Proyecto Final de Carrera

Implantación de un Sistema de Detección de Intrusos en la Universidad de Valencia

Ingeniería Informática
Universidad de Valencia

Emilio José Mira Alfaro
Tutor: Rogelio Montañana

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Problemática

La cantidad de intentos de accesos no autorizados a la información que existe en Internet ha crecido durante estos últimos años. Según el Computer Security Institute, un 70 % de las organizaciones anunciaron al menos un incidente de seguridad durante 2000, frente a un 42 % anunciado en 1996. La mayoría de los expertos piensa que estos números están muy por debajo de la tasa real, puesto que muchas organizaciones evitan dar a conocer sus incidentes y muchas otras ni siquiera los detectan.

Muchas compañías, normalmente por motivos de coste, han migrado información clave a Internet, exponiéndola hacia el exterior. Además, para comodidad de los trabajadores que solicitan teletrabajo, las compañías han tenido que "abrir sus puertas" para permitir la conexión a la intranet de la oficina desde casa. Desafortunadamente, cuando los atacantes comprometen los sistemas de entrada, ellos también tienen acceso a datos de la organización. La incorporación de cortafuegos y redes privadas virtuales (VPNs) para permitir de forma segura que los usuarios externos se puedan comunicar con la intranet de la organización ha aliviado un poco el problema; un cortafuegos con una política correcta puede minimizar el que muchas redes queden expuestas. Sin embargo, los atacantes están evolucionando y aparecen nuevas técnicas como los Troyanos, gusanos y escaneos silenciosos que atraviesan los cortafuegos mediante protocolos permitidos como HTTP, ICMP o DNS. Los atacantes buscan vulnerabilidades en los pocos servicios que el cortafuegos permite y enmascaran sus ataques dentro de estos protocolos, quedando la red expuesta de nuevo.

La cantidad de mensajes publicados en listas de vulnerabilidades como BUGTRAQ ha aumentado de forma exagerada durante los últimos años. Las vulnerabilidades no solo afectan a sistemas tradicionalmente seguros, sino que afectan incluso a sistemas de seguridad: cortafuegos y sistemas de detección de intrusos o IDSs (Intrusion Detection Systems). Esto se debe en parte a un crecimiento del número de auditorías que las empresas de software aplican a sus productos y por el aumento de interés en el campo de la seguridad por parte de los profesionales de la informática.

Aunque muchos escaneos de red y técnicas de ataques son conocidos desde hace varias décadas, no ha sido hasta hace poco tiempo que las herramientas para producir análisis sofisticados a redes han llegado a estar disponibles para las masas. Por ejemplo, el escaneo de puertos que estaba públicamente disponible en los 90 consistía simplemente en intentos de conexiones hacia todos los puertos potencialmente activos. Sin embargo, los escáneres modernos incluyen modos de identificación del

sistema operativo , pueden escanear rangos de direcciones IP enteros de forma aleatoria y oculta, e incluso enviar “escaneos señuelo” para hacer más difícil al objetivo el identificar quien es realmente el origen de la agresión.

Los atacantes también intentan atacar a los sistemas de detección de intrusos, ya sea saturándolos de tráfico a analizar o bien mediante herramientas que les proporcionan información falsa de lo que pasa por la red. El hecho de que los atacantes estén incorporando técnicas anti-IDS a su arsenal, los coloca en situación más ventajosa frente a organizaciones que ni siquiera disponen de un IDS.

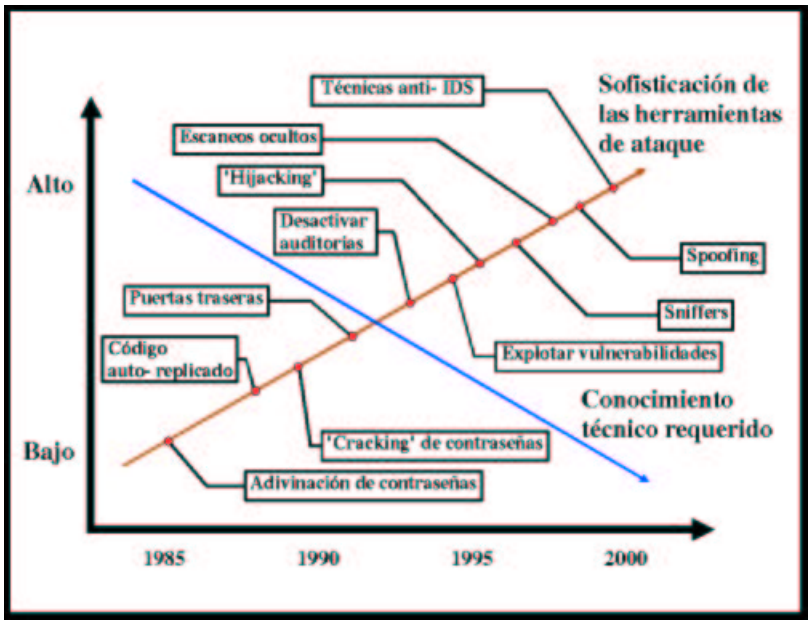

Figura 1.1: Crecimiento de la complejidad de los ataques en relación al conocimiento técnico requerido para su uso [7].

Por otra parte, uno de los problemas principales de la detección de intrusiones corresponde al tiempo que transcurre desde que se descubre una vulnerabilidad y se comenta en diversas listas de seguridad, hasta que las empresas de software comienzan a preparar y desarrollar parches. Durante ese tiempo es probable que aparezca un programa exploit que haga uso de esa vulnerabilidad para irrumpir en un sistema. Hasta que el exploit no es difundido en Internet (lo cual no quiere decir que no sea utilizado), los analista de seguridad no pueden elaborar reglas para detectarlo e incorporarlas a sus IDSs. Se hace necesario, por tanto, un sistema que disminuya este tiempo y detecte automáticamente la existencia de nuevos ataques de forma fiable, que resista las últimas técnicas anti-IDS y que además sea escalable.

1.2. Motivación

La idea de instalar un IDS en la Universidad de Valencia surgió en Octubre de 2001 de la mano de RedIRIS, la red académica española que da acceso a Internet a la mayoría de universidades de España.

Se quería lanzar una experiencia piloto basada en la implantación de un IDS en el troncal de un nodo regional que no tuviese tráfico excesivamente grande pero que fuese significativo al mismo tiempo.

Quedaban por tanto excluidos los nodos de Madrid y Cataluña, siendo los candidatos los de Andalucía y la Comunidad Valenciana.

Por unas causas u otras, las universidades españolas carecen en general de mecanismos de seguridad que hagan frente al aumento del número de ataques que se producen en Internet. Es más, es inviable la instalación de firewalls que limiten el acceso por defecto dejando abiertos solo los servicios necesarios, por lo que su configuración sería la de 'permitir por defecto' y filtrar solo los servicios problemáticos por razones administrativas (correo y web, por ejemplo). Esta tarea se puede realizar en el propio router de entrada y no necesita de hardware específico como el firewall.

Una solución que se le puede dar al problema de la falta de seguridad y que encaja con lo anterior es la instalación de sistemas pasivos que alerten a los administradores en el momento en el que se produzca un ataque. El administrador será conocedor del ataque y dependiendo de la gravedad de éste podrá decidir si avisa al CERT asociado (en nuestro caso CERT-ES) y/o al responsable de la red origen del ataque, o no.

Para facilitar esta tarea a los administradores, RedIRIS propuso el desarrollo de un programa que automatizase el anuncio de alertas al CERT por medio del correo electrónico. El sistema de detección de intrusos se encargaría de elaborar cada día un informe con todas las alarmas producidas durante ese día y enviarlo por correo electrónico al administrador de seguridad. El administrador respondería ese e-mail a la máquina con información adicional de qué alertas debían ser enviadas al CERT y/o al responsable de la red origen del ataque, y cuales no. Por último, la máquina sería la encargada de enviar las alarmas seleccionadas a la dirección de correo adecuada.

RedIRIS proporcionó el hardware necesario consistente en una máquina VA-Linux modelo 2230 FullOn y una tarjeta ATM de FORE Systems modelo PCA-200E.

1.3. Introducción a los Sistemas de Detección de Intrusos

1.3.1. ¿Qué es un Sistema de Detección de Intrusos?

Un Sistema de Detección de Intrusos o IDS (Intrusion Detection System) es una herramienta de seguridad encargada de monitorizar los eventos que ocurren en un sistema informático en busca de intentos de intrusión.

Definimos intento de intrusión como cualquier intento de comprometer la confidencialidad, integridad, disponibilidad o evitar los mecanismos de seguridad de una computadora o red. Las intrusiones se pueden producir de varias formas: atacantes que acceden a los sistemas desde Internet, usuarios autorizados del sistema que intentan ganar privilegios adicionales para los cuales no están autorizados y usuarios autorizados que hacen un mal uso de los privilegios que se les han asignado.

1.3.2. ¿Por qué utilizar un IDS?

La detección de intrusiones permite a las organizaciones proteger sus sistemas de las amenazas que aparecen al incrementar la conectividad en red y la dependencia que tenemos hacia los sistemas de información.

Los IDSs han ganado aceptación como una pieza fundamental en la infraestructura de seguridad de la organización. Hay varias razones para adquirir y usar un IDS:

Prevenir problemas al disuadir a individuos hostiles.

Al incrementar la posibilidad de descubrir y castigar a los atacantes, el comportamiento de algunos cambiará de forma que muchos ataques no llegarán a producirse. Esto también puede jugar en nuestra contra, puesto que la presencia de un sistema de seguridad sofisticado puede hacer crecer la curiosidad del atacante.

Detectar ataques y otras violaciones de la seguridad que no son prevenidas por otras medidas de protección.

Los atacantes, usando técnicas ampliamente conocidas, pueden conseguir accesos no autorizados a muchos sistemas, especialmente a aquellos conectados a redes públicas. Esto a menudo ocurre cuando vulnerabilidades conocidas no son corregidas.

Aunque los vendedores y administradores procuran dar a conocer y corregir estas vulnerabilidades, hay situaciones en las que esto no es posible:

- En algunos sistemas heredados, los sistemas operativos no pueden ser parcheados o actualizados.
- Incluso en los sistemas en los que podemos aplicar parches, los administradores a veces no tienen el suficiente tiempo y recursos para seguir e instalar las últimas actualizaciones necesarias. Esto es un problema común, sobre todo en entornos que incluyen un gran número de hosts con sistemas operativos y hardware variado.
- Los usuarios y administradores pueden equivocarse al configurar sus sistemas.

Un sistema de detección de intrusos puede ser una excelente herramienta de protección de sistemas. Un IDS puede detectar cuando un atacante ha intentado penetrar en un sistema explotando un fallo no corregido. De esta forma, podríamos avisar al administrador para que llevara a cabo un backup del sistema inmediatamente, evitando así que se pierda información valiosa.

Detectar preámbulos de ataques (normalmente pruebas de red y otras actividades).

Cuando un individuo ataca un sistema, lo hace típicamente en fases predecibles. En la primera fase, el atacante hace pruebas y examina el sistema o red en busca de un punto de entrada óptimo. En sistemas o redes que no disponen de un IDS, el atacante es libre de examinar el sistema con un riesgo mínimo de ser detectado. Esto le facilita la búsqueda de un punto débil en nuestra red.

La misma red con un IDS monitorizando sus operaciones le presenta una mayor dificultad. Aunque el atacante puede examinar la red, el IDS observará estas pruebas, las identificará como sospechosas, podrá activamente bloquear el acceso del atacante al sistema objetivo y avisará al personal de seguridad de lo ocurrido para que tome las acciones pertinentes.

Documentar el riesgo de la organización.

Cuando se hace un plan para la gestión de seguridad de la red o se desea redactar la política de seguridad de la organización, es necesario conocer cual es el riesgo de la organización a posibles amenazas, la probabilidad de ser atacada o si incluso ya está siendo atacada.

Un IDS nos puede ayudar a conocer la amenaza existente fuera y dentro de la organización, ayudándonos a tomar decisiones acerca de los recurso de seguridad que deberemos emplear en nuestra red y del grado de cautela que deberemos adoptar al redactar la política de seguridad.

Proveer información útil sobre las intrusiones que se están produciendo.

Incluso cuando los IDSs no son capaces de bloquear ataques, pueden recoger información relevante sobre éstos. Esta información puede, bajo ciertas circunstancias, ser utilizada como prueba en actuaciones legales. También se puede usar esta información para corregir fallos en la configuración de seguridad de los equipos o en la política de seguridad de la organización.

1.3.3. Arquitectura de los IDSs

Existen varias propuestas sobra la arquitectura de los IDSs pero ninguna de ellas se usa mayoritariamente. Esto provoca que los productos de los fabricantes que trabajan con distinta arquitectura puedan difícilmente interoperar entre sí. Según [19, pag. 180]:

> "Si no existe un solo producto mágico que haga todo por nosotros, es mejor que los distintos productos utilizados para establecer nuestra capacidad de detección de intrusos interoperen. Para que estos productos funcionen juntos debe aplicarse un estándar. Los estándares originales fueron el formato 'autopost de AusCERT' y CIDF. En estos momentos, los esfuerzos de estandarización actuales parecen ser IDWG y CVE y posiblemente algunos productos comerciales."

1.3.3.1. CIDF (Common Intrusion Detection Framework)

El Marco de Detección de Intrusos Común fue un primer intento de estandarización de la arquitectura de un IDS. No logró su aceptación como estándar, pero estableció un modelo y un vocabulario para discutir sobre las intrusiones. Mucha gente que trabajó en el proyecto original está fuertemente involucrada en los esfuerzos del Grupo de Trabajo de Detección de Intrusos (Intrusion Detección Working Group, IDWG) del Internet Engineering Task Force (IETF).

Los cuatro tipos básicos de equipos que contempla el CIDF son los siguientes (ver figura 1.2):

- **Equipos E**, o generadores de eventos, son los sensores. Su trabajo es detectar eventos y lanzar informes.

- **Equipos A**, reciben informes y realizan análisis. Pueden ofrecer una prescripción y un curso de acción recomendado.
- **Equipos D**, son componentes de bases de datos. Pueden determinar si se ha visto antes una dirección IP o un ataque por medio de correlación y pueden realizar análisis de pistas.
- **Equipos R**, o equipos de respuesta, pueden tomar el resultado de los equipos E, A y D y responder a los eventos.

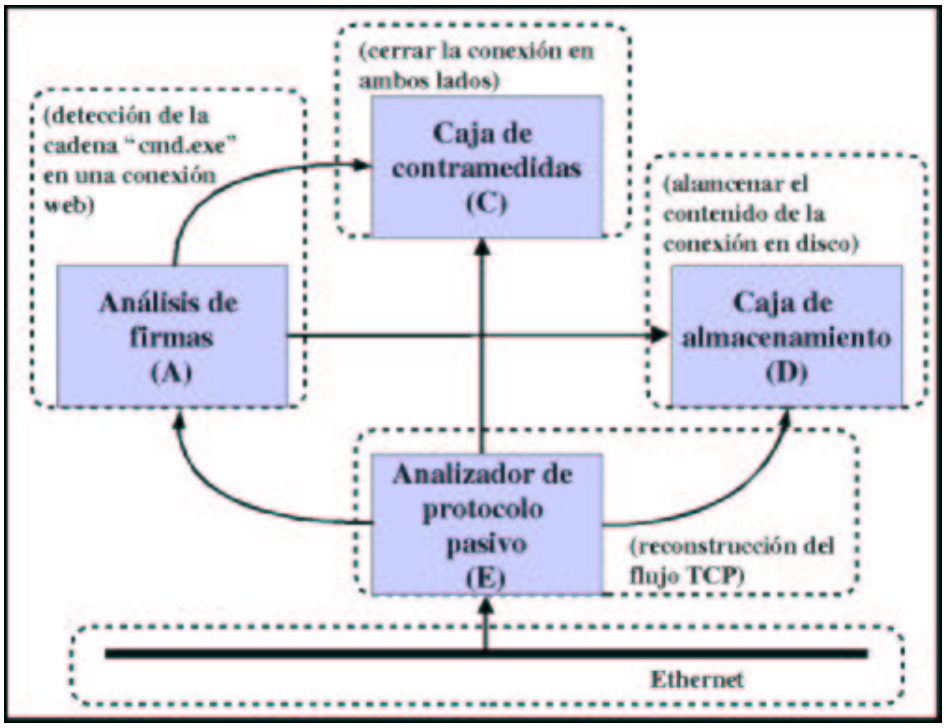

Figura 1.2: Diagrama de bloques de la arquitectura CIDF.

Se puede encontrar más información sobre la arquitectura CIDF en [27].

1.3.3.2. CISL (Common Intrusion Specification Language)

El Lenguaje de Especificación de Intrusiones Común aparece de la necesidad de unir los cuatro tipos de equipos de CIDF. Los diseñadores de CISL pensaron que este lenguaje debería ser capaz al menos de transmitir los siguientes tipos de información:

- **Información de eventos en bruto.** Auditoría de registros y tráfico de red. Sería el encargado de unir equipos E con equipos A.
- **Resultados de los análisis.** Descripciones de las anomalías del sistema y de los ataques detectados. Uniría equipos A con D.
- **Prescripciones de respuestas.** Detener determinadas actividades o modificar parámetros de seguridad de componentes. Encargado de la unión entre equipos A y R.

CISL es un lenguaje bastante complicado de sintaxis parecida a LISP que no llegó a cuajar en la comunidad de seguridad.

1.3.3.3. Autopost de AusCERT

A diferencia de CIDF/CISL, el CERT australiano (AusCERT) desarrolló un sistema de trabajo sencillo que permitía que se analizara y se agregara un informe en una base de datos con tan solo un par de líneas de Perl. La forma que toma el informa de un incidente podría ser la siguiente:

```
Source: 216.36.45.84
Ports: tcp 111
Incident type: Network_scan
re-distribute: yes
timezone: GMT + 1300
reply: no
Time: Web 15 Mar 2000 at 14:01 (UTC)
```

Este sistema tiene una alta interoperabilidad y es muy sencillo de construir y analizar. El problema es que los analistas a menudo necesitan un gran nivel de detalle (una fidelidad alta) acerca del evento, por ejemplo, para análisis forense, y en este modelo toda esa información se perdería. La solución de interoperabilidad que parece ser la elegida es IDWG; según progresa el trabajo, la fidelidad de los datos es el indicador a vigilar más importante.

