilegios de "root", el atacante puede hacer lo que quiera en el sistema. Por ejemplo podría modificar, copiar o borrar datos con las consecuencias que esto puede implicar en empresas como bancos, servicios secretos, etc.

6. Asegurar el regreso. Consiste en la instalación de una "puerta trasera" en el sistema que le permita al atacante volver a entrar fácilmente en él sin tener que llevar a cabo los comprometidos pasos realizados en la primera ocasión. Se trata de crear cuentas de usuarios que pasen desapercibidas pero con privilegios de administrador, programas que se ejecuten al iniciar la máquina, tareas programadas (como poner puertos a la escucha), etc. En esta fase se pueden usar "rootkits" o "troyanos" cuyo caso más conocido es "BackOrifice" que permite el control remoto total del equipo.

7. Borrado de huellas. El atacante debe borrar todo rastro que permita localizarlo, si no el ataque habrá fracasado. Básicamente, las huellas que deja el atacante se registran en los ficheros de "logs" del sistema. Para borrar dichas huellas se emplean también "rootkits". Por "rootkits" se conoce a las herramientas que realizan alguna de las tareas siguientes: creación de puertas traseras para uso futuro, manipulación de los ficheros de "logs" para borrar toda evidencia de ataque, modificación o reemplazo de herramientas del sistema para evitar ser detectado por el administrador, monitorización del tráfico de red o de pulsaciones de teclas y lanzamiento de ataques contra otros sistemas, por ejemplo de denegación de servicio.

2 ESTUDIO Y ANÁLISIS DEL PROBLEMA

2.1 OBJETIVO

El objetivo de este proyecto es el estudio, instalación y configuración de un IDS en la arquitectura de red de la empresa ARSOÉ de Trélazé. Además, la empresa quiere que se aplique en este proyecto una política de ahorro de medios que implicará, por un lado, el estudio e implantación de un IDS de distribución gratuita y, por otro, el aprovechamiento de los equipos disponibles en la empresa.

ARSOÉ es una empresa que alberga una gran base de datos destinada a recoger todas las informaciones genéticas de los animales bovinos de la región. ARSOÉ ofrece un conjunto de servicios a sus adherentes (los ganaderos) para garantizar su acceso a la base de datos y proporcionarles otro tipo de facilidades. Teniendo en cuenta las funciones de ARSOÉ, la disponibilidad de servicio es de vital importancia. Además, una pérdida de la información guardada por ARSOE sería crítica y, por tanto, es necesario un alto nivel de seguridad. A raíz de un ataque sufrido por la empresa en octubre de 2003 que creo una indisponibilidad de sus servidores web durante más de una jornada, se comienza a dar una mayor importante al hecho de obtener información sobre las incidencias de este tipo que se producen en la empresa y se decide implantar un sistema de detección de intrusiones que permita aumentar el nivel de seguridad existente.

2.2 REQUERIMIENTOS DE UN IDS

Antes de estudiar las diferentes posibilidades de instalación, arquitectura y configuración que se nos plantean a la hora de poner en funcionamiento un IDS, nos ayudará estudiar cuáles son las características fundamentales que debe cumplir un IDS en cuanto a instalación, configuración, gestión, tecnología, seguridad, etc. Veamos cuáles son dichas características:

- En cuanto a instalación, configuración y gestión:
    - Interfaz gráfica fácil de manejar.
    - Posibilidad de centralizar las tareas de reinstalación, configuración y actualizaciones mediante una consola de gestión central.

- Posibilidad de crear políticas de seguridad y filtros de alertas que permitan que ciertos sucesos no sean visualizados.

- Posibilidad de crear firmas propias de ataques.

- Las distintas alertas junto con los datos de protocolo y contenido de los paquetes sospechosos deben ser guardados de forma automática en una base de datos.

- En cuanto a la tecnología para la detección de intrusiones:

  - Ensamblado de paquetes IP fragmentados y de segmentos TCP.

  - Ensamblado de sesiones. El sistema debe distinguir los paquetes que pertenecen a una sesión determinada.

  - Normalización del tráfico anormal.

  - Decodificación de protocolos.

  - Detección de violación de protocolos.

  - Detección de ataques en tiempo real.

- Para los mecanismos de notificación de alertas:

  - Deben estar disponibles los siguientes métodos de notificación: envío de alertas por mail, mensajes *syslog* y mensajes de alertas en tiempo real a una consola de monitorización central.

  - Deben tener la capacidad de realizar en tiempo real la resolución de nombres DNS, Netbios, IP y MAC.

  - La información debe ser presentada en un formato fácilmente comprensible incluyendo los siguientes datos: direcciones IP origen y destino, cabeceras IP, protocolos (UDP, TCP, ICMP, etc.), puertos de origen y destino o tipo de paquete ICMP, protocolo de aplicación (HTTP, SMTP, TELNET, FTP, ...) en formato texto, cabeceras TCP, decodificación de protocolo y datos del paquete.

  - La consola o interfaz IDS debe permitir la búsqueda y el análisis de datos dentro de la base de datos, de manera que podamos seleccionar datos, por

ejemplo, por direcciones origen, número de alertas de un mismo tipo, protocolo, prioridad de alertas, etc ...

- En cuanto a la seguridad de los IDS:
    - La comunicación entre los distintos componentes de un IDS debe ser cifrada.
    - Fuerte autentificación utilizando claves.
    - El método de comunicación no debe ofrecer ningún dato sobre el tipo o versión del IDS que se está utilizando.
    - La transmisión por la interfaz que funciona como "sniffer" de datos debe estar prohibida. Es recomendable configurar la tarjeta de red sin una dirección IP definida.
    - No se recomienda el uso de mecanismos de respuesta activos.

2.3 ESTUDIO DE NECESIDADES

El estudio de necesidades viene marcado por el objetivo del proyecto, es decir, la puesta en funcionamiento de un sistema de detección de intrusiones en la red de la empresa. Esto implicará, en primer lugar, conocer la estructura de la red, para luego estudiar dónde colocar cada uno de los sensores, cómo podemos conectarlos a la red o cómo podremos llevar a cabo su comunicación. Por otro lado, deberemos saber que aplicaciones serán necesarias antes de desarrollar la solución. A partir de este punto, nos referiremos a los IDS de software libre (Snort y Prelude-IDS) que son los que se vamos a considerar como posible solución. No obstante, la mayoría de los análisis que hagamos se podrán aplicar de manera general a todos los sistemas de detección de intrusiones.

2.3.1 Descripción de la red de ARSOE

Dentro de ARSOÉ, la red se construye utilizando conmutadores LAN sobre los cuales se configuran múltiples redes locales virtuales (VLAN). Podemos dividir la red de ARSOÉ en 5 VLAN: Internet, DMZ2, DMZ, VLAN interna e Intranet. Todas ellas están separadas por un *firewall* configurado para impedir ciertos accesos no deseados a la red. Este mismo esquema se repite de nuevo formando una estructura redundante, por donde circulará el tráfico en caso de indisponibilidad de la primera como puede observarse en la figura 2.1. Veamos cada una de las partes:

- VLAN Internet. En este segmento de red se encuentran un *router* y un "linkbranch" que se encargan de hacer una primera traslación de direcciones IP de públicas a privadas (NAT) y de realizar un balanceo de carga entre los dos operadores que dan servicio al ARSOE de Trélazé.

- DMZ2. Esta VLAN se encuentra en la parte de la red a través de la cuál se producen los accesos a Internet. Comprende la parte de la red que va del *firewall* hasta el *switch* y el "linkbranch" que se encuentran antes de la salida a Internet.

- DMZ. En esta parte de la red se encuentran todos los servidores de la empresa. Está protegida por el *firewall* y es la parte más crítica o de mayor importancia de la red.

- LAN interna. A esta VLAN se conectan todos los puestos pertenecientes a la empresa en Trelazé.

- Intranet. La Intranet es una red privada de la empresa que comprende la conexión con otras 12 entidades distribuidas por toda Francia.

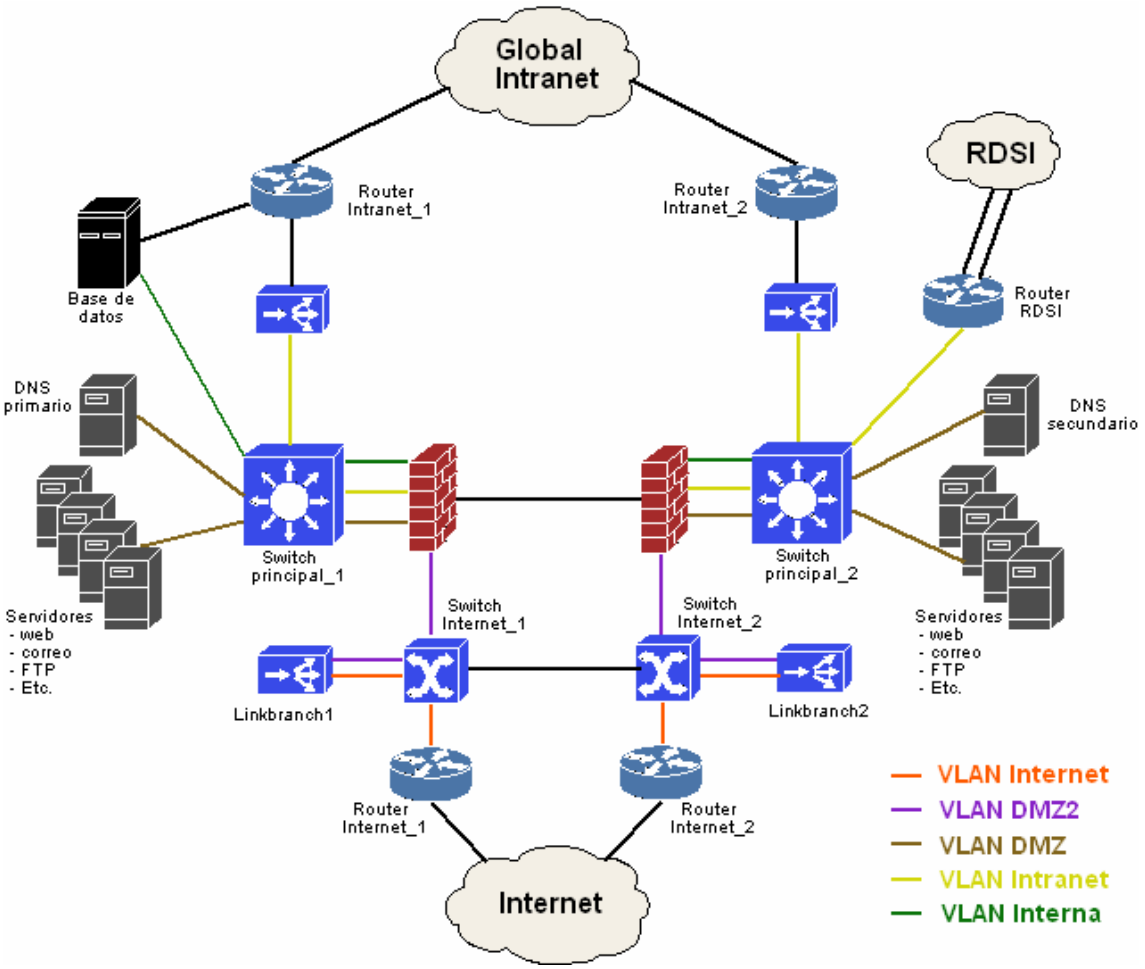


Figura 2.1.- Arquitectura de la red de ARSOÉ

2.3.2 Estudio de las posibles ubicaciones del sensor

Podemos colocar un sensor IDS en cada uno de los segmentos de red que nos interese vigilar. Esto último vendrá dado por la política de seguridad que se quiera aplicar en la empresa. Teniendo en cuenta la estructura de la red se nos presentarían las siguientes posibilidades en cuanto a la ubicación de los sensores: sobre el segmento VLAN Internet, sobre el segmento VLAN DMZ2, en la DMZ, en la zona LAN interna o en la zona VLAN Intranet.

Zona Internet

El tráfico que se puede observar en este segmento de la red corresponde a cualquier dirección IP entre todas aquellas que corresponden a las líneas públicas que la empresa ha contratado, y no solamente a aquellas que realmente están atribuidas. Por tanto, no sería demasiado práctico colocar un sensor en esta zona de la red puesto que se detectaría una cantidad importante de tráfico no interesante que no llegaría a alcanzar realmente nuestra red. Esto resultaría, además, en una gran cantidad de falsos positivos y se necesitaría mucha más potencia en el detector para llegar a analizar el tráfico en esta zona.

Zona DMZ2

Si se coloca un detector en este segmento, se va a analizar solamente el tráfico que está destinado a nuestra red. Sin embargo, el tráfico en esta zona sigue siendo mucho mayor en comparación al tráfico que circula en las zonas protegidas por el *firewall*. Esto significará, por tanto, una cantidad mayor de falsos positivos y el detector deberá estar configurado para trabajar con menor sensibilidad. La ventaja de colocar el sensor IDS en esta zona es que se puede detectar con mayor antelación si se está produciendo una tentativa de ataque, aunque realmente éste no se termine llevando a cabo.

Zona DMZ

Como se ha comentado anteriormente, en esta zona se encuentran todos los servidores de la empresa. Si colocamos un detector IDS en esta parte de la red, se debería configurar con bastante sensibilidad ya que una tentativa de ataque en esta zona tiene muchas más posibilidades de éxito. Es decir, es muy importante evitar los falsos negativos en esta zona. Por otro lado, la cantidad de tráfico a analizar será mucho menor que en las dos zonas anteriores, ya que el *firewall* se encarga de filtrar buena parte de éste. Esto favorecerá el hecho de detectar un menor número de falsos positivos, aunque la configuración del detector con mayor sensibilidad favorecerá el efecto contrario. El problema en este caso es que se

cuenta con menos tiempo para reaccionar ante una tentativa de ataque, ya que el atacante habrá logrado pasar el *firewall*.

Zona LAN Interna

En este caso, el tráfico a analizar es menos interesante puesto que un ataque en esta zona no compromete el funcionamiento de ningún servidor. No obstante, resultaría interesante para detectar si alguien dentro de la organización intenta realizar ataques o si hace un uso de la red no permitido por la política de la empresa (en algunos países como Francia este tipo de vigilancia está prohibida por la ley). La cantidad de tráfico es relativamente pequeña y los niveles de seguridad podrían ser menos exigentes que el caso anterior.

Zona Intranet

Mediante este segmento de la red se permite el acceso a las diferentes entidades pertenecientes a la misma organización. El tráfico a analizar no sería demasiado elevado y el interés radica en detectar actividades sospechosas procedentes de la misma organización.

En la tabla 2.1 se pretende presentar de forma resumida lo comentado anteriormente.

	Internet	**DMZ2**	**DMZ**	**LAN Interna**	**Intranet**
Número de falsos positivos	Muy alto	Alto	Alto-medio	Medio	Alto-medio
Cantidad de tráfico a analizar	Alto	Medio	Medio-bajo	Bajo	Medio-bajo
Importancia del tráfico	Baja	Media	Alta	Baja	Media
Capacidad de reacción ante intento de ataque	Alta	Alta	Baja	Baja	Media
Dificultad de configuración	Alta	Alta	Baja	Media	Media

Tabla 2.1- Posibles ubicaciones de los detectores

En general, el número de falsos positivos es directamente proporcional a la cantidad de tráfico que se analiza y al nivel de sensibilidad de detección con el que se configura el IDS. La configuración del detector será más difícil cuanto menos sensibilidad se requiera, ya que habrá que ajustar de manera más precisa la configuración, de modo que se evite un gran número de falsos positivos. Por último, la importancia del tráfico que se analiza depende directamente de la importancia de los equipos que contenga cada una de las zonas de la red.

2.3.3 Estudio de la conexión a la red

Lo óptimo sería que al conectar un sensor en una de las partes de la red, fuera capaz de escuchar todo el tráfico entrante y saliente de dicha parte. Si la red está configurada haciendo uso de conmutadores, el problema de instalar un IDS, sería que sólo podría detectar el tráfico que va destinado a él. Esto nos hace plantearnos cómo conectar los sensores de manera que puedan monitorizar todo el tráfico que circula por una sección de la red. En principio, se podrían plantear tres alternativas: repetidores, sesiones SPAN y TAPs.

2.3.3.1 Repetidores

Es el método más sencillo para introducir un segmento de vigilancia en la red debido a su principio de funcionamiento. Todo el tráfico que reciben es transmitido por cada uno de sus puertos. Por tanto, sería suficiente conectar un detector IDS a uno de sus puertos para escuchar todo el tráfico del segmento de red al que esté conectado el repetidor. No obstante, presenta muchos inconvenientes. Cada segmento de vigilancia que queramos introducir implicará la colocación de un repetidor. Pues bien, el uso de repetidores puede conllevar:

- Empeoramiento de las prestaciones de la red en cuanto a ancho de banda se refiere, ya que en los repetidores se comparte la carga a transmitir entre todos sus puertos debido al problema de las colisiones.

- Cada repetidor puede suponer un "punto único de fallo" de la red. Es decir, si un repetidor deja de funcionar puede dejar la red indisponible. Esto sería bastante grave puesto que los segmentos de vigilancia se colocan normalmente en zonas estratégicas de la red, donde, o bien el tráfico es importante o los elementos que contiene requieren de una seguridad especial. Por tanto, esta solución no sería del todo coherente.

- Un atacante podría descubrir y comprometer a un IDS conectado a un repetidor puesto que éste permite el flujo de tráfico hacia la red.

2.3.3.2 Sesiones SPAN

Ciertos conmutadores LAN permiten realizar las llamadas sesiones SPAN (Switched Port Analyzer) que consisten en configurar uno de los puertos del *switch* de manera que sea capaz de recibir copias de las tramas enviadas o recibidas por un conjunto de puertos elegidos. Se puede realizar una sesión en modo entrada (sólo se refleja el tráfico que recibe el conmutador), salida (sólo se refleja el tráfico enviado por el conmutador) y entrada-salida (configuración por defecto). Además también podemos configurar una sesión SPAN sobre un

puerto de manera que éste reciba todo el tráfico que circule sobre una VLAN determinada. Por defecto, el puerto SPAN está configurado de tal manera que sólo pueda recibir el tráfico y no enviarlo, aunque se puede configurar también esta opción. Las ventajas de utilizar esta solución son:

- Facilidad de instalación. El IDS puede ser conectado en el conmutador sin modificar la estructura de la red, sin introducir ningún nuevo tipo de *hardware* sobre la red.

- No se requieren cambios en la configuración del *firewall*.

- Se elimina la necesidad de introducir un cable unidireccional, que sólo permita escuchar el tráfico y no enviarlo. El puerto SPAN puede ser configurado para ello.

- No supone ningún "punto único de fallo". Si este puerto se desconfigura o estropea, el tráfico sigue transmitiéndose normalmente a través del conmutador.

Los inconvenientes:

- Sólo se puede realizar SPAN sobre un puerto por conmutador.

- Pueden degradar las prestaciones de los conmutadores (el puerto SPAN puede sobrecargar la memoria del conmutador). El puerto SPAN sería el primero en ser afectado en el caso en el que el conmutador no dispusiera de memoria suficiente para llegar a realizar la copia de tráfico. Por tanto, puede existir la posibilidad de pérdida de paquetes.

- No se pueden recibir las tramas con errores por el puerto SPAN (es decir, paquetes con mayores dimensiones que las normalizadas, con CRC no válido, etc.).

2.3.3.3 TAPs

Un TAP (test access port) es básicamente un dispositivo *hardware* de tres puertos que permite copiar el tráfico entre dos puertos a un tercero, de forma unidireccional (es decir, éste último puerto recibe tráfico, no puede enviarlo). Se suelen conectar sobre cables estratégicos en la red. Es el método más utilizado a la hora de conectar segmentos de vigilancia de tráfico y presenta las ventajas tanto de los repetidores como de los conmutadores. Las ventajas que presentaría esta opción son:

- Hace una copia idéntica del tráfico, es decir, recibe incluso aquellas tramas malformadas. Esto permite que el IDS sea capaz de detectar cualquier tipo de ataque.

- No introduce ningún "punto único de fallo". Es decir, si este componente es desconectado o su funcionamiento falla, el tráfico de red continuará atravesándolo normalmente.

- Se puede configurar para que el tráfico sea unidireccional o bidireccional.

El principal inconveniente de los TAPs es su coste. Puede ser como diez veces el de un repetidor en línea y como el doble del de un conmutador con un puerto SPAN. Esta opción requeriría un buen presupuesto para seguridad por parte de la empresa.

2.3.4 Estudio del sistema operativo

Debemos decidir el sistema operativo sobre el que se va a instalar el sistema de detección de intrusiones.

Snort debe funcionar en todas las plataformas en las que la librería de captura de tráfico "libpcap" pueda hacerlo. La tabla 2.2 muestra las plataformas en las que Snort ha sido compilado con éxito.

Alpha	M68k/PPC	Sparc	x86	Other	
X	X	X	X	X	**Linux**
	X	X	X		**OpenBSD**
X			X		**FreeBSD**
	X		X		**NetBSD**
		X	X		**Solaris**
		X			**SunOS 4.1.X**
				X	**HP-UX**
				X	**AIX**
				X	**IRIX**
X					**Tru64**
	X				**MacOS X Server**
			X		**Win32 - (Win9x/NT/2000)**

Tabla 2.2.- Compatibilidad de Snort con los sistemas operativos y el hardware.

Prelude-IDS sólo funciona en plataformas POSIX. La compatibilidad entre el hardware y los distintos sistemas operativos se muestra en la tabla 2.3.

Alpha	PowerPC	Sparc	x86	Itanium	OS
X	X	X	X	X	**Linux ***
			X		**OpenBSD**
			X		**FreeBSD ****
			X		**NetBSD**
		X			**Sun/Solaris**
	X				**MacOsX**

Tabla 2.3.- Compatibilidad de Prelude-IDS con los sistemas operativos y el hardware.

Un IDS debe beneficiarse de un alto nivel de seguridad dentro de la arquitectura de red. Son los sistemas operativos basados en Unix los que ofrecen mejores prestaciones en materia de seguridad. Según la base de datos SIPS (Security Intelligence Products and Systems), en el año 2002 los sistemas más afectados por vulnerabilidades fueron las distintas versiones de Windows (44%) seguido de Linux (19%) y de BSD y Solaris (9 y 7% respectivamente).

Nosotros vamos a desarrollar la solución utilizando el sistema operativo Linux (RedHat 8.0), que es un sistema basado en Unix y además de libre distribución. De esta manera existirá la posibilidad de poder utilizar tanto Snort como Prelude-IDS. Hay que tener en cuenta que ambos IDS fueron inicialmente diseñados para trabajar sobre plataformas POSIX y, por tanto, se adaptan mejor a este tipo de sistemas operativos.

2.3.5 Estudio de las aplicaciones necesarias

Un IDS no se compone solamente de un detector capaz de analizar el tráfico y enviar alertas. Para que el IDS sea un dispositivo de seguridad práctico necesita cumplir con los requerimientos que hemos mencionado en el punto 2.1. Esto implicará mejorar las prestaciones del IDS, dotarlo de una interfaz o consola que facilite el acceso a los datos de intrusiones detectados y aumentar, en la medida de lo posible, la seguridad del sistema. Para realizar estas tareas es necesaria la utilización de otras aplicaciones, algunas de ellas fundamentales, como el empleo de una base de datos o la utilización de un servidor web.

2.3.5.1 Base de datos

Aunque tanto Snort como Prelude-IDS pueden almacenar los datos en ficheros utilizando distintos formatos de salida, el empleo de una base de datos relacional parece una opción fundamental si se va a utilizar una consola para acceder a la información. Su uso, permitirá el acceso a numerosos datos de intrusión y la posibilidad de seleccionar alertas, efectuar búsquedas, organizar los datos por diversos criterios y realizar operaciones como el borrado o la copia de alertas. Varias aplicaciones explotan los datos de intrusión almacenados en una base de datos para ofrecer al usuario una interfaz gráfica para la administración y el tratamiento de alertas. Ejemplos de dichas aplicaciones son la consola de análisis de datos de intrusión propia de Snort, ACID, y la consola PIWI en el caso de Prelude-IDS.

En el caso de Snort podemos utilizar como base de datos MySQL, PostgreSQL, MSSQL, Oracle y las bases de datos compatibles UnixODBC, mientras que Prelude-IDS sólo puede escribir datos en MySQL, PostgreSQL u Oracle.

En general, el problema que presenta la utilización de una base de datos es que empeorará el rendimiento del IDS en cuanto a velocidad de procesamiento, incluso en redes que presenten un tráfico de datos moderado. Esto ocurre porque la escritura en una base de datos es mucho más lenta que si la salida de datos se realiza hacia un fichero. Este problema se pone de manifiesto sobre todo en el caso de Snort, puesto que en el caso de Prelude-IDS, es un módulo diferente al sensor (Prelude-manager) el que se encarga de realizar esta tarea. Para Snort, existe una aplicación llamada Barnyard que permite liberar al sensor Snort de la escritura en la base de datos. Barnyard utilizará el formato de salida “unified” de Snort (el más rápido), lo interpretará y escribirá los datos de intrusión en la base de datos que se le indique. No obstante, es recomendable que los sensores y la base de datos se encuentren en diferentes máquinas.

En el desarrollo de la solución vamos a utilizar una base de datos que sea de distribución gratuita. PostgreSQL ofrece más posibilidades de operaciones que MySQL y es capaz de trabajar con aplicaciones más complejas. Nosotros utilizaremos MySQL puesto que realmente el almacenamiento de alertas no requiere ni operaciones muy complejas ni un gran número de tablas y además MySQL es más rápida, más fiable y más sencilla de utilizar.

2.3.5.2 Servidor web

La utilización por parte de cualquiera de los dos IDS que estamos considerando de un interfaz gráfico requiere como medio para visualizarlo la instalación de un servidor web. Las consolas ACID y PIWI, escritas respectivamente en PHP y Perl, pueden utilizarse desde prácticamente todos los servidores web incluyendo los conocidos Apache e IIS.

El servidor Apache es el servidor web más popular y será el que utilizaremos puesto que es software libre y existen aplicaciones de Snort que utilizan el código fuente de Apache.

2.3.5.3 Aplicaciones para el cifrado de datos

Si se quiere cumplir con los requisitos de seguridad, a la hora de instalar un IDS es conveniente realizar siempre un cifrado de la información que vayamos a transmitir por la red desde cualquiera de los componentes del IDS. Para realizar esta tarea utilizaremos la librería de encriptación OpenSSL que está compuesta de un conjunto de algoritmos capaces de cifrar casi todas las conexiones TCP. Normalmente, esta librería se utiliza junto a la aplicación Stunnel que permite establecer y mantener sesiones cifradas, es decir, se utiliza para poder comunicar de manera segura servicios que no utilizan por defecto OpenSSL. Stunnel creará un canal seguro utilizando dicha librería de encriptación.

2.3.5.4 Otras aplicaciones

Ya se ha comentado anteriormente algunas aplicaciones cuyo uso es imprescindible, pero existen un gran número de programas que se pueden aprovechar para dotar de mayores prestaciones a los IDS. Algunos de ellos son independientes de dichos IDSs y otros están diseñados expresamente para trabajar con ellos. Algunas de las aplicaciones que nos podrían interesar a lo largo del desarrollo de la implementación del IDS podrían ser:

- Snort-rep. Es un "script" escrito en Perl que permite realizar un informe de las alertas que se han producido utilizando el fichero de *logs* del módulo de salida *syslog* de Snort. El informe que crea está dividido en varias secciones de tal manera que clasifica por un lado los escaneos de puertos y por otro el resto de alertas, y dichas alertas, las muestras de varias maneras: el tanto por cierto de cada tipo de alertas organizadas por prioridad, número de alertas y tipo dependiendo de su proveniencia, a qué dirección destino se dirige cada alerta y finalmente cuáles son los puertos afectados.

- Oinkmaster. Es una aplicación que permite actualizar automáticamente las reglas de Snort cada cierto tiempo y eliminar aquellas que no nos interesan después de cada actualización. Puede mostrar los cambios que se han realizado desde la última actualización lo que ayuda a tener un control del conjunto de reglas. Puesto que Prelude-IDS utiliza este mismo conjunto de reglas, este programa también tiene aplicación en caso de utilización de Prelude-IDS.

- Swatch. Es un programa que analiza diferentes ficheros de *logs* buscando determinados patrones y que podemos configurar para que ejecute una serie de acciones cuando encuentre dichos patrones. Las acciones que puede llevar a cabo son, por ejemplo, enviar un e-mail, hacer saltar una alarma visual o sonora en el sistema o incluso ejecutar otras aplicaciones. Se puede utilizar si nos interesa ejecutar alguna acción cuando se produzca un determinado tipo de ataque.

- Tripwire. Es un analizador de integridad de ficheros. Compara las propiedades de los ficheros indicados en su configuración con la información almacenada sobre ellos en una base de datos que ha creado previamente. Es capaz de indicar cualquier tipo de cambios en dichos ficheros y comunicarlos bien mediante ficheros de *logs* o bien vía mail si se desea. Se puede clasificar como un HIDS.

3 DESARROLLO DE LA SOLUCIÓN

3.1 PERIODO DE ANÁLISIS Y PRUEBA

Antes de tomar la decisión de que solución es la más adecuada para cubrir las necesidades que se plantean en la empresa, es necesario probar las herramientas que se van a utilizar. El objetivo de este periodo de prueba es comparar las prestaciones de los IDS de software libre más relevantes que existen, Snort y Prelude-IDS, así como de observar su modo de funcionamiento, sus partes configurables, y por supuesto, observar las respuestas ante determinadas simulaciones de ataques. Para ello se va a construir una maqueta bastante simple. Se trata de tres ordenadores conectados mediante un repetidor. Uno de ellos va a albergar al NIDS que se quiere probar, en otro se instalarán diferentes herramientas de test de IDS y de simulación de ataques informáticos, y, por último, el tercero estará en modo pasivo y será el objetivo de los ataques.