1.3.3.4. Arquitectura de IDWG (Intrusion Detection Working Group)

El IETF rechazó el enfoque de CIDF seguramente por antipatía a CISL, debido a su complejidad, y creó un grupo de trabajo llamado IDWG (Intrusion Detection Working Group) que tenía como objetivo el de definir formatos y procedimientos de intercambio de información entre los diversos subsistemas del IDS.

Los resultados de este grupo de trabajo serán:

1. Documentos que describan los requerimientos funcionales de alto nivel para la comunicación entre sistemas de detección de intrusos y entre los sistemas de detección de intrusos y sus sistemas de gestión.

2. Un lenguaje común de especificación que describa el formato de los datos.

3. Un marco de trabajo que identifique los mejores protocolos que se pueden usar para la comunicación entre los IDSs y que defina como se mapean en éstos lo formatos de datos.

Actualmente existen tres borradores esperando ser aceptados como estándar. Se puede consultar más información en [11].

1.3.4. Clasificación de los IDSs

Existen varias formas de clasificar los IDSs:

1.3.4.1. Fuentes de información

Existen varias fuentes de las que un IDS puede recoger eventos. Algunos IDSs analizan paquetes de red, capturados del backbone de la red o de segmentos LAN, mientras que otros IDSs analizan eventos generados por los sistemas operativos o software de aplicación en busca de señales de intrusión.

IDSs basados en red (NIDS)

La mayor parte de los sistemas de detección de intrusos están basados en red. Estos IDSs detectan ataques capturando y analizando paquetes de la red. Escuchando en un segmento, un NIDS puede monitorizar el tráfico que afecta a múltiples hosts que están conectados a ese segmento de red, protegiendo así a estos hosts.

Los IDSs basados en red a menudo están formados por un conjunto de sensores localizados en varios puntos de la red. Estos sensores monitor-izan el tráfico realizando análisis local e informando de los ataques que se producen a la consola de gestión. Como los sensores están limitados a ejecutar el software de detección, pueden ser mas fácilmente asegurados ante ataques. Muchos de estos sensores son diseñados para correr en modo oculto, de tal forma que sea más difícil para un atacante determinar su presencia y localización.

Ventajas:

- Un IDS bien localizado puede monitorizar una red grande, siempre y cuando tenga la capacidad suficiente para analizar todo el tráfico.
- Los NIDSs tienen un impacto pequeño en la red, siendo normalmente dispositivos pasivos que no interfieren en las operaciones habituales de ésta.
- Se pueden configurar para que sean muy seguros ante ataques haciéndolos invisibles al resto de la red.

Desventajas:

- Pueden tener dificultades procesando todos los paquetes en una red grande o con mucho trafico y pueden fallar en reconocer ataques lanzados durante periodos de tráfico alto. Algunos vendedores están intentando resolver este problema implementando IDSs completamente en hardware, lo cual los hace mucho más rápidos.
- Los IDSs basados en red no analizan la información cifrada. Este problema se incrementa cuando la organización utiliza cifrado en el propio nivel de red (IPSec) entre hosts, pero se puede resolver con una política de seguridad más relajada (por ejemplo, IPSec en modo túnel).
- Los IDSs basados en red no saben si el ataque tuvo o no éxito, lo único que pueden saber es que el ataque fue lanzado. Esto significa que después de que un NIDS detecte un ataque, los administradores deben manualmente investigar cada host atacado para determinar si el intento de penetración tuvo éxito o no.
- Algunos NIDS tienen problemas al tratar con ataques basados en red que viajan en paquetes fragmentados. Estos paquetes hacen que el IDS no detecte dicho ataque o que sea inestable e incluso pueda llegar a caer.

Quizá el mayor inconveniente de los NIDS es que su implementación de la pila de protocolos de red puede diferir a la pila de los sistemas a los que protege. Muchos sistemas servidores y de escritorio actuales no cumplen en ciertos aspectos los estándares TCP/IP, pudiendo descartar paquetes que el NIDS ha aceptado. Esta inconsistencia de información entre el NIDS y el sistema protegido es la base de la ocultación de ataques que veremos más adelante.

IDSs basados en host (HIDS)

Los HIDS fueron el primer tipo de IDSs desarrollados e implementados. Operan sobre la información recogida desde dentro de una computadora, como pueda ser los ficheros de auditoría del sistema operativo. Esto permite que el IDS analice las actividades que se producen con una gran precisión, determinando exactamente qué procesos y usuarios están involucrados en un ataque particular dentro del sistema operativo.

A diferencia de los NIDSs, los HIDSs pueden ver el resultado de un intento de ataque, al igual que pueden acceder directamente y monitorizar los ficheros de datos y procesos del sistema atacado.

Ventajas:

- Los IDSs basados en host, al tener la capacidad de monitorizar eventos locales a un host, pueden detectar ataques que no pueden ser vistos por un IDS basado en red.
- Pueden a menudo operar en un entorno en el cual el tráfico de red viaja cifrado, ya que la fuente de información es analizada antes de que los datos sean cifrados en el host origen y/o después de que los datos sea descifrados en el host destino.

Desventajas:

- Los IDSs basados en hosts son más costosos de administrar, ya que deben ser gestionados y configurados en cada host monitorizado. Mientras que con los NIDS teníamos un IDS por múltiples sistemas monitorizados, con los HIDS tenemos un IDS por sistema monitorizado.
- Si la estación de análisis se encuentra dentro del host monitorizado, el IDS puede ser deshabilitado si un ataque logra tener éxito sobre la máquina.
- No son adecuados para detectar ataques a toda una red (por ejemplo, escaneos de puertos) puesto que el IDS solo ve aquellos paquetes de red enviados a él.
- Pueden ser deshabilitados por ciertos ataques de DoS.
- Usan recursos del host que están monitorizando, influyendo en el rendimiento del sistema monitorizado.

1.3.4.2. Tipo de análisis

Hay dos acercamientos al análisis de eventos para la detección de ataques: detección de abusos y detección de anomalías. La detección de abusos es la técnica usada por la mayoría de sistemas comerciales. La detección de anomalías, en la que el análisis busca patrones anormales de actividad, ha sido y continua siendo objeto de investigación. La detección de anomalías es usada de forma limitada por un pequeño número de IDSs.

Detección de abusos o firmas

Los detectores de abusos analizan la actividad del sistema buscando eventos que coincidan con un patrón predefinido o firma que describe un ataque conocido.

Ventajas:

- Los detectores de firmas son muy efectivos en la detección de ataques sin que generen un número elevado de falsas alarmas.
- Pueden rápidamente y de forma precisa diagnosticar el uso de una herramienta o técnica de ataque específico. Esto puede ayudar a los encargados de la seguridad a priorizar medidas correctivas.
- Pueden permitir a los administradores de seguridad, sin importar su nivel o su experiencia en este campo, el seguir la pista de los problemas de seguridad de sus sistemas.

Desventajas:

- Solo detectan aquellos ataques que conocen, por lo que deben ser constantemente actualizados con firmas de nuevos ataques.
- Muchos detectores de abusos son diseñados para usar firmas muy ajustadas que les privan de detectar variantes de ataques comunes.

Detección de anomalías

La detección de anomalías se centra en identificar comportamientos inusuales en un host o una red. Funcionan asumiendo que los ataques son diferentes a la actividad normal. Los detectores de anomalías construyen perfiles representando el comportamiento normal de los usuarios, hosts o conexiones de red. Estos perfiles son construidos de datos históricos recogidos durante el periodo normal de operación. Los detectores recogen los datos de los eventos y usan una variedad de medidas para determinar cuando la actividad monitorizada se desvía de la actividad normal. Las medidas y técnicas usadas en la detección de anomalías incluyen:

- Detección de un umbral sobre ciertos atributos del comportamiento del usuario. Tales atributos de comportamiento pueden incluir el numero de ficheros accedidos por un usuario en un periodo de tiempo dado, el numero de intentos fallidos para entrar en el sistema, la cantidad de CPU utilizada por un proceso, etc. Este nivel puede ser estático o heurístico.
- Medidas estadísticas, que pueden ser paramétricas, donde la distribución de los atributos perfilados se asume que encaja con un determinado patrón, o no paramétricas, donde la distribución de los atributos perfilados es aprendida de un conjunto de valores históricos, observados a lo largo del tiempo.
- Otras técnicas incluyen redes neuronales, algoritmos genéticos y modelos de sistema inmune.

Solo las dos primeras se utilizan en los IDSs actuales, el resto son parte de proyectos de investigación.

Ventajas:

- Los IDSs basados en detección de anomalías detectan comportamientos inusuales. De esta forma tienen la capacidad de detectar ataques para los cuales no tienen un conocimiento específico.
- Los detectores de anomalías pueden producir información que puede ser utilizada para definir firmas en la detección de abusos.

Desventajas:

- La detección de anomalías produce un gran numero de falsas alarmas debido a los comportamientos no predecibles de usuarios y redes.
- Requieren conjuntos de entrenamiento muy grandes para caracterizar los patrones de comportamiento normal.

1.3.4.3. Respuesta

Una vez se ha producido un análisis de los eventos y hemos detectado un ataque, el IDS reacciona. Las repuestas las podemos agrupar en dos tipos: pasivas y activas. Las pasivas envían informes a personas, que se encargarán de tomar acciones al respecto, si procede. Las activas lanzan automáticamente respuestas a dichos ataques.

Respuestas pasivas

En este tipo de respuestas se notifica al responsable de seguridad de la organización, al usuario del sistema atacado o a algún CERT de lo sucedido. También es posible avisar al administrador del sitio desde el cual se produjo el ataque avisándole de lo ocurrido, pero es posible que el atacante monitorice el correo electrónico de esa organización o que haya usado una IP falsa para su ataque.

Respuestas activas

Las respuestas activas son acciones automáticas que se toman cuando ciertos tipos de intrusiones son detectados. Podemos estableces dos categorías distintas:

- Recogida de información adicional: consiste en incrementar el nivel de sensibilidad de los sensores para obtener más pistas del posible ataque (por ejemplo, capturando todos los paquetes que vienen de la fuente que originó el ataque durante un cierto tiempo o para un máximo número de paquetes).
- Cambio del entorno: otra respuesta activa puede ser la de parar el ataque; por ejemplo, en el caso de una conexión TCP se puede cerrar la sesión establecida inyectando segmentos TCP RST al atacante y a la víctima o filtrar en el router de acceso o en el firewall la dirección IP del intruso o el puerto atacado para evitar futuros ataques.

1.3.5. Herramientas y complementos

Sistemas de valoración y análisis de vulnerabilidades

Las herramientas de análisis de vulnerabilidades determinan si una red o host es vulnerable a ataques conocidos. La valoración de vulnerabilidades representa un caso especial del proceso de la detección de intrusiones. Los sistemas que realizan valoración de vulnerabilidades funcionan en modo 'batch' y buscan servicios y configuraciones con vulnerabilidades conocidas en nuestra red.

File Integrity Checkers (Controladores de la integridad de los ficheros)

Los 'File Integrity Checkers' son otra clase de herramientas de seguridad que complementan a los IDSs. Utilizan resúmenes de mensajes (*message digest*) u otras técnicas criptográficas para hacer un compendio del contenido de ficheros y objetos críticos en el sistema y detectar cambios, de tal forma que para cualquier cambio del contenido del fichero el compendio sea totalmente distinto y que sea casi imposible modificar el fichero de forma que el compendio sea igual al del fichero original.

El uso de estos controladores es importante, puesto que los atacantes a menudo alteran los sistemas de ficheros una vez que tienen acceso completo a la máquina, dejando puertas traseras que más tarde facilitan su entrada al sistema, esta vez "sin hacer tanto ruido".

El producto freeware Tripwire (`www.tripwiresecurity.com`) es quizá el ejemplo más conocido de este tipo de herramientas.

Honeypots

Son sistemas que están diseñados para ser atacados y que capturan de forma silenciosa todos los movimientos del atacantes. Se usan principalmente para lo siguiente:

- evitan que el atacante pase su tiempo intentado acceder a sistemas críticos.
- recogen información sobre la actividad del atacante.
- permiten al administrador recabar pruebas de quién es el atacante y responda ante su CERT o el administrador del sistema origen de la agresión.

Los honeypots se usan ampliamente para investigar sobre nuevos ataques, y facilitan la incorporación de nuevas firmas en los IDSs. Existen multitud de herramientas que facilitan esta labor de 'análisis forense', entre las que cabe destacar Deception Toolkit (`http://all.net/dtk/`) o ManTrap, de Recourse Tecnologies (`http://www.recourse.com/product/ManTrap/`).

Últimamente han aparecido las Honeynets, redes de honeypots. Para más información se puede consultar `http://project.honeynet.org/`.

1.4. ¿Dónde colocar un IDS?

La decisión de donde localizar el IDS es la primera decisión que hay que tomar una vez que estamos dispuestos a instalar un IDS. De esta decisión dependerá tanto el equipo que usemos, como el software IDS o la base de datos.

1.4.1. Organización

Existen principalmente tres zonas en las que podríamos poner un sensor, tal y como muestra la figura 1.3:

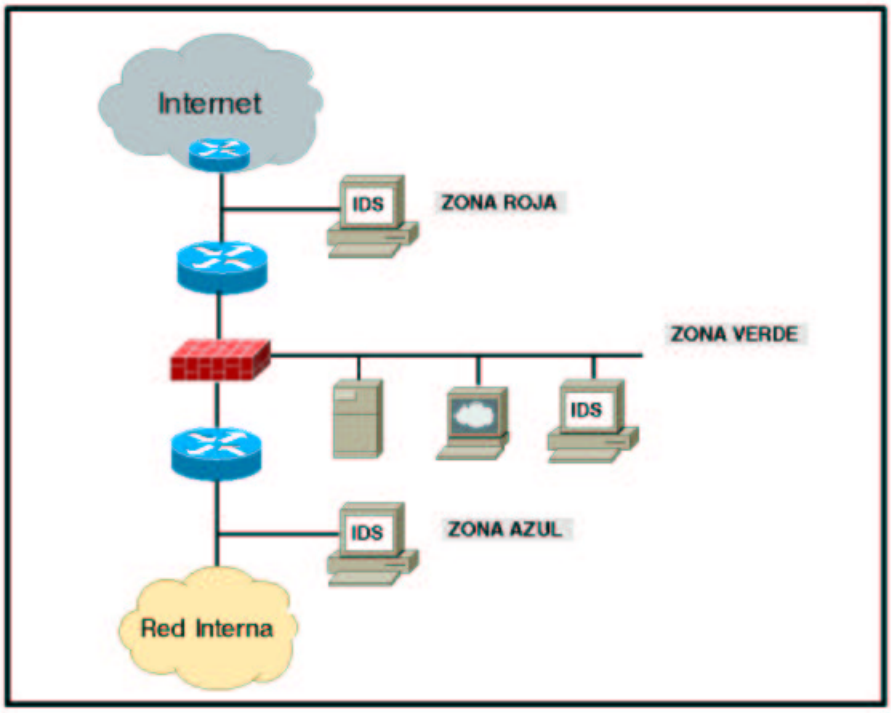

Figura 1.3: Localización de un IDS dentro de una organización.

Veamos las características que presenta cada una de estas zonas:

- **Zona roja**: Esta es una zona de alto riesgo. En esta zona el IDS debe ser configurado para ser poco sensible, puesto que vera todo el tráfico que entre o salga de nuestra red y habrá más posibilidad de falsas alarmas.
- **Zona verde**: El IDS debería ser configurado para tener una sensibilidad un poco mayor que en la zona roja, puesto que ahora, el firewall deberá ser capaz de filtrar algunos accesos definidos mediante la política de nuestra organización. En esta zona aparece un menor número de falsas alarmas que en la zona roja, puesto que en este punto solo deberían estar permitidos accesos hacia nuestros servidores.
- **Zona azul**: Esta es la zona de confianza. Cualquier tráfico anómalo que llegue hasta aquí debe ser considerado como hostil. En este punto de la red se producirán el menor número de falsas alarmas, por lo que cualquier alarma del IDS debe de ser inmediatamente estudiada.

Es importante destacar que la zona azul no es parte de la red interna. Todo lo que llegue al IDS de la zona azul ira hacia el firewall (por ejemplo, si utilizamos un proxy-cache para nuestros usuarios de web) o hacia el exterior. El IDS no escuchará ningún tipo de tráfico interno dentro de nuestra red.

En el caso de tener un IDS escuchando tráfico interno (por ejemplo, colocado entre una VLAN y su router), las falsas alarmas vendrán provocadas en su mayor parte por máquinas internas al acceder a los servidores de nuestra red, por servidores nuestros (DNS sobre todo) y escaneadores de red, por lo que habrá que configurar el IDS para que no sea muy sensible.

1.4.2. ISP

¿Qué ocurre ahora si nuestra organización es un ISP?. Es posible que el trafico a la entrada de este ISP sea demasiado grande como para ser técnicamente imposible instalar un único IDS que lo analice todo. Para estos casos es necesario un sistema de detección de intrusos que pueda separar los sensores de la estación de análisis. Una posible solución podría ser la de instalar un sensor en cada uno de los nodos que conectan físicamente con las organizaciones a las que da servicio el ISP, y que estos sensores envíen las alertas generadas a la estación de análisis (ver figura 1.4).

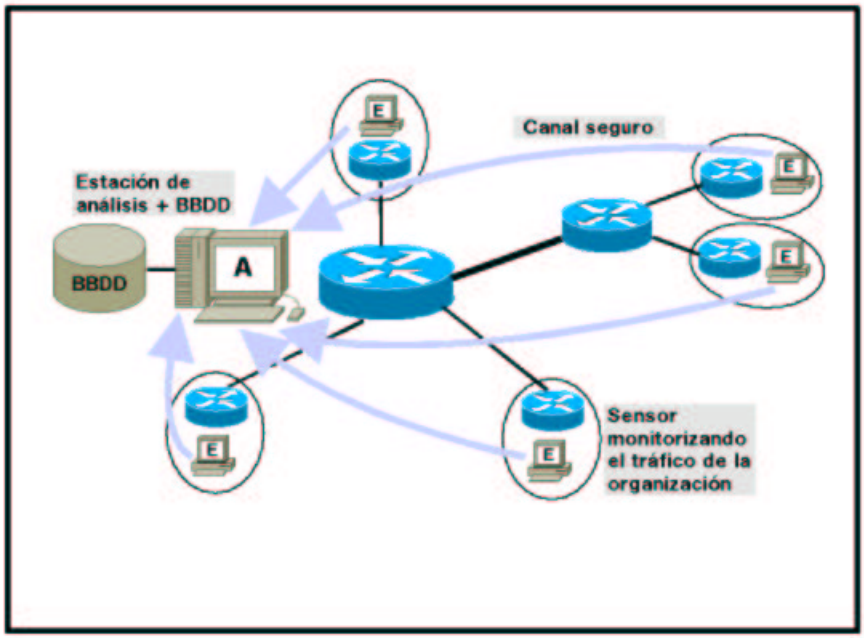

Figura 1.4: Distribución de los sensores dentro de un ISP.

Esta transmisión debe realizarse de forma segura (y esto quiere decir ”cifrada”) y a intervalos regulares, puesto que si el sensor avisa de la alerta nada más ésta se ha producido, un atacante podría monitorizar el tráfico que genera el sensor hacia la estación de análisis y deducir si un ataque ha sido detectado por el sensor o no.

1.5. Limitaciones de los NIDSs

Una de los problemas más importantes de los NIDSs es su incapacidad de reconstruir exactamente lo que está ocurriendo en un sistema que están monitorizando. Como hemos visto, los detectores de intrusos basados en firmas funcionan examinando el contenido de los paquetes que se están transmitiendo por la red. El análisis consiste básicamente en un análisis pasivo del protocolo utilizado, lo cual hace a esta técnica no intrusiva y extremadamente difícil de detectar o evadir.

Algunos de los ataques que los IDSs pueden detectar se pueden ver simplemente analizando los paquetes IP; un intento de ocultar un ataque fragmentando paquetes IP se puede observar examinando el desplazamiento del fragmento dentro de su paquete IP correspondiente. La mayoría de los IDSs comerciales implementan esta función.

Otra técnica más depurada de anti-IDS se basa en los ataques con firmas polimórficas. En este caso, el ataque (por ejemplo, un exploit con un shellcode bien conocido) puede cambiar el juego de instrucciones en ensamblador con una funcionalidad idéntica, pero que generará una firma distinta y por lo tanto no será detectado por el IDS. Se puede dotar a la herramienta hacker de la suficiente inteligencia como para mutar automáticamente el shellcode lanzado en cada uno de sus ataques, haciendo inútil la tarea del detector de intrusos.

Cada componente identificado por el modelo CIDF tiene implicaciones únicas de seguridad, y puede ser atacado de diferentes formas:

- Como único punto de entrada al sistema, las cajas-E actúan como lo ojos y los oídos de un IDS. Un ataque que afecte a estas cajas-E dejará al IDS incapaz de ver lo que realmente ocurre en los sistemas monitorizados.
- Por otra parte, un atacante que conozca el algoritmo de análisis que utiliza la caja-A y descubra un fallo en su implementación será capaz de evadir la detección.
- Un atacante que pueda saturar de información los componentes D puede impedir que se almacene información sobre futuros ataques.
- Si se conoce como engañar a las cajas-C, se podrá seguir atacando una red sin ningún tipo de contramedida.

Los ataques de denegación de servicio también pueden dar al traste con una política de seguridad basada en un IDS. Es entonces cuando hay que decidir si el IDS será 'fail-open' o 'fail-closed'. En el primer caso tenemos que cuando el IDS caiga, la red quedará totalmente abierta a merced de cualquier ataque, mientras que en el segundo caso, el trafico hacia el exterior y viceversa quedará bloqueado. Los NIDSs son inherentemente 'fail-open'.

El problema de los IDSs de código abierto es que el atacante puede examinar el código y determinar que flujo de datos hace que aumente excesivamente la carga computacional, el consumo de memoria o el consumo de disco. El primera provoca un descarte de paquetes que deja ciego al sistema mientras que dura el ataque. El segundo puede incluso abortar el proceso e inhabilitar el IDS, y el tercero puede provocar que no se guarden logs sobre las alertas producidas hasta que se corrija en fallo.

Sin embargo, los ataques anti-IDSs que más estragos causan son los de ’inserción’ y ’evasión’, que veremos a continuación [2].

1.5.1. Inserción

El ataque de inserción se basa en que un IDS puede aceptar paquetes que luego un sistema final va a rechazar. La figura 1.5 da un ejemplo de el ataque.

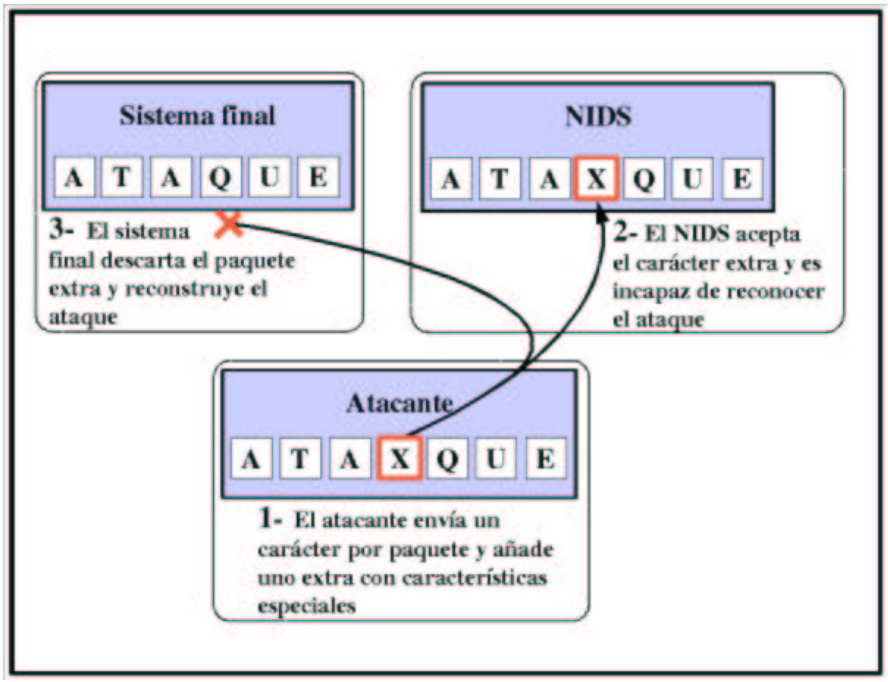

Figura 1.5: Ataque de inserción.

Un atacante envía un flujo de un caracter por paquete, en el cual uno de los caracteres aparecerá solo en el IDS. Como resultado, el IDS y el sistema final reconstruirán dos cadenas distintas. En general, un ataque de inserción ocurre cuando el IDS es menos estricto en procesar un paquete que el sistema final. Una reacción obvia a este problema puede ser la de hacer al IDS tan estricto como sea posible en procesar paquetes leídos de la red; esto podría minimizar los ataque de inserción, Sin embargo, al hacer esto podemos estar dando facilidad a otro ataque, el de evasión, que veremos a continuación.

1.5.2. Evasión

Un sistema final puede aceptar un paquete que un IDS rechace. En la figura 1.6 podemos ver un ejemplo de este ataque:

El ataque de evasión provoca que el IDS vea un flujo diferente que el sistema final. Esta vez, sin embargo, el sistema final toma más paquetes que el IDS, y la información que el IDS pierde es crítica para la detección del ataque.

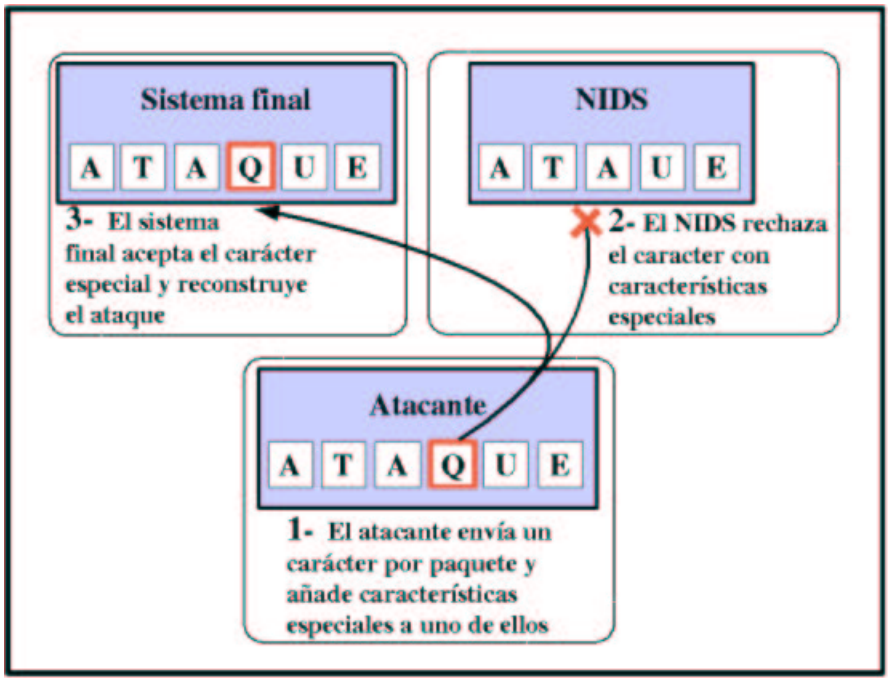

Figura 1.6: Ataque de evasión.

1.5.3. Evasión e inserción en el mundo real.

Muchos protocolos son simples y fáciles de analizar, pero otros son más complejos y requieren considerar muchos otros paquetes antes de determinar si se consideran parte de la transacción actual o no. Para que un monitor de red pueda analizarlos deben tenerse en cuenta mucha información de estado en cada uno de los paquetes. Por ejemplo, para descubrir que hay dentro de una conexión TCP, el monitor debe intentar reconstruir el flujo de datos que se intercambia en la conexión, teniendo en cuenta tanto la reconstrucción en TCP como la reconstrucción de fragmentos IP

Pero, ¿como es posible llevar a cabo ataque de inserción y evasión en TCP/IP?. Hay muchas formas de confundir a un IDS y hacerle que tome o descarte paquetes de forma errónea.

Por ejemplo, supongamos que el IDS se encuentra entre el router de la organización y el router del ISP. Si el atacante envía su paquete que quiere insertar en el IDS con un TTL tal que llegue al IDS con un valor de 1, el IDS lo aceptara, pero será descartado a la entrada de la organización por el router de entrada.

Otra forma es mediante el bit DF (Don’t fragment) de IP y la MTU de la red del sistema atacado. Si en el segmento del IDS la MTU es suficiente como para albergar ese paquete, el IDS lo aceptará, pero si en el sistema final la MTU es demasiado pequeña, el paquete será descartado.

Otro caso es el que aparece cuando al reensamblar fragmentos IP se produce un solapamiento. Dependiendo del sistema operativo, el solapamiento tendrá en cuenta datos antiguos o datos recientes, por lo que puede no coincidir con la forma en la que lo hace el IDS.

El RFC 1323 introduce dos nuevas opciones TCP diseñadas para incrementar el rendimiento de TCP en entornos de alta velocidad. Con estas nuevas opciones surge la posibilidad de que opciones TCP puedan aparecen en paquetes que no son segmentos SYN. El RFC 1323 dice que tales opciones solo pueden aparecer en segmentos no-SYN si la han sido especificadas y aceptadas previamente en esa conexión. Puesto que ciertas implementaciones pueden rechazar segmentos no-SYN conteniendo opciones que no han visto previamente, es importante que el IDS no acepte ciegamente estos paquetes. Por otra parte, algunos sistemas finales pueden simplemente ignorar las opciones y continuar procesando el segmento; si el IDS no determina correctamente como actúa el sistema final que recibe el tráfico, será vulnerable a ataques de inserción o evasión.

Otro problema aparece al reensamblar flujos TCP. Algunos IDS no usan los números de secuencia, simplemente insertan datos en el flujo en el orden en el que los van recibiendo. Esto se suele hacer por razones de eficiencia y rapidez en el IDS, y es vital cuando el tráfico que se tiene que analizar es muy elevado. Estos sistemas son fácilmente engañables y ni siquiera ofrecen seguridad para flujos TCP normales no malintencionados, puesto que los segmentos pueden llegar fuera de orden y esto es algo que está permitido en TCP.

Pero incluso si el IDS chequea los números de secuencia, no hay seguridad de que un segmento dado sea aceptado siempre por igual por el sistema final y el IDS. Por ejemplo, en sistemas como IRIX, HP-UX, AIX, Solaris y BSD, la capa TCP favorece nuevos datos cuando hay un solape en los números de secuencia, pero Windows NT favorece los datos antiguos.

1.6. Actualidad en la detección de intrusiones

1.6.1. Proyectos de investigación

1.6.1.1. Adaptative Intrusion Detection system - AID

El desarrollo del sistema AID (sistema de Detección de Intrusos Adaptativo, [3]) se llevó a cabo en la Universidad de Tecnología de Brandenburgo, entre 1994 y 1996. El sistema fue diseñado para auditorías de red basadas en la monitorización de los hosts presentes en una LAN y está siendo utilizado para la investigación de auditorías de privacidad.

Arquitectura y funcionalidad

El sistema tiene una arquitectura cliente-servidor formada por una estación de monitorización central y varios agentes (servidores) en los hosts monitorizados (ver figura 1.7). La estación central alberga un programa director (cliente) y un sistema experto. Los agentes obtienen los eventos relevantes de los hosts a través de unas funciones de auditoría, para ser convertidos más adelante a un formato de datos independiente al sistema operativo. Es por esto por lo que el sistema soporta entornos UNIX heterogéneos. No se ha llegado a implementar agentes para otra plataformas como Windows NT.

Los datos auditados son transferidos a la estación de monitorización central, cargados en una memoria caché y analizados en tiempo real por un sistema experto.

Por otra parte, el director proporciona funciones para la administración de seguridad de los hosts monitorizados: controla sus funciones de auditoría, pide nuevos eventos mediante consultas seguras y devuelve las decisiones del sistema experto a los agentes. Se usan RPCs seguras para la comunicación entre el director y los agentes.

El sistema experto usa una base de conocimientos con firmas de ataques orientadas a estados, las cuales son modeladas por máquinas deterministas de estado finito e implementadas como secuencias de reglas.

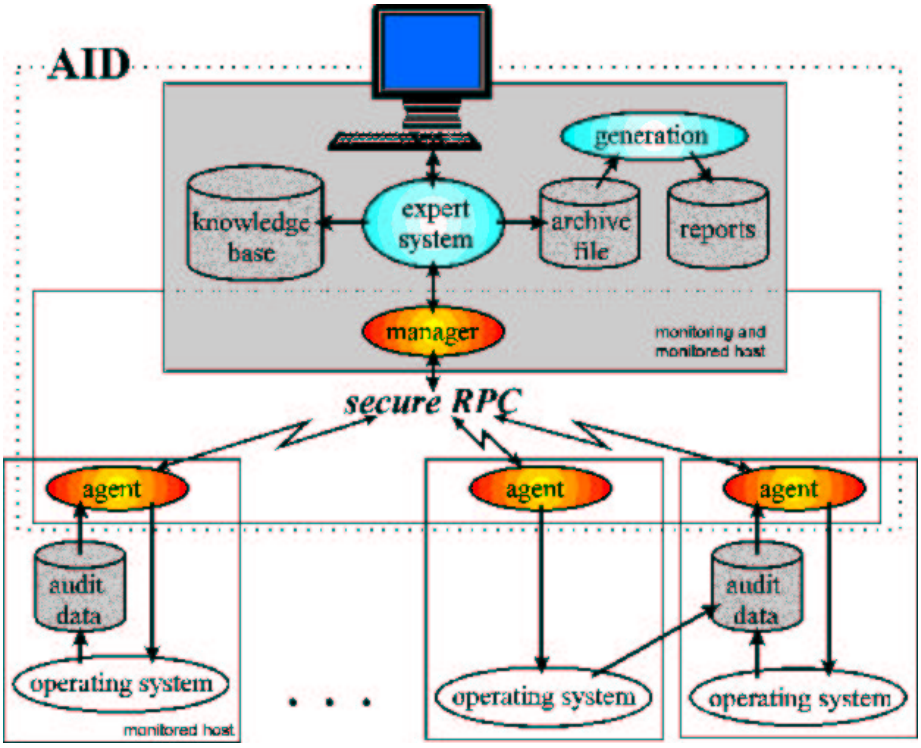

Figura 1.7: Arquitectura del sistema AID.

El administrador puede acceder a las capacidades de monitorización avanzadas por medio de un interfaz de usuario gráfico. Además, el sistema experto archiva datos para análisis forense y crea informes de seguridad al finalizar o bloquear los ataques.

Rendimiento

AID ha sido probado con éxito en entornos de red de área local formados por Sun SPARCstations corriendo Solaris 2.x y TCP/IP. Se implementaron un total de 100 reglas y la base de conocimientos era capaz de detectar 10 escenarios de ataques. En la configuración descrita y bajo suposición de carga normal del sistema en los hosts monitorizados (máximo de 2 usuarios por hosts y sin actualizaciones del sistema en progreso), el sistema experto analiza mas de 2,5 MBytes de eventos por minuto. El test ha mostrado que el prototipo puede soportar hasta 8 hosts.

Pese a ser un sistema antiguo y enfocado como un IDS basado en hosts, fue pionero en su característica multiplataforma al crear un lenguaje unificado para los eventos, siendo los agentes los encargados de la traducción. Además, planteó por primera vez un modelo de reglas en forma de autómata, modelo que utilizarán más tarde otros sistemas de detección de intrusos esta vez basados en red.

1.6.1.2. Agentes Autónomos para la Detección de Intrusiones

El Grupo de Agentes Autónomos para la Detección de Intrusiones (Autonomous Agents for Intrusion Detección Group, [21]) está formado por un número de estudiantes y profesores del CERIAS en la Universidad de Purdue, (Gene Spafford como director), y están estudiando métodos distribuidos para la detección de intrusiones.

Enfoque del grupo

Enfocan el problema de la detección de intrusiones desde un ángulo diferente: en lugar de un diseño monolítico del sistema de detección de intrusos, proponen una arquitectura distribuida que utiliza pequeñas entidades, conocidas como agentes, para detectar comportamientos anómalos o maliciosos. Su diseño aventaja a otros enfoques en términos de escalabilidad, eficiencia, tolerancia a fallos y flexibilidad. Estudian este enfoque desarrollando sistemas que lo implementen y miden su rendimiento y capacidades de detección. De esta forma, esperan ser capaces de conocer las capacidades y limitaciones del enfoque basado en agentes cuando es aplicado a sistemas reales.