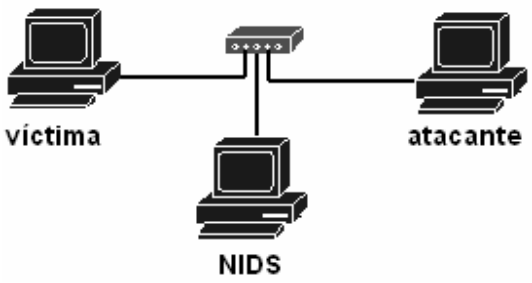


Figura 3.1.- Esquema de la maqueta de pruebas

3.1.1 Procesadores, sistemas operativos y aplicaciones

En los tres ordenadores que componen la maqueta se va a instalar la distribución RedHat 8 del sistema operativo Linux. En esta fase de pruebas no es necesario que los procesadores de la maqueta sean muy potentes. Sus características se detallan a continuación:

- Máquina IDS: procesador intel celeron 800MHz, 256 Mb de memoria RAM y 8 Gb de disco duro.

- Máquina atacante: procesador pentium II 333MHz, 64 Mb de memoria RAM y 3 Gb de disco duro.

- Máquina víctima: procesador pentium II 333MHz , 64 Mb de memoria RAM y 3 Gb de disco duro.

3.1.1.1 Máquina IDS

El objetivo es estudiar la eficiencia de los IDS gratuitos, por lo tanto, los dos programas que vamos a probar serán Snort y Prelude-IDS. Como ya se ha comentado anteriormente, la mejor política de seguridad no es elegir entre implantación de un HIDS o de un NIDS, sino utilizar las ventajas que brindan los dos actuando conjuntamente. Por eso, se instalará también Tripwire (que es un HIDS libre) sobre la máquina IDS y sobre el ordenador atacado para comprobar la integridad de ficheros.

La configuración del IDS que vamos a instalar sobre la maqueta será híbrida, es decir, en el mismo procesador instalaremos tanto el detector como el servidor IDS. Por lo tanto, no hará falta utilizar programas para realizar comunicaciones seguras detector-servidor. La implantación de un IDS sobre una computadora conlleva la instalación de un buen número de aplicaciones que comentaremos a continuación (el manual de instalación del IDS se encuentra en el Anexo I):

- Base de datos. Tanto un IDS como el otro, almacenan información sobre una base de datos. La base de datos que se va a utilizar también debe ser de libre distribución y se ha elegido MySQL.

- Servidor web. Se instalará el servidor web Apache.

- Interfaz. En cuanto a las interfaces, en el caso de Snort, se instalará ACID y en el de Prelude-IDS usaremos PIWI (el frontend en Perl). Accederemos a ambas interfaces mediante el servidor web Apache. ACID utilizará el módulo PHP, mientras que PIWI necesitará los módulos Perl para Apache.

- Otras aplicaciones. Se han buscado otras aplicaciones que aporten más utilidades a un sistema de detección de intrusiones en tiempo real. Se destacan las siguientes:

    - Barnyard. El rendimiento de Snort baja considerablemente cuando se utiliza como módulo de salida una base de datos, es decir, el ancho de banda que es capaz de analizar disminuye. Por este motivo, los mismos programadores de Snort, han desarrollado una herramienta llamada Barnyard, que toma los datos del formato de salida *unified* que genera Snort para encargarse del almacenamiento de datos en distintos recursos, entre ellos el de una base de datos.

    - Tripwire. Como ya se ha comentado anteriormente, *tripwire* es un HIDS con el que se podría complementar la funcionalidad de un NIDS como Snort.

- Swatch. Es un programa que permite realizar notificaciones de alertas en tiempo real. Estas notificaciones pueden ser correos electrónicos, alarmas acústicas e incluso mensajería móvil si contamos con los medios apropiados.

- Módulos de integración de sondas en Prelude. Para poder utilizar Snort como sensor dentro de una arquitectura gobernada por un manager Prelude, es necesario instalar el módulo que permite que ambos sean compatibles. Existen módulos para poder integrar otros tipos de sensores con Prelude, no obstante, en este periodo de pruebas sólo utilizaremos el de Snort.

3.1.1.2 Máquina atacante

En este ordenador se van a instalar diferentes herramientas de simulación de ataques y de test de IDS. El objetivo es comprobar si el IDS reacciona bien ante dichos ataques. Las herramientas más utilizadas son:

- Nmap. Es una herramienta diseñada para explorar y realizar auditorias de seguridad en redes. Permite el escaneo de grandes redes para determinar que servidores se encuentran activos, que servicios ofrecen y saber con cierta exactitud que sistemas operativos se utilizan en dichos *hosts*. Por tanto, Nmap es una herramienta que puede utilizarse por los administradores de redes para determinar el grado de seguridad de una red, pero también por un *hacker* para obtener información de la misma. Un gran número de ataques comienzan utilizando este tipo de técnicas, por lo que es interesante comprobar si el sistema IDS es capaz de detectarlas. Con Nmap se pueden realizar diferentes tipos de escaneos como TCP SYN, escaneo tipo UDP, Stealth FIN, Stealth Xmas o Scan Null.

- Fragroute. Es una herramienta que se utiliza para realizar ataques de fragmentación de paquetes. Fragroute intercepta el tráfico saliente de un equipo y le aplique un número configurable de reglas de fragmentación. El tráfico quedará por tanto fragmentado y manipulado antes de ser enviado a la computadora objetivo.

- Exploit. Podemos encontrar en Internet numerosos “exploit”, pero hay que escoger aquel que se adapte a las características de nuestro sistema. Para realizar la simulación de un ataque utilizando un “exploit” se ha elegido uno llamado “m00-apache-w00t” que es efectivo contra plataformas Linux. Este “exploit” aprovecha un defecto de configuración del módulo *userdir* de Apache y permitirá conocer de manera remota si un nombre en concreto corresponde con el de un usuario del sistema atacado utilizando para ello el protocolo FTP. En la introducción se explicó que un “exploit” es un programa o técnica que se aprovecha de las vulnerabilidades existentes en

determinados sistemas o programas. Los piratas informáticos intentan sacar provecho de esos errores en los sistemas para lograr hacerse con el control de ellos, obteniendo, por ejemplo, los privilegios del administrador.

- Stick. Es un programa que envía paquetes conteniendo las mismas firmas de ataque que IDSs como Snort o Prelude son capaces de detectar con el objetivo de desencadenar muchas alertas en los detectores IDS. Stick es capaz de enviar muchos de estos paquetes por segundo, lo que presumiblemente consumirá muchos recursos del IDS impidiendo que éste responda correctamente (puede perder paquetes). Es decir, esta herramienta puede utilizarse para crear un ataque por denegación de servicio contra un IDS.

- Nessus. Podemos definir Nessus como un escáner de vulnerabilidades, es decir, es una herramienta que permite escanear puertos, pero con la particularidad de que es capaz de descubrir los errores de seguridad de los servicios que se están corriendo en dichos puertos. Por tanto, es un programa que se puede utilizar como herramienta de seguridad, pero también para obtener información importante como paso previo a un ataque. Es interesante entonces, que el IDS pueda detectar si estas prácticas se están llevando a cabo.

3.1.1.3 Máquina víctima

Este ordenador tiene un papel pasivo. En el solamente se ha instalado Tripwire.

3.1.2 **Resultado del test**

No sólo se ha probado como ambos IDS (Snort y Prelude) responden a las diferentes simulaciones de ataque sino que se han estudiado los distintos recursos como por ejemplo las repercusiones de utilizar una base de datos o ficheros de logs o la integración de Snort como sensor de Prelude.

3.1.2.1 Respuesta de los IDS ante las simulaciones de ataques

Vamos a ver como reaccionan ambos detectores IDS ante los diferentes tipos de ataques que se han ensayado.

Nmap

Con esta herramienta pretendemos comprobar la capacidad del IDS de detectar un escaneo de puertos. Los escaneos de puertos utilizan diferentes técnicas en cuanto al protocolo que siguen para llevarse a cabo. Hemos probado diferentes tipos. La tabla 3.1 pretende presentar la respuesta que se obtiene de cada NIDS ante distintos tipos de escaneo Nmap.

Tipo de escaneo	SNORT			PRELUDE-NIDS		
	Detección	Nº de alertas	Alertas diferentes	Detección	Nº de alertas	Alertas diferentes
TCP SYN	SI	8	8	SI	6	6
TCP connect()	SI	8	8	SI	6	6
UDP	SI	29	3	SI	19	4
TCP FIN	SI	821	3	SI	998	3
TCP XMAS	SI	603	3	SI	742	4
TCP NULL	SI	645	3	SI	1203	4
OS fingerprint	SI	12	12	SI	7	7

Tabla 3.1.- Resultados de Snort y Prelude-NIDS ante una simulación de escaneos Nmap.

Puede observarse que el número de alertas emitidas es similar en ambos IDS. Los tipos de escaneo TPC FIN, TCP XMAS, TCP NULL y OS fingerprint se reconocen por ambos NIDS, mientras que para los tres primeros tipos que aparecen en la tabla se emiten diferentes alertas de tipo ICMP y SCAN, de forma que puede adivinarse que se está produciendo un escaneo de puertos, pero no el tipo en concreto.

Fragroute

Ambos NIDS emiten alertas ante un ataque realizado con la herramienta "fragroute". En la tabla 3.2 se muestra el número de alertas obtenidas con cada uno de los IDS.

	SNORT			PRELUDE-NIDS		
	Detección	Nº de alertas	Alertas diferentes	Detección	Nº de alertas	Alertas diferentes
Fragroute + nmap fingerprint	SI	626	11	SI	84	13

Tabla 3.2.- Resultado de Snort y Prelude-NIDS ante la simulación de un ataque con Fragroute

A pesar del intento de enmascaramiento del ataque "nmap fingerprint" utilizando fragmentación de paquetes IP, ambos IDS son capaces de reconstruir de nuevo los paquetes y detectar que se está produciendo dicho ataque. Además Prelude-NIDS emite una alerta que indica la existencia de un ataque por fragmentación IP. La mayoría de las alertas producidas por Snort hacen referencia a errores encontrados en el protocolo de la trama TCP.

Stick

Como ya se ha comentado, Stick es capaz de generar paquetes que contengan los mismos patrones o firmas de ataques que utilizan detectores de intrusiones como Snort o Prelude-NIDS. En principio, podría pensarse que lo ideal sería que el IDS emitiera una alerta correspondiente a cada una de las firmas que Stick es capaz de transmitir a la red. Sin embargo, Stick puede enviar cientos de estas firmas por segundo y se emplea principalmente como una herramienta para sobrecargar de trabajo a los sistemas de detecciones de intrusiones y provocar una situación de denegación de servicio en los mismos. Por tanto, lo ideal sería que ante este tipo de actividad, a la que muchas veces se le llama "stealth", los IDS fueran capaces de, o bien bloquearla, o bien, ignorarla.

En este caso, se obtiene un resultado totalmente diferente dependiendo del IDS que se esté utilizando. Ambos NIDS emiten un número elevado de alertas (que depende del tiempo de ejecución de Stick). Snort solamente genera repetidamente dos tipos de alertas ("Invalid UDP header, legth field < 8" y "Unknown Datagram decoding problem!") originadas por el decodificador de Snort que encuentra irregularidades en el paquete IP. Sin embargo, Prelude-NIDS analiza cada paquete de manera que emite la alerta correspondiente a cada una de las firmas que Stick simula en los paquetes que envía.

Aunque muchos libros y artículos aseguran que tanto Snort como Prelude-NIDS son inmunes a este tipo de ataque, lo cierto es que ambos generan una cantidad importante de alertas. No obstante, en esta etapa de pruebas se verifica que esta carga adicional de trabajo no implica la pérdida de paquetes o una sobrecarga de la memoria del sistema.

Ping

Ambos IDS, en su configuración más sensible, son capaces de detectar si se está realizando PING en un sistema. Esto no será práctico cuando el sistema se aplique a condiciones de tráfico real.

	SNORT			PRELUDE-NIDS		
	Detección	Nº de alertas	Alertas diferentes	Detección	Nº de alertas	Alertas diferentes
PING	SI	2	2	SI	2	2

Tabla 3.3.- Resultado de Snort y Prelude-NIDS cuando se realiza "ping".

Exploit

Hemos realizado la prueba para dos casos, uno en el que ninguno de los usuarios de los que utiliza el "exploit" coincide con alguno de la máquina víctima y otro en el que coincide uno de ellos. Los datos de detección obtenidos por ambos NIDS son idénticos.

Exploit Apache	SNORT			PRELUDE-NIDS		
	Detección	Nº de alertas	Alertas diferentes	Detección	Nº de alertas	Alertas diferentes
0 usuarios encontrados	SI	3	3	SI	3	3
1 usuario encontrado	SI	4	3	SI	4	3

Tabla 3.4.- Resultado de Snort y Prelude-NIDS ante un ataque utilizando un "exploit".

Nessus

Se ha ejecutado "Nessus" desde la máquina atacante de manera que analice si existen vulnerabilidades tanto en la máquina víctima como en la que contiene el IDS. Si bien lo ideal sería obtener una alerta que indicará que se está produciendo un análisis de vulnerabilidades, se obtienen numerosas alertas de distinto tipo que pueden hacer sospechar que se está llevando a cabo ese tipo de actividad. El comportamiento de ambos NIDS es similar.

	SNORT			PRELUDE-NIDS		
	Detección	Nº de alertas	Alertas diferentes	Detección	Nº de alertas	Alertas diferentes
Nessus	SI	2891	numerosas	SI	2866	numerosas

Tabla 3.5.- Resultado de Snort y Prelude-NIDS cuando se ha producido un análisis de vulnerabilidades con la herramienta Nessus.

3.1.2.2 Prueba de prestaciones

Ninguna de las pruebas que hemos realizado ha provocado una situación de saturación en los IDS. Sería lógico pensar que, si existe una situación en el que el tráfico a analizar es elevado y además provoca que el sistema de detección de intrusiones emita numerosas alertas, se produzca una situación de saturación. Aunque, evidentemente, el segmento de red que existe en esta maqueta no puede transmitir una gran cantidad de tráfico, se ha intentado saturar de trabajo los detectores con herramientas como Stick o Nessus. Los informes de tráfico obtenidos por ambos IDS pueden observarse en las figuras 3.2 y 3.3. En el caso de Snort se obtienen también estadísticas en cuanto al tipo de protocolo, a la fragmentación de paquetes y a la reconstrucción de sesiones TCP.

```
-*> Snort! <*-
Version 2.0.5 (Build 98)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)

===============================================================================
Snort analyzed 40121 out of 40121 packets, dropping 0(0.000%) packets

Breakdown by protocol:                Action Stats:
    TCP: 25408      (95.730%)         ALERTS: 5188
    UDP: 425        (1.059%)          LOGGED: 9318
   ICMP: 123        (0.307%)          PASSED: 0
    ARP: 40         (0.100%)
  EAPOL: 0          (0.000%)
   IPv6: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 498        (1.241%)
DISCARD: 2051       (5.112%)
===============================================================================
Wireless Stats:
Breakdown by type:
    Management Packets: 0          (0.000%)
    Control Packets:    0          (0.000%)
    Data Packets:       0          (0.000%)
===============================================================================
Fragmentation Stats:
Fragmented IP Packets: 1121       (2.794%)
    Fragment Trackers: 550
   Rebuilt IP Packets: 494
   Frag elements used: 4498
Discarded(incomplete): 0
   Discarded(timeout): 548
  Frag2 memory faults: 0
===============================================================================
TCP Stream Reassembly Stats:
        TCP Packets Used: 35904      (89.489%)
         Stream Trackers: 18097
          Stream flushes: 61
           Segments used: 62
   Stream4 Memory Faults: 0
===============================================================================
```

Figura 3.2.- Informe del tráfico detectado por Snort.

En el informe que se obtiene de Prelude-NIDS se observa que no existe pérdida de paquetes, pero un tanto por ciento de ellos es descartado por el *kernel* probablemente porque encuentra algún tipo de error en los paquetes. Además puede verse el tiempo empleado para procesar los paquetes recogidos.

```
*** Capture stats (not accurate if > 2e32-1 packet) ***

[eth0]: 44287 packets analyzed (prelude-nids counted).
[eth0]: 58334 packets received by filter (pcap counted).
[eth0]: 14047 (24.08%) packets dropped by the kernel (pcap counted).
[eth0]: 14047 (19.41%) packets dropped by the kernel (snort counted
(buggy)).
[all]: 44287 packets received by filter.

It took 2131.708622s real time to process 44287 packets.
It took 3.630000s CPU  time to process 44287 packets.

*** System stats (not accurate if > 2e32-1 packet) ***

Average cpu time by packet : 0.000082s, 0.081965ms, 81.965362us.
Page reclaims = 24725
Page faults = 487
Swap = 0

*** Plugin stats (not accurate if used > 2e32-1 times) ***

Waiting for asynchronous operation to finish.
```

Figura 3.3.- Informe del tráfico detectado por Prelude-NIDS.

También se ha comprobado el buen funcionamiento de las consolas para representación de datos de intrusión. Ya se hizo referencia a sus características en el punto 1.6.3.4.

3.1.3 Conclusiones

Después de realizar las mismas pruebas con ambos tipos de sensores se llega a las siguientes conclusiones:

- Ambos IDS son capaces de responder a todas las simulaciones de ataque en este escenario.

- La reacción de ambas herramientas es prácticamente semejante. En cuanto al motor de detección por comparación de reglas ambos se comportan idénticamente, presentando el mismo tipo y número de alertas. Esto es lógico, puesto que los dos utilizan la misma base de datos de firmas. La diferencia se da en cuanto a los preprocesadores o módulos de decodificación con los que cuentan. No obstante, en las pruebas realizadas en la

maqueta, esta diferencia sólo ha quedado de manifiesto en aquellas realizadas con Stick (como se ha comentado anteriormente). Decir que comportamiento es más o menos óptimo es una cuestión subjetiva. En cuanto al número de alertas, si bien teóricamente se supone que Prelude-IDS debiera enviar más, se observa que la cantidad es similar en ambos casos.

- El sensor Snort cuenta con un número mayor de preprocesadores y módulos de salida, y además, todos estos elementos tienen más opciones de configuración que aquellos de Prelude-NIDS. Es decir, resulta mucho más complicado filtrar alertas con el sensor Prelude-NIDS que si utilizamos Snort.

- En principio se observa que el sensor Prelude-NIDS tiene un tiempo de respuesta mejor, es decir, es más rápido. Esto se puede explicar debido a que, en el caso de Snort, es el mismo proceso el que se encarga del análisis de tráfico y de gestionar el almacenamiento de alertas. Sin embargo, la estructura modular de Prelude-IDS descarga de este trabajo al detector, siendo el manager quien realiza el almacenamiento de datos. Esta diferencia de rendimiento se reduce al utilizar la aplicación Barnyard que descarga a Snort de realizar este tipo de tareas.

- La estructura modular de Prelude-IDS permitirá un uso más sencillo del IDS, sobre todo si nos planteamos la implementación de un sistema de detección de intrusiones distribuido (varios sensores). Es más rápida y permite realizar cambios en su estructura más fácilmente. Por ejemplo, si hay sensores remotos, Prelude-IDS se ocupa automáticamente de realizar una comunicación cifrada entre ellos y el manager usando la librería *ssl*, mientras que en el caso de Snort debemos instalar otras aplicaciones que realicen esta tarea. Pero esto lo veremos más adelante. Además presenta la capacidad de poder integrar distintos tipos de sensores.

3.2 ELECCIÓN DE UNA SOLUCIÓN

Una vez verificada la utilidad de los IDS de libre distribución se debe decidir que solución se va a implementar en la red de la empresa. Esta solución debe cumplir con los siguientes objetivos:

- VLANs dentro de la red que nos interesa vigilar. Se quiere vigilar la zona de la DMZ2 para analizar todo el tráfico dirigido a la red de la empresa antes de ser filtrado por el *firewall,* y también, la zona de la DMZ dado que es la zona de la red donde se encuentran los equipos más importantes. El cumplimiento de este objetivo implica la utilización de una solución distribuida, es decir, debemos utilizar al menos dos sensores. Si se consigue que esta solución funcione correctamente, se podría añadir

más tarde un tercer sensor que se ocupará de analizar el tráfico que proviene de la Intranet.

Se quiere continuar la política de ahorro de medios, comenzada desde que se plantea solamente estudiar los IDS de software libre de distribución gratuita. Por tanto, a la hora de elegir el tipo de conexión de los sensores a la red de la empresa, optaremos por aquel que altera en menor medida la estructura de ésta y que, además, es más sencillo de llevar a cabo, es decir, la conexión utilizando puertos SPAN en los conmutadores LAN.

- Flexibilidad. Como se ha podido observar a lo largo de esta memoria, ambos IDS funcionan aceptablemente. Ocurre que hay características que presentan uno de los dos que no tiene el otro y que podrían ser interesantes. Estas características se ponen de manifiesto sobre todo en cuánto al tratamiento de datos que realizan las interfaces gráficas de uno y otro (véase comparación entre interfaces en el punto 1.6.3.4) y en cuanto a la capacidad de centralización de datos provenientes de diferentes sensores con la que cuenta la estructura modular de Prelude-IDS (Snort necesita de otras aplicaciones para realizar esta tarea). Por eso, se decide finalmente instalar ambos tipos de IDS, dotando de mayor flexibilidad la solución y pudiendo elegir a posteriori entre uno de los siguientes tipos de funcionamiento:

    - Utilizar una estructura Snort únicamente. Es decir, todos los sensores serían Snort. Estos sensores almacenarían los datos de alertas en la base de datos MySQL utilizando una conexión segura con ayuda de herramientas como Stunnel y OpenSSL. Accederíamos a dichos datos mediante la interfaz específica para Snort ACID.

      Las aplicaciones de notificación de alertas suelen utilizar los ficheros de *logs*. En este caso, si queremos centralizar los *logs* tendríamos que configurar otra aplicación externa a Snort (*syslog_ng*). No obstante, puesto que la estructura distribuida no cuenta con un gran número de sensores se puede considerar tratar los ficheros de *logs* en cada uno de los detectores por separado.

      Por otro lado, puesto que Snort es solamente un NIDS, si queremos utilizar un HIDS para dotar de mayor seguridad los sistemas, debemos utilizar Tripwire y mirar los informes por separado.

    - Utilización únicamente de Prelude-IDS. Consistiría en utilizar como NIDS y HIDS los sensores Prelude diseñados para ello (prelude-nids y prelude-lml). La gran ventaja de esta opción es que todos los datos estarán por defecto centralizados en el servidor, puesto que los sensores transmiten los datos de

alertas al manager de Prelude (prelude-manager) situado en el servidor y es éste quién se encarga de almacenarlos en la base de datos y crear los ficheros de *logs*. Además, la comunicación entre detectores y servidor se haría de forma segura automáticamente utilizando la librería OpenSSL. El problema sería que la interfaz de presentación de datos para Prelude (PIWI) es menos potente que aquella que utiliza Snort.

- Utilización mixta Prelude-Snort. Esta opción consistiría en utilizar un manager Prelude en el servidor y sensores Snort, con lo que aprovecharíamos las ventajas del modelo de centralización de datos de Prelude y la mayor potencia del detector Snort. Además, debemos tener en cuenta, que siempre que utilicemos Prelude, existirá la posibilidad de utilizar también otros tipos de sensores como Nessus. La desventaja es la utilización de la interfaz PIWI.

- Facilidad de acceso a los datos de intrusión. El administrador del sistema debe contar con una interfaz que le facilite la consulta de los datos de intrusión. Como ya se ha comentado, esto será posible gracias a la utilización conjunta de la consola de los IDS, ACID y PIWI junto con la base de datos MySQL y el servidor Apache. Por otro lado, sería conveniente automatizar un poco la gestión de incidencias de manera que el administrador pueda recibir por correo cada determinado periodo de tiempo un resumen de las alertas que se han producido, a fin de que se pueda hacer una valoración de la gravedad de los sucesos. Para ello se procederá a la instalación de dos programas o "scripts", uno encargado de realizar los informes y el otro encargado de enviarlos. Dicha instalación se puede ver con detalle en el Anexo I.

- Seguridad en la transmisión de cualquier dato de intrusión. Se instalarán las aplicaciones necesarias para que toda comunicación entre los detectores y el servidor y todo acceso a éstos desde cualquier otra máquina se realicen de manera segura. Los tres tipos de comunicaciones remotas que debemos cifrar son:

  - Acceso a los sistemas operativos donde están instaladas las aplicaciones IDS. Estos accesos se realizarán mediante SSH que permite abrir sesiones sobre máquinas remotas de manera segura cifrando los datos de autentificación y de sesión.

  - Acceso a la consola de datos de intrusión. Se requerirá el acceso a la consola utilizando una conexión "http" segura (https). Para ello se instalará Apache activando el módulo OpenSSL.

  - Comunicación entre el servidor y los detectores remotos. En el caso de utilización de Prelude-IDS, es la propia aplicación la que se encarga de cifrar

el canal de comunicación utilizando OpenSSL. Para realizar la comunicación entre un detector Snort y el servidor se utilizará Stunnel para cifrar el canal con la ayuda de SSL.

3.2.1 Ubicación del servidor

El servidor será el equipo que centralizará los datos en la base de datos MySQL y que tendrá instalado el servidor web Apache para poder acceder a dichos datos mediante interfaces gráficas como ACID o PIWI. También será donde resida el manager en el caso de utilizar Prelude. Este equipo, por tanto, deberá estar comunicado con el resto de sensores existentes en la red.

Puesto que la ubicación física tanto de los sensores como del servidor será una sala de máquinas, deberemos estudiar cuál es la ubicación más adecuada para el servidor dentro de la red, de manera que podamos acceder desde un puesto situado en la LAN interna vía SSH. Puesto que queremos hacer un aprovechamiento de los equipos, se decide colocar dicho servidor junto con uno de los sensores, lo que reduce las posibilidades de ubicación de dicho servidor a la DMZ2 o a la DMZ.

Entre las dos posibilidades anteriores, la única solución posible es la de colocarlo en la DMZ. Esto se explica debido a que la configuración del *firewall* no permite que llegue tráfico directamente desde la DMZ2 a la LAN interna. El tráfico proveniente de Internet llega a la red interna a través de la DMZ (ver esquema). Por tanto, desde la zona LAN interna sólo podremos mantener una comunicación vía SSH directa si el equipo está situado en la DMZ.

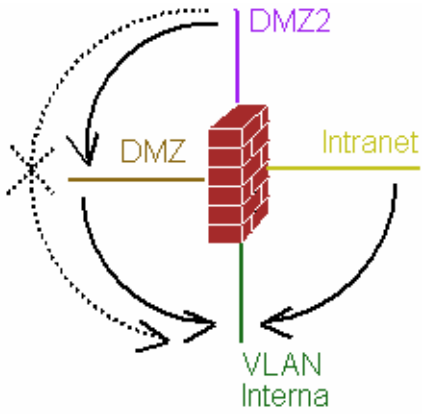


Figura 3.4.- Flujo de tráfico en el *firewall*.

Aunque no es recomendable instalar el sensor y el servidor en una misma máquina por razones de eficiencia, se ha pensado que el sensor situado en la DMZ no tendrá que analizar una cantidad excesiva de tráfico puesto que gran parte de éste se habrá filtrado antes en el

firewall y, por tanto, la pérdida de rendimiento será casi despreciable. Además utilizaremos un procesador lo suficientemente potente como para poder realizar ambas funciones sin problema. Puestos a situar el servidor en la misma máquina que un sensor, se ve también justificada la ubicación del servidor junto con el sensor de la DMZ. Otro motivo es que el servidor se encuentra más protegido ante cualquier posible ataque en la zona DMZ que en la DMZ2.

3.2.2 Síntesis de la solución

En resumen, la solución que se ha elegido consiste en utilizar dos sensores, un primero colocado sobre la DMZ2 y un otro sobre la DMZ. Junto con este último sensor y en el mismo equipo se instalará el servidor del IDS. Los sensores se conectarán a cada VLAN aprovechando la capacidad de realizar sesiones SPAN en los conmutadores LAN. Todas las posibles conexiones entre o con cualquiera de las dos máquinas se harán de manera segura. Sobre cada uno de los sensores tendremos la posibilidad de utilizar tanto Snort como Prelude-IDS.

3.3 EVOLUCIÓN DE LA SOLUCIÓN

No se ha tratado de implementar directamente la solución sino que se ha llevado a cabo por pasos. Esto ayuda a depurar más fácilmente los problemas que puedan ir surgiendo a lo largo de la instalación del sistema. Se empezó probando la instalación de un solo sensor en la red de la empresa para luego añadir el segundo y llegar hasta la solución propuesta.

3.3.1 Instalación del primer sensor

En esta primera fase se va a utilizar el mismo equipo que tenía instalada la configuración híbrida cuando se han realizado las pruebas en la maqueta.

Siguiendo la solución que se va a adoptar, debemos conectar el sensor sobre la VLAN DMZ2. Para ello tenemos que realizar una sesión SPAN sobre el conmutador donde está conectada la VLAN DMZ2 mediante el interfaz de configuración de dicho equipo, de tal manera que se configurará uno de los puertos para que sea capaz de recibir todo el tráfico que entra y sale de la VLAN DMZ2.

Podría pensarse que en principio no hace falta instalar nada más, sin embargo, sería interesante colocar otra tarjeta de red al equipo una vez que pretendemos conectarlo a un puerto SPAN. La razón principal es que, por razones de seguridad, no podemos enviar datos a través de una tarjeta de red que está conectada a un puerto SPAN. Esto implica, por un lado,

la incomunicación del sensor (aunque por el momento no estamos considerando una estructura distribuida), y por otro el desaprovechamiento de ciertas aplicaciones como la resolución de direcciones IP (para tener, por ejemplo, una noción más clara de donde provienen los ataques) o la notificación de alertas vía mail.

Por tanto, finalmente, el equipo quedaría conectado al conmutador ubicado en la zona DMZ2 mediante dos tarjetas de red. Una de ellas conectada a un puerto SPAN de la VLAN DMZ2 y la otra a otro puerto al que se le ha asignado una dirección IP dentro de esa misma VLAN DMZ2.

Se ha observado que la conexión del IDS al conmutador utilizando un puerto SPAN no sobrecarga la memoria de éste. Se ha comprobado que el consumo de memoria por parte del conmutador está alrededor de un 20% de la capacidad disponible.

3.3.2 Instalación del servidor más el segundo sensor

Para la instalación del servidor más el sensor es conveniente utilizar un equipo más potente. Hay que tener en cuenta que el servidor debe manejar un número importante de datos en tiempo de real, además de realizar todo el tratamiento de los datos almacenados para presentarlos de diferentes maneras a través de las consolas de datos.

El equipo que utilizaremos para albergar toda la parte del servidor tendrá las siguientes características: procesador IBM Intel Xeon 2,8G, 36,4 G de disco duro y otro disco de copia de seguridad de la misma capacidad y 512 M de memoria RAM.