La arquitectura AAFID

Fue propuesta en 1994 por Crosbie y Spafford. La idea consistía en usar agentes autónomos para realizar detección de intrusiones, y sugirieron que los agentes podrían evolucionar automáticamente usando programación genética de tal forma que el sistema de detección de intrusiones automáticamente se ajustaría y evolucionaría conforme al comportamiento del usuario. La idea de usar programación genética no fue nunca implementada. Sin embargo, la idea de utilizar agentes para la detección de intrusiones fue evolucionando, y entre 1995 y 1996 la arquitectura AAFID fue desarrollada en el laboratorio COAST. La arquitectura inicial tuvo una estructura jerárquica que permanece en la actualidad y fue usada para implementar el primer prototipo del sistema. Desde 1997 hasta hoy, la arquitectura AAFID evolucionó con la incorporación de filtros y la separación entre el interfaz de usuario y el monitor. La nueva arquitectura ha sido utilizada par el crear un último prototipo que se estudia en la actualidad. En la figura 1.8podemos ver los cuatro componentes principales de la arquitectura: agentes, filtros, transceivers y monitores. Nos referiremos a cada uno de estos componentes como "entidades AAFID" o simplemente "entidades", y a la totalidad del sistema de detección de intrusiones como "sistema AAFID".

Un sistema AAFID puede estar distribuido sobre cualquier número de hosts en una red. Un agente es una entidad independiente que monitoriza ciertos aspectos de un host y los notifica al transceiver apropiado. Por ejemplo, un agente podría estar buscando un largo número de conexiones telnet a un host protegido, y considerar su ocurrencia como sospechosa. El agente generaría una alerta que sería enviada al transceiver apropiado. El agente no tiene la autoridad de lanzar directamente una alarma, sino que son los transceivers los que, combinando notificaciones de distintos agentes, conocen el estado actual de la red y son capaces de saber cuando existe realmente peligro.

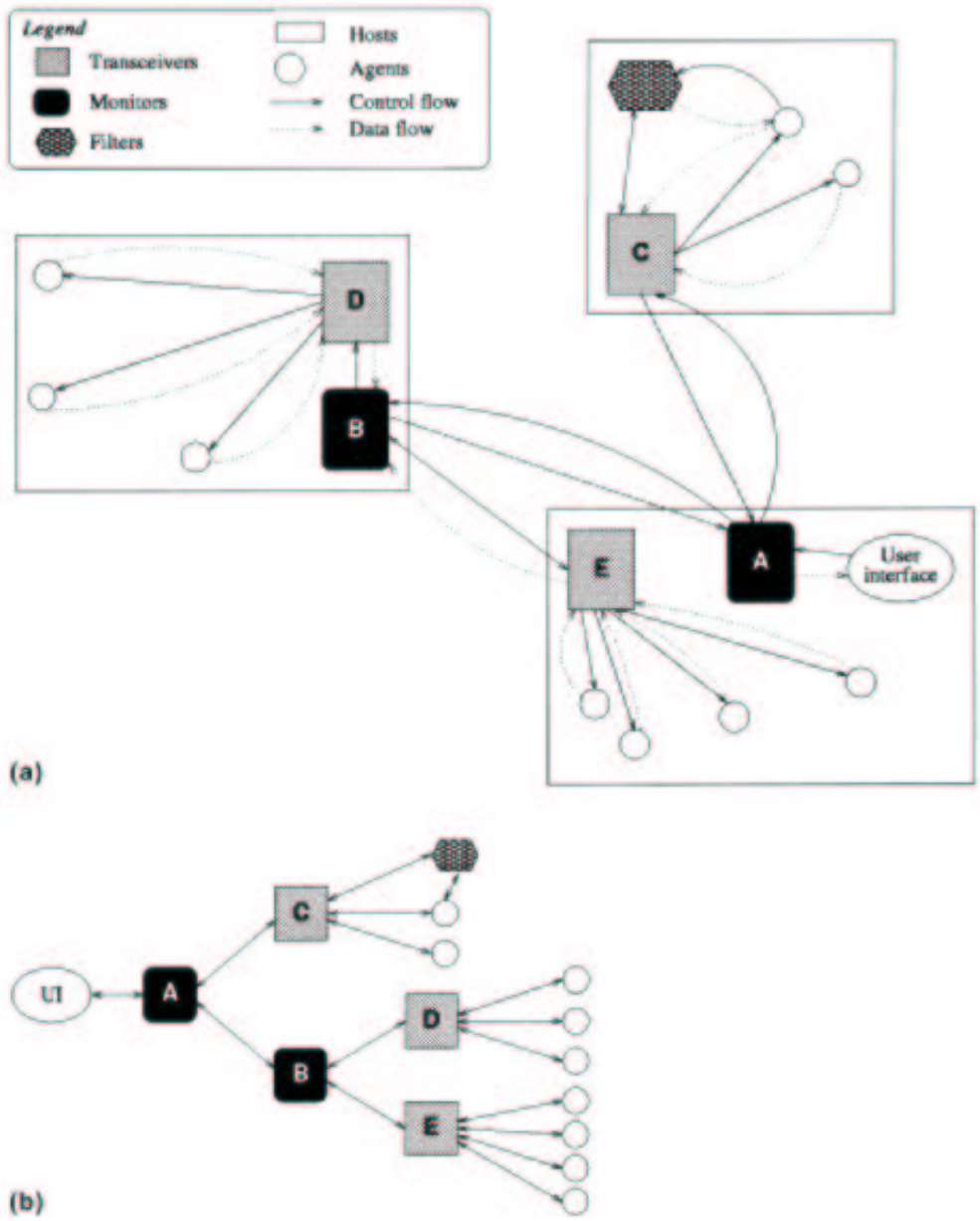

Figura 1.8: Representación física y lógica de un ejemplo de sistema AAFID. (a) Distribución física de los componentes en un ejemplo de sistema AAFID, mostrando los agentes, filtros, transceivers y monitores, al igual que la comunicaciones y canales de control entre ellos. (b) Organización lógica del mismo AAFID mostrando la comunicación jerárquica de los componentes. Las flechas bidireccionales representan flujo de datos y control entre las entidades.

Los filtros se encargan de la selección de datos y de la abstracción de éstos hacia los agentes. Cada host puede contener cualquier número de agentes que monitoricen eventos interesantes que ocurran en él. Los agentes pueden usar filtros para obtener datos de una forma independiente al sistema, lo cual permite la portabilidad de agentes a distintas plataformas simplemente adoptando el filtro adecuado.

En la arquitectura AAFID original, cada agente era responsable de obtener los datos que necesitaba. Cuando el primer prototipo fue implementado, este acercamiento mostró los siguientes problemas:

- En un único sistema, puede haber mas de un agente que necesite datos de la misma fuente. Teniendo a cada agente leyendo los datos del mismo origen se duplicaba el trabajo de lectura en los ficheros.
- Puede haber agentes que puedan proporcionar una función útil bajo diferentes versiones de UNIX, o incluso bajo diferentes arquitecturas (como Windows NT). Sin embargo, los datos requeridos por el agente pueden ser localizados en diferentes lugares en cada sistema y pueden ser almacenados en diferentes formatos. Esto significa tener que escribir un agente diferente para cada sistema, que sepa donde encontrar el dato y como leerlo.

Ambos problemas fueron resueltos con la incorporación de los filtros, una capa software que independiza a los agentes del tipo de sistema en el que están corriendo y gestiona los recursos de éste de manera óptima.

Los transceivers son el interfaz de comunicación externa de cada host. Tienen dos papeles: control y procesamiento de datos. Para que un hosts sea monitorizado por un sistema AAFID, debe haber un transceiver corriendo en ese host.

Los monitores son las entidades de más alto nivel en la arquitectura AAFID. Reciben información de todos los transceivers que ellos controlan y pueden hacer correlaciones a alto nivel y detectar eventos que ocurren en diferentes hosts. Los monitores tienen la capacidad de detectar eventos que pueden pasar desapercibidos por los transceivers.

Implementación de AAFID

El primer prototipo fue desarrollado entre 1995 y 1996, basado en la primera especificación de la arquitectura AAFID. Este prototipo fue implementado por una combinación de programas escritos en C, Bourne shell, AWK y Perl. Su principal objetivo era el de tener algo tangible de todo lo desarrollado hasta el momento y enfrentarse a las primeras decisiones de diseño. La primera implementación fue solamente e\ aleada internamente en el laboratorio de COAST.

La segunda implementación incorpora los cambios más recientes en la arquitectura, como son los filtros. Esta implementación es conocida como AAFID2 y fue lanzada en septiembre de 1998. La primera release de AFFID2, que incluía el sistema base y algunos agentes, fue testeada bajo Solaris.

La segunda release pública fue lanzada en septiembre de 1999. Fueron añadidos nuevos mecanismo de procesamiento de eventos y fue testeada en Linux y Solaris. El prototipo AFFID2v2 está implementado completamente en Perl5, lo cual lo hace fácil de instalar, ejecutar y portar a diferentes plataformas. Sólo ha sido probado en entornos UNIX, pero está en proceso de ser portado a Windows NT.

El objetivo del diseño de AAFID2v2 es que sea sencillo de experimentar y extremadamente flexible. Fue desarrollado usando características de programación orientada a objetos de Perl5, lo cual

permite que su código sea fácilmente reutilizable. AAFID2v2 también incluye una herramienta para la generación de código para desarrollar nuevos agentes.

1.6.1.3. Detección de intrusiones basada en grafos (GrIDS - Graph-based Intruson Detection System), Universidad de California, Davis.

GrIDS [26] ha sido diseñado para detectar ataques automatizados a gran escala en sistemas de red. El mecanismo que se utiliza es el de construir grafos de actividad que representen la estructura causal de actividades distribuidas a gran escala. Forma parte del proyecto "Intrusion Detection for Large Networks" (`http://seclab.cs.ucdavis.edu/arpa/arpa.html`), fundado por ARPA.

Los nodos de un grafo de actividad corresponden a los hosts en un sistema, mientras que los arcos corresponden a la actividad de red entre estos hosts. La actividad en una red monitorizada se puede modelar mediante grafos que representen dicha actividad. Estos grafos son entonces comparados con patrones conocidos de actividades intrusivas y hostiles, y si presentan un grado de similitud superior a un umbral, se dispara una alarma o un procedimiento reactivo.

Como ejemplo, en la figura 1.9podemos ver el grafo en forma de árbol que generaría una actividad tipo gusano, que infecta máquinas de forma exponencial.

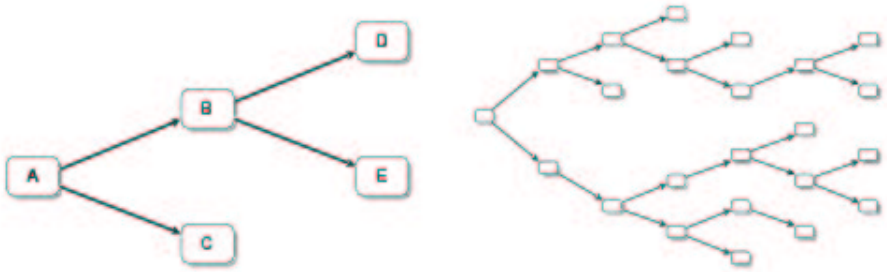

Figura 1.9: (a) El inicio del grafo de un gusano. (b) Una vista más extensa del mismo gusano.

GrIDS modela una organización como una jerarquía de departamentos. Para que esto sea fácilmente configurable, incorpora un interfaz "drag and drop" que permite reconfigurar la jerarquía incluso durante la actividad del sistema. Cada departamento en la jerarquía tiene un módulo GrIDS propio y construye y evalúa grafos de actividad dentro del departamento. Cuando una actividad cruza los limites departamentales es pasada al siguiente nivel en la jerarquía de resolución. Este nivel construirá grafos reducidos en los cuales los nodos son enteramente subdepartamentales en lugar de simples hosts. Características más complejas del grafo completo pueden ser preservadas en el grafo reducido mediante la incorporación de atributos.

Este acercamiento hace de GrIDS un diseño escalable. Muchos de los esfuerzos van dirigidos a conseguir que la agregación de departamentos en la jerarquía global sea un mecanismo escalable, y permitir la configuración dinámica del sistema de forma que continúe siendo funcional cuando se llegue a una red grande.

A pesar de esto, GrIDS debería ser tomado como una prueba de concepto en lugar de un sistema acabado apropiado para producción. Todavía no es posible asegurar la integridad de las comunicaciones entre módulos GrIDS, por lo que no se puede prevenir que un atacante reemplace partes del GrIDS con código maligno. El prototipo tampoco es resistente a ataques de denegación de servicio, disrupción del protocolo de tiempo de red en el que se basa, o fallos en las redes o computadores en los que corre. Otro problema que aparece es que no detecta intrusiones pequeñas, lentas o ambas.

Detección de intrusos para redes con un gran número de hosts

Como ya hemos comentado, el objetivo de este proyecto es desarrollar tecnología de detección de intrusiones para redes de área extensa. La investigación avanza en dos direcciones:

- Tamaño: se desean manejar miles o decenas de miles de hosts, en lugar de los cientos actualmente posibles.
- Protección de la infraestructura: se desea proteger a los routers, servidores de dominio, etc, además de hacerlo con los hosts de producción.

Como características deseables del sistema se incluyen:

- Interoperabilidad con tecnología de gestión de la red (especialmente SNMP)
- Independencia de sistemas operativos
- Extensibilidad a nuevos componentes de red y servicios.

En estos momentos, el proyecto se encuentra a medio camino. Se han investigado la mayoría de problemas subyacentes y se está trabajando en integrar la experiencia acumulada en un prototipo real.

1.6.2. Productos comerciales

1.6.2.1. Dragon - Enterasys Networks

El IDS de Enterasys Networks, Dragon [7], toma información sobre actividades sospechosas de un sensor denominado Dragon Sensor y de un módulo llamado Dragon Squire que se encarga de monitorizar los logs de los firewalls y otros sistemas. Esta información es enviada a un producto denominado Dragon Server para posteriores análisis y correlaciones. Cada componente tiene ventajas que compensan con debilidades de otro, por ejemplo, el sensor Dragon Sensor es incapaz de interpretar tráfico codificado de una sesión web SSL, pero el producto Dragon Squire es capaz de recocer los logs del servidor web y pasárselos a la máquina de análisis. Veamos un poco más en detalle cada uno de estos componentes.

Dragon Sensor - Network IDS

El sensor de Dragon monitoriza una red en busca de evidencias de actividades hostiles. Cuando éstas ocurren, envía informes junto con un registro de análisis forense al servidor Dragon, el cual lo analiza y lo almacena durante largo tiempo.

Características del sensor:

- Detección de actividades sospechosas tanto mediante firmas como mediante técnicas basadas en anomalías.

- Decodificación robusta a nivel de aplicación: el sensor tiene un conocimiento avanzado de los protocolos a nivel de aplicación, evitando muchos falseos que los atacantes utilizan para burlar los IDSs, como por ejemplo: .

  - En HTTP nos podemos referir al fichero "SECRET.TXT" por "SECRET%2eTXT". Si la firma solo busca la primera cadena, será incapaz de detectar un acceso a ese fichero si el atacante utiliza caracteres %.
  - Espacio en blanco y borrado de caracteres en Telnet y FTP.
  - Codificación SNMP null-byte.
  - Otros.

- Incorporación de filtros para disminuir los falsos positivos.
- Monitorización de redes de alta velocidad (100 Mb/s)
- Las técnicas para evitar falseos anti-NIDS incluyen técnicas para evitar inserción y evasión, tales como:

  - Ignorar paquetes con un TTL pequeño.
  - Ignorar paquetes mayores que la MTU de la red con el bit DF activado.
  - Otros

- Honeypots virtuales: el sensor incluye una variedad de características que permiten al administrador del IDS colocar numerosas trampas para escáneres de red y atacantes. Por ejemplo, el sensor tiene la habilidad de etiquetar direcciones IP individuales o bloques CIDR y capturar todo el tráfico dirigido a ellos. Si este rango de IPs no corresponde a ningún host de nuestra red, seguramente serán intentos de ataques y escaneos.

Los sistemas operativos soportados por el sensor son:

- Linux
- Solaris (Sparc y x86)
- HP-UX
- FreeBSD
- OpenBSD

El sensor incluye dos tarjetas de red a 100Mb/s y una única interfaz Gigabit Ethemet. Está diseñado para agregar tráfico desde múltiples enlaces T3 o E3 o para núcleos Gigabit que requieran de monitorización IDS. Provee más de 1300 firmas. Estas firmas pueden ser elaboradas por el usuario final, aunque debido a la flexibilidad y potencia de su sintaxis la labor es complicada y requiere de cursos de entrenamiento.

Dragon Squire

Dragon Squire se encarga de monitorizar los logs de sistemas de producción y firewalls, en busca de evidencias de actividad maliciosa o sospechosa. Está basado en firmas al igual que Dragon Sensor, y a diferencia de otros IDSs basados en host, puede monitorizar los logs de la aplicaciones que corren en el host, tales como servidores web, mail o FTP.

Servidor Dragon - Consolas de análisis

Muchos de los IDS comerciales están muy limitados en cuanto a los análisis que pueden realizar sobre los datos que capturan. Muchas veces se incorpora una tercera herramienta que ofrece información básica sumarizada y poco más.

El servidor Dragon intenta ofrecer herramientas más similares a la forma en la que los analistas de IDSs hacen su trabajo. Incluye tres aplicaciones web para análisis que soportan análisis de eventos en tiempo real, correlación de tendencias e inspección detallada de cada evento y su información asociada.

Normalmente, los administradores de los IDSs no se sientan delante de sus monitores en espera de alarmas, sino que configuran sus IDSs para enviar alertas al centro de operaciones de red. Si se detecta una alerta, un administrador puede usar varios interfaces web distintos para analizar tales eventos. También se incluye una interfaz de línea de comandos, puesto que muchos administradores encuentran este tipo de interfaces más eficiente. Dragon incluye un completo conjunto de herramientas de línea de comandos para facilitar el análisis de eventos sin el uso de un explorador.

1. **Consola de análisis forense.**

   La consola de análisis forense es un conjunto de aplicaciones que procesan datos de eventos usando herramientas de línea de comandos y nos permiten ver a bajo nivel información sobre los paquetes que hicieron saltar las alarmas. También es posible la visualización mediante un interfaz gráfico basado en web, que nos permite almacenar eventos y ordenarlos en base a parámetros relevantes.

2. **Consola en tiempo real**

   Proporciona una aplicación de alta velocidad para analizar varios millones de eventos con la misma funcionalidad que nos proporciona una página web (ver figura 1.10). Se basa en el programa "rts" o "real-time shell", que lee nuevos eventos de los sensores y los almacena en un buffer circular. Almacenar un millón de eventos en el anillo solo toma unos 24MB de memoria, por lo que dedicar un servidor para correr solo la consola en tiempo real puede proporcionar una capacidad de varios millones de eventos en memoria.

   La aplicación web que envuelve al binario "rts" proporciona una funcionalidad muy similar a la consola de análisis forense, pero, adicionalmente, muestra gráficas de las estadísticas en el tráfico. Casi todas las herramientas en la consola de tiempo real incluyen una opción en vivo que permite el refresco automático cada 1,5 o 15 minutos.

3. **Consola de correlaciones**

   La consola de correlaciones de Dragon (ver figura 1.11) se utiliza para responder consultas correladas. Para ello se utilizan bases de datos SQL, entre ellas MySQL, Oracle, Sybase o MS-SQL.

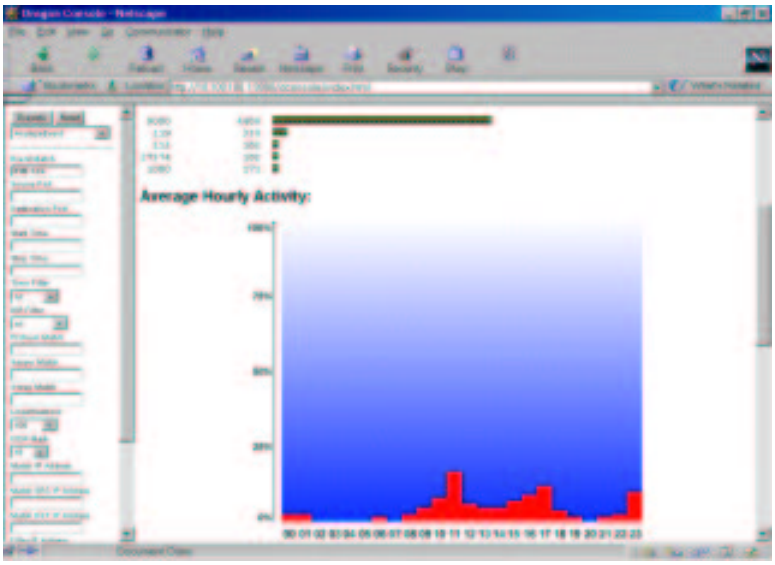

Figura 1.10: Pantalla de la consola en tiempo real en Dragon.

Utiliza consultas SQL para buscar eventos de forma eficiente con un criterio de búsqueda complicado. Para cada consulta se utiliza un sofisticado applet para mostrar la ocurrencia de los resultados con más positivos en un periodo de tiempo seleccionado de antemano.

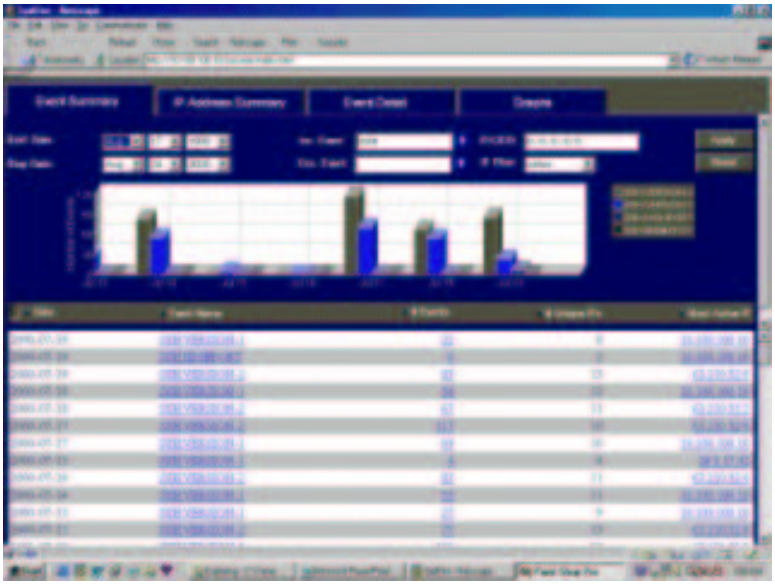

Figura 1.11: Consola de correlaciones en Dragon.

1.6.2.2. NetRanger - Cisco Systems

El sistema de detección de intrusos de Cisco, conocido formalmente por Cisco NetRanger [5], es una solución para detectar, prevenir y reaccionar contra actividades no autorizadas a través de la red.

Productos

Cisco IDS Host Sensor v2.0 es capaz de identificar ataques y prevenir accesos a recursos críticos del servidor antes de que ocurra cualquier transacción no autorizada. Esto lo hace integrando sus capacidades de respuesta con el resto de sus equipos, como veremos más adelante.

La versión más reciente actualmente del sensor de cisco es la v3.0, que incluye un mecanismo de actualización automática de firmas, un lenguaje robusto que permite a los clientes escribir sus propias firmas y extensiones al módulo de respuestas que añaden soporte para la familia de firewalls Cisco PIX y para los conmutadores Cisco Catalyst®.

Cisco Secure IDS incluye dos componentes: Sensor y Director. Las "herramientas" de red de alta velocidad Cisco Secure IDS Sensors analizan el contenido y el contexto de los paquetes individuales para determinar si se autoriza su tráfico. Si se detecta una intrusión, como por ejemplo un ataque de pruebas SATAN (System Administrators Tool for Analyzing Networks), un barrido de pings o si una persona que tiene acceso a información confidencial envía un documento que contiene una palabra código de propiedad, los sensores de Cisco Secure IDS pueden detectar el uso incorrecto en tiempo real, enviar alarmas a una consola de gestión de Cisco Secure IDS Director para la representación geográfica y sacar al agresor de la red.

Dado que el sistema Cisco Secure IDS incorpora funciones de respuesta pro-activas en Sensor, mediante la modificación de las listas de control de acceso (ACL) de los routers Cisco los usuarios pueden configurar el sistema para rechazar o eliminar automáticamente ciertas conexiones. Esta característica puede ser temporal o, si se desea, se puede mantener indefinidamente. El resto del tráfico de la red funcionará normalmente: sólo se eliminará de forma rápida y eficaz el tráfico no autorizado de los usuarios internos o de los intrusos externos. Así se consigue que los operadores de seguridad tengan un poder de actuación sobre toda la red para detener rápidamente la utilización inadecuada de recursos o el acceso de intrusos a la red.

Cisco Secure IDS Sensors

El sensor de Cisco se presenta en tres formatos distintos dependiendo de las necesidades de la organización:

- Modulo IDS Catalyst® 6000: diseñado para integrar la funcionalidad IDS directamente dentro del conmutador permitiendo al usuario monitorizar tráfico directamente del backplane del conmutador en lugar de utilizar módulos software.

  Rendimiento:

  - Monitoriza 100 Mbps de tráfico.
  - Aproximadamente 47.000 paquetes por segundo.

Figura 1.12: Módulo sensor para el conmutador Catalyst®6000.

- IDS-4250: Soporta hasta 500 Mb/s sin paralelizar y puede ser utilizado para monitorizar tráfico en una red Gigabit y para tráfico atravesando conmutadores usados para agregar tráfico entre numerosas subredes.

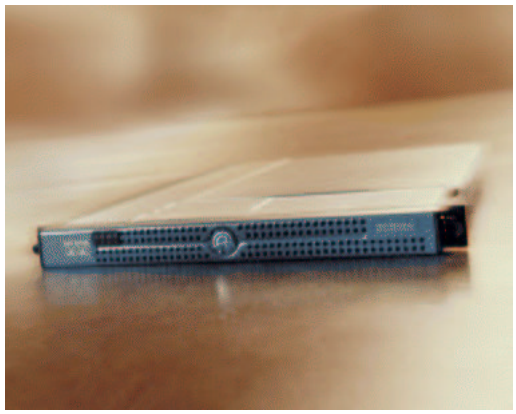

Figura 1.13: Sensor Cisco 4250.

- IDS-4235: Este sensor está optimizado para monitorizar 200 Mb/s de tráfico y es especialmente adecuado para monitorizar tráfico en puertos SPAN (Switched Port Analyzer) y segmentos Fast Ethernet. También es adecuado para monitorizar múltiples entornos T3.
- IDS-4210: Está optimizado para monitorizar entornos de 45 Mb/s aunque también es adecuado para múltiples T1/El, T3 y entornos Ethernet.

Gestores de seguridad y monitorización:

Para la gestión de las alertas y de los dispositivos de seguridad, Cisco dispone de los siguientes sistemas:

- Cisco Secure Policy Manager: CSPM es el sistema de gestión y monitorización más moderno de Cisco. Es un software basado en Windows que facilita la gestión y configuración de componentes de seguridad, entre otros sus IDSs. El componente IDS de CSPM proporciona la capacidad de gestionar y configurar remotamente dispositivos de forma robusta basándose en una política de seguridad y provee un mecanismo de monitorización y gestión de alertas en tiempo real.

- Cisco IDS Unix Director: El Director es una aplicación software basada en UNIX que se ejecuta en entornos HP-OpenView. Esta solución proporciona capacidad de configuración de dispositivos IDS y monitorización de eventos. Es un sistema más antiguo y carece de muchas innovaciones que tiene CSPM.

1.6.2.3. Internet Security Systems - RealSecure®

RealSecure® Network Sensor

RealSecure® [12] proporciona detección, prevención y respuestas a ataques y abusos originados en cualquier punto de la red. Entre las respuestas automáticas a actividades no autorizadas se incluyen el almacenar los eventos en una base de datos, bloquear una conexión, enviar un mail, suspender o deshabilitar una cuenta en un host o crear una alerta definida por el usuario.

El sensor de red rápidamente se ajusta a diferentes necesidades de red, incluyendo alertas específicas por usuario, sintonización de firmas de ataques y creación de firmas definidas por el usuario. Las firmas son actualizables automáticamente mediante la aplicación X-Press Update. El sensor de red puede ser actualizado de una versión a otra posterior sin problema, asegurando así la última versión del producto.

Todos los sensores son centralmente gestionados por la consola RealSecure® SiteProtector (ver figura 1.14), incluyen do la instalación automática, desarrollo y actualizaciones.

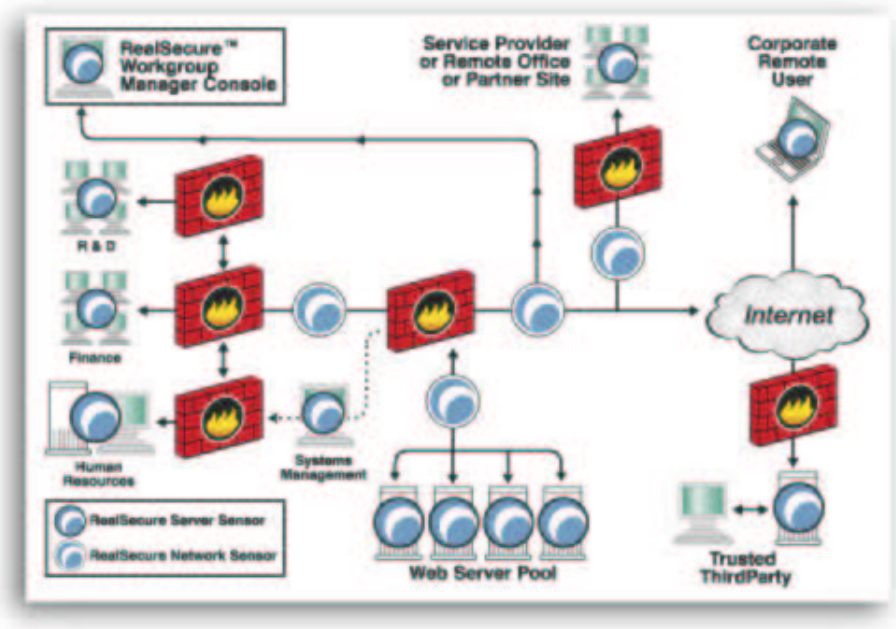

Figura 1.14: Ejemplo de la distribución de sensores basados en host y en red en RealSecure®.

RealSecure® Sentry proporciona detección de intrusiones en tiempo real. Tiene mecanismos de respuesta a comportamientos sospechosos en un segmento de red. Los drivers de captura de paquetes a

alta velocidad funcionan sin causar el más mínimo impacto en la red. Están disponibles en full duplex, multipuerto y Gigabit.

RealSecure® Guard (ver figura 1.15) es un filtro que protege segmentos de red, incluyendo sistemas de producción críticos o conexiones. El tráfico que atraviesa el sistema Guard, es analizado en tiempo real en búsqueda de evidencia de ataques o abusos. Si se detecta un comportamiento anormal, Guard bloquea el ataque e impide que pase a través del otro interfaz.

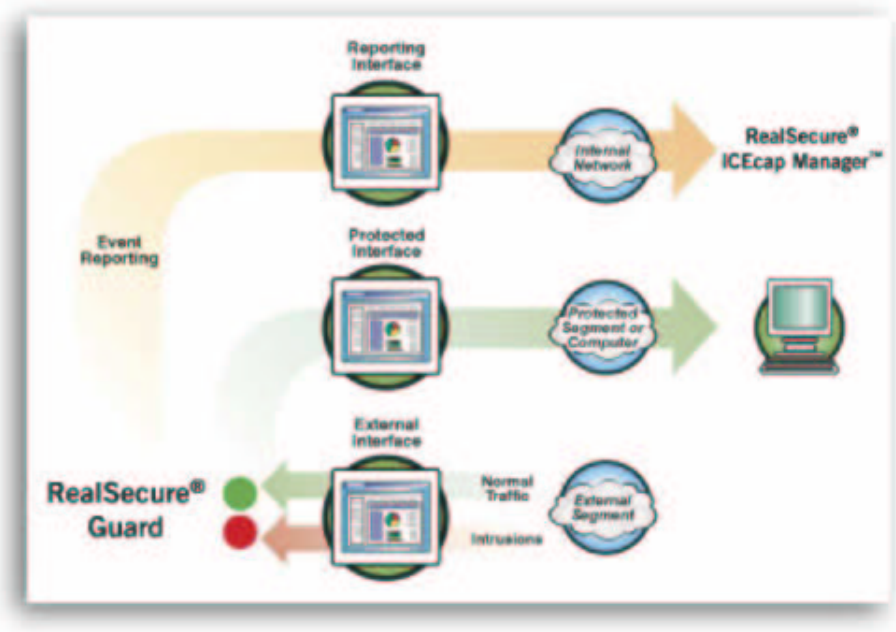

Figura 1.15: Funcionamiento de RealSecure® Guard.

1.6.2.4. Snort

Snort [22]es un IDS en tiempo real desarrollado por Martin Roesch y disponible bajo GPL. Se puede ejecutar en máquinas UNIX y Windows. Es el número uno en sistemas de detección de intrusos en este momento[11. Dispone actualmente de unas 1.600 reglas y de multitud de aplicaciones para el análisis de sus alertas.

En un principio fue diseñado para cumplir los requerimientos de un IDS ligero para uso como prototipo. Era pequeño y flexible, pero poco a poco ha ido creciendo e incorporando funcionalidades que solo estaban presentes en los IDSs comerciales, aunque actualmente sigue estando por detrás de muchos de estos sistemas.

En Snort no es posible separar el componente de análisis y los sensores en máquinas distintas. Sí que es posible ejecutar Snort atendiendo a varios interfaces a la vez (cada uno podría estar monitorizando lugares distintos dentro de una red), puesto que se basa en la librería *pcap*[1], pero esto no

[1]Una librería muy utilizada, creada por Van Jacobson, Craig Leres y Steven McCanne, en Lawrence Berkeley Nacional

permite ningún reparto de carga en el proceso de análisis. Es más, ni aun disponiendo de una máquina multiprocesador es posible (actualmente) hacer balanceo de carga por CPU. Esto es así porque la opción de compilación multihilo de Snort está todavía en fase de pruebas y no es muy estable, lo que obliga a ejecutar Snort como un proceso monolítico. Esto será posible a partir de la versión 2.0.

La arquitectura de Snort se enfocó par ser eficiente, simple y flexible. Snort está formado por tres subsistemas: el decodificador de paquetes, la máquina de detección y el subsistema de alerta y logs. Estos subsistemas, como ya hemos mencionado antes, corren por encima de la librería libpcap, la cual es portable y proporciona mecanismos de filtrado y captura de paquetes.

Decodificador de paquetes

Soporta gran variedad de protocolos de capa de enlace bajo TCP/IP, tales como Ethemet, SLIP, PPP y ATM. Es el encargado de organizar los paquetes conforme van pasando por la pila de protocolos. Cada subrutina del decodificador ordena de una forma distinta los paquetes, formando una estructura de datos basada en punteros por encima del tráfico real capturado. Esta estructura de datos será la que guíe al motor de análisis para el posterior análisis.

Motor de detección

Snort mantiene sus reglas de detección en un lista enlazada bidimensional, cuya estructura vemos en la figura 1.16. La lista base se denomina 'Chain Header' y la que deriva de ésta se llama 'Chain Option'.

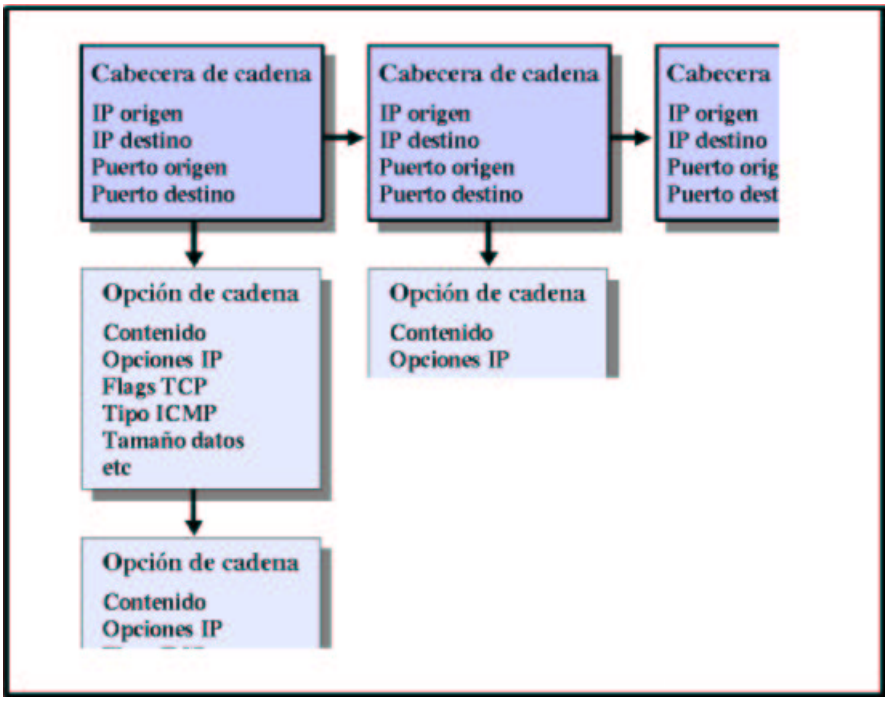

Figura 1.16: Estructura lógica del encadenamiento de reglas en Snort.