Esta máquina se situará dentro de la DMZ y deberá contar como la anterior, y por razones similares, con dos tarjetas de red. Una de ellas será la que utilizará el sensor para recibir todo el tráfico de ésta VLAN a través de un puerto SPAN configurado a tal efecto en el conmutador correspondiente, y la otra se conectará al mismo *switch* y tendrá asignada una dirección IP de la subred DMZ. A través de esta última tarjeta de red se realizarán las comunicaciones entre el servidor y el sensor situado en la DMZ2.

La instalación de un equipo que comprende tanto un servidor IDS como un sensor viene explicada en el ANEXO I.

3.3.3 Comunicaciones y seguridad en el modelo distribuido

En el modelo distribuido los equipos quedarían conectados a la red como se muestra en la figura 3.5.

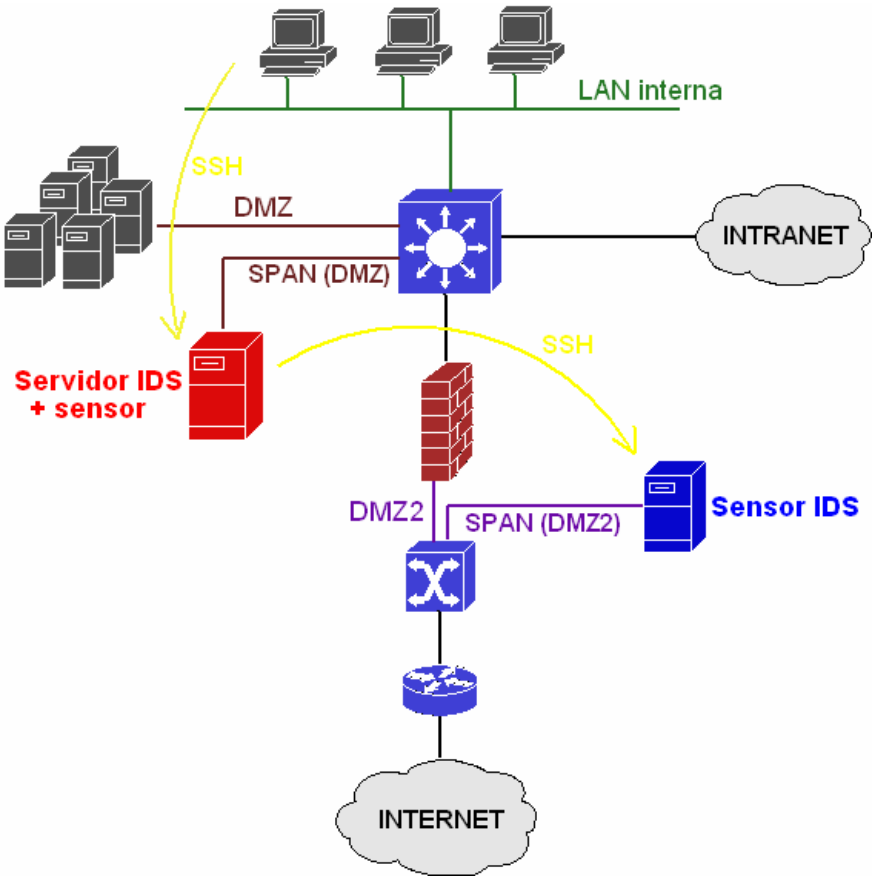


Figura 3.5.- Conexión de los detectores a la red.

A la hora de tratar con un sistema distribuido, el primer problema que se plantea es como realizar las comunicaciones. En un esquema de red como el que presenta la empresa, esto va a quedar muy marcado por la configuración del *firewall*. Tenemos dos necesidades principales en cuanto a comunicación entre equipos:

- Es necesario poder acceder a ambos equipos remotamente, ya que no sería práctico tener que desplazarse a la sala de máquinas cada vez que nos interese hacer alguna operación sobre ellos. Como se ha comentado anteriormente, la configuración del *firewall* no permite acceder desde la LAN interna a la zona del *firewall* por donde salen los datos dirigidos hacia Internet, es decir, la DMZ2. Pero sí que podemos acceder a la DMZ y desde la DMZ a la DMZ2. Puesto que, en principio, no hay ningún problema para tomar el control remoto vía SSH de una máquina situada en la DMZ, utilizaremos este último equipo para tomar a su vez el control vía SSH de la otra máquina ubicada en la DMZ2.

- Es necesario lograr una comunicación entre el sensor situado en la DMZ2 y el servidor. El *firewall* sólo deja pasar a la red de la empresa el tráfico que cumple con unas ciertas características en cuanto a direcciones IP y puertos de destino. Por tanto,

si queremos comunicar el sensor con el servidor, debemos saber que puertos son necesarios para establecer dicha comunicación y configurar el *firewall* para dejar pasar el tráfico dirigido a dichos puertos. En nuestro caso, estos puertos serán:

- Puerto 22: SSH. Para permitir que se pueda tomar el control del sensor desde el servidor.

- Puerto 3307: MySQL seguro. Es el puerto que utilizamos para transmitir los datos de manera segura (cifrados) desde el sensor a la base de datos en el servidor.

- Puerto 5553. Es el puerto que utiliza Prelude-IDS para registrar los sensores.

- Puerto 5554. Es el puerto que utiliza Prelude-IDS para realizar la comunicación entre manager y sensores.

Por otro lado, es importante que toda transmisión de datos se haga de manera cifrada, tanto la transmisión de datos de los sensores al servidor como el acceso a los datos del servidor utilizando la aplicación Apache. Dos razones importantes son:

- Privacidad de los datos. De esta manera evitamos que un posible intruso pueda ver los datos y averigüe, por ejemplo, que estamos haciendo uso de un detector de intrusiones.

- Disponibilidad del IDS. El modo de funcionamiento de cualquiera de los dos IDS que vamos a instalar es por comparación de firmas de ataques, es decir, busca si las características y contenido de un paquete corresponde con alguna de las firmas de ataques almacenadas en la base de datos de reglas. Sucede que muchas de esas reglas o firmas consisten en encontrar una cadena de caracteres determinada dentro un paquete. Esa cadena de caracteres que hace que el IDS envíe una alerta, coincide en muchos casos con el nombre con el que el IDS transmite la alerta para notificarla. Si las comunicaciones no están cifradas, esta situación creará un bucle de alertas puesto que el IDS está escuchando la red y volverá a ver dicha cadena de caracteres. Por ejemplo, si el IDS utiliza una regla que provoca la emisión de una alerta cada vez que encuentra la palabra “EXPLOIT Apache” y el tráfico no está cifrado, el IDS emitirá otra vez la misma alerta esta vez a causa de la notificación anterior (si la notificación en sí contiene esa misma palabra). Esta situación puede llegar a saturar el servidor y dejarlo indisponible. Es imprescindible, por tanto, cifrar toda comunicación de alertas.

  Este problema se soluciona utilizando la aplicación Stunnel junto con OpenSSL, para cuya instalación se hace referencia de nuevo al Anexo I.

En definitiva, la estructura de comunicación entre las diferentes aplicaciones dentro del sistema distribuido se muestra en la figura 3.6.

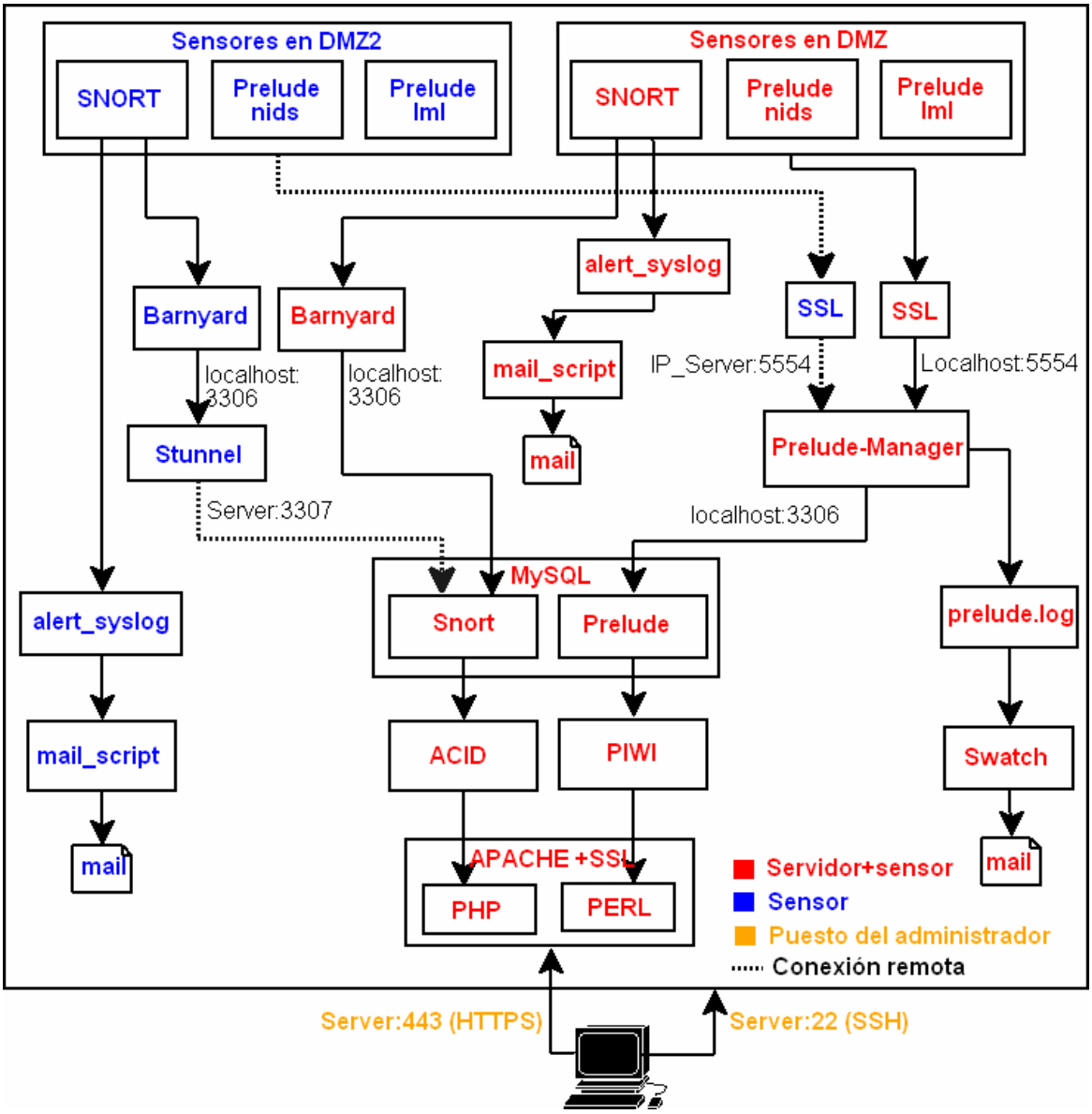


Figura 3.6.- Relación y comunicación entre las diferentes aplicaciones en la arquitectura distribuida.

3.3.4 Otras medidas de seguridad

Nuestro sistema de detección de intrusiones debe ser inaccesible para los atacantes. En el punto anterior, hemos visto la importancia de cifrar las comunicaciones. Esta medida también puede ayudar a ocultar la existencia de un IDS. No obstante, se ha pensado que la manera más segura de proteger el IDS, es impidiendo su acceso a Internet, de manera que nadie desde el exterior de la empresa sea capaz de poder acceder a cada uno de los detectores.

Como se ha comentado en el capitulo anterior, en la estructura de la red se encuentran dos puntos de NAT, uno en el *firewall* y otro en el *router* que da acceso a la red Internet. Esta segunda conversión de direcciones no se efectúa para ninguna de las dos máquinas que contienen los detectores IDS. Esta medida permite que ambos equipos sean inaccesibles desde la parte Internet, pero también presentará inconvenientes como la incapacidad de realizar actualizaciones automáticas de reglas a través de Internet o el envío de notificaciones a organizaciones como el CERT.

Otras medidas de seguridad que se han llevado a cabo en el sistema son las propias del proceso de instalación del IDS bajo el sistema operativo RedHat 8 (ver ANEXO I) como la actualización y corrección de vulnerabilidades en las aplicaciones del sistema, la eliminación de servicios y usuarios inútiles, la prohibición del acceso al sistema como administrador y la creación de usuarios mediante SUDO, la protección de ficheros importantes, etc.

4 CONFIGURACIÓN Y OPTIMIZACIÓN

4.1 CONFIGURACIÓN DE LOS SENSORES

Una vez que el sistema de detección está instalado y es capaz de analizar el tráfico y transmitir alertas, queda realizar la tarea más complicada que consiste en configurar los sensores de manera que se minimice el número de falsos positivos evitando caer en los falsos negativos. Es muy difícil decidir que regla crea falsos positivos y más complicada aún la decisión de eliminarla o no. Puede que una regla cree un 90% de falsos positivos, pero el 10% de acierto que resta sea tan interesante que no nos interese quitar esa regla. En otras ocasiones podemos pensar que la regla no corresponde a un ataque lo suficientemente importante como para que merezca la pérdida un 10% de falsos negativos. Por supuesto, y por defecto, siempre es preferible obtener muchos falsos positivos que dejar de detectar ataques reales.

Debido a que el proceso de decisión descrito anteriormente no es fácil, la optimización de la configuración es un proceso lento. La capacidad de detección y, por tanto, el número de falsos positivos o negativos que obtengamos, depende sobre todo de la configuración de los preprocesadores y de las reglas o firmas de ataques. Es preferible partir de una configuración muy sensible que genere una cantidad importante de falsos positivos, pero que sea más fiable a la hora de no fallar en la detección de ataques reales. Se verá como el sensor Snort presenta muchos más elementos configurables que Prelude. A raíz de la observación en el tiempo de los datos de alertas que se obtengan, iremos cambiando la configuración del IDS. Dicha configuración referida al último día de estancia en la empresa queda reflejada en el Anexo II.

4.1.1 Configuración del sensor Snort

El fichero de configuración de Snort se divide en cuatro partes: configuración de las variables de red (descripción de la red en términos de direcciones IP), configuración de preprocesadores, configuración de módulos de salida y configuración de las reglas a tener en cuenta.

Veamos como ha ido evolucionando la configuración de cada una de estas partes a lo largo del desarrollo del proyecto:

- Configuración de las variables de red. Se comenzó con una definición de variables de red lo más ambigua posible para detectar el rango más amplio de ataques. Dichas variables estaban definidas de tal manera que tanto la red interna como la externa

pudieran ser cualquier dirección IP ($HOME_NET=any, $EXTERNAL_NET=any), e igualmente las variables referidas a los servidores (todas ellas inicializadas a valor *any*). No obstante, pronto se decide que no merece la pena detectar ciertas tentativas de ataques del interior de la red hacia el exterior, ya que en su mayoría serían falsos positivos y además, es prácticamente imposible que se lleven a cabo con éxito debido a la configuración de la red. Por tanto, se modifican las variables de red de tal manera que la descripción de la red queda reflejada como sigue:

- *$HOME_NET=192.168.25.0/24* y *$EXTERNAL_NET=any*. Esta configuración evita detectar ciertos ataques procedentes de nuestra red hacia el exterior. Sin embargo, si que detectarán todos aquellos que se dirigen hacia nuestra red.

- *$DNS_SERVERS=192.168.25.100, 192.168.25.101*; *$SMTP=$HOME_NET; $HTTP_SERVERS=$HOME_NET; $SQL_SERVERS=$HOME_NET*. En general, conviene saber cuando se está intentado realizar un ataque, aunque, por ejemplo, un ataque SQL tenga como destino un equipo que carece de este servicio. Esto significaría que el ataque no se podrá llevar a cabo, pero desvela las intenciones del atacante. Por ello, se decide inicializar los servidores con el valor de toda la red interna, a excepción del servidor DNS que se inicializará con la dirección IP real dentro de la red. Se decide tomar esta medida debido a que los servidores DNS son una de las principales fuentes de falsos positivos.

- *$HTTP_PORTS 80, $SHELLCODE_PORTS !80, $ORACLE_PORTS 1521*. Esta es la definición de puertos que tomaremos.

- Configuración de los preprocesadores. La configuración de los preprocesadores queda de la siguiente manera:

  - Frag2. Mantenemos este preprocesador activado para poder detectar los ataques basados en la fragmentación IP. Es recomendable cambiar un poco los parámetros que vienen por defecto en la configuración puesto que los atacantes pueden aprovechar el conocimiento y realizar los ataques de manera que no sean detectables. Cambiaremos:

    - *timeout = 65*. Son los segundos durante los que un fragmento será guardado. Si después de este tiempo no se ha producido un ensamblado de paquetes, dicho fragmento será suprimido.

    - *min_ttl = 3*. Todo paquete con el campo TTL menor que 3 no se tendrá en cuenta.

- *ttl_limit = 8*. Indica la diferencia máxima que puede haber entre los campos TTL de los paquetes que se van a reensamblar.

- *Stream4*. Con este preprocesador Snort puede seguir el estado del flujo TCP lo que le permite detectar ciertos ataques de recuperación de información (ayuda a identificar las firmas de ataque repartidas en varios paquetes). Vamos a activar o cambiar los siguientes parámetros:

  - *detect_scan*. Activa la función de detección de rastreo de puertos con la que cuenta este preprocesador.

  - *disable_evasion_alerts*. Esta opción desactiva una parte de alertas correspondiente a tentativas de intrusión poco utilizadas hoy en día. Por ejemplo, detecta el solapamiento TCP. Se ha desactivado este tipo de detección porque de otra manera se obtienen una gran cantidad de falsos positivos.

  - *min_ttl = 3*. Siguiendo el mismo razonamiento que el preprocesador anterior.

  - *ttl_limit = 8*. Mismo caso que con el preprocesador anterior.

  - *timeout = 35*. Razonamiento similar al del preprocesador anterior.

- *Stream4_reassemble*. Este preprocesador estará activado utilizando la opción *both* que permitirá ensamblar las sesiones establecidas tanto por parte del cliente como por parte del servidor.

- *http_decode*. Este preprocesador se encarga de detectar el tráfico HTTP anormal y lo normaliza de manera que el motor de detección lo pueda interpretar. Se activarán todas las opciones de manera que pueda detectarse todo método de codificación sospechosa en una petición URI. Son las siguientes opciones:

  - *port_list = 80*. Puertos sobre los que se utiliza el protocolo HTTP. ARSOÉ sólo utiliza el puerto 80 para las aplicaciones web.

  - *unicode*. Normaliza todas las cadenas “unicode” a formato ASCII.

  - *iis_alt_unicode*. Se detecta la codificación “%u” (método propietario de codificación de URI HTTP de los servidores web IIS).

- *double_encode*. Detecta los intentos de codificación doble en un URI, es decir, ataques que utilizarían dos niveles de codificación y, por tanto, sería necesario normalizar dos veces dicho URI.

- *iis_flip_slash*. Esta opción permite decodificar ciertos separadores de directorios propietarios de Microsoft ("\" por "/").

- *full_whitespace.* Decodifica "/t" como un espacio para el caso de servidores web Apache.

- *rpc_decode*. Funciona de manera análoga al preprocesador anterior pero con el protocolo RPC. Sólo podemos indicar como parámetros la lista de puertos RPC a normalizar. Dejaremos los valores establecidos por defecto, 111 y 32771.

- *bo*. Este preprocesador detecta "Back orifice" que es un caballo de Troya de control remoto para los sistemas Windows. No cuenta con ningún parámetro configurable.

- *telnet_decode*. Realiza la misma función que los preprocesadores *http_decode* y *rpc_decode* pero con el protocolo Telnet. No utiliza ningún argumento.

- *portscan*. Es capaz de detectar paquetes UDP y TCP SYN yendo hacia un número configurable de puertos en menos de un número también configurable de segundos. Por defecto, considera escaneo de puerto si hay intentos de conexión con 4 puertos en menos de 3 segundos. Nosotros vamos a utilizar los siguientes parámetros:

  - *$HOME_NET*. Con esta opción consideramos sólo los escaneos sobre nuestra red.

  - Se emitirá una alerta por escaneo de puertos si se detecta un barrido de 6 puertos en 3 segundos.

  - *portscan.log*. Es el nombre del fichero donde almacenaremos la información que proporcione este preprocesador. Siguiendo la configuración de Snort, dicho fichero estará situado en */var/log/snort/*.

- *Portscan_ignorehosts*. Permite ignorar los escaneos de puertos provenientes de determinadas direcciones IP o redes. En nuestro caso eliminaremos los escaneos provenientes de las siguientes redes:

- *62.39.0.0/16*. Se van a filtrar los escaneos provenientes de la red designada por esta dirección IP puesto que corresponde a uno de los proveedores de servicio de ARSOÉ y es normal que se detecte una actividad similar a la de los escaneos de puertos.

Los preprocesadores que desactivaremos serán:

- *conversacion*, *portscan2* y *portscan2_ignorehosts*. Trabajan los tres en el mismo objetivo, detectar escaneos de puertos. Por el momento los dejaremos desactivamos puesto que se ha comprobado que generan una gran cantidad de falsos positivos y además ya utilizamos el preprocesador *portscan* para realizar prácticamente la misma tarea.

- *arpspoof* y *arpspoof_detect_host*.

- Configuración de los módulos de salida. Se van a utilizar los siguientes módulos de salida de Snort:

  - Módulo alert_syslog. Este módulo de salida será utilizado por la aplicaciones de notificaciones de alertas vía mail.

  - Módulos unified. Configuraremos Snort de tal manera que guarde los datos en un fichero utilizando este formato. Estos ficheros serán utilizados por la aplicación Barnyard para escribir en la base de datos.

  - Módulo *alert_prelude*. Activaremos este módulo de salida si queremos enviar las alertas a *Prelude-manager* en el caso en que se trabaje con una arquitectura Prelude-IDS.

- Configuración de las reglas aplicables. Se trata de eliminar aquellas reglas que consideramos que provocan una gran cantidad de falsos positivos o que no proporcionan información interesante. Enumeramos a continuación dichas reglas (recordemos que el fichero de reglas se encuentra en */etc/snort/):*

  - Fichero de reglas icmp-info.rules. Existen dos ficheros de reglas que hacen referencia exclusivamente al uso del protocolo ICMP, *icmp.rules* e *icmp-info.rules*. El primero contiene reglas que representan diversas huellas de herramientas de scanner, pero el segundo contiene reglas que informan del tipo de actividad ICMP. Este último genera una gran cantidad de alertas y dado

que la información que nos ofrece no parece demasiado interesante, se decide eliminar este fichero.

- Regla “BAD-TRAFFIC loopback traffic”. Esta regla que se encuentra en el fichero *bad-traffic.rules* genera multitud de falsos positivos.

- Regla “WEB-CGI calendar access”. Esta regla ubicada en el fichero *web-cgi.rules* genera muchos falsos positivos y conviene quitarla. Se sabe que las alertas que se producen se deben a la existencia de una página en uno de los servidores web cuyo nombre contiene la palabra “calendar”.

- Regla “ICMP PING speedera”. Esta regla no representa en sí un ataque, el problema es que algún atacante enmascarara su “ping” bajo esta forma, pero es algo bastante improbable. Sin embargo, se producen una cantidad importante de estas alertas y por ello la eliminaremos.

Las reglas anteriores se han suprimido tanto en el detector ubicado en la DMZ como en aquel situado en la DMZ2. En este último se suprimirán además otras dos reglas:

- Regla “MS-SQL Worm propagation attempt”. Esta regla provoca una cantidad considerable de alertas. Aunque puedan representar verdaderos ataques, no se dispone de servidores SQL en la red. Por tanto, se decide desactivar esta regla.

- Regla “MS-SQL version overflow attempt". Esta regla se suprimirá por el mismo motivo que la anterior.

- Filtros aplicados. Se pueden crear reglas que sirven para filtrar determinado tipo de tráfico. Vamos a editar estos filtros en el fichero *local.rules* (ver anexo III). El contenido de este fichero será diferente dependiendo del detector. Para el detector situado en la DMZ:

  - Filtro para suprimir las alertas provocadas por el sitio web www.snort.org. Éste impedirá que se obtengan alertas cuando se realiza una consulta de información en la base de datos de reglas de Snort (*pass tcp 199.107.65.177/32 any -> any any*).

  - Filtro para eliminar todas las alertas provenientes de la supervisión (*pass tcp 192.168.25.200 any -> any any*, *pass udp 192.168.25.200 any -> any anypass* e *icmp 192.168.25.200 any -> any any*).

- Filtro para evitar las alertas provenientes del "*proxy cache"* destinadas al puerto 9935 del "*proxy antivirus"* (*pass tcp 192.168.25.95 any -> 192.168.25.90 9935*).

- Filtro para eliminar ciertas alertas relativas al protocolo POP3 que provienen de la propia VLAN DMZ de ARSOÉ, es decir, son alertas que se producen debido a un tráfico interno entre distintos servidores de la red (ver anexo III).

- Filtros para no tener en cuenta ciertas alertas « P2P GNUTella GET » provocadas por los administradores de la red (ver anexo III).

- Filtros para no tener en cuenta ciertas alertas « SCAN Squid Proxy attempt » provocadas por los administradores de la red (ver anexoIII).

Para el detector situado en la DMZ2:

- Filtro para suprimir las alertas provocadas por la consulta de información en la base de datos de reglas de Snort (*pass tcp 199.107.65.177/32 any -> any any*).

4.1.2 Configuración del sensor Prelude-NIDS

El fichero de configuración de Prelude-NIDS tiene menos elementos configurables, pudiéndolo dividir en tres partes: variables de la red, configuración de reglas y configuración de módulos de detección. Las dos primeras partes se configuran en el fichero */usr/local/etc/prelude-nids/ruleset/prelude.rules* y la tercera en el fichero */usr/local/etc/prelude-nids/prelude-nids.conf*.

Si se usa como sensor Prelude-NIDS, se intentará reflejar en su configuración las mismas medidas que se comentaron en el apartado anterior para Snort:

- Variables de red. Utilizaremos las mismas definiciones de variables que el caso de Snort.

- Configuración de reglas. Puesto que Prelude-NIDS utiliza la misma base de datos de reglas que Snort, suprimiremos de la configuración las mismas reglas que hemos comentado en el apartado anterior.

- Módulos de detección. Los módulos de detección son equivalente a los preprocesadores de Snort. Hay muchos de ellos que realizan las mismas funciones. A continuación los preprocesadores activados:

- [PreludeNIDS]. Este módulo debe estar activado y cuenta con un parámetro que está inicializado con la dirección del manager de Prelude: *manager-addr = 192.168.25.50*.

- [Tcp-Reasm]. Es el módulo equivalente a *Stream4_reassemble* en Snort. Cuenta con los mismos parámetros, por lo que utilizaremos solamente el argumento *both*.

- [SnortRules]. Este módulo sirve para poder tomar en cuenta la configuración de reglas de Snort. Solamente debemos indicar el camino hasta el fichero de configuración de reglas y variables : *ruleset=/usr/local/etc/prelude-nids/ruleset/prelude.rules*

- [ScanDetect]. Este preprocesador está diseñado para detectar los escaneos de puertos. Vamos a utilizar los siguientes valores para estos parámetros:

  - *high-port-cnx-count = 50*.

  - *low-port-cnx-count = 5*.

  - *cnx-ttl = 60*.

- [RpcMod]. Detecta el tráfico RPC anormal y lo normaliza. Configuramos los puertos RPC: *port-list = 111 32771*.

- [TelnetMod]. Normaliza los caracteres de las sesiones Telnet.

Por otro lado, se desactivarán los módulos de detección siguientes:

- [Shellcode]. Este módulo permite la detección de código polimórfico pero consume demasiados recursos de la CPU, por lo que no lo utilizaremos.

- [Debug]. Este módulo envía una alerta por cada paquete. Lo desactivamos porque esta opción no sería práctica por motivos evidentes.

- [HTTPMod]. Este módulo de detección que se encargaría de detectar el tráfico anormal y normalizarlo, hemos comprobado que genera una cantidad intratable de alertas que se han considerado falsos positivos. Por tanto, hemos decidido desactivarlo.

- [ArpSpoof].

4.2 CONFIGURACIÓN DE OTRAS APLICACIONES

La configuración del resto de aplicaciones necesarias para la puesta en marcha de un IDS no afecta en nada a la optimización del funcionamiento del mismo, sino que es imprescindible para el funcionamiento del sistema IDS en sí. Dicha configuración puede verse en el Anexo I (manual de instalación).

5 MANTENIMIENTO

El mantenimiento de un sistema de detección de intrusiones es indispensable. Hace falta realizar tareas de optimización y de actualización periódicamente para que el IDS no se convierta en una herramienta inútil. Debemos ocuparnos de llevar a cabo los siguientes trabajos:

- Actualización de reglas. Es una tarea muy importante. Numerosos ataques nuevos son descubiertos prácticamente a diario y la manera de detectarlos es incluir las nuevas representaciones de estos ataques en la base de datos de reglas de nuestro sistema. Existen aplicaciones que permiten actualizar automáticamente los ficheros de reglas pero para utilizarlos sería necesaria la conexión a Internet. Como se ha comentado anteriormente en esta memoria, la posibilidad de conectar el sistema a Internet se había descartado por lo que deberemos realizar la actualización de reglas manualmente. Cuanto más frecuente se hagan estas actualizaciones se dispondrá de un mayor nivel de seguridad en el sistema. Como mínimo se deberá hacer una vez al mes. Podemos encontrar los ficheros de reglas actualizados en http://www.snort.org/dl/rules/. En nuestro sistema, las reglas se encontraban en el directorio */etc/snort/*. Al reemplazar las reglas antiguas por las más actuales debemos tener cuidado de no suprimir las modificaciones que habíamos hecho (reglas suprimidas o filtros creados con el objetivo de optimizar el funcionamiento del IDS).

- Optimización del IDS. Se recomienda analizar periódicamente el comportamiento del sistema. Es decir, es importante estudiar asiduamente las alertas que el IDS emite con el objetivo de reducir el número de falsos positivos, evitar falsos negativos y ajustar cada vez de forma más precisa la detección a nuestras necesidades. Esto se traducirá en una mejora de las prestaciones del IDS y facilitará a posteriori la gestión del administrador.