Laboratory (Universidad de California). Se puede descargar en ftp://ftp.ee.lbl.gov/libpcap.tar.Z

Cuando llega un paquete al motor de detección, éste busca en la lista 'Chain Header' de izquierda a derecha la primera coincidencia. Después, buscará por la lista 'Chain Option' si el paquete cumple las opciones especificadas. Si aparece alguna coincidencia no seguirá buscando y se tomarán las acciones correspondientes. En caso contrario, buscará en el siguiente nodo de 'Chain Header'.

El motor de búsqueda tiene capacidad para la inserción de módulos plug-in, que pueden utilizarse para dos cosas:

- Permiten hacer búsquedas más complicadas en los paquetes o flujos TCP cuando la sintaxis de las reglas no lo permite. A estos módulos se les llama preprocesadores, y los más utilizados son el detector de escaneos de puertos, el defragmentador, el reconstructor de flujo TCP y el Spade, un motor de detección de anomalías aun en fase muy primitiva.
- Permite definir modos de alerta y log no nativos a Snort, como bases de datos (MySQL, Oracle, PostgreSQL, MS-SQL), SNMP traps, CSV o salida en formato XML, por ejemplo.

Subsistema de alerta y log

Actualmente hay tres sistemas de log y cuatro de alerta. Las opciones de log pueden ser activadas para almacenar paquetes en forma decodificada y entendible por humanos o en formato tcpdump. El formato decodificado se usa para un análisis rápido y el tcpdump es mucho más rápido de almacenar y proporciona un mayor rendimiento. Las alertas se pueden realizar por syslog, en ficheros texto mediante dos formatos distintos (rápido y completo) y NETBIOS para avisar a máquinas Microsoft Windows. Avisan del tipo de ataque detectado y ofrece información adicional como IP origen y destino, fecha y hora de la detección y campo de datos.

Snort dispone de un mecanismo que optimiza considerablemente su rendimiento. Puesto que normalmente se quiere un sistema de back-end potente como una base de datos SQL para hacer correlaciones de los ataques, las escrituras de los logs suelen ser muy costosas. Al ser Snort un proceso monolítico, mientras que se encuentra escribiendo en la base de datos es incapaz de hacer otras cosas, como procesar el tráfico de entrada. Existe una herramienta llamada Barnyard que consigue la división de estas tareas: por una parte tendremos un proceso dedicado a la decodificación y al análisis, y por otra tendremos un proceso que escribe en la base de datos. Lógicamente estos procesos necesitan comunicarse, pero esta comunicación está optimizada para que sea mucho más rápida que la escritura en una base de datos compleja como pueda ser Oracle o MS-SQL.

1.6.2.5. Shadow

Fue desarrollado como respuesta a los falsos positivos de un IDS anterior, NID. La idea era construir una interfaz rápida que funcionara bien en una DMZ caliente (una DMZ que sufre mucho ataques). La interfaz permitiría al analista evaluar gran cantidad de información de red y decidir de qué eventos informar.

No es en tiempo real. Los diseñadores de Shadow sabían que no iban a estar disponibles para vigilar el sistema de detección de intrusos 7 días a las semana, 24 horas al día. Shadow almacena el tráfico de red en una base de datos y se ejecuta por la noche. Los resultados esperan a que el analista llegue a la mañana siguiente.

Al no ser en tiempo real, es inviable que Shadow analice él contenido de los paquetes, por lo qué tan solo se centra en las cabeceras. Esto le hace inútil para análisis forense.

Capítulo 2

Análisis de requisitos y diseño del sistema

2.1. Objetivos y requerimientos del sistema

Los objetivos del proyecto por parte de RedIRIS fueron los siguientes:

- Evitar que otras organizaciones tengan que avisar de ataques producidos desde máquinas de la Universidad de Valencia (UV): una vez se ha comprometido una máquina, los atacantes suelen aprovechar el ancho de banda que la Universidad tiene hacia el exterior para lanzar nuevos ataques. Cuando las organizaciones atacadas tienen IDSs o firewalls se producen quejas, las cuales deben evitarse.
- Envío de incidencias de forma automatizada al CERT-ES: se debía implementar un software para automatizar la gestión de incidencias, de forma que se pudiese avisar al CERT de la organización sobre los ataques que se habían producido dependiendo de su nivel de importancia. Además, el software debería estar escrito en un lenguaje portable como Perl o PHP.
- Conocimiento por parte del CERT de las fuentes más frecuentes de ataques: es útil conocer cuales son las organizaciones origen de los ataques.
- Instalar un sensor a la entrada de la Universidad: desde un principio se pensó que el sistema monitorizaría todo el tráfico que entraba y salía de la Universidad, y se localizaría en el PVC ATM que une el router de acceso de RedIRIS con el de la Universidad de Valencia.

Además de estos objetivos, se pensó que también serían útiles los siguientes:

- Protección en la medida de lo posible de la red frente a ataques externos: dado que en general, las universidades españolas carecen de medidas de seguridad, se hace importante la detección de los ataques entrantes. Esto será útil no solo para avisar a los responsables de las máquinas sobre el intento de penetración por si éste hubiese tenido éxito, sino también para obtener detalles mediante análisis forense en el caso de que un ataque requiriese investigación adicional a posteriori.
- Conocimiento del grado de riesgo de la Universidad: útil para la creación de una política de seguridad genérica.
- Elaborar una política de seguridad para el sistema.

2.2. Posibles ubicaciones del sensor

La localización de los sensores es una cuestión que depende en gran medida de la infraestructura de la red en la que estemos trabajando. En el caso de la red de la Universidad de Valencia podemos distinguir dos partes: la red interna y el acceso a Internet. Dependiendo del tráfico que queramos analizar nos centraremos más en una parte o en otra.

La red interna se divide básicamente en tres infraestructuras correspondientes a los tres campus que forman la Universidad: Burjasot, Blasco Ibañez y Naranjos. Cada uno de estos campus dispone de un comnutador ATM connectado con los otros dos formando una topología mallada. Para mejorar la fiabilidad se utiliza PNNI como routing ATM. Se usa LAN Emulation para el transporte de tráfico LAN entre los tres campus.

Dentro de cada campus se utilizan redes Ethernet conmutadas en todos los edificios a excepción de algunos hubs, y se han configurado múltiples VLANs que separan el tŕafico entre centros. En el caso del campus de Burjasot, todas las VLANs se conectan al conmutador central que dispone de una tarjeta de conmutación a nivel 3 que permite el routing entre ellas. Esta tarjeta de routing tiene una interfaz ATM por la cual conecta al conmutador ATM de la Universidad, modelo LightStream 1010 de Cisco, que le da acceso al resto del campus y al exterior.

La conexión a Internet se realiza por medio de RedIRIS, cuyo POP (Point Of Presence) en la Comunidad Valenciana está situado en el campus de Burjasot, concretamente en el edificio del Servicio de Informática. La conexión al exterior se realiza mediante un PVC ATM entre el router del campus de Burjasot y un router de RedIRIS en el POP. Este PVC atraviesa dos conmutadores ATM, el propio de la Universidad y otro propiedad de RedIRIS.

Localizaremos el sensor en el punto donde queramos analizar el tráfico y los dispositivos de la red nos lo permitan. A continuación describimos las posibles ubicaciones del sensor, comentando las ventajas e inconvenientes de cada una de ellas.

2.2.1. PVC multipunto

Como hemos visto antes, el acceso de la Universidad hacia el exterior se hace por medio de un PVC ATM entre el router de acceso y el de RedIRIS. Éste podría ser un lugar donde poner el sensor para que capturase una copia de todo el tráfico que entra y sale.

La forma de hacerlo sería modificar la configuración en uno de los conmutadores ATM del PVC punto a punto para que sea multipunto. El problema que aparece es que no es posible establecer un PVC multipunto bidireccional, por lo que hay que dividir el PVC existente en dos, uno para cada sentido. Ahora bien, si asignamos en cada uno de los routers un subinterfaz por PVC tenemos rutas asimétricas, y eso no funciona bien con el protocolo de routing multicast que se usa actualmente, PIM-SM.

Para resolver este inconveniente, se puede utilizar un truco en la configuración de los routers Cisco que consiste en asignar al mismo subinterfaz los dos PVCs utilizando uno para enviar paquetes al otro router y el otro para recibirlos. Los detalles de la implementación los veremos en el capítulo siguiente.

2.2.2. Splitter ATM

Un splitter de fibra óptica está formado por tres latiguillos de fibra unidos en forma de Y. Se utiliza normalmente para bifurcar un mismo haz de luz a dos dispositivos distintos.

La ventaja de este dispositivo es la facilidad con la que se puede replicar el tráfico de un mismo PVC; tal y como vemos en la figura 2.1, basta con sustituir los dos latiguillos de fibra óptica de los routers de ambos extremos por dos splitters y estaremos recibiendo todo el tráfico entre ambos routers sin necesidad de modificar la configuración de los equipos.

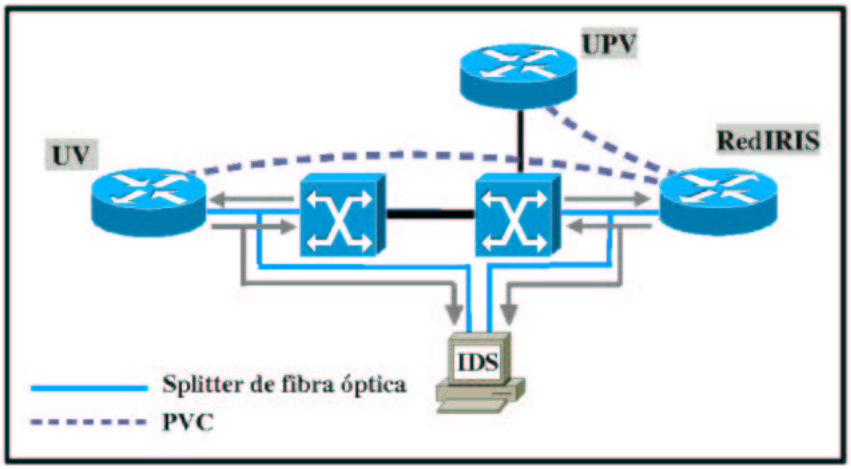

Figura 2.1: Uso de splitters de fibra óptica para el replicado de tráfico.

Los inconvenientes son varios. Primero, el uso de splitters aumenta la atenuación de la señal, aunque en nuestro caso esto no es un factor importante por tratarse siempre de distancias cortas. Segundo, las dos fibras que llegan al IDS son de entrada, por lo que deberemos usar dos tarjetas ATM en el IDS, lo cual encarece el sistema y lo complica. Tercero, necesitamos tarjetas ATM con interfaces de fibra, más caras que las que usan un conector eléctrico. Y cuarto, no podemos discriminar tráfico por VC sino solo por interfaz. Esto plantea problemas cuando discurren varios VCs por una misma interfaz y no se pueden o no se quieren analizar todos ellos.

2.2.3. Conmutadores LAN

Otra opción es la de utilizar los conmutadores LAN para dirigir el tráfico que queramos analizar hacia el sensor. En los conmutadores de Cisco esto lo podemos hacer mediante lo que se conoce como una sesión SPAN (Switched Port Analyzer) [6]. Esta característica está soportada en las series Catalyst® 2900, 3500, 5000 y 6000.

Una sesión SPAN es una asociación entre un conjunto de puertos 'elegidos' y un puerto de destino. El conmutador envía al puerto de destino una copia de las tramas enviadas y/o recibidas por los puertos elegidos. Para la configuración de SPAN, los puertos elegidos y el puerto de destino han de estar normalmente en el mismo conmutador. Se puede activar o desactivar una sesión SPAN mediante la interfaz de línea de comandos o por comandos SNMP. SPAN solo funciona en interfaces Ethernet y no está soportado en interfaces ATM.

El puerto de destino no aceptará por defecto tráfico entrante hacia el conmutador, aunque se puede configurar para que sí que lo acepte. En ese caso el tráfico entrante será conmutado como si de un puerto estándar se tratara, esto es, por la VLAN a la que pertenece. Sólo se permite un puerto destino por sesión SPAN; un mismo puerto no puede ser destino para múltiples sesiones SPAN. Por último, un puerto destino no puede ser a la vez elegido en la misma sesión SPAN.

Se puede monitorizar uno o varios puertos en una misma sesión SPAN en modo de entrada (sólo se copia el tráfico que recibe el conmutador), salida (sólo se copia el tráfico que envía) o ambos (configuración por defecto), pero la elección ha de ser igual para todos los puertos. También es posible 'elegir' una VLAN entera, lo que significa que todos los puertos de esa VLAN son elegidos para la sesión SPAN. A esto se le conoce como VSPAN. Si el puerto destino pertenece a una de las VLANs monitorizadas es automáticamente eliminado de la lista de los puertos elegidos.

En las sesiones VSPAN, los puertos *trunk* son incluidos como puertos elegidos, pero solo se copiará el tráfico que pertenezca a la VLAN especificada en la sesión.

También es posible establecer sesiones SPAN remotas mediante lo que se conoce como RSPAN. En RSPAN, los puertos elegidos y el de destino están distribuidos a través de múltiples conmutadores, permitiendo monitorización remota de cualquier conmutador a través de la red. RSPAN solo está disponible en la serie de conmutadores Catalyst 6000.

2.2.4. Concentradores

Los concentradores son dispositivos que encajan muy bien en la monitorización de tráfico por su característica inherente de reenviar las tramas por todos sus puertos.

El problema con los concentradores es que tan solo se pueden utilizar para Ethernet 10/100Mbps y obligan a que el tráfico sea de 100Mbps máximo half-duplex. Sin embargo, esta limitación de velocidad puede resultar incluso ventajosa para el IDS puesto que si su interfaz es de 100Mbps estaremos asegurando que nunca se verá desbordada con la suma del tráfico de entrada y de salida.

Si lo que queremos es analizar el tráfico entre dos conmutadores y decidimos instalar un hub en medio, debemos tener en cuenta si el enlace que los une está configurado como trunk, ya que de ser así el encapsulado 802.1Q confundiría al IDS y sería incapaz de interpretar el tráfico que le llega si no le damos soporte para el etiquetado de tramas.

2.2.5. Taps

Los taps son dispositivos que permiten hacer derivaciones de cables de fibra óptica o eléctricos. Existen dispositivos tanto para Fast como para Gigabit Ethernet, con capacidad de convertir la señal de óptica a eléctrica.

En el caso de que queramos monitorizar el tráfico que transcurre por un enlace 100-Base-FX, por ejemplo, vimos que colocando un hub limitábamos el tráfico a 100Mbps half-duplex, lo cual puede ser demasiado restrictivo. En la figura 2.2 podemos ver como lo haríamos con un tap.

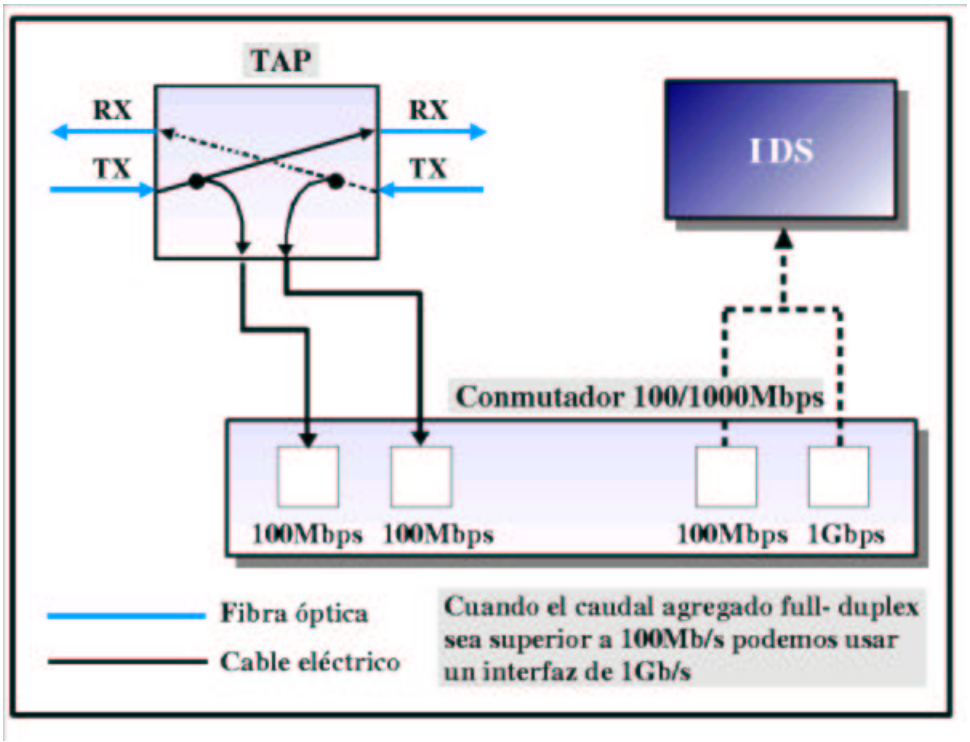

Figura 2.2: Utilización de un tap 100-Base FX a TX y monitorización en 100Mbps ó 1Gbps.

El tráfico transmitido por cada fibra es dirigido al conmutador por un cable eléctrico distinto. En el conmutador podemos configurar una sesión SPAN para que copie el tráfico agregado al puerto donde tenemos el IDS, lo que permite aprovechar en la medida de lo posible la capacidad del conmutador para hacer buffering del tráfico. Cuando el tráfico agregado sea superior a 100Mbps de forma continuada, podemos cambiar el puerto destino de la sesión SPAN a uno de 1Gbps.

Si lo que queremos es monitorizar un enlace Gigabit, podemos utilizar un tap para Gigabit combinado con un balanceador de carga para IDSs tal y como muestra la figura 2.3:

El tap envía el tráfico de cada fibra a un puerto distinto del conmutador (la conexión de recepción de cada puerto Gigabit del conmutador al tap es simulada). El balanceador de carga distribuye tráfico de alta velocidad sobre enlaces hacia los IDSs de 100Mbps . El balanceador de carga hace un reparto por conexiones TCP, para que el IDS sea capaz de reensamblar la sesión TCP. Pero deberá ser más inteligente, por ejemplo, para hacer posible que los IDSs detecten escaneos de puertos. Si el reparto de segmentos SYN fuese por orden secuencial es posible que, si existen multitud de IDSs, el total de segmentos SYN por segundo que recibe cada IDS sea inferior al umbral fijado por el administrador, cuando en realidad no lo sea. Para que sea posible detectar escaneos de puertos, el balanceador de carga debe enviar intentos de conexión repetidos por una misma IP origen al mismo IDS.

Esto solo es cierto cuando el sensor y la máquina de análisis corren en el mismo proceso. En caso contrario, todos los sensores podrían compartir una misma máquina de análisis y ésta sí que sería capaz de detectar los escaneos al almacenar todos los segmentos SYN de forma centralizada. Para ello bastaría con añadir en los sensores una regla que generase una alarma cada vez que se recibiese

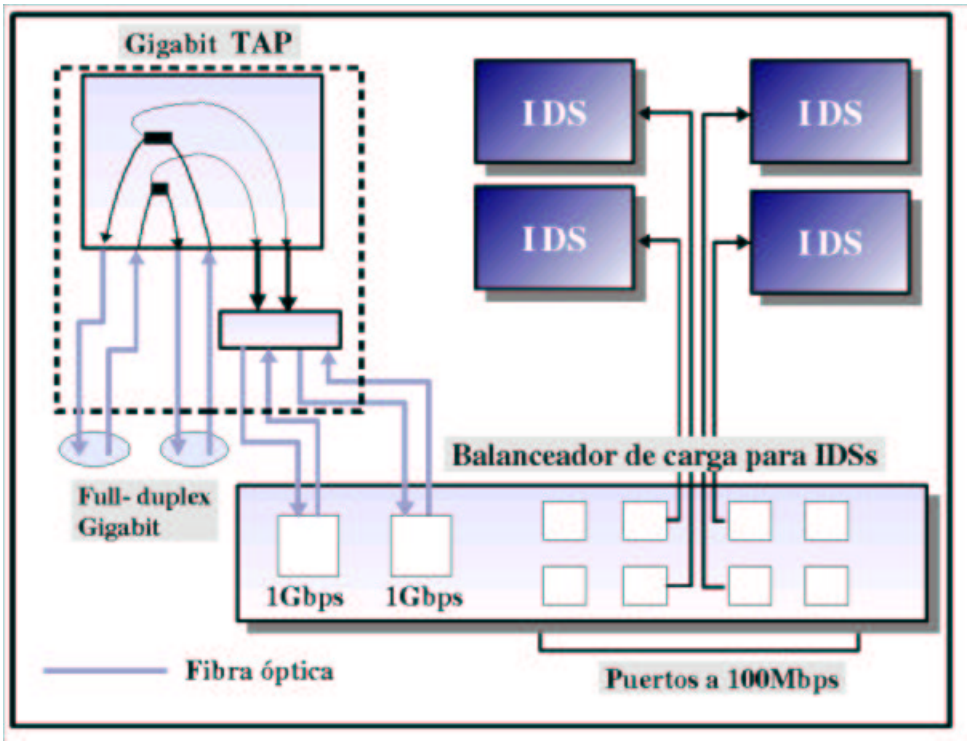

Figura 2.3: Utilización de un tap para Gigabit combinado con un balanceador de carga para IDSs.

un segmento SYN.

2.2.6. Router

En los routers de Cisco no es posible hacer una copia del tráfico tal y como hacíamos en los conmutadores, por lo que esta opción no es viable.

2.3. Elección del sistema

La mayor parte del sistema fue fijado de antemano por RedIRIS: la máquina y el NIDS. El resto (base de datos y software adicional) quedaba a la elección del alumno bajo la condición de que fuese funcional y flexible.

2.3.1. Comentarios acerca de la máquina

La máquina es un VA-Linux modelo 2230 FullOn. La configuración inicial era la siguiente:

- Procesador dual Pentium III (Coppermine) stepping 3, 800MHz.
- 1024 MB de memoria RAM.
- Controlador SCSI Adaptec 7896.
- Disco duro Quantum 'ATLAS V 36 WLS', 40 GB.

- Tarjeta Ethernet Intel 82557 (Ethernet Pro 100), HWaddr 00:D0:B7:A7:F9:2D.
- Tarjeta Dual Ethernet Intel 82557 (Ethernet Pro 100), HWaddr 00:03:47:22:71:E8, 00:03:47:22:71:E9.

Como vemos esta máquina dispone de la arquitectura típica de PC. El chasis es un rack de 2U lo que permite situarlo cómodamente en la sala de máquinas de una organización.

Esta configuración se amplió más tarde (febrero de 2002) con la inclusión de una tarjeta ATM OC-3 con intrefaz en cobre (UTP-5), modelo FORE PCA-200E.

2.3.2. Comentarios acerca del NIDS

El NIDS a utilizar debía ser Snort. RedIRIS optó por este sistema dada su alta relación calidad/precio, principalmente. Puesto que Snort es software de código abierto (*Open Source* bajo GPL), es gratuito y no necesita de licencia alguna para su uso en producción.

Existen en el mercado varias soluciones propietarias que han demostrado mejor rendimiento que Snort, como IIS, NetRanger o Dragon, pero debido a su alto coste se descartaron.

Snort es un NIDS potente disponible en máquinas UNIX y Windows, con mas de 1600 firmas en continua evolución y multitud de aplicaciones para el análisis de las alertas que produce, lo cual le hace el IDSs más utilizado en la actualidad [19], pero esto no quiere decir que sea el IDS más funcional.

Uno de los problemas principales que encontramos es la imposibilidad de aprovechar la capacidad de una máquina multiprocesador, aunque según los desarrolladores esta carencia se solucionará en la próxima versión, la 2.0; a día de hoy (julio de 2002) la versión más reciente de Snort, la 1.8.7, se ejecuta como un proceso secuencial. Esto deja un procesador libre en el sistema, lo cual permite por otro lado realizar labores de mantenimiento y actualización sin casi pérdida de rendimiento por parte del Snort.

Otro problema fundamental de Snort es la imposibilidad de separar el sensor de la máquina de análisis, lo que significa que tampoco es posible el uso de repartidores de carga en enlaces troncales de alta velocidad, cosa que sí que se puede hacer con muchos de los IDSs comerciales que hay en el mercado.

Una posible solución a este problema es hacer que los IDSs Snort colocados en segmentos de red distintos compartan la misma base de datos de forma remota, lo cual se asemeja mucho a un IDS distribuido pero falla, por ejemplo, al no detectar escaneos de puertos a IPs no consecutivas con respecto al segmento al que pertenecen. Supongamos que nuestra red está dividida en dos subredes y queremos monitorizarlas mediante dos IDSs Snort centralizando las alarmas en una base de datos remota. Si decidimos que el umbral para generar una alerta de escaneo de puertos es de 6 o más conexiones TCP por segundo entre las mismas máquinas, si el atacante envía 5 conexiones por segundo a dos máquinas que pertenecen a subredes distintas, ninguno de estos IDSs generará una alerta a pesar de las 10 conexiones por segundo que se están produciendo hacia nuestra red.

2.3.3. Elección de la base de datos.

Para la gestión de los incidentes se optó por usar una base de datos relacional que permitiera hacer consultas complejas y facilitara el análisis al responsable de seguridad.

Es conveniente situar la base de datos en una máquina distinta a la que corre Snort por razones de eficiencia, pero dado que la máquina facilitada por RedIRIS era potente y que Snort tan solo puede aprovechar un procesador simultáneamente quedando el otro libre, se optó por situar Snort y la base de datos en la misma máquina.

Entre las bases de datos con soporte por Snort están MySQL, PostgreSQL, Oracle y MSSQL. Puesto que se había decidido correr el servidor de la base de datos en la misma máquina que en Snort, la opción de MSSQL quedó descartada. Entre el resto y siguiendo la línea de software libre, los candidatos fueron MySQL y PostgreSQL. A favor de PostgreSQL teníamos el soporte para subconsultas, algo que facilita mucho el manejo de consultas complejas.

Finalmente se optó por MySQL por una razón fundamental: en principio, la base de datos no iba a tener un gran número de entradas, tan solo las alertas generadas y algunas falsas alarmas. Dado que MySQL es más rápida al trabajar con tablas con pocas entradas, se decidió el uso de esta base de datos. El inconveniente de no poder realizar subconsultas no es tal, ya que el administrador utilizará una aplicación para realizar las consultas de forma fácil y transparente a la base de datos.

2.3.4. Elección de software adicional

Para el análisis de las alertas por parte del administrador de seguridad es necesaria una herramienta que interactúe con el back-end del sistema. En Snort existen multitud de aplicaciones a tal efecto, desde simples analizadores léxicos en forma de scripts en Perl que transforman el fichero texto con todas las alarmas a un documento HTML que podemos visualizar con el navegador, hasta complejos programas escritos en PHP que realizan consultas en nuestra base de datos y permiten interactuar con ella por medio de *clics* de ratón.

Entre los candidatos se encontraban Demarc (`www.demarc.com`), IDS Center y SnortReport. Demarc fue desestimado por no ser gratuito, y entre IDS Center y SnortReport se optó por SnortReport por utilizar MySQL como back-end.

SnortReport es un conjunto de scripts PHP, por lo que para su uso necesitamos un servidor web en el propio host. Se eligió Apache como servidor web por ser el de mayor difusión en entornos Linux.

Esta elección nos ha traído dos nuevo problemas. Por una parte añadimos un servicio (el servidor web) que en un principio era innecesario para un IDS, lo cual aumenta el esfuerzo de administración de la máquina y añade una potencial vulnerabilidad en el puerto 80, aunque esto último puede ser corregido si utilizamos un *tcpwrapper* o un firewall interno, añadiendo eso sí más configuración sobre la máquina y privando al administrador de hacer una gestión remota de los incidentes de seguridad. Por otra parte hemos cargado la máquina con un nuevo servicio, el servidor web. Esta vez, en cada consulta, la máquina deberá repartir dos procesadores entre el Snort, MySQL y Apache (que deberá ejecutar el script PHP para devolver HTML), aunque una posible solución a este último problema sería la de dar mayor prioridad a Snort sobre el resto de procesos en ejecución.

Por último y puesto que nuestro back-end es una base de datos, utilizaremos Barnyard para optimizar el proceso de almacenamiento de las alertas. Este software, que se ejecuta en un proceso independiente, hace de intermediario entre Snort y MySQL, con lo que podremos descargar el procesador que corre Snort y realizar en paralelo la escritura en la base de datos.

2.4. Software a implementar

Al inicio del proyecto el software a implementar consistía únicamente en un sistema de gestión automática de alertas. Pronto se vió la necesidad de conocer el rendimiento al que podía llegar la máquina, por lo que se decidió diseñar un software que permitiese monitorizar varios parámetros para determinar el nivel de carga de la máquina, y extrapolar así el tráfico a partir del cual el sistema empezaría a descartar paquetes.

2.4.1. Gestor de incidencias

El gestor de incidencias debía ser un programa que facilitara la gestión de las incidencias al responsable de seguridad, con las siguientes características:

- La gestión de las incidencias se haría vía correo electrónico. Por una parte, el gestor recogería información sobre los incidentes ocurridos durante el día, en un momento en el que la máquina estuviese poco cargada (a las 4 a.m., por ejemplo). Estos incidentes serían enviados por e-mail al responsable de seguridad, el cual marcaría de alguna forma en el mail cuales son los incidentes que se quieren reportar al CERT y cuales al administrador de la red que los ha causado.
- Se ha de tener la posibilidad de indicar que no queremos que nos envíe en el mail unas alertas determinadas, sin tener para ello que eliminarlas del conjunto de reglas de Snort. La forma de indicarlo podría ser mediante la edición de un fichero texto, como es norma en los sistemas tipo Unix.
- Debe ser portable para su fácil migración a otras plataformas, escrito en un lenguaje interpretado tipo script (Perl, PHP) y con la máxima independencia sobre la base de datos que utilice el NIDS.

Se decidió implementar el gestor de incidencias en Perl, lenguaje que compila en plataformas Unix y Windows sin necesidad de hacer cambio alguno en el código fuente, y que dispone de módulos de bases de datos (conocidos como DBI, o *Database Interconnection*) que permiten el cambio de base de datos en los programas con muy pocas modificaciones.

2.4.2. Benchmark

Como se ha comentado antes, la necesidad de una herramienta que midiese el rendimiento del sistema surgió durante la fase de diseño del proyecto.

La idea consistía en una aplicación gráfica flexible que permitiera recoger contadores variados del sistema, en concreto, los paquetes por segundo recibidos por el interfaz de red del IDS de todas las fuentes posibles para poder hacer comparaciones, los paquetes por segundo recibidos y los procesados por Snort, el tamaño en memoria del proceso y la CPU que ha consumido. La aplicación tomaría cada

intervalo de tiempo una muestra de todos estos contadores, y ofrecería la posibilidad de cambiar este intervalo para la visualización de las gráficas, obteniendo de esta forma un suavizado natural. Además podría incorporar otras técnicas de suavizado más avanzadas.

Todos estos datos servirían más tarde para la elección de la máquina apropiada que necesita nuestra red. Por ejemplo, el descarte de paquetes es una evidencia de que la máquina no es lo suficientemente potente, ya sea por la CPU o por otras causas. Es posible que la CPU de la que disponemos sea suficiente para el tráfico y las firmas que tenemos configuradas, pero si la cantidad de memoria física es menor de la que necesita Snort[1], se puede producir un exceso de fallos de página que haga caer el rendimiento de Snort drásticamente.

También nos dará una idea de lo útil que será Snort en nuestra red; si el descarte de paquetes es frecuente producirá un gran número de falsos negativos, lo cual puede aconsejar al uso de un repartidor de carga para IDSs, por ejemplo.

2.5. Diseño del sistema

El diseño del sistema y del software a desarrollar se ha hecho mediante UML (*Unified Modeling Language*). Se ha modelado el sistema completo mediante un diagrama de clases, que describe las partes principales que lo componen y como interacciona cada una de ellas.

El software se ha modelado mediante diagramas de secuencia, donde vemos la ordenación temporal de los mensajes entre los objetos. De esta forma describimos la vista dinámica de las aplicaciones.

En el apéndice A se pueden encontrar los diagramas de diseño del sistema y de las aplicaciones.

2.6. Presupuesto

El presupuesto de este proyecto lo podemos encontrar en el apéndice A.

[1]El consumo de memoria en Snort suele ser alto cuando el número de conexiones TCP es alta, puesto que debe mantener estado para todas ellas. Además, las estructuras de datos que utiliza Snort para reordenar paquetes IP y reensamblar fragmentos son grandes y complejas, ya que el objetivo fundamental del diseño no fue minimizar el tamaño en memoria del proceso sino maximizar la velocidad del motor de búsqueda.

Capítulo 3

Desarrollo

3.1. Creación de una política de seguridad

Como objetivo adicional del proyecto se decidió la creación de una política de seguridad en la que enmarcar el sistema. Para ello nos hemos basado en un documento bastante popular para la creación de políticas de seguridad [10]. Dependiendo del riesgo existente en la Universidad, los administradores deberán elegir una de las siguientes tres políticas. Por supuesto, se pueden desechar algunos puntos e incorporar otros nuevos, es totalmente flexible a cambios y se puede adecuar a las necesidades de la organización.

3.1.1. Riesgo bajo

Implementación

- Los procesos de logging de las aplicaciones y los sistemas operativos deben estar activados en todos los hosts y servidores.
- Las funciones de alarma y alerta de los firewalls y otros dispositivos de control de acceso al perímetro deben estar activados.

Administración

- Se debe instalar software para el chequeo de la integridad de los sistemas de ficheros en los firewalls y otros sistemas de control de acceso al perímetro.
- Se deben revisar diariamente los logs en los sistemas de control de acceso al perímetro.
- Se deben revisar semanalmente los logs de los hosts y servidores que se encuentran en la red interna.
- Se debe entrenar a los usuarios para que avisen de cualquier anomalía en el rendimiento del sistema a los administradores.
- Todos los problemas que reciban los administradores serán revisados en busca de síntomas que indiquen actividad intrusiva. Los síntomas sospechosos deberán ser comunicados al personal de seguridad.

3.1.2. Riesgo medio

Implementación

- Los procesos de logging de las aplicaciones y los sistemas operativos deben estar activados en todos los hosts y servidores.
- Las funciones de alarma y alerta de los firewalls y otros dispositivos de control de acceso al perímetro deben estar activados.
- Todos los servicios críticos deben ser monitorizados por herramientas adicionales como Tripwire o *tcpwrappers* instaladas apropiadamente, como suplemento de la actividad de logging que incorpora el sistema operativo.

Administración

- Se debe instalar software para el chequeo de la integridad de los sistemas de ficheros en los firewalls y otros sistemas de control de acceso al perímetro.
- Se deben revisar diariamente los logs en los sistemas de control de acceso al perímetro.
- Se deben revisar semanalmente los logs de los hosts y servidores que se encuentran en la red interna.
- Se debe entrenar a los usuarios para que avisen de cualquier anomalía en el rendimiento del sistema a los administradores.
- Todos los problemas que reciban los administradores serán revisados en busca de síntomas que indiquen actividad intrusiva. Los síntomas sospechosos deberán ser comunicados al personal de seguridad.
- Las herramientas HIDS como Tripwire serán revisadas periódicamente.
- El personal de seguridad de red establecerá relaciones con otras organizaciones dedicadas a la respuesta de incidentes, como FIRST, y compartirán información relevante como incidentes y vulnerabilidades.
- Cuando se produzca una intrusión, a menos que los sistemas críticos hayan sido comprometidos, la organización intentará primero recavar pruebas sobre los intrusos antes de reparar los sistemas, buscando más información de quién y cómo se produjo la intrusión. Esta persona debe ser entrenada en las vías legales para reportar una intrusión.
- Se permitirá que determinados agujeros de seguridad queden sin corregir controladamente para el estudio de los atacantes y sus técnicas (’honeypots’).

3.1.3. Riesgo alto

Implementación

- Los procesos de logging de las aplicaciones y los sistemas operativos deben estar activados en todos los hosts y servidores.

- Las funciones de alarma y alerta de los firewalls y otros dispositivos de control de acceso al perímetro deben estar activados.

- Todos los servicios críticos deben ser monitorizados por herramientas adicionales como Tripwire o *tcpwrappers* instaladas apropiadamente, como suplemento de la actividad de logging que incorpora el sistema operativo.

- Todos los servicios críticos deben tener además herramientas redundantes de detección de intrusos instaladas, la cuales operen con principios diferentes a los de las herramientas primarias ya instaladas. Por ejemplo, si el HIDS primario es Tripwire, el secundario se podría basar en detección de anomalías.