- Parchear vulnerabilidades. Las vulnerabilidades en las aplicaciones son probablemente el medio más utilizado de entrar y atacar un sistema. Aunque el sistema IDS que hemos implementado no es accesible desde el exterior de la red de la empresa, sería prudente parchear frecuentemente las posibles vulnerabilidades existentes. Las páginas web de las aplicaciones distribuyen normalmente los parches necesarios para las vulnerabilidades encontradas en dichos programas.

- Rotación de ficheros de "logs". Tras observar durante un periodo de tiempo la cantidad de información que genera el IDS, se llega a la conclusión de que no existe ningún problema en cuanto a capacidad en las máquinas que se utilizan. Sin embargo,

hay que tenerlo en cuenta. Una frecuencia de rotación de ficheros de “logs” de un año es más que suficiente. Debemos también realizar una rotación de los datos almacenados en la base de datos. Esto último permitirá descargar de trabajo la consola, que aumentará la velocidad de tratamiento de los datos. Los ficheros a rotar serían:

- Ficheros en */var/log/snort*
- Ficheros en */var/log/snort-old*
- Ficheros en */var/log/snort-week*
- Ficheros en */var/log/snort-month*
- Ficheros en */var/log/prelude.log* y */var/log/prelude-xml.log*

6 RESULTADOS

Se van a mostrar los resultados obtenidos por ambos detectores siguiendo varios criterios: lista de alertas obtenidas ordenadas por su frecuencia, puertos más afectados por los ataques, tipos de ataques más frecuentes, etc. Los datos se han tomado en los siguientes periodos de tiempo:

- Para el detector situado en la DMZ2: del 18 de marzo al 19 de mayo del 2004.

- Para el detector situado en la DMZ: del 30 de abril al 19 de mayo del 2004.

Hay que tener en cuenta que durante estos periodos de tiempo se han realizado tareas de optimización del funcionamiento del IDS, es decir, se ha procedido a la eliminación de determinadas firmas de ataque o a la realización de filtros para evitar obtener aquellas alertas que se han considerado como falsos positivos. No obstante, dichas alertas aparecerán en los resultados que se van a mostrar a continuación puesto que han formado parte de los datos de detección.

En la tabla 6.1 se detalla la lista de alertas obtenidas por ambos sensores. En la tabla se especifica el nombre de cada una de las alertas, el número detectado de cada una de ellas, el número de sensores que han detectado dichas alertas, y el número de direcciones IP origen y destino relacionadas con cada una de estas alertas.

Alertas	**Nº de alertas**	**Nº de sondas**	**Nº de IPs origen**	**N de IPs destino**
ICMP PING NMAP	89756 (34%)	2	17776	22
P2P GNUTella GET	75399 (28%)	2	47	92
SCAN Squid Proxy attempt	16848 (6%)	2	85	12
SHELLCODE x86 NOOP	10827 (4%)	2	499	20
ICMP Destination Unreachable (Undefined Code!)	10337 (4%)	1	90	8
MS-SQL Worm propagation attempt	7986 (3%)	1	4180	11
WEB-IIS cmd.exe access	6720 (3%)	2	125	21
SHELLCODE x86 unicode NOOP	5765 (2%)	2	3	2
WEB-IIS WEBDAV nessus safe scan attempt	4047 (2%)	2	2953	14
ICMP Destination Unreachable (Communication Administratively Prohibited)	3269 (1%)	2	66	19
WEB-MISC robots.txt access	3095 (1%)	2	270	10
SCAN SOCKS Proxy attempt	2730 (1%)	2	125	15
WEB-MISC WebDAV search access	2463 (1%)	2	879	14
VIRUS OUTBOUND .doc file attachment	1825 (1%)	2	84	88

ICMP Large ICMP Packet	1690 (1%)	2	94	57
WEB-ATTACKS mail command attempt	1548 (1%)	2	2	44
WEB-MISC Transfer-Encoding: chunked	1450 (1%)	1	14	5
ICMP Time-To-Live Exceeded in Transit (Undefined Code!)	1447 (1%)	1	118	8
ICMP PING CyberKit 2.2 Windows	1055 (0%)	2	65	22
ATTACK-RESPONSES 403 Forbidden	967 (0%)	2	58	35
WEB-FRONTPAGE /_vti_bin/ access	954 (0%)	2	76	14
WEB-IIS %2E-asp access	950 (0%)	1	1	93
Virus - Possible pif Worm	793 (0%)	1	423	2
SHELLCODE x86 inc ebx NOOP	766 (0%)	2	81	17
SCAN Proxy (8080) attempt	757 (0%)	1	43	25
SCAN nmap TCP	655 (0%)	1	19	6
VIRUS OUTBOUND .exe file attachment	625 (0%)	2	108	32
WEB-CGI redirect access	602 (0%)	2	2	154
WEB-MISC ?open access	544 (0%)	1	1	20
WEB-IIS unicode directory traversal attempt	499 (0%)	2	23	14
WEB-CLIENT Javascript URL host spoofing attempt	464 (0%)	2	31	2
SCAN Proxy Port 8080 attempt	459 (0%)	2	16	28
WEB-IIS unicode directory traversal attempt	417 (0%)	2	21	10
WEB-MISC /doc/ access	413 (0%)	2	2	53
WEB-IIS _mem_bin access	391 (0%)	2	24	18
WEB-IIS unicode directory traversal attempt	390 (0%)	2	23	10
VIRUS OUTBOUND bad file attachment	311 (0%)	1	190	3
ICMP superscan echo	300 (0%)	2	19	22
POP3 PASS overflow attempt	264 (0%)	1	2	1
WEB-MISC Apache Chunked-Encoding worm attempt	264 (0%)	2	1	10
WEB-IIS unicode directory traversal attempt	260 (0%)	2	20	10
PORN ejaculation	249 (0%)	2	42	2
WEB-MISC Chunked-Encoding transfer attempt	245 (0%)	2	3	9
WEB-IIS view source via translate header	245 (0%)	2	18	21
POP3 TOP overflow attempt	231 (0%)	1	1	1
POP3 DELE overflow attempt	230 (0%)	1	1	1
PORN masturbation	207 (0%)	2	45	2
PORN dildo	206 (0%)	2	26	3
PORN BDSM	202 (0%)	2	56	2
WEB-MISC http directory traversal	197 (0%)	2	29	22
WEB-IIS iissamples access	195 (0%)	2	1	3
WEB-MISC login.htm access	195 (0%)	2	2	5
POP3 STAT overflow attempt	192 (0%)	1	1	1
POP3 USER overflow attempt	192 (0%)	1	1	1
DNS zone transfer TCP	178 (0%)	2	3	4
WEB-CGI scriptalias access	145 (0%)	2	2	12
POLICY FTP anonymous login attempt	145 (0%)	2	36	29
SHELLCODE x86 NOOP	142 (0%)	2	31	7
WEB-IIS CodeRed v2 root.exe access	125 (0%)	2	26	14

Virus - Possible scr Worm	124 (0%)	1	65	1
BACKDOOR typot trojan traffic	118 (0%)	1	22	10
WEB-IIS ISAPI .ida attempt	118 (0%)	2	74	13
WEB-IIS nsiislog.dll access	117 (0%)	2	27	14
BAD-TRAFFIC udp port 0 traffic	108 (0%)	1	2	9
WEB-CGI count.cgi access	107 (0%)	2	2	28
snort_decoder: Truncated Tcp Options	103 (0%)	2	30	13
ICMP PING Delphi-Piette Windows	89 (0%)	1	8	9
ICMP PING Windows	81 (0%)	1	37	5
WEB-MISC .htaccess access	76 (0%)	1	1	17
WEB-PHP viewtopic.php access	67 (0%)	1	1	17
WEB-CGI yabb access	66 (0%)	1	1	4
SMTP rcpt to command attempt	65 (0%)	2	10	6
VIRUS OUTBOUND .com file attachment	63 (0%)	2	44	3
WEB-CGI csh access	61 (0%)	1	1	10
ICMP PING speedera	60 (0%)	2	33	2
MISC source port 53 to <1024	59 (0%)	1	6	11
INFO FTP No Password	56 (0%)	2	6	3
DNS zone transfer UDP	55 (0%)	2	11	2
MS-SQL version overflow attempt	54 (0%)	1	4	11
PORN free XXX	54 (0%)	2	15	2
WEB-MISC http directory traversal	51 (0%)	2	2	2
WEB-CGI formmail access	42 (0%)	2	10	6
WEB-IIS srchadm access	42 (0%)	2	1	3
SMTP rcpt to sed command attempt	42 (0%)	1	25	5
SHELLCODE x86 0x90 NOOP unicode	41 (0%)	1	2	1
WEB-MISC backup access	40 (0%)	2	3	7
WEB-MISC Lotus Notes .exe script source download attempt	40 (0%)	1	1	7
VIRUS OUTBOUND .scr file attachment	25 (0%)	1	26	3
WEB-IIS iisadmin access	25 (0%)	2	1	3
spp_stream4: SYN FIN Stealth Scan	36 (0%)	1	3	10
WEB-IIS SAM Attempt	35 (0%)	2	2	4
BAD-TRAFFIC loopback traffic	34 (0%)	1	1	11
MISC MS Terminal server request (RDP)	31 (0%)	1	3	3
WEB-IIS ISAPI .printer access	31 (0%)	2	7	18
WEB-CGI icat access	30 (0%)	2	2	1
WEB-IIS Unicode2.pl script (File permission canonicalization)	30 (0%)	2	1	3
spp_stream4: Retransmission	30 (0%)	1	8	5
POLICY FTP anonymous (ftp) login attempt	29 (0%)	1	6	1
WEB-MISC ftp attempt	28 (0%)	2	3	7
WEB-CGI db2www access	27 (0%)	1	1	2
INFO Connection Closed MSG from Port 80	27 (0%)	2	9	2
PORN anal sex	26 (0%)	2	7	2
WEB-PHP phpBB privmsg.php access	25 (0%)	2	2	1
WEB-MISC Domino domcfg.nsf access	24 (0%)	1	1	4
WEB-FRONTPAGE shtml.exe access	24 (0%)	2	3	10
ICMP traceroute	23 (0%)	1	7	4

SMTP HELO overflow attempt	23 (0%)	2	17	2
WEB-MISC apache DOS attempt	21 (0%)	2	16	9
WEB-FRONTPAGE author.exe access	20 (0%)	1	2	2
WEB-ATTACKS python access attempt	19 (0%)	2	2	1
WEB-CGI wrap access	18 (0%)	2	2	6
snort_decoder: TCP Data Offset is less than 5!	18 (0%)	1	7	6
ICMP Echo Reply (Undefined Code!)	18 (0%)	1	9	4
WEB-CGI calendar access	18 (0%)	2	6	3
SHELLCODE x86 stealth NOOP	18 (0%)	2	7	3
WEB-CGI cgiwrap access	18 (0%)	2	2	2
snort_decoder: Tcp Options found with bad lengths	17 (0%)	2	11	3
WEB-MISC search.dll access	17 (0%)	2	2	5
ICMP PING Microsoft Windows	15 (0%)	1	1	9
WEB-CGI htsearch access	15 (0%)	2	2	5
WEB-IIS iisadmpwd attempt	15 (0%)	2	1	3
WEB-PHP Advanced Poll popup.php access	14 (0%)	1	1	6
WEB-IIS _vti_inf access	14 (0%)	2	3	10
spp_stream4: Stealth Activity Detected	14 (0%)	1	6	7
VIRUS OUTBOUND .bat file attachment	14 (0%)	2	7	6
WEB-ATTACKS id command attempt	12 (0%)	2	3	5
spp_bo: Back Orifice Traffic Detected	12 (0%)	1	2	11
ICMP Source Quench	12 (0%)	1	3	3
WEB-MISC bad HTTP/1.1 request, Potentially worm attack	11 (0%)	2	1	11
WEB-CLIENT Outlook EML access	11 (0%)	2	2	6
WEB-CGI campus access	10 (0%)	2	2	4
BAD-TRAFFIC tcp port 0 traffic	10 (0%)	1	9	8
WEB-PHP read_body.php access attempt	10 (0%)	1	1	1
SHELLCODE x86 setgid 0	10 (0%)	2	3	3
DNS named version attempt	9 (0%)	2	3	5
WEB-CGI search.cgi access	9 (0%)	2	2	6
DDOS shaft client to handler	9 (0%)	1	2	2
PORN oral sex	9 (0%)	2	5	2
WEB-MISC Domino names.nsf access	8 (0%)	1	1	3
POLICY SMTP relaying denied	8 (0%)	1	6	1
SNMP public access udp	8 (0%)	1	1	8
ATTACK-RESPONSES id check returned root	8 (0%)	1	5	1
snort_decoder: TCP Data Offset is longer than payload!	8 (0%)	1	1	8
WEB-MISC nc.exe attempt	8 (0%)	2	2	5
WEB-CGI adcycle access	7 (0%)	2	2	4
VIRUS OUTBOUND .dll file attachment	6 (0%)	1	2	2
WEB-IIS anot.htr access	6 (0%)	2	1	3
WEB-MISC /etc/passwd	6 (0%)	2	1	3
WEB-CGI /cgi-bin/ access	6 (0%)	2	2	3
WEB-MISC telnet attempt	6 (0%)	2	2	4

VIRUS OUTBOUND .vbs file attachment	6 (0%)	1	6	1
WEB-MISC /home/ftp access	5 (0%)	1	1	2
WEB-MISC sadmind worm access	5 (0%)	1	1	5
WEB-ATTACKS mail command attempt	5 (0%)	2	2	4
WEB-PHP test.php access	5 (0%)	1	2	2
WEB-MISC perl post attempt	5 (0%)	1	1	1
WEB-CGI finger access	5 (0%)	2	2	4
SMTP Content-Transfer-Encoding overflow attempt	4 (0%)	2	3	4
WEB-IIS fpcount attempt	4 (0%)	2	2	4
WEB-IIS fpcount access	4 (0%)	2	2	4
PORN nude celeb	4 (0%)	2	2	2
WEB-CGI /cgi-bin/ls access	4 (0%)	1	1	4
WEB-MISC intranet access	4 (0%)	1	1	1
INFO FTP Bad login	4 (0%)	1	1	1
MISC MS Terminal Server no encryption session initiation attmept	4 (0%)	1	1	1
WEB-ATTACKS netcat command attempt	4 (0%)	1	1	2
WEB-MISC net attempt	4 (0%)	1	1	3
WEB-CGI download.cgi access	4 (0%)	1	1	2
VIRUS OUTBOUND .ini file attachment	4 (0%)	2	4	3
WEB-CLIENT Microsoft emf metafile access	4 (0%)	2	2	2
WEB-CGI swc access	3 (0%)	1	1	1
ICMP PING BSDtype	3 (0%)	1	3	1
PORN hardcore anal	3 (0%)	1	2	1
WEB-CGI bash access	3 (0%)	1	1	2
WEB-ATTACKS rm command attempt	3 (0%)	2	2	1
WEB-IIS achg.htr access	3 (0%)	2	1	3
ATTACK-RESPONSES Invalid URL	3 (0%)	1	2	1
WEB-MISC guestbook.pl access	3 (0%)	1	1	2
WEB-IIS webhits access	3 (0%)	2	1	3
WEB-MISC /.... access	3 (0%)	2	1	3
WEB-CGI test.bat access	3 (0%)	1	1	3
WEB-IIS ISAPI .idq attempt	3 (0%)	1	1	1
spp_stream4: NULL Stealth Scan	3 (0%)	1	1	1
ATTACK-RESPONSES Microsoft cmd.exe banner	3 (0%)	1	2	2
WEB-PHP php.exe access	3 (0%)	2	1	3
WEB-IIS /iisadmpwd/aexp2.htr access	3 (0%)	2	1	3
PORN nude cheerleader	2 (0%)	1	1	1
WEB-MISC .htpasswd access	2 (0%)	1	1	1
WEB-MISC cybercop scan	2 (0%)	2	2	1
FTP REST overflow attempt	2 (0%)	1	2	1
VIRUS OUTBOUND .hta file attachment	2 (0%)	1	2	1
PORN fuck movies	2 (0%)	2	1	2
PORN alt.binaries.pictures.tinygirls	2 (0%)	1	1	1
FTP command overflow attempt	2 (0%)	1	2	1
WEB-IIS ISAPI .ida access	2 (0%)	1	1	1
WEB-MISC cat%20 access	2 (0%)	1	1	1

WEB-CGI cvsweb.cgi access	2 (0%)	1	1	2
WEB-MISC cross site scripting attempt	1 (0%)	1	1	1
ATTACK-RESPONSES directory listing	1 (0%)	1	1	1
WEB-FRONTPAGE posting	1 (0%)	1	1	1
WEB-MISC handler access	1 (0%)	1	1	1
ATTACK-RESPONSES id check returned userid	1 (0%)	1	1	1
MS-SQL ping attempt	1 (0%)	1	1	1
PORN hot young sex	1 (0%)	1	1	1
WEB-MISC counter.exe access	1 (0%)	1	1	1
WEB-IIS .cnf access	1 (0%)	1	1	1
ICMP L3retriever Ping	1 (0%)	1	1	1
WEB-IIS .asp chunked Transfer-Encoding	1 (0%)	1	1	1
spp_stream4: NMAP Fingerprint Stateful Detection	1 (0%)	1	1	1
FTP large SYST command	1 (0%)	1	1	1
WEB-FRONTPAGE shtml.dll access	1 (0%)	1	1	1
WEB-ATTACKS cc command attempt	1 (0%)	1	1	1
WEB-CGI phf access	1 (0%)	1	1	1
PORN alt.binaries.pictures.erotica	1 (0%)	1	1	1
WEB-MISC Cisco /%% DOS attempt	1 (0%)	1	1	1
TFTP Get	1 (0%)	1	1	1
VIRUS OUTBOUND .sys file attachment	1 (0%)	1	1	1
WEB-ATTACKS kill command attempt	1 (0%)	1	1	1
DDOS TFN Probe	1 (0%)	1	1	1
WEB-CGI dcboard.cgi access	1 (0%)	1	1	1

Tabla 1.6.- Clasificación de los diferentes tipos de alertas obtenidas.

Actualmente existen más de 2300 firmas de ataques. Desde la consola ACID se puede acceder a la base de datos de firmas de Snort para obtener información sobre las causas que han provocado una determinada alerta. Vamos a analizar brevemente las alertas que aparecen con más frecuencia:

- "ICMP PING NMAP". Es la alerta que se obtiene con más frecuencia. Indica que probablemente se está produciendo un escaneo de puertos con la herramienta Nmap. No obstante, muchas de estas alertas son falsos positivos. Hay que tener en cuenta que el IDS enviará esta alerta si detecta el "ping" típico de Nmap que se caracteriza porque el tamaño de su campo de datos es cero, y pueden existir otros programas o sistemas que generen este mismo tipo de "ping".

- "P2P GNUTella GET". Esta alerta indica que existe actividad de un cliente "Pear-to-Pear" (P2P). Éste es el protocolo que se utiliza por aplicaciones de descarga de ficheros compartidos en red, algunos "chats", etc. En definitiva, alerta de que se están

produciendo ciertas actividades que pueden estar prohibidas por la política de empresa.

- "SCAN Squid Proxy attempt". Indica que se está produciendo un escaneo en la red.

- "SHELLCODE x86 NOOP". Se obtiene esta alerta cuando se detectan instrucciones NOP para la arquitectura Intel x86.

- ICMP Destination Unreachable (Undefined Code!). Esta alerta se produce cuando un paquete ICMP que viaja por la red no encuentra el destino. En algunos casos, puede significar que se está produciendo un ataque por denegación de servicio. Como se ha visto en el apartado 4.1, esta regla fue eliminada debido a que se producen numerosas alertas de este tipo y normalmente es sólo mera información sobre el paquete ICMP.

- "MS-SQL Worm propagation attempt". Alerta de que el gusano "Slammer" intenta comprometer un servidor Microsoft SQL aprovechando una vulnerabilidad de este último. Esta alerta indica que se está produciendo dicha tentativa de ataque, pero nunca se llevará a cabo con éxito puesto que no existe este tipo de servidor en nuestra red.

- "WEB-IIS cmd.exe access". Indica que el atacante puede estar intentando obtener acceso "root" utilizando una vulnerabilidad en un sistema con el servicio Microsoft IIS. Es muy común este tipo de alerta y es de vital importancia tener parcheadas las vulnerabilidades en dicho servidor.

Podemos agrupar las alertas obtenidas en categorías y volver a realizar un esquema de probabilidad que mostraría más intuitivamente aquellos tipos de ataques preferidos por los piratas informáticos. La figura 6.1 muestra los tipos de ataques más frecuentes.

Los ataques más numerosos están clasificados bajo el nombre "attempted-recon" que hace referencia a los ataques perpetrados con el objetivo de obtener diferente tipo de información del sistema. Aquí se incluiría todo tipo de escaneo. "Policy-violation" es el segundo tipo de ataque más frecuente y recoge aquellas actividades que pueden estar prohibidas por una determinada política de empresa como las descargas de Internet o el chateo. A continuación el grupo "shellcode-detect" recoge todas las alertas emitidas debido a la existencia de código ejecutable en algún paquete. También puede observarse un número importante de alertas debidas a ataques basados en aplicaciones web. Un tipo de ataque que presenta bastante importancia es el clasificado como "attempted-admin" pues representa los ataques llevados a cabo con la intención de ganar los privilegios del administrador. "suspicious-filename-detect" indica la detección de un fichero sospechoso, es decir, de virus

en general y, "misc-attack" y "misc-activity" indican la presencia de diferentes actividades que realizan un uso malicioso de los protocolos de red.

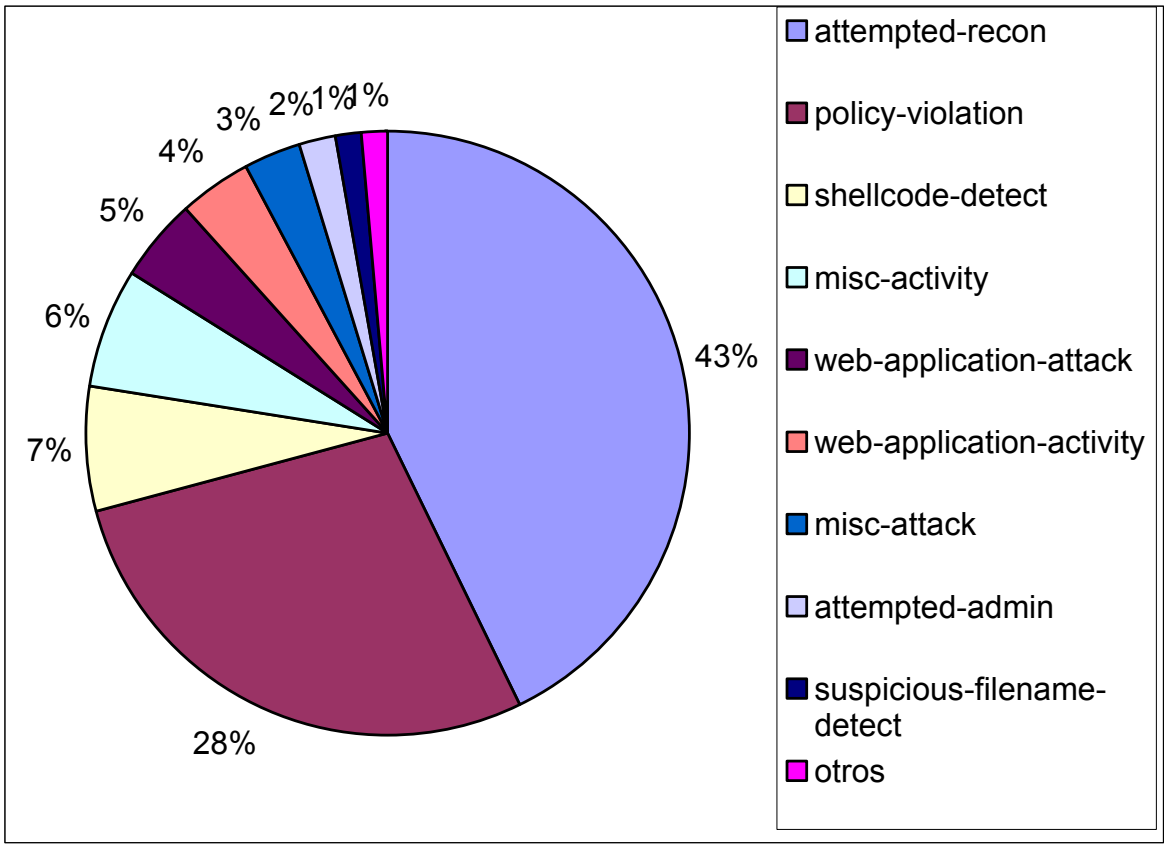


Figura 6.1.- Tipos de ataque más frecuentes.

En la tabla 6.2 se muestra los puertos preferidos para llevar a cabo un ataque según esta experiencia.

Puerto	**Servicio**	**% de alertas**
9935/tcp	no asignado	29,44%
3128/tcp	squid, Squid Proxy Server	27,79%
80/tcp	www, HTTP	25,07%
1434/tcp	MS-SQL, Microsoft-SQL	5,24%
25/tcp	SMTP, Simple Mail Transfer	2,88%
1080/tcp	SOCKS	1,77%
443/tcp	HTTPS	1,35%
8080/tcp	www, HTTP	1,28%
1020/tcp	no asignado	0,86%
110/tcp	POP3	0,73%
1019/tcp	no asignado	0,60%
1016/tcp	no asignado	0,52%
1017/tcp	no asignado	0,51%

Tabla 6.2.- Puertos más atacados

En la tabla de arriba se observa que los puertos destinos que aparecen en un mayor número de alertas son los tcp 9935 y 3128. La mayoría de las alertas relacionadas con estos dos puertos provienen de la red interna de la empresa y se producen debido a determinadas operaciones de los administradores de la red. En el punto 4.1.1 se ha visto como se realizaron reglas para filtrar este tipo de tráfico. Por tanto, podemos concluir que la mayoría de los ataques se intentan llevar a cabo accediendo a los servicios web, seguido de las aplicaciones SQL y de la utilización del protocolo de correo SMTP.

7 AMPLIACIONES FUTURAS

Cuando se ha planteado la puesta en marcha de un IDS en una arquitectura de red, ha sido prioritaria la instalación de una sonda IDS fuera del perímetro del *firewall* para conseguir una detección anticipada, y también, la colocación de una sonda en la parte DMZ de la red (que contiene todos los servidores) para detectar los ataques que han conseguido atravesar el *firewall* y que son críticos en este segmento tan importante de la red. Con la solución que se ha implementado a lo largo de este proyecto, seriamos capaces de detectar las actividades maliciosas en las dos partes de la red que se viene de comentar. Sin embargo, quedaría aún mucho que hacer en cuanto a la mejora de la detección de incidencias en la red. Desde el principio, la idea de instalar en los equipos los componentes de Prelude-IDS tenía como objetivo dejar el sistema IDS abierto a posibles ampliaciones futuras puesto que permite fácilmente la integración de nuevas aplicaciones. Algunas ampliaciones del sistema IDS existente pueden ser:

- Aumento del número de sondas NIDS. Una vez que tenemos ya instalado un sistema IDS con dos sondas y un servidor y sabemos como gestionar la comunicación entre ellos, añadir sondas en otros segmentos de la arquitectura de la red e integrarlas en el sistema IDS existente no resultaría complicado. Ampliar el número de sondas sólo implicaría un problema de recursos en el sentido de que harían falta más máquinas. Podría considerarse la colocación de un nuevo sensor sobre la LAN interna y otro en la parte de Intranet.

- Refuerzo del IDS con una parte HIDS. Un sistema de detección de intrusiones consta tanto de una parte NIDS como de otra HIDS. Sería interesante instalar sondas HIDS en todos los servidores críticos. Esto permitirá saber si un intruso ha conseguido entrar en un sistema y las operaciones que está realizando. Hay dos tipos de HIDS, los analizadores de integridad de ficheros (detectan si ha habido modificaciones en el sistema de ficheros) y los analizadores de ficheros de *logs* (analizan los ficheros de *logs* buscando determinadas firmas). Prelude-IDS permitirá añadir fácilmente estos dos tipos de sondas. Prelude-lml es una sonda donde se pueden configurar los ficheros de *logs* que son interesantes de vigilar y Prelude-FIC será capaz de realizar un análisis de la integridad del sistema de ficheros. Por supuesto, Prelude-IDS puede integrar otros tipos de sensores HIDS (no propios a Prelude) como Samhain.

- Explotación de recursos existentes. Existen ciertos dispositivos en la red que cuentan con una parte IDS. Concretamente, se podría configurar la parte IDS disponible en la máquina *Linkbranch* (figura 2.1).

- Instalación de otros tipos de sonda. Como se ha repetido varias veces, Prelude-IDS permite integrar otras sondas con diferentes funcionalidades. Se podría plantear la utilización de otras aplicaciones relacionadas con la seguridad informática como Nessus (escáner de vulnerabilidades, es decir, buscan si existen vulnerabilidades en las aplicaciones que utilizan los sistemas en una red) o Libsafe (sonda que reacciona de manera activa contra ataques de desbordamiento de buffer). Existen muchas más aplicaciones. Podemos echar un vistazo en http://prelude-ids.org.

8 CONCLUSIONES

De la realización de este proyecto se puede llegar a las siguientes conclusiones:

- Se ha instalado una solución NIDS que funciona correctamente en cuanto a que es capaz de analizar todo el tráfico que circula por los segmentos estratégicos de la infraestructura de red de la empresa proporcionando información valiosa sobre los incidentes que se han producido contra la seguridad de los sistemas informáticos en general. No obstante, aplicaciones IDS como Snort y Prelude-IDS todavía presentan ciertas carencias y su comportamiento no es del todo ideal ante cierto tipo de ataques como los "stealth" (véase los apartados 6 y 1.7.1).

- Por un lado, se ha facilitado la tarea del administrador gracias a la puesta en marcha de un programa que envía informes diarios, semanales y mensuales de las alertas que se han producido. De otro lado, el IDS requerirá un trabajo importante de mantenimiento y actualización, puesto que hemos sacrificado la posibilidad de realizar automáticamente la mayoría de estas tareas a costa de hacer inaccesible los equipos IDS desde el exterior para dotarlos de un mayor nivel de seguridad.

- Es imprescindible la actualización de la base de datos de firmas de ataque o reglas que utiliza el IDS con una frecuente considerable, si no, el sistema podría quedar inservible.

- Los resultados obtenidos por los IDS ayudarán a tomar nuevas medidas como la realización de filtros en el *firewall* para evitar el tráfico considerado como sospechoso o la actualización o el cambio de los sistemas o aplicaciones que resulten ser un objetivo potencial de ataque.

- Se ha optado por una solución flexible que permitirá seguir aumentando la seguridad de la red y los sistemas gracias a la posibilidad de centralizar en el mismo servidor con el que ya contamos incidencias detectadas por otros dispositivos de seguridad (HIDS, verificadores de integridad de ficheros y vulnerabilidades, etc.).

- El campo de la seguridad informática es muy amplio y se debe seguir estudiando y trabajando en la consecución de la misma. Aunque un IDS ayudé en la detección de ataques y facilité gran cantidad de datos para llegar hasta el intruso, no significa que dote a la red de un nivel de seguridad absoluto. La seguridad informática requiere un alto nivel de dedicación que implicaría como tareas imprescindibles la optimización constante de los filtros de tráfico en el *firewall*, la optimización del IDS y la

actualización frecuente de las aplicaciones con el objetivo de parchear sus posibles vulnerabilidades.

9 ANEXOS

9.1 ANEXO I: MANUAL DE INSTALACIÓN

9.1.1 INSTALACIÓN DEL SISTEMA OPERATIVO. REDHAT 8.0

La instalación de RedHat no presenta ningún problema siguiendo los pasos de la herramienta gráfica de instalación de RedHat 8.0. No obstante, durante la etapa de instalación, y debido al tipo de aplicación que vamos a instalar, es conveniente seguir estos consejos:

1. Elegir inglés como idioma para el sistema operativo. Esto evitará problemas asociados a la compatibilidad con las aplicaciones que se utilizaran, puesto que están escritas para trabajar en inglés.

2. Particionamiento con Disk Druid. Se realizarán las particiones siguiendo estos criterios:

    a) /boot: 100 M.

    b) <Swap>: 1024 M.

    c) /usr: aproximadamente el 50% de la capacidad con un mínimo de 3072 M. Es importante ya que esta carpeta contendrá la mayoría de las aplicaciones.

    d) /var: aproximadamente un 30 %, mínimo 2048 M. Es importante, ya que una aplicación IDS puede generar muchos ficheros de *logs* dependiendo de como esté configurada.

    e) /: aproximadamente un 15%.

    f) /temp: más o menos un 5%.

3. No instalar paquetes como editores o herramientas de programación.

9.1.2 MEDIDAS DE SEGURIDAD PREVIAS A LA INSTALACIÓN DE LA APLICACIÓN

Antes de comenzar con la instalación del sistema de detección de intrusiones, realizaremos una serie de tareas encaminadas a conseguir que el sistema sea más seguro.

9.1.2.1 Actualización y corrección de vulnerabilidades en el sistema operativo RedHat 8.0

Por seguridad, es necesario mantener actualizado el sistema. Para ello, es necesario bajarse de la red los archivos que encontremos en http://rhn.redhat.com/errata/rh8-errata.html. Guardamos estos archivos en una carpeta. Dentro de este directorio ejecutamos:

```
# rpm –Fvh *
```

9.1.2.2 Eliminación de servicios inútiles

Se recomienda parar ciertos procesos que no se utilizaran:

```
# /etc/rc.d/init.d/apmd stop
# /etc/rc.d/init.d/portmap stop
# /etc/rc.d/init.d/nfslock stop
```

Después desinstalamos servicios que no serán necesarios para este sistema con el comando `rpm –e –nodeps` como: apmd, dosfstools, eject, hotplug, ipchains, ksymoops, kudzu, lokkit, mailcap, pcutils, redhat-logos, redhat-release, setserial, procmail, cups, firstboot, isdn, netfs, pcmcia y sgi_fam.

9.1.2.3 Eliminación de usuarios y grupos de usuarios inútiles

Eliminamos los usuarios y grupos de usuarios inútiles con los comandos *userdel* y *groupdel*. Por ejemplo, usuarios inútiles serían: adm, lp, shutdown, halt, mail, uucp, operator, games, gopher, ftp, rpc, vcsa, nscd, mailnull, rpcuser, nfsnobody y pcap. Como grupos inútiles: adm, lp, news, mail, uucp, games y dip. Podemos verificar que el fichero */etc/passwd* contiene usuarios importantes como root, bin, daemon, sync y nobody.

9.1.2.4 Creación de usuarios con SUDO

Debemos crear los usuarios que tendrán acceso al servidor y cada uno con los permisos apropiados. Esto lo vamos a hacer utilizando SUDO. Para ello hay que editar el fichero de configuración utilizado *visudo*. Por tanto:

```
# visudo
```

En este caso, crearemos un grupo de administradores que tendrán todos los permisos excepto el de modificar la contraseña root. Definiremos los usuarios pertenecientes a este grupo. El archivo de configuración de sudo quedaría de la siguiente manera:

```
# User alias specification
User_Alias ADMINISTRATEURS = oaracine, bmoubreaux, fcducamp,
gddionneau, veletin, herlegal

# Cmnd alias specification
Cmnd_Alias ROOTMOD=/usr/bin/passwd root, /usr/sbin/userdel
root, /usr/sbin/userdel -r root, /usr/sbin/adduser root

# Defaults specification

# User privilege specification
root    ALL=(ALL) ALL
ADMINISTRATEURS         ALL= ALL, !ROOTMOD
```

9.1.2.5 Otras medidas de seguridad

- Proteger ficheros importantes. Es importante que los ficheros que contienen usuarios y claves estén bien asegurados. Podemos, por tanto, protegerlos para que no puedan ser modificados:

  ```
  # chattr +i /etc/passwd
  # chattr +i /etc/shadow
  # chattr +i /etc/group
  # chattr +i /etc/gshadow
  ```

  Hay que tener en cuenta que cada vez que deseemos creer un usuario habría que realizar la operación inversa sobre los ficheros anteriores con el comando `chattr -i`.

- Asegurar SSH. Podemos impedir que se acceda a nuestro servidor como usuario root vía SSH y que no se pueda acceder con contraseñas vacías. Para ello, editamos el fichero */etc/ssh/sshd_config*, de manera que aparezcan las siguientes líneas:

```
Protocol 2
PermitRootLogin no
PermitEmptyPasswords no
```

9.1.3 Instalación del IDS

Snort presenta muchas ventajas, como ya se ha comentado a lo largo del desarrollo de esta memoria, pero también Prelude-IDS presenta funcionalidades de las que carece Snort y que podrían ser interesantes. Por este motivo se va a explicar el proceso de instalación de ambos IDS. Se verá la instalación de la parte del servidor IDS y el sensor como un todo, es decir, se instalarán ambos sobre el mismo procesador. Instalar solamente un sensor o cliente es mucho más sencillo y se comentará más adelante.

9.1.3.1 Descarga de aplicaciones

Antes de comenzar con la instalación, es recomendable bajarse de la red todos los archivos necesarios. Seguiremos los siguientes pasos:

1. Crear un directorio donde guardar todos los archivos de instalación. Yo lo crearé bajo la carpeta */root* y lo llamaré *IDSinstall*:

```
# mkdir /root/IDSinstall
```

2. Los archivos comunes para la instalación de ambos IDS serían los siguientes:

   - MySQL 4.0.18: http://mysql.crihan.fr/Downloads/MySQL-4.0/mysql-4.0.18.tar.gz
   - Apache 1.3.29: http://apache.crihan.fr/dist/httpd/apache_1.3.29.tar.gz
   - mod_ssl-2.8.16-1.3.29: http://www.modssl.org/source/mod_ssl-2.8.16-1.3.29.tar.gz
   - Libpcap 0.7.2 : http://www.tcpdump.org/release/libpcap-0.7.2.tar.gz

3. Si queremos instalar sólo el IDS Snort deberemos descargar los archivos que se indican a continuación. Las URL que aparecen pueden facilitar esta tarea:

    - Snort 2.0.5: http://www.snort.org/dl/snort-2.0.5.tar.gz
    - PHP 4.3.4: http://www.php.net/distributions/php.4.3.4.tar.gz
    - ADODB 3.90 : http://phplens.com/lens/dl/adodb390.tgz
    - ACID 0.9.6b23 : http://acidlab.sourceforge.net/acid-0.9.6b23.tar.gz
    - Zlib 1.1.4 : http://www.gzip.org/zlib
    - JPGraph 1.13 : http://www.aditus.nu/jpgraph/downloads/jpgraph-1.13.tar.gz
    - Barnyard 0.1.0: http://www.snort.org/dl/barnyard/barnyard-0.1.0.tar.gz

4. Si además, queremos instalar Prelude-IDS harían falta los siguientes archivos:

    - Libprelude 0.8.10: http://www.prelude-ids.org/download/releases/libprelude-0.8.10.tar.gz
    - Prelude-manager 0.8.10: http://www.prelude-ids.org/download/releases/prelude-manager-0.8.10.tar.gz
    - Prelude-nids 0.8.6: http://www.prelude-ids.org/download/releases/prelude-nids-0.8.6.tar.gz
    - Prelude-lml 0.8.6: http://www.prelude-ids.org/download/releases/prelude-lml-0.8.6.tar.gz
    - Mod_perl 1.29: http://www.online-mirror.org/apache/perl/mod_perl-1.29.tar.gz
    - Piwi 0-8-latest: http://www.prelude-ids.org/download/snapshots/0-8/piwi-0-8-latest.tar.gz

5. Si queremos utilizar Prelude-IDS como manager en una estructura distribuida, pero utilizar como sensor Snort, haría falta descargar el siguiente archivo:

- Prelude-Snort-reporting-patch: http://www.prelude-ids.org/download/releases/prelude-snort-reporting-patch-0.2.6.tar.gz

6. Si queremos utilizar una estructura distribuida sin utilizar Prelude-manager como gestor de la centralización (es decir, Snort solamente) debemos también descargar:

   - Stunnel-4.05: http://www.stunnel.org/download/stunnel/src/stunnel-4.05.tar.gz

7. Para otras aplicaciones como la de notificación de alertas vía mail (para sensores Snort):

   - Mailsend: http://www.muquit.com/muquit/software/mailsend/mailsend_src_1.05.zip
   - Snort-rep : http://people.ee.ethz.ch/~dws/software/snort-rep/pub/snort-rep-1.10.tar.gz

9.1.4 Instalación de aplicaciones comunes

Básicamente, los elementos comunes a ambas instalaciones (Snort y Prelude-IDS) son la librería *libpcap*, la base de datos y el servidor web. Muchas aplicaciones (Prelude, Apache y Stunnel) utilizan la librería OpenSSL. Esta librería se encuentra entre los paquetes que se instalan por defecto con la distribución de la RedHat 8.0 y será la que utilicemos.

9.1.4.1 Instalación de libpcap

Libpcap es la librería de captura de paquetes que utiliza tanto Snort como Prelude-IDS. Para instalarla, partiendo desde el directorio de descargas (*/root/IDSinstall*):

```
# tar -xvzf libpcap-0.8.1.tar.gz
# cd libpcap-0.8.1
# ./configure
# make
# make install
# cd ..
```

9.1.4.2 Instalación de MySQL

Para realizar la instalación y configuración de la base de datos llevaremos a cabo los siguientes pasos:

1. Crear el usuario y grupo para MySQL:

```
# groupadd mysql
# useradd -g mysql mysql
```

2. Editar en el archivo */root/.bash_profile* la siguiente línea:

```
PATH=$PATH:$HOME/bin:/usr/local/mysql/bin
```

3. Desde el directorio donde se han descargado todos los archivos de instalación, ejecutamos los siguientes comandos:

```
# tar -xvzf mysql-4.0.18.tar.gz
# cd mysql-4.0.18
# ./configure --prefix=/usr/local/mysql
# make
# make install

# scripts/mysql_install_db

# chown -R root /usr/local/mysql
# chown -R mysql /usr/local/mysql/var
# chgrp -R mysql /urs/local/mysql
# cp support-files/my-medium.cnf  /etc/my.cnf
```

4. Añadimos las líneas "/usr/local/mysql/lib/mysql" y "/usr/local/lib" al fichero */etc/ld.so.conf*. Una vez modificado, ejecutamos el comando:

```
# ldconfig -v
```

5. Para comprobar que la base de datos funciona, ejecutar:

```
# /usr/local/mysql/bin/mysqld_safe --user=mysql &
```

Se deberá pulsar *intro* para volver al *prompt*. Si no aparece ningún mensaje de error, podemos comprobar que el proceso está activo utilizando el comando:

```
# ps –ef | grep mysql
```

6. Si MySQL funciona, podemos querer que se ejecute automáticamente cada vez que arranquemos el sistema. Para ello:

```
# cp support-files/mysql.server  /etc/init.d/mysql
# cd /etc/rc3.d
# ln –s ../init.d/mysql  S85mysql
# ln –s ../init.d/mysql  K85mysql
# cd /etc/rc5.d
# ln –s ../init.d/mysql  S85mysql
# ln –s ../init.d/mysql  K85mysql
# cd ../init.d
# chmod 755 mysql
```

9.1.4.3 Instalación de APACHE

Apache será utilizado como servidor web tanto por la consola de presentación de alertas de Snort (ACID) como por la de Prelude-IDS (PIWI). ACID está escrito en PHP y PIWI en Perl, por lo que necesitaremos que Apache interprete bien estos dos lenguajes. Además queremos que también trabaje en modo seguro (https) utilizando el módulo_ssl.

Vamos a ver los pasos que hay que seguir para lograr que el servidor Apache trabaje con los tres módulos comentados anteriormente:

1. Preparación de Apache. Desde el directorio de descargas :

```
# tar –xvzf  apache-1.3.29.tar.gz
# cd apache-1.3.29
# ./configure –prefix=/usr/local/apache
# cd ..
```

2. Instalamos el módulo perl ejecutando los siguientes comandos:

```
# tar –xvzf mod_perl-1.29.tar.gz
# cd mod_perl-1.29
#   perl   Makefile.PL   APACHE_SRC=../apache-1.3.29/      DO_HTTP=1
USE_APACI=1 EVERYTHING=1
# make
```

```
# make test
# make install
# cd ..
```

3. Instalamos el módulo ssl:

```
# tar -xvzf mod_ssl-2.8.16-1.3.29.tar.gz
# cd mod_ssl-2.8.16-1.3.29
# ./configure --with-apache=../apache-1.3.29
# cd ..
```

4. Instalación de Apache:

```
# cd apache-1.3.29
# ./configure --prefix=/usr/local/apache --enable-module=ssl --
enable-shared=ssl --enable-module=so --activate-
module=src/modules/perl/libperl.a --enabled-shared=perl
# make
# make certificate TYPE=custom
```

Esta última línea, va a generar la clave privada que utilizará la aplicación Apache. Deberemos contestar a ciertas preguntas y recordar la contraseña que debemos introducir, puesto que se pedirá cada vez que se vaya a iniciar el servidor Apache. Continuamos con:

```
# make install
# cd ..
```

5. Instalación de PHP:

```
# tar -xvzf php-4.3.4.tar.gz
# cd php-4.3.4.tar.gz
#./configure --prefix=/usr/local/apache/php --with-
apxs=/usr/local/apache/bin/apxs --with-config-file-
path=/usr/local/apache/php --enable-sockets -with-
mysql=/usr/local/mysql --with-zlib-dir=/usr/local --with-gd
# make
# make install
# cp php.ini-dist  /usr/local/apache/php/php.ini
```

6. Probar si Apache funciona. Para iniciar la aplicación ejecutar el siguiente comando:

```
# /usr/local/apache/bin/apachectl startssl
```

Esta orden, antes de iniciar el servicio *https*, nos pedirá la contraseña para acceder a la cable privada que habíamos definido anteriormente. De este modo podemos utilizar el servidor web tanto en modo normal, por el puerto 80, como en modo seguro (puerto 443). Utilizando un navegador web comprobamos si la aplicación funciona (https://IPdelhost o bien http://IPdelhost). Para pararla:

```
# /usr/local/apache/bin/apachectl stop
```

7. Configuración de Apache. Debemos realizar ciertas modificaciones en el fichero de configuración de Apache. Para ello editamos el fichero */usr/local/apache/conf/httpd.conf* de manera que aparezcan las siguientes líneas:

```
# Example:
# LoadModule foo_module libexec/mod_foo.so
LoadModule perl_module        libexec/libperl.so

<Directory "/usr/local/apache/htdocs/piwi/">
     Options ExecCGI
     AddHandler cgi-script .pl
</Directory>
```

Un poco más abajo, debemos encontrar *DirectoryIndex*, donde introduciremos *index.php* e *index.perl* de manera que tengamos algo así:

```
# DirectoryIndex: Name of the file or files to use as a
pre-written HTML
# directory index.  Separate multiple entries with
spaces.
#
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.html.var
index.pl
</IfModule>
```

Por último, también debe aparecer en el fichero de configuración:

```
# AddType allows you to tweak mime.types without
#actually editing it, or to
# make certain files to be certain types.
#
AddType application/x-tar .tgz
AddType image/x-icon .ico
AddType application/x-httpd-php .php
```

8. Hacemos seguro el directorio de Apache a través del servidor web. Para ello:

```
# mkdir /usr/local/apache/passwords
# /usr/local/apache/bin/htpasswd -c
/usr/local/apache/passwords/apache apache
```

En la expresión anterior hemos llamado apache al usuario que tendrá acceso a cualquiera de las aplicaciones que se ejecuten bajo Apache y nos preguntará la contraseña para acceder. Para que la autentificación funcione correctamente hay que introducir en el archivo *httpd.conf* (ubicado en */usr/local/apache/conf*) las siguientes líneas:

```
<Directory "/usr/local/apache/htdocs">
      AuthType Basic
      AuthName "Apache"
      AuthUserFile /usr/local/apache/passwords/apache
      Require user apache
      SSLRequireSSL
</Directory>
```

En este caso, el usuario que se requiere para acceder es apache. La expresión "SSLRequireSSL" indica que sólo podemos acceder de manera segura utilizando *https*.

Para comprobar que la autentificación funciona podemos reiniciar el servicio http (https) y volver a acceder mediante el navegador a la dirección de nuestro host.

9. Comprobemos si PHP funciona. Para ello, vamos a crear un fichero al que llamaremos *test.php* con la siguiente línea como contenido: `<?php phpinfo(); ?>`. Colocaremos dicho archivo en el directorio */usr/local/apache/htdocs*/. Iniciaremos Apache (`# /usr/local/apache/bin/apachectl startssl`) y mediante un navegador web comprobaremos que metiendo el URL https://IPdelhost/test.php aparece una pantalla en modo gráfico de información sobre PHP.

10. Esperaremos a instalar la consola en Perl para comprobar si el módulo Perl funciona con Apache.

9.1.5 Instalación del NIDS SNORT

En este apartado se seguirán los pasos para instalar y configurar de forma básica un servidor IDS Snort.

9.1.5.1 Instalación de zlib

Zlib es una librería de compresión necesaria para la consola de Snort, ACID. Para su instalación, partiendo del directorio de descargas:

```
# tar –xvzf zlib-1.2.1.tar.gz
# cd zlib-1.2.1
# ./configure
# make
# make test
# make install
# cd ..
```

9.1.5.2 Instalación de Snort

A continuación se van a explicar los pasos a seguir para conseguir la instalación de la sonda Snort. Esta sonda puede configurarse para que utilice diferentes módulos de salida. Cabe la posibilidad de utilizar la sonda Snort como un sensor dentro de una arquitectura distribuida Prelude. Vamos a considerar aquí esa posibilidad puesto que, si después no queremos hacer uso de este módulo, sólo habría que deshabilitarlo en el fichero de configuración de Snort, snort.conf. Seguimos los siguientes pasos:

1. Crear el usuario y grupo de Snort:

```
# groupadd snort
# useradd –g snort snort
```

2. Creamos un directorio dónde guardar las reglas y ficheros configurables de Snort y otro dónde se guardarán los ficheros de *logs*.

```
# mkdir /etc/snort
# mkdir /var/log/snort
```

3. Seguimos los siguientes pasos para descomprimir los archivos necesarios y proceder a su instalación:

```
# tar –xvzf snort-2.0.5.tar.gz
# tar –xvzf snort-prelude-reporting-patch-0.2.6.tar.gz
# cd snort-2.0.5
# patch –p0 < /root/snortinstall/snort-2.0.x-prelude.diff
# sh ./autogen.sh
# ./configure --with-mysql=/usr/local/mysql
# make
# make install
```

4. Instalación de reglas y ficheros de configuración:

```
# cd rules
# cp * /etc/snort
# cd ../etc
# cp snort.conf /etc/snort
# cp * /etc/snort
```

5. Modificaciones en el fichero de configuración /etc/snort/snort.conf:

    a) var RULE_PATH /etc/snort/

    b) De momento, insertar un caracter '#' para comentar la línea:

    ```
    #output alert_prelude: async,
    classification_file=/etc/snort/prelude-
    classification.config
    ```

    Esto desactivará el módulo de salida Prelude, que no podemos utilizar puesto que no se ha procedido a la instalación del IDS Prelude. Más adelante se explicará como utilizarlo.

    c) Descomentar la línea de "output database" de manera que quede de la siguiente manera:

```
output database: log, mysql, user=snort
password=contraseña dbname=snort host=localhost
```

6. Establecer permisos para los usuarios *root* y *snort* en la base de datos MySQL y crear la base de datos que utilizará Snort (snort). También debemos crear otra base de datos que podrá utilizar también ACID para realizar diferentes operaciones (snortarchive).

```
# /usr/local/mysql/bin/mysql

mysql> SET PASSWORD FOR root@localhost=PASSWORD('contraseña');

mysql> create database snort;

mysql> create database snortarchive;

mysql>grant INSERT,SELECT on root.* to snort@localhost;

mysql>SET PASSWORD FOR snort@localhost=PASSWORD('contraseña');

mysql>grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to
snort@localhost;

mysql>grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to
snort;

mysql>grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to
snortarchive;

mysql>exit
```

Desde el directorio fuente de Snort ejecutamos los siguientes comandos:

```
# /usr/local/mysql/bin –u root –p < ./contrib/create_mysql snort
Enter password:

# /usr/local/mysql/bin –u root –p < ./contrib/create_mysql
snortarchive
Enter password:
```

Para instalar también las tablas extras en la base de datos Snort:

```
# cd contrib
# zcat snortdb-extra.gz | /usr/local/mysql/bin/mysql -p snort
Enter password:

# zcat snortdb-extra.gz | /usr/local/mysql/bin/mysql -p snortarchive
Enter password:
```

7. Comprobamos si la base de datos *snort* se ha instalado correctamente:

```
# /usr/local/mysql/bin/mysql -p
>Enter password:
mysql> SHOW DATABASES;
```

Debe aparecer algo como esto:

```
+--------------+
| Database     |
+--------------+
| mysql
| snort
| snortarchive
| test
+--------------+

mysql>connect snort;

mysql>SHOW TABLES;
```

Debe aparecer:

```
+------------------+
| Tables_in_snort |
+------------------+
| data
| detail
| encoding
| event
| flags
| icmphdr
| iphdr
| opt
```

```
| protocols
| reference
| reference_system
| schema
| sensor
| services
| sig_class
| sig_reference
| signature
| tcphdr
| udphdr
+------------------+
```

8. Si queremos que Snort se inicie automáticamente cuanto arranquemos el servidor entonces debemos realizar lo siguiente (partiendo del directorio fuente de Snort):

   ```
   # cp contrib/S99snort  /etc/init.d/snort
   ```

   en este fichero cambiamos las siguientes líneas para que queden así:

   ```
   CONFIG=/etc/snort/snort.conf
   SNORT_GID=snort
   ```

   y continuamos con:

   ```
   # cd /etc/init.d
   # chmod 755 snort
   # cd /etc/rc3.d
   # ln -s ../init.d/snort  S99snort
   # ln -s ../init.d/snort  K99snort
   # cd /etc/rc5.d
   # ln -s ../init.d/snort  S99snort
   # ln -s ../init.d/snort  K99snort
   ```

9. Probablemente las reglas que vienen junto con la versión de Snort no estén actualizadas. Convendrá actualizarlas frecuentemente. Se pueden encontrar los ficheros de reglas actualizados en la dirección "http://www.snort.org/dl/rules/". Sólo habría que descargar el archivo comprimido, descomprimirlo y guardar las nuevas reglas donde antes hemos situado las otras, es decir, en */etc/snort/*.

9.1.5.3 Instalación de JPGraph

Para instalar esta librería gráfica utilizada por ACID partimos del directorio donde hemos descargado todos los archivos:

```
# cp jpgraph-1.14.tar.gz  /usr/local/apache/htdocs
# cd /usr/local/apache/htdocs
# tar –xvzf jpgraph-1.14.tar.gz
# rm –rf jpgraph-1.14.tar.gz
# cd jpgraph-1.14
# rm –rf README
# rm –rf QPL.txt
```

9.1.5.4 Instalación de ADODB

Desde el directorio de descarga empezamos ejecutando los siguientes comandos:

```
# cp adodb420.tgz  /usr/local/apache/htdocs
# cd /usr/local/apache/htdocs
# tar –xvzf adodb420.tgz
# rm –rf adodb420.tgz
```

9.1.5.5 Instalación y configuración de ACID

Para la instalación y configuración de ACID seguimos los siguientes puntos:

1. Instalación partiendo del directorio de descarga:

```
# cp acid-0.9.6b23.tar.gz  /usr/local/apache/htdocs
# cd /usr/local/apache/htdocs
# tar –xvzf acid-0.9.6b23.tar.gz
# rm –rf acid-0.9.6b23.tar.gz
```

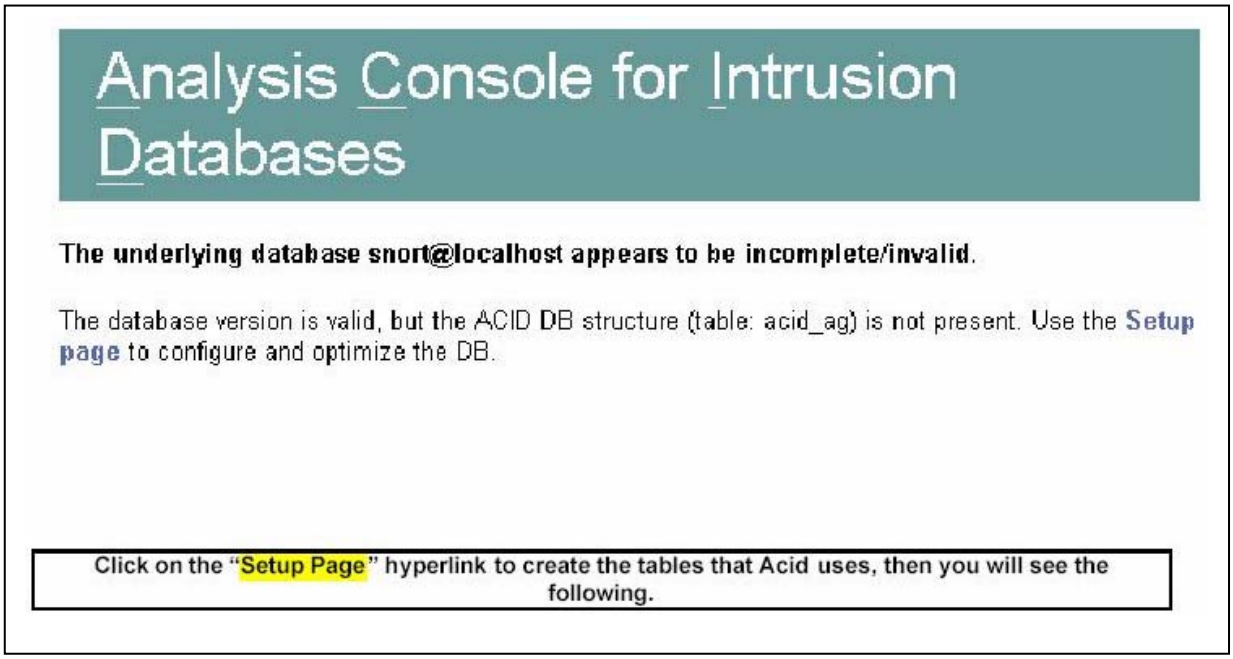
2. Ir hasta el directorio */usr/local/apache/htdocs/acid* y editamos el fichero de configuración *acid_conf.php*. Deberá parecerse a esto:

```
$DBlib_path = "/usr/local/apache/htdocs/adodb";

/* The type of underlying alert database
 *
 *  MySQL       : "mysql"
 *  PostgresSQL : "postgres"
 *  MS SQL Server : "mssql"
 */
$DBtype = "mysql";

/* 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
 *   - $alert_password : password of the DB user
 *
 *  This information can be gleaned from the Snort
database
 *  output plugin configuration.
*/
$alert_dbname   = "snort";
$alert_host     = "localhost";
$alert_port     = "";
$alert_user     = "snort";
$alert_password = "contraseña";

/* Archive DB connection parameters */
$archive_dbname   = "snortarchive";
$archive_host     = "localhost";
$archive_port     = "";
$archive_user     = "snort";
$archive_password = "contraseña";
```

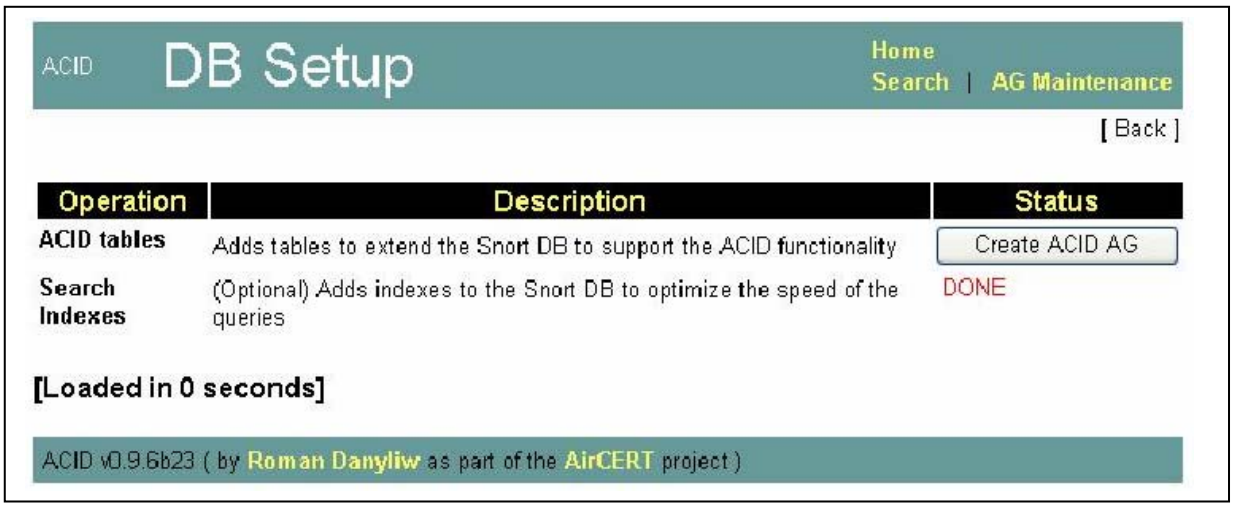
Un poco más abajo:

```
$ChartLib_path = "/usr/local/apache/htdocs/jpgraph-
1.14/src";

/* File format of charts ('png', 'jpeg', 'gif') */
$chart_file_format = "png";
```

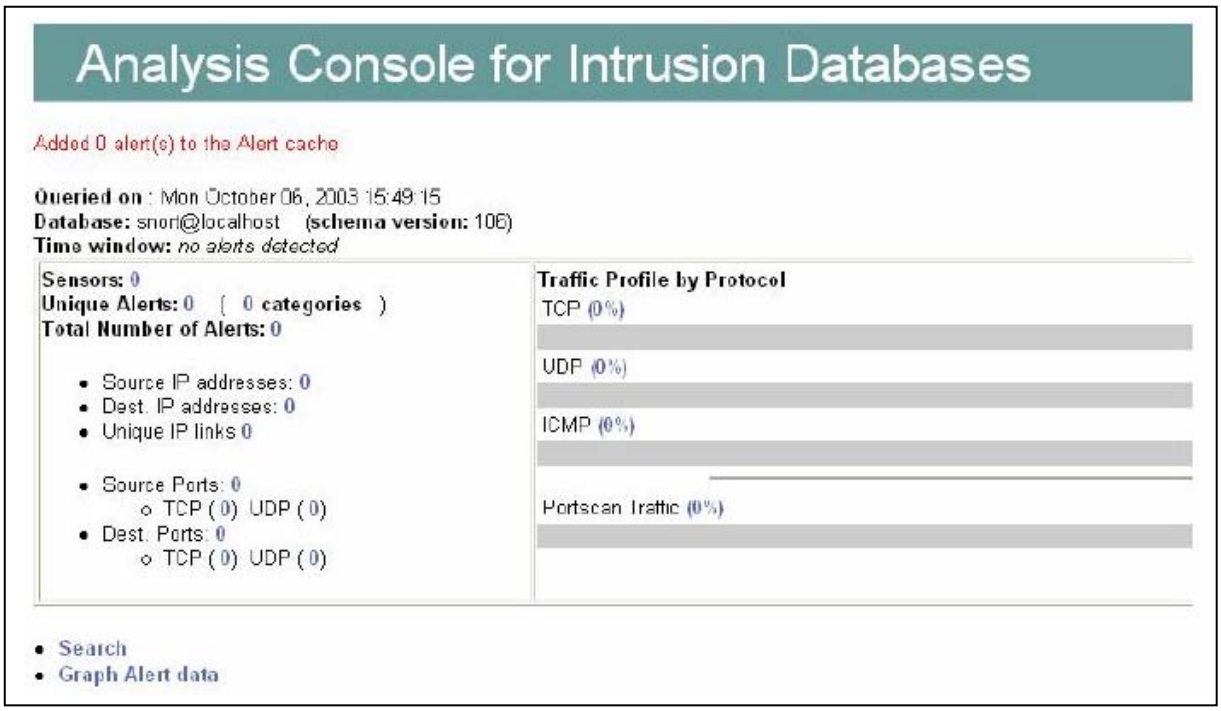
Guardamos los cambios, iniciamos Apache y desde un navegador accedemos a ACID (http://IPdelhost/acid/acid_main.php). Deberá aparecer algo como esto en el navegador:



Hacer clic en el enlace "Setup page":



Hacer clic en el icono "Create Acid AG" para crear tablas propias a ACID en la base de datos. La próxima vez que introduzcamos en el navegador http://IPdelhost/acid/ deberá aparecer la página principal de ACID:



Si queremos comprobar que todo funciona correctamente, hay que ejecutar Snort y verificar que no da ningún error cuando intenta acceder a la base de datos. Verificar también que está barriendo el tráfico (para ello se puede utilizar la opción –v o tras dejarlo un rato en funcionamiento, pararlo y ver el informe de captura de tráfico que aparece). La manera más simple de ejecutar Snort es:

```
# /usr/local/bin/snort -c /etc/snort/snort.conf -i eth1
```

Por supuesto, el mejor modo de probar su funcionamiento es simulando un ataque, de manera que podamos comprobar incluso utilizando la consola de datos de Snort que todo el conjunto de aplicaciones que hemos instalado antes funcionan correctamente.

9.1.5.6 Instalación de Barnyard

Barnyard es una aplicación, creada específicamente para Snort, que interpreta el formato de salida de datos *unified* para después guardarlo, entre otros recursos, en una base de datos, lo que descargaría de mucho trabajo al detector Snort. Para su correcto funcionamiento, a parte de la instalación, debemos cambiar ciertos parámetros de la configuración:

1. Instalación. Partimos, como siempre, del directorio donde hemos descargados los archivos comprimidos:

```
# tar -xvzf barnyard-0.1.0.tar.gz
```

```
# cd barnyard-0.1.0
#./configure --enable-mysql --with-mysql-
includes=/usr/local/mysql/include/mysql --with-mysql-
libraries=/usr/local/mysql/lib/mysql
```

Si al realizar este último paso obtenemos al final el siguiente mensaje de error:

```
ERROR: unable to find mysqlclient library
```

deberemos modificar el fichero *configure.in* (en la línea 285) y cambiar donde pone *mysql_connect* por *mysql_real_connect*. Para que estos cambios tengan efecto:

```
# aclocal
# autoconf
# autoheader
# automake
```

Volvemos a comenzar de nuevo con el proceso de ejecución:

```
# ./configure --enable-mysql --with-mysql-
includes=/usr/local/mysql/include/mysql --with-mysql-
libraries=/usr/local/mysql/lib/mysql
# make
# make install
```

Guardamos el archivo de configuración de Barnyard en la misma carpeta que el fichero de configuración de Snort:

```
# cp etc/barnyard.conf  /etc/snort/
```

2. Configuración. En el archivo de configuración es conveniente rellenar de manera apropiada los campos que aparecen a continuación (hará falta descomentar algunos):

```
config hostname: nombredelhost

config interface: eth1            # interfaz donde se
esta sniffando

config filter: not port 22

output log_dump: /var/log/snort/dump.log

output log_acid_db: mysql, database snort, server
localhost, user snort, password contraseña, detail full
```

También debemos descomentar la línea que permite tomar como referencia temporal la hora local del sistema:

```
# use localtime instead of UTC (*not* recommended because
of timewarps)
config localtime
```

9.1.6 Utilización básica de Snort

En este apartado se pretende explicar de manera básica los comandos en línea y los cambios necesarios en los ficheros de configuración para llevar a cabo una utilización simple del IDS.

En cuanto al fichero de configuración de Snort, como ya sabemos, está dividido en 4 partes. La primera parte permite la configuración de las variables de nuestra red, la segunda esta dedicada a la configuración de los preprocesadores que utiliza Snort, la tercera permite el uso de diferentes recursos para el almacenamiento de las alertas y, por último, la parte en donde configuramos las reglas que se tendrán en cuenta.

Por supuesto, los módulos de salida de Snort permiten numerosas posibilidades de almacenamiento de *logs*, pero si queremos hacer uso de la consola ACID es imprescindible activar la salida hacía la base de datos MySQL:

```
output database: log, mysql, user=snort password=contraseña
dbname=snort host=localhost
```

Si queremos descargar a Snort de esta tarea es conveniente utilizar Barnyard. Por tanto, deberemos comentar esta opción para desactivarla. En cambio, tendremos que activar en *snort.conf* los módulos de salida que almacenarán los *logs* en formato *unified*, es decir:

```
output alert_unified: filename snort.alert, limit 256
output log_unified: filename snort.log, limit 256
```

Nos aseguramos que la salida hacia la base de datos esté activada en el fichero de configuración de Barnyard, barnyard.conf:

```
output log_acid_db: mysql, database snort, server localhost, user
snort, password contraseña, detail full
```

Sólo faltan las órdenes en línea de comandos para ejecutar Snort y Barnyard. Èstas serían respectivamente:

```
# /usr/local/bin/snort –i eth1 –c /etc/snort/snort.conf –D -o
```

donde con la opción –i indicamos la tarjeta de red que queremos rastrear, con –c la ubicación del fichero de configuración de Snort, con –D la ejecución en modo demonio, y –o es una opción necesaria si se quieren tomar en cuenta los posibles filtros creados (reglas que comienzan por *pass*), y:

```
# Barnyard -c /etc/snort/barnyard.conf -d /var/log/snort/ -s
/etc/snort/sid-msg.map -g /etc/snort/gen-msg.map -f snort.log -w
/var/log/snort/checkpoint -D
```

donde con la opción –c introducimos la ubicación del fichero de configuración de Barnyard, con –d indicamos donde se encuentran los ficheros en formato *unified*, con –s y –f indicamos el camino para llegar a los ficheros *sid-msg.map* y *gen-msg.map* respectivamente (servirán para que Barnyard interprete de que alertas se trata), con –w se indica el camino a un fichero que será un punto de control, es decir, un puntero que indicará hasta que alerta se leyó en el fichero tipo *unified* la última vez que Barnyard fue ejecutado y –D hace alusión a la ejecución de la aplicación en modo demonio. Si se han seguido los pasos de configuración anteriores, las alertas no sólo se guardarán en la base de datos sino también en un archivo de *logs* ubicado en la carpeta de *logs* de Snort llamado *dump.log*. Esto es conveniente, ya que se guardarán los datos en formato largo en un fichero y tendremos otro medio de acceder a ellos en caso de que se produjera algún problema en la base de datos.

Si no hay problema, podremos visualizar las alertas que detectemos utilizando la consola ACID.

9.1.7 Diferencias en la instalación y configuración en el caso de una arquitectura distribuida Snort

Si utilizamos más de un sensor, situados en diferentes máquinas, hay que cambiar ciertos parámetros en los ficheros de configuración, además de instalar una otra herramienta, Stunnel, que permitirá establecer una comunicación cifrada entre los sensores y el servidor con el que se comunican:

1. Mientras que en una máquina destinada a ser el servidor se deben instalar todos lo programas que han aparecido hasta el momento, en un procesador en donde sólo nos interesa albergar un detector Snort, el número de aplicaciones a instalar se reduce considerablemente. Estas serían las aplicaciones a instalar en el detector:

   - Libpcap. Podemos seguir los mismos pasos que en el caso del servidor.

   - Snort. Seguimos los pasos del 1 al 5 del apartado de instalación de Snort.

   - Barnyard. Seguir instalación en el apartado de instalación de Barnyard.

   - Cliente MySQL. Habrá que realizar sólo la instalación de la parte cliente de MySQL:

     ```
     # tar –xvzf mysql-4.0.18.tar.gz
     # cd mysql-4.0.18
     # ./configure --prefix=/usr/local/mysql --without-server
     # make
     # make install
     ```

   - Stunnel como cliente. Se verá en el siguiente punto.

2. Instalación de Stunnel. Desde el directorio de descargas:

   ```
   # tar –xvzf stunnel-4.05.tar.gz
   # cd stunnel-4.05
   # ./configure
   # make
   # make install
   ```

   Para que el intercambio de información sea cifrado cada servidor Stunnel debe poseer una clave privada. Esta clave se encuentra sobre un fichero *.pem* que permite a

Stunnel definir su identidad. Por defecto, todas las instalaciones de Stunnel vienen con la misma clave. Por lo tanto será conveniente cambiarla.

En el caso de utilización de Snort en forma distribuida, la comunicación remota se produce entre el servidor y el cliente MySQL. El proceso de instalación de Stunnel en ambas máquinas es el mismo. La diferencia estará en la configuración, que se podrá modificar en el fichero *stunnel.conf*. Antes de crear este fichero realizamos los siguientes pasos:

```
# groupadd stunnel
# useradd –g stunnel stunnel
# cd /usr/local/etc/stunnel
# cp stunnel.conf-sample stunnel.conf
# chmod 644 stunnel.conf
# openssl req –new –out mail.pem –keyout mail.pem –nodes –x509 –days
365
```

En este punto, aparecen una serie de preguntas. Algunas de ellas podemos contestarlas en blanco. Otras son importantes y debemos contestarlas correctamente como "FQDN of your server" donde deberemos insertar el nombre del *host* donde hemos instalado Stunnel. Los siguientes pasos serían:

```
# chown stunnel:stunnel mail.pem
# chmod 600 mail.pem
# mkdir /usr/local/etc/rc.d
# cp /root/snortinstall/stunnel-4.0.5/tools/stunnel.init
/usr/local/etc/rc.d/stunnel.sh
# mkdir /var/tmp/stunnel
# chown stunnel:stunnel /var/tmp/stunnel
```

Hasta ahora, se han realizado los mismos pasos en la parte del servidor y en la del cliente. Sin embargo los ficheros de configuración serán diferentes en un caso y en el otro. Para el servidor, el fichero *stunnel.conf* debe ser así:

```
# Sample stunnel configuration file
# Copyright by Michal Trojnara 2002

# Comment it out on Win32
cert = /usr/local/etc/stunnel/mail.pem
chroot = /var/tmp/stunnel
# PID is created inside chroot jail
pid = /stunnel.pid
setuid = stunnel
setgid = stunnel

# Workaround for Eudora bug
#options = DONT_INSERT_EMPTY_FRAGMENTS

# Authentication stuff
#verify = 2
# don't forget about c_rehash CApath
# it is located inside chroot jail:
#CApath = /certs
# or simply use CAfile instead:
#CAfile = /usr/local/etc/stunnel/certs.pem
# CRL path or file (inside chroot jail):
#CRLpath = /crls
# or simply use CAfile instead:
#CRLfile = /usr/local/etc/stunnel/crls.pem

# Some debugging stuff
#debug = 7
#output = stunnel.log

# Use it for client mode
#client = yes

# Service-level configuration

[3306]
accept = 3307
connect = 3306
```

En el cliente el fichero de configuración deberá ser como este:

```
# Sample stunnel configuration file
# Copyright by Michal Trojnara 2002

# Comment it out on Win32
cert = /usr/local/etc/stunnel/mail.pem
chroot = /var/tmp/stunnel
# PID is created inside chroot jail
pid = /stunnel.pid
setuid = stunnel
setgid = stunnel

# Workaround for Eudora bug
#options = DONT_INSERT_EMPTY_FRAGMENTS

# Authentication stuff
#verify = 2
# don't forget about c_rehash CApath
# it is located inside chroot jail:
#CApath = /certs
# or simply use CAfile instead:
#CAfile = /usr/local/etc/stunnel/certs.pem
# CRL path or file (inside chroot jail):
#CRLpath = /crls
# or simply use CAfile instead:
#CRLfile = /usr/local/etc/stunnel/crls.pem

# Some debugging stuff
#debug = 7
#output = stunnel.log

# Use it for client mode
client = yes

#foreground = yes

# Service-level configuration

[3307]
accept = 3306
connect = IP_SERVIDOR:3307
```

Para ejecutar Stunnel utilizamos el script stunnel.sh:

```
# /usr/local/etc/rc.d/stunnel.sh start
```

3. Cambios en la configuración de otras aplicaciones. Los únicos cambios que hay que realizar son los que se refieren a la conexión remota entre servidor y sensores. Cuando el sensor no está situado en la misma máquina física que el servidor, hace falta modificar convenientemente los parámetros para establecer la comunicación entre un cliente MySQL y un servidor, y que por defecto toman el valor *localhost*. Por eso este

cambio, solamente atañe a las líneas de salida hacia una base de datos en los ficheros de configuración de Snort o de Barnyard, según cuál se esté utilizando para llevar los datos a MySQL. Lo lógico sería indicar la dirección física del servidor donde se encuentra la base de datos MySQL. No obstante, como utilizamos Stunnel para establecer una comunicación cifrada, la dirección que debemos indicar en el parámetro "server" es la física del propio detector:

```
output log_acid_db: mysql, database snort, server ip_detector,
user snort, password contraseña, detail full
```

Esto es así porque Stunnel escucha en *localhost* todas las conexiones por el puerto que se configura en stunnel.conf (en este caso 3306) y las transmite de manera cifrada hacía el servidor que le hayamos indicado. Debemos iniciar primero la aplicación Stunnel, para que cuando, bien Snort o bien Barnyard comiencen a transmitir los datos, el canal esté ya cifrado.

Si llegados a este punto intentamos establecer una comunicación con la base de datos de Snort, obtendremos un error de conexión, puesto que no hemos dado los permisos correspondientes al usuario snort@ip_detector. Por tanto, en el servidor MySQL debemos realizar esta tarea. Accedemos a la base de datos y:

```
mysql>grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to
snort@ip_detector;
```

Ahora no debería existir ningún problema para poder llevar a cabo la transmisión de datos hacia el servidor.

9.1.8 Instalación de Prelude-IDS

El IDS híbrido Prelude consta de 4 módulos importantes: libprelude, prelude-manager, prelude-nids y prelude-lml, además de una consola para visualizar los resultados de la detección.

9.1.8.1 Instalación de Libprelude

Partiendo del directorio de descargas:

```
# tar -xvzf libprelude-0.8.10.tar.gz
# cd libprelude-0.8.10
```

```
# ./configure
# make
# make install
```

9.1.8.2 Instalación de Prelude-manager

De nuevo:

```
# tar –xvzf prelude-manager-0.8.10.tar.gz
# cd prelude-manager-0.8.10
# ./configure
# make
# make install
```

9.1.8.3 Instalación de Prelude-nids

```
# tar –xvzf prelude-nids-0.8.6.tar.gz
# cd prelude-nids-0.8.6
# ./configure
# make
# make install
```

9.1.8.4 Instalación de Prelude-lml

```
# tar –xvzf prelude-lml.0.8.6.tar.gz
# cd prelude-lml.0.8.6
# ./configure
# make
# make install
```

Si el proceso de instalación no se puede completar porque no se encuentra la librería pcre, entonces habría que descargarse dicha librería de la red. Podemos buscar el paquete rpm (http://www.rpmfind.net) para RedHat 8 e instalarlo como sigue:

```
# rpm –ivh pcre-3.9-5.rpm
```

y después recomenzar con la instalación de prelude-lml.

9.1.8.5 Creación de la base de datos

Teniendo en cuenta que la instalación de MySQL ya se ha realizado, ahora deberemos crear la base de datos que utilizará Prelude. Dentro del directorio *prelude-manager* ejecutamos:

```
# ./prelude-manager-db-create.sh
```

Deberemos introducir el nombre de la base de datos para Prelude, el administrador (introducir root) y un usuario y contraseña para Prelude (pueden ser cualquiera).

Se puede comprobar que se ha creado la base de datos de la misma manera que se hizo en el caso de Snort.

9.1.8.6 Instalación de PIWI

Para la instalación de PIWI (el frontend de Prelude desarrollado en Perl) es necesario haber realizado previamente la instalación de los siguientes componentes: Perl, Date-Calc, CGI, DBI y DBD-mysql. Normalmente, la aplicación Perl viene instalada por defecto. El proceso de instalación de PIWI incluye por tanto la de estos módulos:

1. Instalación del módulo Date-Calc:

```
# tar –xvzf Date-Calc-5.3.tar.gz
# cd Date-Calc-5.3
# perl Makefile.PL
# make
# make test
# make install
# make realclean
```

2. Instalación del módulo CGI:

```
# tar –xvzf CGI.pm-3.04
# cd CGI.pm-3.04
# perl Makefile.PL
# make
# make test
# make install
# make realclean
```

3. Instalación del módulo DBI:

```
# tar –xvzf DBI-1.40
# cd DBI-1.40
# perl Makefile.PL
# make
# make test
# make install
# make realclean
```

4. Instalación del módulo DBD-mysql:

```
# tar –xvzf DBD-mysql-2.9003
# cd DBD-mysql-2.9003
# perl Makefile.PL
# make
# make test
# make install
# make realclean
```

5. Instalación de PIWI:

```
# cp piwi-0-8-lastest.tar.gz  /usr/local/apache/htdocs
# cd /usr/local/apache/htdocs
# tar –xvzf  piwi-0-8-lastest.tar.gz
# rm –rf piwi-0-8-lastest.tar.gz
```

   Es necesario que el usuario de apache (nobody) tenga los permisos de la carpeta /generated:

```
# chown –R nobody.nobody /usr/local/apache/htdocs/piwi/generated
```

6. Configuración. Tenemos que modificar el fichero de configuración de PIWI, config.pl de tal manera que podamos acceder a la base de datos para Prelude:

```
# Database :
       $conf{'dbtype'} = 'mysql'; # mysql / Pg
       $conf{'dbname'} = 'prelude';
       $conf{'dbhost'} = 'localhost';
#      $conf{'dbport'} = 5432; # default mysql port is
3306 / pgsql 5432 (only uncomment if using Postgres)
#      $conf{'dboptions'} = 'mysql_compression=1'; #
(only uncomment with mysql)
       $conf{'dblogin'} = 'prelude_user';
       $conf{'dbpasswd'} = 'contraseña';
```

PIWI permite el borrado de alertas de la base de datos pero para ello debe conectarse con el perfil de "admin" y no con el de "guest" como hace por defecto. Si queremos que "guest" pueda también suprimir alertas en la base de datos debemos editar el fichero /usr/local/apache/piwi/Profiles/guest.user:

```
# User full-name :
FullName=guest
 # IP Access mask. 255.255.255.255 means any IP :
IPAccess=255.255.255.255

 # Delete privilege : (if 'none', can't delete any alert)
priv_delete=all
 # User privilege : (if 'none', can't
create/modify/delete any user profile)
priv_user=all
 # Process privilige: (if 'none', can't start processing
of alerts)
priv_process=none
 # Acknowledgement privilege :
priv_ack=all
```

7. Instalación de otros componentes. Cabe la posibilidad de instalar otros módulos como Geo::IP (permite a PIWI conocer la nacionalidad asignada a una dirección IP y lo utiliza para hacer estadisticas) o PDF::API2 (se podrían transmitir alertas utilizando ficheros PDF). Para llevar a cabo su instalación sólo habría que llevar a cabo los mismos pasos que en los primeros puntos.

8. Comprobación. Podemos comprobar el funcionamiento de PIWI accediendo mediante un navegador a la carpeta https://IPdelhost/piwi. Deberemos ver la siguiente página:

Hay una opción que permite comprobar si nos falta algún módulo. Para ello ir a https://IPdelhost/piwi/test y aparecerá un mensaje similar a este:

Mandatory perl modules :

perl : You have perl v5.6.0 or newer
Socket : Network routines, name resolution ok
CGI : You have the CGI module
DBI : You have generic DB access module
+ DBD : You have, at least, one specific DB access module
Date::Calc : Date manipulations

Optionnal perl modules :

Geo::IP : Country look-up for an IP address
Ghostscript : Ghostscript (postscript generation tool) has been found
PDF::API2 : PDF::API2 (PDF report generation) has not been found

---

Additional tests :

generated/ directory perms : generated/ sub-dir is writable by this webserver
Profiles/ directory perms : Profiles/ sub-dir is readable by this webserver
DB access configuration : this script could have access to prelude DB
password-protection : piwi directory is password protected. look at Docs/user_file_format.txt to create profiles with privilege separation

9.1.9 Utilización básica de Prelude

En este apartado vamos a ver los pasos necesarios para poner en funcionamiento una estructura Prelude-IDS. Es decir, como configurarlo para que acceda a la base datos, como realizar la comunicación entre los sensores y el manager y como poner finalmente el sistema en ejecución:

1. Configuración de Prelude-manager. Debemos indicar al manager que base de datos debe utilizar. Esto se configura en el archivo */usr/local/etc/prelude-manager/prelude-manager.conf*. Editamos este fichero de manera que los siguientes campos sean como sigue:

```
[MySQL]

# Host the database is listening on.
dbhost = localhost;

# Name of the database.
dbname = prelude;

# Username to be used to connect the database.
dbuser = prelude_user;

# Password used to connect the database.
dbpass = contraseña;
```

2. Registro de sensores. Cuando queremos registrar un sensor para que sea reconocido por el manager y pueda comunicarse con él, debemos ejecutar dos comandos simultáneamente:

   a) por parte del manager:

   ```
   # manager-addusser
   ```

   b) por parte del sensor:

   ```
   # sensor-adduser -s <sensorname> -u <uid> -m <manager address>
   ```

   Debemos seguir las instrucciones que aparezcan al ejecutar estos comandos. Veamos, por ejemplo, como registramos el sensor prelude-nids. Si primero ejecutamos manager-adduser:

```
# manager-adduser
Generated one-shot password is "ajg4pde3".
Waiting for install request from Prelude sensors ...
```

Desde otro terminal debemos ejecutar:

```
# sensor-adduser -s prelude-nids -m 127.0.0.1 -u 0
Enter registration one shot password : ajg4pde3
Please confirm one shot password : ajg4pde3
connecting to Manager host (127.0.0.1:5553)... Succeeded.
Username to use to authenticate : nids
Please enter a password for this user : contraseña
Please re-enter the password (comfirm) :
Plaintext account creation succeed with Prelude Manager.
Allocated ident for prelude-nids@ids1: 954839305001359532.
```

Este proceso debe repetirse para registrar todos los sensores que queramos utilizar (prelude-nids, prelude-lml, snort ...). Sólo debemos hacer este proceso de identificación una vez, justo antes de utilizar el sensor con Prelude por primera vez.

Si queremos utilizar como sensor snort utilizando Prelude como manager, debemos primero activar el módulo para Prelude en el fichero de configuración de Snort, *snort.conf*:

```
output alert_prelude: async,
classification_file=/root/snortinstall/snort-
2.0.5/etc/prelude-classification.config
```

y registramos el sensor de la misma manera que acabamos de hacer con prelude-nids, sustituyendo este nombre por el de "snort" en el comando correspondiente.

3. Ejecución de Prelude-IDS. Debemos ejecutar el manager y también cada uno de los sensores. El siguiente ejemplo sería utilizando prelude-manager y prelude-nids:

   Inicializamos el manager:

```
# prelude-manager
- Initialized 2 reporting plugins.
- Initialized 1 database plugins.
- Subscribing Prelude NIDS data decoder to active decoding
  plugins.
```

```
- Initialized 1 decoding plugins.
- Subscribing TextMod to active reporting plugins.
- Subscribing MySQL to active database plugins.
- sensors server started (listening on 127.0.0.1:5554).
```

Inicializamos el sensor:

```
# prelude-nids -i eth0
- Initialized 3 protocols plugins.
- Initialized 5 detections plugins.
- HttpMod subscribed for "http" protocol handling.
-  Done  loading  Unicode  table  (663  Unichars,  0  ignored,  0
with errors).
- RpcMod subscribed for "rpc" protocol handling.
- TelnetMod subscribed for "telnet" protocol handling.
- ArpSpoof subscribed to : "[ARP]".
- ScanDetect subscribed to : "[TCP,UDP]".
- Signature engine added 890 and ignored 2 signature.
- Connecting to Unix prelude Manager server.
- Plaintext authentication succeed with Prelude Manager.
- Initializing packet capture.
```

Si la conexión entre manager y sensor se ha producido correctamente debería aparecer en el manager algo como esto:

```
[unix] - accepted connection.
[unix] - plaintext authentication succeed.
[unix] - sensor declared ident 954839305001359532.
```

9.1.10 Diferencias en la configuración en el caso de una arquitectura distribuida Prelude

Una arquitectura distribuida Prelude implica la existencia de varios sensores (ubicados en máquinas diferentes) y al menos un manager que gobernará todos los datos provenientes de estos sensores.

Al igual que en una arquitectura distribuida Snort, si utilizamos un procesador solamente como sensor o detector Prelude tendremos que instalar menos componentes. La

arquitectura Prelude es muy modular y como ya se ha comentado permite integrar diferentes sensores (prelude-nids, prelude-lml, snort, nessus, ...).

En un detector Prelude se debe instalar siempre como base la librería *libprelude* y después los sensores que deseemos utilizar. Para ello se pueden seguir los pasos que ya se han comentado para realizar sus instalaciones respectivas.

Los cambios que se deben realizar en cuanto a la ejecución de comandos y registro de sensores en un sistema distribuido tienen que ver con el cambio de dirección IP del manager y son los siguientes:

1. Cambios en los ficheros de configuración. Habrá que cambiar las siguientes líneas en estos ficheros de configuración:

    a) */usr/local/etc/prelude-sensors/sensors-default.conf* :

    ```
    manager-addr =  IP_del_manager;
    ```

    b) en el caso de utilizar prelude-nids habrá que cambiar también su fichero de configuración */usr/local/etc/prelude-nids/prelude-nids.conf*:

    ```
    manager-addr =  IP_del_manager;
    ```

    c) si utilizamos prelude-lml modificaremos */usr/local/etc/prelude-lml/prelude-lml.conf*:

    ```
    manager-addr =  IP_del_manager;
    ```

    d) en el sistema donde se encuentre el manager también debemos modificar el fichero de configuración de éste en */usr/local/etc/prelude-manager/prelude-manager.conf*:

    ```
    sensors-srvr =  IP_del_manager;
    ```

2. Cambios en el registro de sensores. Para registrar un sensor se sigue prácticamente el mismo proceso explicado anteriormente. Sólo cambiamos la dirección IP en la que se encuentra el manager, de manera que, por ejemplo, en el caso de querer registrar un sensor prelude-nids ubicado en un sistema diferente al del manager, la línea de comandos sería:

    ```
    # sensor-adduser -s prelude-nids -m IP_del_manager -u 0
    ```

En el otro extremo, desde la máquina donde se encuentra el manager debemos ejecutar igual que antes:

```
# manager-adduser
```

La diferencia ahora, será que generará una clave privada para poder establecer después una conexión segura entre sensor y manager usando SSL.

Si utilizamos los comandos para iniciar manager y sensor la comunicación remota debería funcionar correctamente.

9.1.11 Instalación de otras aplicaciones

En este apartado vamos a seguir los pasos necesarios para que se puedan llevar a cabo aplicaciones como son la notificación de alertas vía mail.

9.1.11.1 Notificación de alertas vía mail

La notificación de alertas vía mail permitirá comunicar al administrador un resumen de todas las alertas que se han producido durante un determinado periodo de tiempo. Evidentemente, no es práctico enviar una notificación por mail cada vez que se produzca una alerta. Para poner en funcionamiento dicha aplicación tenemos que hacer uso de dos programas: *mailsend* y *snort-rep*. Estos son los pasos que debemos seguir:

1. Instalación de mailsend. Este programa servirá para enviar correos dentro de la red local utilizando el protocolo smtp. Desde el directorio de descargas:

```
# unzip mailsend_src_1.05.zip
# cd mailsend
# ./configure
# make
```

2. Podemos comprobar si el programa funciona sólo con ejecutarlo:

```
# ./mailsend
SMTP server address/IP: 192.168.25.150
```

```
Domain: mondomain.com
From: moi@mondomail.com
To: gema.detoro@arsoe-trelaze.com
Carbon copy:
Blind Carbon copy:
Subject: test
=================================================================
Type . in a new line and press Enter to end the message, CTRL+C to
abort
=================================================================
test
.
Mail sent successfully
```

Todas los parámetros anteriores se pueden introducir mediante opciones por línea de comando y utilizando un fichero que contenga los datos.

3. Instalación de snort-rep. Este programa se compone fundamentalmente de dos *scripts*. Uno de ellos (snort-rep) se encarga de realizar un informe de alertas utilizando como base el fichero de la salida *syslog* de Snort. El otro *script* (snort-rep-mail) utiliza el anterior, y lo que hace básicamente es preparar la información para enviarla por mail. Para instalarlo:

```
# tar –xvzf snort-rep-1.10.tar.gz
# cd snort-rep-1.10
```

Dentro de esta carpeta, se encuentran ambos *scripts* ya compilados.

4. Modificaciones. El *script* snort-rep-mail está escrito para enviar alertas vía mail a un determinado usuario del equipo donde está instalado. Pero esto no es lo que nosotros queremos hacer (por eso hemos instalado mailsend), ya que es necesario enviar los correos a través de la red, y, por tanto, deberemos realizar ciertas modificaciones. También modificaremos algunas líneas de código en el otro *script* (snort-rep) debido a que no realiza bien, por ejemplo, el tratamiento de prioridades en el informe. Por tanto, comenzaremos por copiar dichos *scripts* en otros ficheros con nombre diferente:

```
# cp snort-rep reporter
# cp snort-rep-mail report2file
```

Debemos crear también un par de ficheros y una carpeta:

- Fichero report.txt. Uno de los cambios fundamentales de la aplicación es que, en lugar de enviar un informe directamente a un usuario, guarda dicho informe en un fichero (report.txt). Este fichero será utilizado más tarde por la aplicación mailsend como contenido de los mensajes que envíe. Este fichero debe estar inicialmente vacío:

  # touch report.txt

- Fichero snort-rep.local-nets. Este fichero indicará cual es nuestra red local.

  # vi /etc/snort-rep.local-nets

  Insertar en nuestro caso: 192.168.y.0/24

- Directorio snort-old. Cada vez que se realice un informe, se almacenará en este directorio el archivo de *logs* utilizado para hacerlo. Deberemos crearlo:

  # mkdir /var/log/snort-old

- Fichero alert. Éste es el fichero que utilizará Snort para escribir las alertas vía *syslog*. Su camino será: */var/log/snort/alert*

Vamos a empezar modificando el *script* que hemos llamado *report2file* con ayuda de un editor. Serían estas:

- Referidas a la definición de variables. Hay un cierto número de variables que se refieren a directorios o a ficheros y que debemos inicializar correctamente:

```
my $vault = "/var/log/snort-old"; # where to place old snort logs
my $logname = "/var/log/snort/alert"; # snort log-file
my $snort_rep = '/root/IDSinstall/snort-rep-1.10/reporter'; #
reporter_path
```

Por otro lado, debemos definir una variable que contenga el fichero donde queremos escribir:

```
my $reportfile = "/root/IDSinstall/snort-rep-1.10/report.txt";
```

- Código a añadir. Vamos a añadir dos trozos de código. El primero reinicia el servicio syslog, lo que permite a Snort seguir almacenando alertas después de que el *script* cree de nuevo el fichero de alertas donde Snort debe empezar a cargar los *logs* de nuevo:

```
system "/etc/init.d/syslog restart";
```

Por otro lado, añadiremos las siguientes líneas para que el *script* escriba el informe en el fichero creado anteriormente a tal efecto:

```
if($text) {
        open (SALIDA,">$reportfile")||die "ERROR: I can't open
$informe\n";
        print SALIDA $text;
        close (SALIDA);}
```

- Código a eliminar. Debemos eliminar o comentar todo el código referido al envío por mail del informe y la parte de html. En definitiva el fichero debe ser así:

```
#!/usr/bin/perl -w

# This is an example script to generate e-mail with daily snort reports
# It rotates the snort.log file, generates a report in both text and html
# (as MIME multipart/alternate) and sends it to root.
#
# It uses snort-rep (http://people.ee.ethz.ch/~dws/software/snort-rep)

use strict;
use POSIX qw(strftime);
#use MIME::Lite;

$ENV{PATH}='/usr/bin:/bin';

my $reportfile = "/root/IDSinstall/snort-rep-1.10/report.txt";
my $vault   = "/var/log/snort-old"; # where to place old snort logs
my $logname = "/var/log/snort/alert"; # snort log-file
my $snort_rep = '/root/IDSinstall/snort-rep-1.10/reporter'; # snort-rep path
my @snort_rep_args =                          # snort-rep arguments
        qw(
        --narrow
        --resolve
        --local-file=/etc/snort-rep.local-nets
        --remove-name='\.ethz\.ch'
        );

sub TodayStr()
{
        return strftime("%Y%m%d",gmtime);
}

sub mv($$)
{
        my $from = shift;
        my $to = shift;
        # rename original
        rename "$from", "$from.$$" or die "ERROR: can't rename $from to
$from.$$\n";
        # copy
        system "cp $from.$$ $to.$$" and exit 1;
        # delete original
         unlink "$from.$$" or die "ERROR: can't remove $from.$$: $!\n";
         # rename new
         rename "$to.$$", $to or die "ERROR: can't rename $to.$$ to $to:
$!\n";
}
```

```
(stat($logname))[7]>0 or exit;

my $i=0;
my $base = "$vault/snort-".TodayStr();
while(-e "$base.$i") {
        $i++;
}
my $lastmoved = "$base.$i";
mv($logname, $lastmoved);
system "touch $logname";
system "/etc/init.d/syslog restart";
#system "kill -HUP `cat /etc/syslog.pid`"; # this is for Solaris...

#my $msg = MIME::Lite->new(
#       From => 'root',
#       To => 'root',
#       Subject => '[snort] report',
#       Type => 'multipart/alternative',
#       Datestamp => 0,
#);

# make report
push @snort_rep_args, '--text';
push @snort_rep_args, '--html';
my $text = '';
my $html = '';
my $cmd = "$snort_rep ".join(' ',@snort_rep_args)." $lastmoved";
open(REPORT, "$cmd|") or die "can't execute $snort_rep: $!\n";
my $is_text=1;
while(<REPORT>) {
        if($is_text and /^<<<<<</) { $is_text=0; next; }
        if($is_text) { $text .= $_; }
        else         { $html .= $_; }
}
close(REPORT);
#print "$text\n";
if($text) {
        open (SALIDA,">$reportfile")||die "ERROR: No puedo abrir el fichero
$reportfile\n";
        print SALIDA $text;
        close (SALIDA);}

#$msg->attach(
#       Type => 'TEXT',
#       Data => $text,
#);
#$msg->attach(
#       Type => 'text/html',
#       Data => $html,
#);
#$msg->scrub;

# send it
#$msg->send;
```

El otro fichero que debemos modificar es el que ahora hemos llamado *reporter*. No eliminaremos ninguna parte del código y sólo haremos las siguientes modificaciones:

- En la línea 552, cambiar el valor de inicialización de las variables a éste:

```
my $default_priority_med  = 2;
my $default_priority_high = 1;
```

- Entre la línea 655 y 687 debemos cambiar el código de manera que quede así:

```
for (my $i = $prio_high ; $i <= 5 ; $i++) {
        $PRIO{$i} = {
            'order' => 0,
            'text'  => 'HIGH',
            'html'  => '<FONT COLOR="#BB0000">high</FONT>'
        };
    }

    for (my $i = $prio_med ; $i <= 5 ; $i++) {
        $PRIO{$i} = {
            'order' => 1,
            'text'  => 'med',
            'html'  => '<FONT COLOR="#FF8800">med</FONT>'
        };
    }

    for (my $i = 3 ; $i <= 5 ; $i++) {
        $PRIO{$i} = {
            'order' => 1,
            'text'  => 'low',
            'html'  => 'low'
        };
    }
```

5. Crear un *script*, que recoja los dos programas anteriores, para enviar vía mail el informe de alertas. Es muy sencillo, le hemos llamado *sendreport* y sería como:

```
#!/bin/bash

/root/IDSinstall/snort-rep-1.10/report2file
/root/IDSinstall/mailsend/mailsend -d snort.com -smtp
192.168.50.150 -f IDSsnort1@snort.com -t gema.detoro@arsoe-
trelaze.com -sub "rapport" -m /root/IDSinstall/snort-rep-
1.10/report.txt
```

6. Activar la salida syslog de Snort. Para ello, debemos descomentar la línea correspondiente en el fichero de configuración de Snort, *snort.conf*:

```
output alert_syslog: LOG_AUTH LOG_ALERT
```

Ahora, tenemos que modificar el fichero de configuración de syslog */etc/syslog.conf*, de manera que todos los mensajes que provengan de Snort los guarde en el fichero correspondiente. Es decir, tenemos que añadir la línea siguiente:

```
# Mensajes de snort
auth.alert                              /var/log/snort/alert
```

Después de la modificación de este fichero, reiniciamos el servicio *syslog*:

```
# service syslog restart
```

7. Ejecutar el envío de informe cada cierto periodo de tiempo. Si queremos enviar los informes de manera periódica nos podemos ayudar de la aplicación *cron*. Por ejemplo, si queremos enviar un informe diario, debemos realizar los siguientes pasos:

```
# cd /etc/cron.daily/
# vi sendreport
```

y editar:

```
/root/IDSinstall/snort-rep-1.10/sendreport
```

En el fichero */etc/crontab* podemos configurar el momento del día en el que queremos que se ejecuten las aplicaciones que hay dentro de la carpeta *cron.daily*. Si, por ejemplo, queremos que se ejecute a las 2:05 horas de la mañana, en dicho fichero:

```
# run-parts
01 * * * * root run-parts /etc/cron.hourly
05 2 * * * root run-parts /etc/cron.daily
20 3 * * 0 root run-parts /etc/cron.weekly
45 4 1 * * root run-parts /etc/cron.monthly
```

Ahora, tenemos que reiniciar el servicio *cron*. Para ello:

```
# service crond restart
```

También podemos hacer que la frecuencia de envío de informes sea semanal o mensual. Esto implicaría hacer el mismo el proceso:

- Creamos otros dos directorios para guardar los *logs*, uno semanal y otro mensual:

```
# mkdir /var/log/snort-weekly
# mkdir /var/log/snort-mouthly
```

- En el fichero de configuración de *syslog* añadimos otros dos ficheros de salida:

```
# Mensajes de snort
auth.alert                      /var/log/snort/alert
auth.alert                      /var/log/snort/alertweek
auth.alert                      /var/log/snort/alertmonth
```

- Cambiar en el fichero *report2file* las asignaciones de variables por los valores que ahora corresponden y crear dos nuevos *scripts*: *report2fileweek* y *report2filemonth*.

- Realizar una operación similar con el fichero *sendreport*. Es decir, modificamos el fichero cambiando *report2file* por *report2file** y creamos dos nuevos ficheros *sendreportweek* y *sendreportmonth*.

- Finalmente, crear otros dos ficheros *sendreportweek* y *sendreportmonth*, que contengan respectivamente:

```
/root/IDSinstall/snort-rep-1.10/sendreportweek
```

```
/root/IDSinstall/snort-rep-1.10/sendreportmonth
```

Guardamos estos ficheros dentro de los directorios */etc/cron.weekly* y */etc/cron.month* respectivamente. En el fichero */etc/crontab* configuramos el momento dentro de la semana o el mes en el que queremos que se ejecuten las aplicaciones dentro de dichas carpetas.

9.2 ANEXO II : FICHEROS DE CONFIGURACIÓN

9.2.1 Fichero de configuración de Snort : snort.conf

9.2.1.1 snort.conf en el detector situado en la DMZ2

```
#--------------------------------------------------
#   http://www.snort.org     Snort 2.0.0 Ruleset
#     Contact: snort-sigs@lists.sourceforge.net
#--------------------------------------------------
# $Id: snort.conf,v 1.124 2003/05/16 02:52:41 cazz Exp $
#
##################################################
# This file contains a sample snort configuration.
# You can take the following steps to create your
# own custom configuration:
#
#  1) Set the network variables for your network
#  2) Configure preprocessors
#  3) Configure output plugins
#  4) Customize your rule set
#
##################################################
# Step #1: Set the network variables:
#
# You must change the following variables to reflect
# your local network. The variable is currently
# setup for an RFC 1918 address space.
#
# You can specify it explicitly as:
#
# var HOME_NET 10.1.1.0/24
#
# or use global variable $<interfacename>_ADDRESS
# which will be always initialized to IP address and
# netmask of the network interface which you run
# snort at.  Under Windows, this must be specified
# as $(<interfacename>_ADDRESS), such as:
# $(\Device\Packet_{12345678-90AB-CDEF-1234567890AB}_ADDRESS)
#
# var HOME_NET $eth0_ADDRESS
#
# You can specify lists of IP addresses for HOME_NET
# by separating the IPs with commas like this:
#
# var HOME_NET [10.1.1.0/24,192.168.1.0/24]
#
# MAKE SURE YOU DON'T PLACE ANY SPACES IN YOUR LIST!
#
# or you can specify the variable to be any IP address
# like this:

var HOME_NET 192.168.50.0/24
```

```
# Set up the external network addresses as well.
# A good start may be "any"

var EXTERNAL_NET any

# Configure your server lists.  This allows snort to only look for attacks
# to systems that have a service up.  Why look for HTTP attacks if you are
# not running a web server?  This allows quick filtering based on IP addresses
# These configurations MUST follow the same configuration scheme as defined
# above for $HOME_NET.

# List of DNS servers on your network
var DNS_SERVERS [192.168.50.100,192.168.50.101]

# List of SMTP servers on your network
var SMTP_SERVERS $HOME_NET

# List of web servers on your network
var HTTP_SERVERS $HOME_NET

# List of sql servers on your network
var SQL_SERVERS $HOME_NET

# List of telnet servers on your network
var TELNET_SERVERS $HOME_NET

# Configure your service ports.  This allows snort to look for attacks
# destined to a specific application only on the ports that application
# runs on.  For example, if you run a web server on port 8081, set your
# HTTP_PORTS variable like this:
#
# var HTTP_PORTS 8081
#
# Port lists must either be continuous [eg 80:8080], or a single port [eg 80].
# We will adding support for a real list of ports in the future.

# Ports you run web servers on
var HTTP_PORTS 80

# Ports you want to look for SHELLCODE on.
var SHELLCODE_PORTS !80

# Ports you do oracle attacks on
var ORACLE_PORTS 1521

# other variables
#
# AIM servers.  AOL has a habit of adding new AIM servers, so instead of
# modifying the signatures when they do, we add them to this list of
# servers.
var AIM_SERVERS
[64.12.24.0/24,64.12.25.0/24,64.12.26.14/24,64.12.28.0/24,64.12.29.0/24,64.12.161.0/24,64.12.163.0/
24,205.188.5.0/24,205.188.9.0/24]

# Path to your rules files (this can be a relative path)
var RULE_PATH /etc/snort

# Configure the snort decoder:
# ============================
#
```

```
# Stop generic decode events:
#
# config disable_decode_alerts
#
# Stop Alerts on experimental TCP options
#
# config disable_tcpopt_experimental_alerts
#
# Stop Alerts on obsolete TCP options
#
# config disable_tcpopt_obsolete_alerts
#
# Stop Alerts on T/TCP alerts
#
# config disable_ttcp_alerts
#
# Stop Alerts on all other TCPOption type events:
#
# config disable_tcpopt_alerts
#
# Stop Alerts on invalid ip options
#
# config disable_ipopt_alerts


# Configure the detection engine
# ================================
#
# Use a different pattern matcher in case you have a machine with very
# limited resources:
#
# config detection: search-method lowmem


##################################################
# Step #2: Configure preprocessors
#
# General configuration for preprocessors is of
# the form
# preprocessor <name_of_processor>: <configuration_options>

# frag2: IP defragmentation support
# ------------------------------
# This preprocessor performs IP defragmentation.  This plugin will also detect
# people launching fragmentation attacks (usually DoS) against hosts.  No
# arguments loads the default configuration of the preprocessor, which is a
# 60 second timeout and a 4MB fragment buffer.

# The following (comma delimited) options are available for frag2
#    timeout [seconds] - sets the number of [seconds] than an unfinished
#                        fragment will be kept around waiting for completion,
#                        if this time expires the fragment will be flushed
#    memcap [bytes] - limit frag2 memory usage to [number] bytes
#                     (default:  4194304)
#
#    min_ttl [number] - minimum ttl to accept
#
#    ttl_limit [number] - difference of ttl to accept without alerting
#                         will cause false positves with router flap
#
```

```
# Frag2 uses Generator ID 113 and uses the following SIDS
# for that GID:
#  SID     Event description
# -----   -------------------
#   1      Oversized fragment (reassembled frag > 64k bytes)
#   2      Teardrop-type attack

preprocessor frag2: timeout 65, min_ttl 3, ttl_limit 8

# stream4: stateful inspection/stream reassembly for Snort
#---------------------------------------------------------------------
# Use in concert with the -z [all|est] command line switch to defeat
# stick/snot against TCP rules.  Also performs full TCP stream
# reassembly, stateful inspection of TCP streams, etc.  Can statefully
# detect various portscan types, fingerprinting, ECN, etc.

# stateful inspection directive
# no arguments loads the defaults (timeout 30, memcap 8258608)
# options (options are comma delimited):
#   detect_scans - stream4 will detect stealth portscans and generate alerts
#                  when it sees them when this option is set
#   detect_state_problems - detect TCP state problems, this tends to be very
#                       noisy because there are a lot of crappy ip stack
#                       implementations out there
#
#   disable_evasion_alerts - turn off the possibly noisy mitigation of
#                       overlapping sequences.
#
#
#   min_ttl [number]       - set a minium ttl that snort will accept to
#                       stream reassembly
#
#   ttl_limit [number]     - differential of the initial ttl on a session versus
#                        the normal that someone may be playing games.
#                        Routing flap may cause lots of false positives.
#
#   keepstats [machine|binary] - keep session statistics, add "machine" to
#                       get them in a flat format for machine reading, add
#                       "binary" to get them in a unified binary output
#                       format
#   noinspect - turn off stateful inspection only
#   timeout [number] - set the session timeout counter to [number] seconds,
#                   default is 30 seconds
#   memcap [number] - limit stream4 memory usage to [number] bytes
#   log_flushed_streams - if an event is detected on a stream this option will
#                       cause all packets that are stored in the stream4
#                       packet buffers to be flushed to disk.  This only
#                       works when logging in pcap mode!
#
# Stream4 uses Generator ID 111 and uses the following SIDS
# for that GID:
#  SID     Event description
# -----   -------------------
#   1      Stealth activity
#   2      Evasive RST packet
#   3      Evasive TCP packet retransmission
#   4      TCP Window violation
#   5      Data on SYN packet
#   6      Stealth scan: full XMAS
#   7      Stealth scan: SYN-ACK-PSH-URG
```

```
# 8      Stealth scan: FIN scan
# 9      Stealth scan: NULL scan
# 10     Stealth scan: NMAP XMAS scan
# 11     Stealth scan: Vecna scan
# 12     Stealth scan: NMAP fingerprint scan stateful detect
# 13     Stealth scan: SYN-FIN scan
# 14     TCP forward overlap

preprocessor stream4: detect_scans, disable_evasion_alerts, min_ttl 3, ttl_limit 8, timeout 35

# tcp stream reassembly directive
# no arguments loads the default configuration
#   Only reassemble the client,
#   Only reassemble the default list of ports (See below),
#   Give alerts for "bad" streams
#
# Available options (comma delimited):
#   clientonly - reassemble traffic for the client side of a connection only
#   serveronly - reassemble traffic for the server side of a connection only
#   both - reassemble both sides of a session
#   noalerts - turn off alerts from the stream reassembly stage of stream4
#   ports [list] - use the space separated list of ports in [list], "all"
#                  will turn on reassembly for all ports, "default" will turn
#                  on reassembly for ports 21, 23, 25, 53, 80, 143, 110, 111
#                  and 513

preprocessor stream4_reassemble: both

# http_decode: normalize HTTP requests
# ------------------------------------
# http_decode normalizes HTTP requests from remote
# machines by converting any %XX character
# substitutions to their ASCII equivalent. This is
# very useful for doing things like defeating hostile
# attackers trying to stealth themselves from IDSs by
# mixing these substitutions in with the request.
# Specify the port numbers you want it to analyze as arguments.
#
# Major code cleanups thanks to rfp
#
# unicode         - normalize unicode
# iis_alt_unicode  - %u encoding from iis
# double_encode    - alert on possible double encodings
# iis_flip_slash   - normalize \ as /
# full_whitespace  - treat \t as whitespace ( for apache )
#
# for that GID:
#  SID     Event description
# -----   ------------------
#   1      UNICODE attack
#   2      NULL byte attack

preprocessor http_decode: 80 unicode iis_alt_unicode double_encode iis_flip_slash full_whitespace

# rpc_decode: normalize RPC traffic
# ---------------------------------
# RPC may be sent in alternate encodings besides the usual
# 4-byte encoding that is used by default.  This preprocessor
# normalized RPC traffic in much the same way as the http_decode
# preprocessor.  This plugin takes the ports numbers that RPC
```

```
# services are running on as arguments.
# The RPC decode preprocessor uses generator ID 106
#
# arguments: space separated list
# alert_fragments - alert on any rpc fragmented TCP data
# no_alert_multiple_requests - don't alert when >1 rpc query is in a packet
# no_alert_large_fragments - don't alert when the fragmented
#                            sizes exceed the current packet size
# no_alert_incomplete - don't alert when a single segment
#                       exceeds the current packet size

preprocessor rpc_decode: 111 32771

# bo: Back Orifice detector
# -------------------------
# Detects Back Orifice traffic on the network.  Takes no arguments in 2.0.
#
# The Back Orifice detector uses Generator ID 105 and uses the
# following SIDS for that GID:
#  SID     Event description
# -----   ------------------
#   1       Back Orifice traffic detected

preprocessor bo

# telnet_decode: Telnet negotiation string normalizer
# ---------------------------------------------------
# This preprocessor "normalizes" telnet negotiation strings from
# telnet and ftp traffic.  It works in much the same way as the
# http_decode preprocessor, searching for traffic that breaks up
# the normal data stream of a protocol and replacing it with
# a normalized representation of that traffic so that the "content"
# pattern matching keyword can work without requiring modifications.
# This preprocessor requires no arguments.
# Portscan uses Generator ID 109 and does not generate any SID currently.

preprocessor telnet_decode

# Portscan: detect a variety of portscans
# ---------------------------------------
# portscan preprocessor by Patrick Mullen <p_mullen@linuxrc.net>
# This preprocessor detects UDP packets or TCP SYN packets going to
# four different ports in less than three seconds. "Stealth" TCP
# packets are always detected, regardless of these settings.
# Portscan uses Generator ID 100 and uses the following SIDS for that GID:
#  SID     Event description
# -----   ------------------
#   1       Portscan detect
#   2       Inter-scan info
#   3       Portscan End

preprocessor portscan: $HOME_NET 6 3 portscan.log

# Use portscan-ignorehosts to ignore TCP SYN and UDP "scans" from
# specific networks or hosts to reduce false alerts. It is typical
# to see many false alerts from DNS servers so you may want to
# add your DNS servers here. You can all multiple hosts/networks
# in a whitespace-delimited list.
#
preprocessor portscan-ignorehosts: 62.39.0.0/16
```

```
# arpspoof
#--------------------------------------
# Experimental ARP detection code from Jeff Nathan, detects ARP attacks,
# unicast ARP requests, and specific ARP mapping monitoring.  To make use
# of this preprocessor you must specify the IP and hardware address of hosts on # the same layer 2
segment as you.  Specify one host IP MAC combo per line.
# Also takes a "-unicast" option to turn on unicast ARP request detection.
# Arpspoof uses Generator ID 112 and uses the following SIDS for that GID:
#  SID     Event description
# -----   -------------------
#   1       Unicast ARP request
#   2       Etherframe ARP mismatch (src)
#   3       Etherframe ARP mismatch (dst)
#   4       ARP cache overwrite attack

#preprocessor arpspoof
#preprocessor arpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00

# Conversation
#------------------------------------------
# This preprocessor tracks conversations for tcp, udp and icmp traffic.  It
# is a prerequisite for running portscan2.
#
# allowed_ip_protcols 1 6 17
#      list of allowed ip protcols ( defaults to any )
#
# timeout [num]
#      conversation timeout ( defaults to 60 )
#
#
# max_conversations [num]
#      number of conversations to support at once (defaults to 65335)
#
#
# alert_odd_protocols
#      alert on protocols not listed in allowed_ip_protocols
#
#preprocessor conversation: allowed_ip_protocols all, timeout 60, max_conversations 3000
#
# Portscan2
#------------------------------------------
# Portscan 2, detect portscans in a new and exciting way.  You must enable
# spp_conversation in order to use this preprocessor.
#
# Available options:
#       scanners_max [num]
#       targets_max [num]
#       target_limit [num]
#       port_limit [num]
#       timeout [num]
#       log [logdir]
#
#preprocessor portscan2: scanners_max 256, targets_max 1024, target_limit 5, port_limit 20, timeout
60

# Too many false alerts from portscan2? Tone it down with
# portscan2-ignorehosts!
#
# A space delimited list of addresses in CIDR notation to ignore
```

```
#
# preprocessor portscan2-ignorehosts: 10.0.0.0/8 192.168.24.0/24
#

# Experimental Perf stats
# ----------------------
# No docs. Highly subject to change.
#
# preprocessor perfmonitor: console flow events time 10

##################################################################
# Step #3: Configure output plugins
#
# Uncomment and configure the output plugins you decide to use.
# General configuration for output plugins is of the form:
#
# output <name_of_plugin>: <configuration_options>
#
# alert_syslog: log alerts to syslog
# ----------------------------------
# Use one or more syslog facilities as arguments.  Win32 can also
# optionally specify a particular hostname/port.  Under Win32, the
# default hostname is '127.0.0.1', and the default port is 514.
#
# [Unix flavours should use this format...]
output alert_syslog: LOG_AUTH LOG_ALERT
#
# [Win32 can use any of these formats...]
# output alert_syslog: LOG_AUTH LOG_ALERT
# output alert_syslog: host=hostname, LOG_AUTH LOG_ALERT
# output alert_syslog: host=hostname:port, LOG_AUTH LOG_ALERT

# log_tcpdump: log packets in binary tcpdump format
# -------------------------------------------------
# The only argument is the output file name.
#
# output log_tcpdump: tcpdump.log

# alert_prelude: Reporting to Prelude Manager
#
# parameters:
#       async - enables asynchronous (multithreadedthreaded) reporting mode

#                       file from Prelude distribution
#
output alert_prelude: classification_file=/etc/snort/prelude-classification.config


# database: log to a variety of databases
# ---------------------------------------
# See the README.database file for more information about configuring
# and using this plugin.
#
# output database: log, mysql, user=root password=test dbname=db host=localhost
# output database: alert, postgresql, user=snort dbname=snort
# output database: log, unixodbc, user=snort dbname=snort
# output database: log, mssql, dbname=snort user=snort password=test

# unified: Snort unified binary format alerting and logging
# ---------------------------------------------------------
```

```
# The unified output plugin provides two new formats for logging
# and generating alerts from Snort, the "unified" format.  The
# unified format is a straight binary format for logging data
# out of Snort that is designed to be fast and efficient.  Used
# with barnyard (the new alert/log processor), most of the overhead
# for logging and alerting to various slow storage mechanisms
# such as databases or the network can now be avoided.
#
# Check out the spo_unified.h file for the data formats.
#
# Two arguments are supported.
#    filename - base filename to write to (current time_t is appended)
#    limit    - maximum size of spool file in MB (default: 128)
#
output alert_unified: filename snort.alert, limit 128
output log_unified: filename snort.log, limit 128

# You can optionally define new rule types and associate one or
# more output plugins specifically to that type.
#
# This example will create a type that will log to just tcpdump.
# ruletype suspicious
# {
#   type log
#   output log_tcpdump: suspicious.log
# }
#
# EXAMPLE RULE FOR SUSPICIOUS RULETYPE:
# suspicious $HOME_NET any -> $HOME_NET 6667 (msg:"Internal IRC Server";)
#
# This example will create a rule type that will log to syslog
# and a mysql database.
# ruletype redalert
# {
#   type alert
#   output alert_syslog: LOG_AUTH LOG_ALERT
#   output database: log, mysql, user=snort dbname=snort host=localhost
# }
#
# EXAMPLE RULE FOR REDALERT RULETYPE
# redalert tcp $HOME_NET any -> $EXTERNAL_NET 31337 \
#   (msg:"Someone is being LEET"; flags:A+;)

#
# Include classification & priority settings
#

include classification.config

#
# Include reference systems
#

include reference.config

#####################################################################
# Step #4: Customize your rule set
#
# Up to date snort rules are available at http://www.snort.org
#
```

```
# The snort web site has documentation about how to write your own
# custom snort rules.
#
# The rules included with this distribution generate alerts based on
# on suspicious activity. Depending on your network environment, your
# security policies, and what you consider to be suspicious, some of
# these rules may either generate false positives ore may be detecting
# activity you consider to be acceptable; therefore, you are
# encouraged to comment out rules that are not applicable in your
# environment.
#
# Note that using all of the rules at the same time may lead to
# serious packet loss on slower machines. YMMV, use with caution,
# standard disclaimers apply. :)
#
# The following individuals contributed many of rules in this
# distribution.
#
# Credits:
#   Ron Gula <rgula@securitywizards.com> of Network Security Wizards
#   Max Vision <vision@whitehats.com>
#   Martin Markgraf <martin@mail.du.gtn.com>
#   Fyodor Yarochkin <fygrave@tigerteam.net>
#   Nick Rogness <nick@rapidnet.com>
#   Jim Forster <jforster@rapidnet.com>
#   Scott McIntyre <scott@whoi.edu>
#   Tom Vandepoel <Tom.Vandepoel@ubizen.com>
#   Brian Caswell <bmc@snort.org>
#   Zeno <admin@cgisecurity.com>
#   Ryan Russell <ryan@securityfocus.com>
#
#==================================================
# Include all relevant rulesets here
#
# shellcode, policy, info, backdoor, and virus rulesets are
# disabled by default.  These require tuning and maintance.
# Please read the included specific file for more information.
#==================================================

include $RULE_PATH/local.rules
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/rpc.rules
include $RULE_PATH/rservices.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/tftp.rules

include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-coldfusion.rules
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-frontpage.rules
include $RULE_PATH/web-misc.rules
include $RULE_PATH/web-client.rules
include $RULE_PATH/web-php.rules
```

```
include $RULE_PATH/sql.rules
include $RULE_PATH/x11.rules
include $RULE_PATH/icmp.rules
include $RULE_PATH/netbios.rules
include $RULE_PATH/misc.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/oracle.rules
include $RULE_PATH/mysql.rules
include $RULE_PATH/snmp.rules

include $RULE_PATH/smtp.rules
include $RULE_PATH/imap.rules
include $RULE_PATH/pop2.rules
include $RULE_PATH/pop3.rules

include $RULE_PATH/nntp.rules
include $RULE_PATH/other-ids.rules
include $RULE_PATH/web-attacks.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/shellcode.rules
include $RULE_PATH/policy.rules
include $RULE_PATH/porn.rules
include $RULE_PATH/info.rules
#include $RULE_PATH/icmp-info.rules
include $RULE_PATH/virus.rules
include $RULE_PATH/chat.rules
include $RULE_PATH/multimedia.rules
include $RULE_PATH/p2p.rules
include $RULE_PATH/experimental.rules
```

9.2.1.2 snort.conf en el detector situado en la DMZ

El fichero de configuración snort.conf de la máquina ubicada en la DMZ es igual que el anterior salvo:

- La definición de las variables de red que están definidas para reflejar las direcciones de esta VLAN.

- El preprocesador *portscan* está desactivado.

9.2.2 Fichero de configuración de Barnyard : Barnyard.conf

El fichero barnyard.conf presenta la misma configuración en ambos detectores salvo las variables referentes a las direcciones IP de cada sistema.

```
#--------------------------------------------------------------
#   http://www.snort.org    Barnyard 0.1.0 configuration file
#          Contact: snort-barnyard@lists.sourceforge.net
#--------------------------------------------------------------
# $Id: barnyard.conf,v 1.1.1.1 2002/12/02 20:51:35 andrewbaker Exp $
```

```
##################################################
# Currently you want to do two things in here: turn on
# available data processors and turn on output plugins.
# The data processors (dp's) and output plugin's (op's)
# automatically associate with each other by type and
# are automatically selected at run time depending on
# the type of file you try to load.
##################################################

# Step 0: configuration declarations
# To keep from having a commandline that uses every letter in the alphabet
# most configuration options are set here

# enable daemon mode
# config daemon

# use localtime instead of UTC (*not* recommended because of timewarps)
config localtime

# set the hostname (currently only used for the acid db output plugin)
config hostname: snort

# set the interface name (currently only used for the acid db output plugin)
config interface: eth1

# set the filter (currently only used for the acid db output plugin)
config filter: not port 22

# Step 1: setup the data processors

# dp_alert
# -------------------------
# The dp_alert data processor is capable of reading the alert (event) format
# generated by Snort's spo_unified plug-in.  It is used with output plug-ins
# that support the "alert" input type.  This plug-in takes no arguments.
processor dp_alert


# dp_log
# -------------------------
# The dp_log data processor is capable of reading the log format generated
# by Snort's spo_unified plug-in.  It is used with output plug-ins
# that support the "log" input type.  This plug-in takes no arguments.
processor dp_log


# dp_stream_stat
# -------------------------
# The dp_stream_stat data processor is capable of reading the binary output
# generated by Snort's spp_stream4 plug-in.  It is used with output plug-ins
# that support the "stream_stat" input type.  This plug-in takes no arguments.
processor dp_stream_stat


# Step 2: setup the output plugins

# alert_fast
#-------------------------
# Converts data from the dp_alert plugin into an approximation of Snort's
# "fast alert" mode.  Argument: <filename>
```

```
output alert_fast: /var/log/snort/alertfast.log

# log_dump
#----------------------------
# Converts data from the dp_log plugin into an approximation of Snort's
# "ASCII packet dump" mode.  Argument: <filename>

output log_dump: /var/log/snort/dump.log

# alert_html (experimental)
#--------------------------
# Creates a series of html pages about recent alerts
# Arguments:
#   [webroot] - base directory for storing the html pages
#
# Example:
#   output alert_html:  /var/www/htdocs/op_alert_html
#   output alert_html:  /var/www/htdocs/op_alert_html

# alert_csv (experimental)
#--------------------------
# Creates a CSV output file of alerts (optionally using a user specified format)
# Arguments:  filepath [format]
#
# The format is a comma-seperated list of fields to output (no spaces allowed)
# The available fields are:
#   sig_gen         - signature generator
#   sig_id          - signature id
#   sig_rev         - signatrue revision
#   sid             - SID triplet
#   class           - class id
#   classname       - textual name of class
#   priority        - priority id
#   event_id        - event id
#   event_reference - event reference
#   ref_tv_sec      - reference seconds
#   ref_tv_usec     - reference microseconds
#   tv_sec          - event seconds
#   tv_usec         - event microseconds
#   timestamp       - prettified timestamp (2001-01-01 01:02:03) in UTC
#   src             - src address as a u_int32_t
#   srcip           - src address as a dotted quad
#   dst             - dst address as a u_int32_t
#   dstip           - dst address as a dotted quad
#   sport_itype     - source port or ICMP type (or 0)
#   sport           - source port (if UDP or TCP)
#   itype           - ICMP type (if ICMP)
#   dport_icode     - dest port or ICMP code (or 0)
#   dport           - dest port
#   icode           - ICMP code (if ICMP)
#   proto           - protocol number
#   protoname       - protocol name
#   flags           - flags from UnifiedAlertRecord
#   msg             - message text
#   hostname        - hostname (from barnyard.conf)
#   interface       - interface (from barnyard.conf)
#
# Examples:
#   output alert_csv: /var/log/snort/csv.out
```

```
#   output alert_csv: /var/log/snort/csv.out
timestamp,msg,srcip,sport,dstip,dport,protoname,itype,icode
#   output alert_csv: csv.out  timestamp,msg,srcip,sport,dstip,dport,protoname,itype,icode


# alert_syslog
#------------------------------
# Converts data from the alert stream into an approximation of Snort's
# syslog alert output plugin.  Same arguments as the output plugin in snort.

output alert_syslog: LOG_AUTH LOG_ALERT

# log_pcap
#------------------------------
# Converts data from the dp_log plugin into standard pcap format
# Argument: <filename>

#output log_pcap: /var/log/snort/logpcap

# acid_db
#--------------------------------
# Available as both a log and alert output plugin.  Used to output data into
# the db schema used by ACID
# Arguments:
#      $db_flavor          - what flavor of database (ie, mysql)
#      sensor_id $sensor_id - integer sensor id to insert data as
#      database $database   - name of the database
#      server $server       - server the database is located on
#      user $user           - username to connect to the database as
#      password $password   - password for database authentication
#output alert_acid_db: mysql, database snort, server 192.168.50.170, user snort, password duende,
detail full
output log_acid_db: mysql, database snort, server 192.168.50.170, user snort, password duende, detail
full
```

9.2.3 Ficheros de configuración de Prelude

Los ficheros de configuración de Prelude serán iguales para ambos sistemas (salvo las variables que hacen referencia a las direcciones IP de cada sistema). El fichero *prelude-manager.conf* sólo se encuentra en la máquina situada en la DMZ donde está instalado el servidor IDS, y, por tanto, el manager de Prelude-IDS.

9.2.3.1 prelude-manager.conf

```
[Prelude Manager]

# Address where the sensors serber is listening on.
# if value is 127.0.0.1 (or is resolved as being 127.0.0.1),
# it mean the Manager server will be listening via a local (UNIX)
# socket.
#
# format : address:port
#
```

```
sensors-srvr = 192.168.25.101;
#sensors-srvr = 192.168.149.152;

# Address where the administrative server is listening on.
# if value is "unix", it mean the report server is listening
# on the same machine via a local (UNIX) socket.
#
# format : address:port
#
# admin-srvr = 0.0.0.0:5555;


# If you want the message caught by this manager to be relayed.
# You can use boolean AND and OR to make the rule.
#
# relay-manager = x.x.x.x || y.y.y.y && z.z.z.z
#
# This mean the emission should occur on x.x.x.x or, if it fail,
# on y.y.y.y and z.z.z.z (if one of the two host in the AND fail,
# the emission will be considered as failed involving saving the
# message locally).


####################################
# Here start plugins configuration #
####################################

[MySQL]

# Host the database is listening on.
dbhost = localhost;

# Name of the database.
dbname = prelude;

# Username to be used to connect the database.
dbuser = prelude;

# Password used to connect the database.
dbpass = motdepas;



#
# The Textmod plugin allow to report alert as text
# in a file. Or to dump theses alert to stderr.
#
# The default logfile for this plugin is /var/log/prelude.log
#

[TextMod]
#
# Tell Textmod to output to stderr
# stderr;
#

logfile = /var/log/prelude.log;
```

```
#
# The Xmlmod plugin allow to report alert as IDMEF XML
# in a file. Or to dump theses alert to stderr.
#
# The default logfile for this plugin is /var/log/prelude-xml.log
#

[XmlMod]
#
# Tell Xmlmod to output to stderr
# stderr;
#
#
# Tell Xmlmod to disable output file buffering.
# This will prevent XML alerts to be truncated and thus make real-time
# parsing easier.
#
# disable-buffering;
#
#
# Tell Xmlmod to check generated XML against IDMEF DTD
# check-dtd;
#

logfile = /var/log/prelude-xml.log;



# [Debug]
#
# Print the value of each element.
# verbose;
#
# Be aggressive, print strings even if consistency checks fail
# (may lead to crash).
# aggressive;
#
# Use wide format for lists.
# wide-format;
```

9.2.3.2 sensors-default.conf

```
# This is the default configuration for sensors that use libprelude.
# Entry in this configuration file are overriden by entry directly
# provided by the sensor configuration file.


# Try to connect on a Manager listening on 127.0.0.1.
#
# manager-addr = x.x.x.x:port || y.y.y.y && z.z.z.z
#
# This mean the emission should occur on x.x.x.x:port or, if it fail,
# on y.y.y.y and z.z.z.z (if one of the two host in the AND fail,
# the emission will be considered as failed involving saving the
# message locally).

manager-addr = 192.168.50.150;
```

```
#
# Optionnal data gathered with the alert.
#
# node-name = Name of the equipment;
# node-location = Location of the equipment;
# node-category = Type of node:
#
# unknown          Domain unknown or not relevant
# ads              Windows 2000 Advanced Directory Services
# afs              Andrew File System (Transarc)
# coda             Coda Distributed File System
# dfs              Distributed File System (IBM)
# dns              Domain Name System
# hosts            Local hosts file
# kerberos         Kerberos realm
# nds              Novell Directory Services
# nis              Network Information Services (Sun)
# nisplus          Network Information Services Plus (Sun)
# nt               Windows NT domain
# wfw              Windows for Workgroups


# Address contained by this Node,
# You may have several address.
#
# [Node Address]
#
# address = Address of the equipment;
# netmask = Netmask for this address;
# vlan-name = Name of the Virtual LAN to which the address belongs;
# vlan-num = Number of the Virtual LAN to which the address belongs;
#
# category = Type of address represented;
#
# Permitted values for category are (default to unknown) :
#
# unknown          Address type unknown
# atm              Asynchronous Transfer Mode network address
# e-mail           Electronic mail address (RFC 822)
# lotus-notes      Lotus Notes e-mail address
# mac              Media Access Control (MAC) address
# sna              IBM Shared Network Architecture (SNA) address
# vm               IBM VM ("PROFS") e-mail address
# ipv4-addr        IPv4 host address in dotted-decimal notation (a.b.c.d)
# ipv4-addr-hex    IPv4 host address in hexadecimal notation
# ipv4-net         IPv4 network address in dotted-decimal notation, slash, significant bits (a.b.c.d/nn)
# ipv4-net-mask    IPv4 network address in dotted-decimal notation, slash, network mask in dotted-
decimal notation (a.b.c.d/w.x.y.z)
# ipv6-addr        IPv6 host address
# ipv6-addr-hex    IPv6 host address in hexadecimal notation
# ipv6-net         IPv6 network address, slash, significant bits
# ipv6-net-mask    IPv6 network address, slash, network mask
```

9.2.3.3 prelude-nids.conf

```
#############################################
# Configuration for the Prelude NIDS Sensor  #
#############################################

[Prelude NIDS]

# Address where the Prelude Manager Server is listening on.
# if value is "127.0.0.1", the connection will occur throught
# an UNIX socket.
#
# This entry is disabled. The default is to use the entry
# located in sensors-default.conf... You may overwrite the
# default address for this sensor by uncommenting this entry.
#
manager-addr = 192.168.50.150;


# Set this entry if you want Prelude NIDS to use a specific user.
#
# user = prelude;



[Tcp-Reasm]

#
# TCP stream reassembly option
#
# Only analyse TCP packet that are part of a stream,
# this defeat stick/snot against TCP signatures.
#
# statefull-only;

#
# Only reassemble TCP data sent by the client (default).
#
# client-only;

#
# Only reassemble TCP data sent by the server.
#
# server-only;

#
# Reassemble TCP data sent by client and server.
#
 both;

#
# Only reassemble data to specific port (default is to reassemble everything).
#
# If this option is used with the statefull-only option, packet that are not
# going to theses specified port will be analyzed anyway.
#
# port-list = 1 2 3 4;


####################################
```

```
# Here start plugins configuration #
####################################

[SnortRules]

ruleset=/usr/local/etc/prelude-nids/ruleset/prelude.rules;


[ScanDetect]

# Number of connection attempt to get from the same
# host and targeted on different port before the scan
# detection plugin issue an alert.
#
high-port-cnx-count = 50;
low-port-cnx-count = 5;


# Window of time without getting any activity the scan
# detection plugin should wait before issuing an alert
# for a given host.
#
cnx-ttl = 60;



# [Shellcode]
#
# This plugin allow for polymorphic shellcode detection.
# It may consume a lot of CPU time, so it's disabled by
# default. Uncomment the section name to enable it, or
# specify --shellcode on the command line.

nops_raise_alert = 60;

#
# If a port-list is specified, the Shellcode plugin
# will only analyse data going to theses port (when
# the protocol used have have dst port).
#
# port-list = 1 2 3 4;



# [Debug]
#
# This plugin issue an alert for each packet.
# Carefull to the loging activity it generate.


#[HttpMod]
#
# Normalize HTTP request.
# The "codepage-file" option contains the name of the file containing
# Unicode to ASCII convertion tables for WIN32 machines.
#
# The "codepage-number" option is the codepage number your WIN32 servers use.
#
#
# end-on-param:
```

```
# Stop parsing the URL when we meet a parameter.
#
# double-encode:
# Check for encoded '%' character.
#
# max-whitespace:
# Maximum number of whitespace allowed before URL begin.
#
# flip-backslash:
# Change '\' to '/' when parsing URL.
#

double-encode;
flip-backslash;
max-whitespace = 10;
codepage-file = /usr/local/etc/prelude-nids/unitable.txt;
codepage-number = 437;


port-list = 80 8080;


[RpcMod]
#
# Decode RPC traffic, Also provide the RPC rule key.
#
port-list = 111 32771;


[TelnetMod]
#
# Normalize telnet negotiation character
#
port-list = 23 21;


#[ArpSpoof]
#
# Search anomaly in ARP request.
#
# The "directed" option will result in a warn each time an ARP
# request is sent to an address other than the broadcast address.
#
# directed;
# arpwatch=<ip> <macaddr>;
```

9.3 ANEXO III : FICHERO DE FILTROS : local.rules

9.3.1 local.rules en el detector situado en la DMZ2

```
# $Id: local.rules,v 1.5 2001/12/19 18:40:05 cazz Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.

pass tcp 192.168.25.95 any -> 192.168.25.90 9935
pass tcp 192.168.25.200 any -> any any
pass udp 192.168.25.200 any -> any any
pass icmp 192.168.25.200 any -> any any
pass tcp 199.107.65.177/32 any -> any any
pass tcp 192.168.25.10 any -> 192.168.25.94 110 (msg:"POP3 PASS overflow attempt";
flow:to_server,established; content:"PASS"; nocase; isdataat:50,relative;
pcre:"/^PASS\s[^\n]{50}/smi"; reference:cve,CAN-1999-1511; reference:nessus,10325;
classtype:attempted-admin; sid:1634; rev:10;)
pass tcp 192.168.25.10 any -> 192.168.25.94 110 (msg:"POP3 TOP overflow attempt";
flow:to_server,established; content:"TOP"; nocase; isdataat:10,relative; pcre:"/^TOP\s[^\n]{10}/smi";
classtype:attempted-admin; sid:2109; rev:3;)
pass tcp 192.168.25.10 any -> 192.168.25.94 110 (msg:"POP3 DELE overflow attempt";
flow:to_server,established; content:"DELE"; nocase; isdataat:10,relative;
pcre:"/^DELE\s[^\n]{10}/smi"; classtype:attempted-admin; sid:2111; rev:3;)
pass tcp 192.168.25.10 any -> 192.168.25.94 110 (msg:"POP3 STAT overflow attempt";
flow:to_server,established; content:"STAT"; nocase; isdataat:10,relative;
pcre:"/^STAT\s[^\n]{10}/smi"; classtype:attempted-admin; sid:2110; rev:3;)
pass tcp 192.168.25.10 any -> 192.168.25.94 110 (msg:"POP3 USER overflow attempt";
flow:to_server,established; content:"USER"; nocase; isdataat:50,relative;
pcre:"/^USER\s[^\n]{50,}/smi"; reference:bugtraq,789; reference:cve,CVE-1999-0494;
reference:nessus,10311; classtype:attempted-admin; sid:1866; rev:9;)
pass tcp 192.168.25.25 any -> 192.168.25.92 3128 (msg:"P2P GNUTella GET";
flow:to_server,established; content:"GET "; depth:4; classtype:policy-violation; sid:1432; rev:5;)
pass tcp 192.168.25.41 any -> 192.168.25.92 3128 (msg:"P2P GNUTella GET";
flow:to_server,established; content:"GET "; depth:4; classtype:policy-violation; sid:1432; rev:5;)
pass tcp 192.168.25.184 any -> 192.168.25.92 3128 (msg:"P2P GNUTella GET";
flow:to_server,established; content:"GET "; depth:4; classtype:policy-violation; sid:1432; rev:5;)
pass tcp 192.168.25.10 any -> 192.168.25.92 3128 (msg:"P2P GNUTella GET";
flow:to_server,established; content:"GET "; depth:4; classtype:policy-violation; sid:1432; rev:5;)
pass tcp 192.168.25.10 any -> 192.168.25.90 9935 (msg:"P2P GNUTella GET";
flow:to_server,established; content:"GET "; depth:4; classtype:policy-violation; sid:1432; rev:5;)
pass tcp 192.168.25.25 any -> 192.168.25.92 3128 (msg:"SCAN Squid Proxy attempt"; flags:S,12;
flow:stateless; classtype:attempted-recon; sid:618; rev:8;)
pass tcp 192.168.25.41 any -> 192.168.25.92 3128 (msg:"SCAN Squid Proxy attempt"; flags:S,12;
flow:stateless; classtype:attempted-recon; sid:618; rev:8;)
pass tcp 192.168.25.184 any -> 192.168.25.92 3128 (msg:"SCAN Squid Proxy attempt"; flags:S,12;
flow:stateless; classtype:attempted-recon; sid:618; rev:8;)
pass tcp 192.168.25.10 any -> 192.168.25.92 3128 (msg:"SCAN Squid Proxy attempt"; flags:S,12;
flow:stateless; classtype:attempted-recon; sid:618; rev:8;)
```

9.3.2 local.rules en el detector situado en la DMZ

```
# $Id: local.rules,v 1.5 2001/12/19 18:40:05 cazz Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.

pass tcp 199.107.65.177/32 any -> any any
```

9.4 ANEXO IV: PLIEGO DE CONDICIONES

Para la realización del proyecto no se ha exigido ningún tipo de condición técnica específica. De hecho, uno de los objetivos del proyecto es el estudio de las aplicaciones más apropiadas para lograr la puesta en marcha en un entorno real de un sistema de detección de intrusiones. No obstante, la empresa ha impuesto dos requisitos:

- Utilización de software y aplicaciones de distribución gratuita.

- Aprovechamiento de los recursos ya existentes en la empresa en cuanto a dispositivos o máquinas se refiere.

El primero de los requisitos limita las posibilidades de búsqueda y estudio a los IDS de código libre y distribución gratuita. Toda aplicación necesaria para el funcionamiento del IDS o para mejorar su rendimiento y prestaciones deberá ser también de distribución gratuita.

En la empresa se dispone de los siguientes materiales para la instalación del IDS y el desarrollo del proyecto:

- Servidor IBM Intel Xeon 2,8G, 36,4 G de disco duro con disco de copia de seguridad de la misma capacidad, 512 M de memoria RAM y dos tarjetas de red ethernet.

- Procesador intel celeron 800MHz, 256 Mb de memoria RAM y 8 Gb de disco duro con una tarjeta de red ethernet.

- 2 procesadores pentium II 333MHz, 64 Mb de memoria RAM y 3 Gb de disco duro con una tarjeta de red ethernet.

- Tarjetas de red ethernet y cables.

9.5 ANEXO V: ORGANIZACIÓN DEL PROYECTO, RECURSOS

La realización del proyecto se ha llevado cabo por una persona con contrato de prácticas, aunque ha contado con la supervisión de otra. La distribución del trabajo realizado por la persona encargada de desarrollarlo es la mostrada en la siguiente tabla:

	Nombre de tarea	Duración	Comienzo	Fin
1	Búsqueda de información	10 días	lun 24/11/03	vie 05/12/03
2	Estudio de diferente sofware IDS	5 días	lun 08/12/03	vie 12/12/03
3	Estudio de la instalación de un IDS y de otras aplicaciones necesarias	5 días	lun 15/12/03	vie 19/12/03
4	Montaje de maqueta e instalación del sistema operativo RedHat Linux	1 día?	lun 05/01/04	lun 05/01/04
5	Instalación del IDS en maqueta: Snort, Prelude-IDS y otras aplicaciones requeridas	5 días	mar 06/01/04	lun 12/01/04
6	Selección e instalación de herramientas de simulación de ataques	5 días	mar 13/01/04	lun 19/01/04
7	Test y configuración de los IDS en maqueta	5 días	mar 20/01/04	lun 26/01/04
8	Estudio de la red del ARSOÉ de Trélazé (estructura, servidores, etc.)	2 días	mar 27/01/04	mié 28/01/04
9	Estudio de una solución IDS (nº detectores y ubicación, servidor IDS, etc.)	8 días	jue 29/01/04	lun 09/02/04
10	Instalación del servidor IDS y un detector en una máquina	2 días	mar 10/02/04	mié 11/02/04
11	Conexión de un detector en la zona DMZ2	1 día?	jue 12/02/04	jue 12/02/04
12	Conexión de la máquina servidor+detector en la DMZ	1 día?	vie 13/02/04	vie 13/02/04
13	Establecer comunicación distante del detector en DMZ2 con el servidor en DMZ	2 días	lun 16/02/04	mar 17/02/04
14	Estudio e instalación de aplicaciones para comunicaciones seguras entre componentes	3 días	mié 18/02/04	vie 20/02/04
15	Observación de alertas obtenidas. Configuración y optimización del IDS	65 días	lun 23/02/04	vie 21/05/04
16	Instalación de aplicación para envío de informes vía mail	5 días	lun 12/04/04	vie 16/04/04
17	preparación de la documentación	20 días	lun 19/04/04	vie 14/05/04
18	transferencia de competencias	5 días	lun 17/05/04	vie 21/05/04

En total se ha dedicado 120 días laborables (8 horas diarias) a la realización del proyecto.

Por otro lado, no ha sido necesaria la compra de equipos, materiales o software relacionados con el proyecto, sino que se han aprovechado los recursos materiales ya existentes en la empresa. Esto implica que a la hora de calcular el presupuesto del proyecto, éste coincida prácticamente con el salario total de la persona contratada en prácticas (3000 €).

10 BIBLIOGRAFÍA

Libros y manuales de texto

- Jack Koziol. *Snort 2*. CampusPress 2003.
- Jay Beale, James C. Foster, Jeffrey Posluns & Brian Caswell. *Snort 2.0 Intrusion Detection*. Syngress Publishing, 2003.
- Anónimo. *Sécurité Optimale. Ecrit par un hacker*. CampusPress 1999.
- Anónimo. *Maximun Linux Security*. Sams Publishing 2001.
- Craig Hunt. *TCP/IP Administration de réseau*. Editions O'Reilly. Paris 1998.
- D. Brent Chapman and Elisabeth D. Zwicky. *Firewalls*. Editions O'Reilly 1996.
- Simson Garfinkel & Gene Spafford. *Practical UNIX & Internet Security*. O'Reilly, second edition, April 1996.
- Antonio Villalón Huerta. *Seguridad en Unix y redes*. Versión 2.1. Julio 2002. http://es.tldp.org/Manuales-LuCAS/doc-unixsec/unixsec-html/unixsec.html
- Emilio José Mira Alfaro. *Implantación de un Sistema de Detección de Intrusiones en la Universidad de Valencia*. Proyecto fin de carrera. Universidad de Valencia, 2002.
- Gabriel Verdejo Álvarez. *Capitulo 3: Seguridad en Redes IP: IDS*.

Publicaciones y artículos

- Patrick Harper. *Snort Install Manual. Snort, Apache, PHP, MySQL and ACID Install on RH9.0*. June 2003.
- Keith Tokash. *How to Setup and Secure Snort, MySQL and ACID on FreeBSD 4.7 Release*. January 2003.

- Detmar Liesen. *Requirements for Enterprise-Wide Scaling Intrusion Detection Products. A Criteria Catalog for IT Executives, IDS Users and Vendors*. 2002.

- Brian Laing. *How To Guide-Implementing a Network Based Intrusion Detection System*. Internet Security Systems, 2000.

- Javier Fernández-Sanguino Peña. *Sistemas de detección de intrusos: carencias actuales y nuevas tecnologías*. SIC, Junio 2002.

- Frédéric Bordage. *Détection d'intrusions: prévenir plutôt que guérir*. Décision Micro & Réseau, mai 2003.

- Gonzalo Álvarez y Luis Cornide. *Anatomía de un ataque hacker. La metodología de ataque seguida por los hackers explicada paso a paso.* Revista iWorld, Junio 2001.

- Ramón Aguilera Santiago y Bernardo Hernández González. *I Curso de Especialista Universitario en Seguridad Informática*. 2002.

- Pravin Kothari. *Intrusion Detection Interoperability and Standardization*. GSEC Assignment Version 1.3. February 2002.

- Krzysztof Zaraska. *Prelude IDS: Current State and Development Perspectives*. March 2004.

- Tim Buchheim, Michael Erlinger, Ben Feinstein, Greg Matthews, Roy Pollock, Joseph Betser and Andy Walther. *Implementing the Intrusion Detection Exchange Protocol*.

- Anup K. Ghosh and Aaron Schwartzbard. *A Study in using Neural Networks for Anomaly and Misuse Detection.* Proceedings of the 8th USENIX Security Symposium. Washington, August 1999.

- Hervé Debar & Benjamín Morin. *Intrusion Detection, Evaluation of the diagnostic capabilities of commercial intrusion detection systems*.

- John Reuning. *Applying Information Retrieval Techniques to Event Log Analysis for Intrusion Detection*. January 2004.

- Jed Pickel & Roman Danyliw. *Enabling Automated Detection of Security Events that affect Multiple Administrative Domains*. 2000.

- Marion Bates & William Stearns. *Setting up automatic alerting in your Unix environment.* January 2001.

- *Escaneando con Nmap. Técnicas de scan de puertos*. Maty & Asociados, 2002. http://www.nautopia.net/archives/es/varios_redes/nmap/escaneando_con_nmap.php

- *Técnicas básicas de los Nukes*. Panda Software. http://www.zonavirus.com/Detalle_Articulo.asp?Articulo=86.

- Nicolas Six. *Les IDS : panorama du marché*. Julio 2002. http://solutions.journaldunet.com/0207/020716_sonde_intrusion_4.shtml

- Victor E. Cappuccio. *Sistemas de detección de intrusos*. Enero 2002. www.vcappuccio.freeservers.com

- *Guía breve de Tripwire*. http://es.tldp.org/Tutoriales/GUIA_TRIPWIRE/guia_tripwire.html

- *Tipos de ataque*. http://webs.ono.com/usr026/Agika2/3internet/ataques.htm

- Samuel Dralet. *Détection et tolérance d'intrusions.* Mayo 2004. *http://www.miscmag.com/articles/index.php3?page=108*

- Federico Castanedo Sotela. *Ports Scanners.* http://www.terra.es/personal2/federico2/portscan.pdf

Enlaces de Internet

- IDS de código libre :
  - http://www.snort.org/
  - http://www.prelude-ids.org/
- IDS comerciales:
  - http://www.enterasys.com/products/ids/
  - http://www.cisco.com/en/US/products/hw/vpndevc/ps4077/ps2110/

- http://www.cisco.com/warp/public/cc/pd/sqsw/sqidsz/index.shtml
- http://www.enterasys.com/products/ids/DSHSS-xxx/
- http://www.nwfusion.com/reviews/1004revu.html
- http://www.iss.net/products_services/enterprise_protection/rsnetwork/sensor.php
- http://www.iss.net/products_services/enterprise_protection/rsserver/protector_server.php
- http://www.mimestar.com/products/

- Enlaces de seguridad:
  - http://www.cert.org/
  - http://www.whitehats.com
  - http://www.securityfocus.com
  - http://cve.mitre.org
  - http://icat.nist.gov