- En los puntos de concentración de la red se instalaran NIDSs que monitoricen el tráfico en busca de patrones conocidos de ataques.

Administración

- Se debe instalar software para el chequeo de la integridad de los sistemas de ficheros en los firewalls y otros sistemas de control de acceso al perímetro.

- Se deben revisar diariamente los logs en los sistemas de control de acceso al perímetro.

- Se deben revisar semanalmente los logs de los hosts y servidores que se encuentran en la red interna.

- Se debe entrenar a los usuarios para que avisen de cualquier anomalía en el rendimiento del sistema a los administradores.

- Todos los problemas que reciban los administradores serán revisados en busca de síntomas que indiquen actividad intrusiva. Los síntomas sospechosos deberán ser comunicados al personal de seguridad.

- Las herramientas HIDS como Tripwire serán revisadas periódicamente.

- El personal de seguridad de red establecerá relaciones con otras organizaciones dedicadas a la respuesta de incidentes, como FIRST, y compartirán información relevante como incidentes y vulnerabilidades.

- Cuando se produzca una intrusión, a menos que los sistemas críticos hayan sido comprometidos, la organización intentará primero recavar pruebas sobre los intrusos antes de reparar los sistemas, buscando más información de quién y cómo se produjo la intrusión. Esta persona debe ser entrenada en las vías legales para reportar una intrusión.

- El NIDS será revisado periódicamente para su correcto funcionamiento.

- La organización intentará perseguir a los intrusos pero no permitirá que no se corrijan agujeros de seguridad para el estudio de los atacantes.

3.2. Configuración y testeo de la máquina

3.2.1. Actualización del sistema

La máquina venia con una versión primitiva de Linux, la RH6.2. Para las primeras pruebas de monitorización (que utilizaban la tarjeta Fast Ethernet), no se requirió la actualización del software. Fue en febrero cuando se hizo la primera actualización del núcleo de la versión 2.2.18pre11 a las 2.4.5, que incluía controladores para la tarjeta ATM. Además, incluía un uso más eficiente de las interrupciones, pudiendo asignar una CPU a la atención de una interrupción dada, con lo que, en teoría, se aprovecharía la localidad espacial del controlador y su código no tendría que copiarse continuamente entre las caches de los procesadores, mejorando el rendimiento.

La asignación de IRQs a CPUs la podemos configurar por ejemplo con los siguientes comandos:

```
# cd /proc/irq/15
# echo 02 >smp_affinity
```

En este caso, hemos asignado la interrupción 15 al procesador 2. El 02 que aparece en el comando hay que leerlo en binario, es decir, 10. Al estar el segundo bit a 1 significa que la CPU 2 será la elegida. En el caso de tener un multiprocesador de 16 CPUs, si queremos que la IRQ 44 sea ejecutada solo por los primeros procesadores utilizaremos lo siguiente:

```
# cd /proc/irq/44
# echo 0f >smp_affinity
```

Para la actualización del núcleo se tuvo en cuenta que la máquina era VA-Linux. Esta empresa distribuye sus máquinas con una versión modificada del núcleo. Los parches se pueden descargar gratuitamente de su página web (`www.va-linux.com`), pero no están al día con respecto a los núcleos de Linux en `www.kernel.org`, por lo que no se pudo actualizar al último núcleo de entonces que era el 2.4.18.

Para la actualización se siguieron estos pasos:

- Descarga e instalación de la versión 2.4.5 del núcleo.
- Descarga e instalación de los parches VA-Linux de `ftp://ftp.vafoftware.com/pub/kernel/2.4.5/`.
- Parcheado del núcleo:

  ```
  # cd /usr/src/linux
  # for f in *.patch; do patch -p1 <$f; done
  ```

- Copia del fichero `.config`:

  ```
  # cp ../config/i686-smp-piii .config
  ```

- Configuración del núcleo para darle soporte al interfaz ATM

  ```
  # make xconfig
  ```

- Compilación e instalación

```
# make dep && make bzImage
<edición de /etc/lilo.conf>
# lilo
<reiniciar equipo>
# cd /usr/src/linux
# make modules && make modules_install
```

Además del núcleo, hubo que actualizar la librería `libpcap`, de la que hace uso Snort para la captura de tráfico. La versión que traía la RH6.2 era la 0.4, que no soporta la capa de enlace con identificador 18, esto es, ATM. La versión más reciente era la 0.7.1, y para instalarla se siguieron los siguientes pasos:

- Descargar `libpcap` 0.7.1 de `www.tcpdump.org`.
- Compilar e instalar la librería:

```
$ ./configure --prefix=/usr
$ make
# make install
```

- Construcción de la librería dinámica libpcap.so a partir de la librería estática libpcap.a

```
$ gcc -shared -fPIC -o lib.so pcap-linux.c pcap.c inet.c \
gencode.c optimize.c nametoaddr.c etherent.c savefile.c \
bpf_filter.c bpf_image.c bpf_dump.c scanner.c grammar.c version.c
```

Pese a todas estas actualizaciones, el 1 de marzo de 2002 fue necesario un cambio de versión en la distribución de Linux. Se pasó de RH6.2 a RH7.2, ya que se necesitaba la librería de C glibc2.2.4 para las herramientas ATM. Al ser una librería tan ligada a la distribución, se optó por dar el paso que con las actualizaciones anteriores se había intentado evitar.

3.2.2. Configuración de red

Para la primera fase del proyecto (monitorización de un servidor proxy y una VLAN), la máquina disponía de dos interfaces Fast Ethernet: uno de gestión con una IP pública que permitía hacer una administración remota del IDS, y otro de monitorización sin dirección IP, del que Snort monitorizaba y analizaba el tráfico. Se utilizó la tarjeta simple para la gestión y un interfaz de la doble para monitorización. Para la administración remota se habilitó el servicio SSH. Se eligió SSH frente a Telnet por permitir comunicación codificada y evitar así que un atacante averiguase los passwords del sistema escuchando la conexión.

En febrero se pasó a la monitorización del acceso a Internet de la Universidad de Valencia. Para ello, RedIRIS envió una tarjeta ATM FORE PCA-200E. Hubo que quitar la tarjeta doble Fast Ethernet porque no había espacio suficiente en el chasis del equipo para albergar las dos tarjetas.

Para el uso de ATM sobre Linux, además de los controladores de la tarjeta en el núcleo, se necesitaban un conjunto de herramientas que permitían la creación de interfaces y su asociación a PVCs.

La versión elegida de estas herramientas fue la 2.4.0 y se pude descargar de `http://linux-atm.sourceforge.net/dist.php`. El proyecto que las mantiene se llama “ATM on Linux” [24].

El paquete contiene, entre otros, los siguientes comandos:

- `atmsigd`: demonio de señalización UNI, necesario para establecer SVCs.
- `ilmid`: demonio que implementa la asignación de direcciones ATM.
- `atmarpd`: demonio que implementa el protocolo ATMARP especificado en RFC1577 y RFC1755.
- `atmarp`: comando que asigna un PVC a un interfaz.

A modo de ejemplo, para la configuración de un PVC punto a punto con VPI/VCI 0.120 entre la interfaz ATM de la máquina Linux con dirección IP 192.168.1.6/30 y un host o router con la 192.168..1.5/30, haríamos:

1. `# atmarpd -b`
2. `# atmarp -c atm0`
3. `# ifconfig atm0 192.168.1.6 netmask 255.255.255.252 up`
4. `# atmarp -s 192.168.1.5 0.120`

En 1 lanzamos el demonio ARP, que no tiene nada que ver con el servidor ARP de IPOA, sino que es utilizado por el comando `atmarp` para la resolución de direcciones IP. En 2 creamos la interfaz `atm0` de la que podremos leer paquetes IP. Con esto conseguimos que las aplicaciones como Snort sigan funcionando de forma transparente, al no tener que entenderse con la capa ATM. En el paso 3 le damos una IP al interfaz y en 4 añadimos en la tabla ARP de ATM una entrada con la asignación del PVC 0.120 a la IP 192.168.1.5; como esta dirección pertenece a la misma subred que 192.168.1.6, sabe que la puede encontrar por la interfaz `atm0`.

La elección del VP 0 del PVC se ha hecho porque esta tarjeta ATM no soporta otros valores de VP, lo que puede dar más de un quebradero de cabeza al comprobar que no recibimos tráfico si elegimos otro valor distinto de 0.

Por último comentar que como hemos utilizado únicamente PVCs no es necesario el demonio de señalización `atmsigd` ni tampoco el de protocolo de asignación de direcciones en ATM, `ilmid`.

3.2.3. Seguridad de la máquina

Puesto que el IDS es capaz de monitorizar todo el trafico que la Universidad tiene hacia el exterior, una intrusión en él puede tener consecuencias desastrosas para la red, por lo que se hace necesario reforzar la seguridad de la máquina.

3.2.3.1. Eliminación de servicios innecesarios.

Se han eliminado todos los servicios que lanza la instalación por defecto de RH7.2 a excepción del servidor SSH, para hacer posible la administración remota. Además, y por necesidades del módulo de Perl que interactúa con la base de datos, CPAN, se ha lanzado el servidor de base de datos MySQL escuchando en un puerto TCP. El último servicio que se configuró fue el servidor web Apache escuchando del puerto 80. Para evitar accesos no autorizados, se protegió el acceso a la página por defecto por contraseña.

Más tarde, y como luego comentaremos, dejó de ser necesario el servidor web, por lo que se eliminó su entrada de los scripts de arranque.

3.2.3.2. Instalación de Tripwire

Tripwire (`www.tripwiresecurity.com`) es una aplicación que detecta cualquier cambio en los ficheros del sistema. Para ello crea primero una base de datos con los ficheros que hemos seleccionado y su compendio criptográfico. El administrador puede luego comprobar la integridad del sistema volviendo a calcular los compendios y comparándolos con los originales, previamente almacenados en un lugar seguro fuera de la máquina.

Después de la instalación de la nueva versión del sistema operativo, se instaló Tripwire y se almacenó la base de datos generada en un diskette. Para que la comparación entre compendios sea automática, se puede montar el diskette en modo solo lectura (no montando la unidad en modo solo lectura, sino físicamente, moviendo la pestaña del diskette) y ejecutando periódicamente el motor de Tripwire con una utilidad como *cron*. El problema es que esto puede ser descubierto por un posible atacante que logre entrar a la máquina como superusuario antes de que notemos su intrusión, y podría forzar a algún tipo reacción violenta por parte del atacante.

3.2.3.3. Configuración del firewall interno

Linux incorpora un firewall interno llamado Netfilter (`http://netfilter.samba.org`) que se ha configurado de forma que permite:

- Registrar cada intento de conexión TCP entrante en un fichero de logs (esto lo podemos hacer con Snort, como veremos a continuación).
- Rechazar con segmentos RST cualquier intento de conexión por el interfaz `eth0` a un puerto distinto al SSH.
- Rechazar con ICMP Port Unreachable cualquier datagrama UDP.
- Descartar todos los segmentos TCP y datagramas UDP que entren por el interfaz de monitorización.

Enviando segmentos RST e ICMPs Port Unreachable conseguimos simular un puerto cerrado, por lo que la máquina no se verá como filtrada desde el exterior y no tendrá tanta importancia.

Por último, aunque el interfaz `atm0` es únicamente de monitorización, si desde fuera de la universidad se intenta una conexión al IDS, ésta llega primero por el interfaz ATM y más tarde por el FE. Si

se hace a un puerto distinto del SSH, la interfaz FE la rechazará pero entrará al núcleo por el ATM, por lo que responderá a ese intento de conexión dependiendo del estado del puerto. Para evitar esta situación hemos añadido reglas que descarten paquetes hacia el IDS en el interfaz ATM. Estas reglas añaden el inconveniente de que el IDS no detectará ataques dirigidos a él mismo, pero esto no tiene porque ser un problema cuando el resto de medidas de seguridad está al día, como la actualización de los servicios y la revisión frecuente de los logs.

En el apéndice B, apartado Netfilter, se encuentra la secuencia de comandos *iptables* que se ha configurado en el IDS.

3.2.4. Configuración de Snort

Snort lo podemos descargar de `www.snort.org/dl/snort-1.8.7.tar.gz` (la última versión). Para instalarlo seguiremos estos pasos:

- Descomprimir con `tar -zxf snort-1.8.7.tar.gz`
- Configurar con `./configure --with-mysql --enable-flexresp`
- `make && make install`

El fichero principal de configuración de Snort se llama `snort.conf` y contiene los preprocesadores a cargar y sus parámetros, los plugins activados en salida y los ficheros de reglas a incluir en el motor de búsqueda.

Existen multitud de tipos de reglas en Snort, entre los que cabe destacar:

- Respuesta a ataques: estas reglas normalmente producen alertas cuando una máquina han sido comprometida, por ejemplo, devolviendo el identificador de usuario como root, o devolviendo un listado del directorio en servidores web.
- Exploits: estas reglas son actualizadas con frecuencia y evidencian más claramente un ataque intrusivo. Entre ellas, las más conocidas son el exploit al servidor SSH, al servidor de impresión en RedHat 7.0 y el exploit por overflow al servidor IMAP.
- DDoS y DoS: Detecta ataques DDoS como Stacheldraht, Trin00, TFN y DoS como Teardrop, Land-to-Land y Winnuke.
- Virus: un compendio algo anticuado (y en busca de alguien que se encargue de mantenerlo) de casi unos 100 virus.
- Política: firmas dedicadas a ayudar a mantener la política de seguridad de una organización. Detectan actividad de aplicaciones P2P, IRC y telepresencia, entre otras.
- Web-IIS: un total de 91 firmas en Snort 1.8.7beta2 dedicadas exclusivamente al servidor IIS de Microsoft.
- Shellcode: basadas en el shellcode común entre múltiples exploits de uso público, como operaciones NOOP y llamadas al sistema en distintas plataformas hardware y software. Estas los patrones de estas firmas son buscados en todos los puertos, por lo que activando estas firmas (no cargadas por defecto) el rendimiento cae estrepitosamente.

- NetBIOS: actividad sospechosa de NetBIOS, como acceso al directorio '...' o propagación del gusano Nimda.

Se han utilizado las reglas que vienen con Snort por defecto en la instalación, unas 1700 en la versión 1.8.7beta2, y se han cargado todas (cosa que no viene por defecto) en `snort.conf` como peor de los casos de carga máxima que la máquina deberá soportar. Más tarde, cuando el IDS sea gestionado por administradores de la red de la Universidad como sistema de producción real, se pueden eliminar el conjunto de reglas que no se considere oportuno.

Se añadió una regla nueva que registraba cada intento de conexión. La regla es la siguiente:

```
alert tcp $EXTERNAL_NET any ->$HOME_NET any (msg:"TCP SYN received";
flags:S; classtype:misc; sid:3324; rev:1;)
```

También se modificaron algunas reglas para que respondieran a ataques, como es el caso de las reglas que detectaban la presencia de un CodeRed o el Nimda. Estas reglas necesitan del módulo flexresp, incluido en Snort en la etapa de configuración con `./configure`. Como ejemplo, a la siguiente regla que detecta un acceso al fichero root.exe por parte del gusano CodeRed v2, se le ha configurado para que cierre la conexión establecida en ambos extremos, lo cual evita que el gusano siga su infección.

```
alert tcp $EXTERNAL_NET any ->$HTTP_SERVERS 80 (msg:"WEB-IIS CodeRed
v2 root.exe access"; flags: A+; uricontent:"scripts/root.exe?"; re-
sp: rst_all; nocase; classtype:web-application-attack; sid: 1256;
rev:2;)
```

Para conseguir esto, Snort utiliza una técnica conocida como *spoofing* y muy utilizada por los intrusos, que consiste en falsear la dirección IP origen de los paquetes que enviamos, algo trivial en IPv4, pero que no suele funcionar muy bien cuando monitorizamos medios dedicados muy rápidos, ya que para cuando Snort ha enviado su TCP RST con el número de secuencia adecuado, los hosts implicados en la conexión ya han avanzado en su conversación y descartan los segmentos que Snort les envió.

Durante el tiempo en el que se estuvo monitorizando la VLAN de informática el 90 % del tiempo dedicado a la administración del IDS se dedicó a eliminar las reglas que generaban excesivas falsas alarmas. Estas reglas solían pertenecer al grupo que se encuentra en `icmp-info.rules`, `icmp.rules` y `misc.rules`. y frecuentemente eran ICMPs Echo y Reply procedentes de un monitorizador de red llamado huron.ci.uv.es, o ICMPs Port Unreachable procedentes del resto de hosts hacia huron, como resultado de un puerto UDP cerrado monitorizado por huron. También aparecían numerosos escaneos de puertos falsos con destino a los servidores Proxy y DNSs, por lo que al final se optó por deshabilitar el preprocesador spp_portscan, porque llenaba innecesariamente la base de datos y producía mucha basura.

No se puede eliminar la regla directamente sin antes ver que produce demasiadas falsas alarmas. Por ejemplo, eliminar todas las reglas que generen alertas con tráfico ICMP no sería válido, porque muchas veces, sobre todo para análisis forense, se necesita la máxima inforamación posible. Si estamos intentado seguir la pista a un atacante, es posible que éste haya hecho pings a nuestras máquinas antes. Estos pings no suponen un ataque a nuestra red, pero a veces se usan para recabar información sobre nuestra red, por lo que es una buena idea guardarlos en nuestra base de datos.

Algunas veces, (en versiones beta normalmente), Snort caía sin dar ningún tipo de mensaje. Al hacerlo dejaba sin protección la red de la Universidad, por lo que se creó un envolvente que lanzaba Snort cuando éste había caído y lo notificaba en un fichero de logs.

Para mejorar el rendimiento de Snort, se ha utilizado un programa llamado Barnyard que se encarga de liberar a Snort del trabajo que supone escribir las alertas y los logs en un formato dado, por ejemplo, en la base de datos. Barnyard lee ficheros de salida de Snort en formato unificado y los traduce a la salida que queramos. En nuestro caso, esa transformación consiste en la inserción de los logs en MySQL. Para esta configuración, debemos añadir en `snort.conf` estas líneas:

```
output alert_unified: filename snort.alert, limit 128
output log_unified: filename snort.log, limit 128
```

Con esto le decimos que las alertas y logs los escriba al fichero snort.alert y snort.log respectivamente y que lo haga en formato unificado. Ahora basta con lanzar Barnyard diciéndole que lea de esos ficheros y que haga la transformación a MySQL. Esto se lo indicamos en el fichero `barnyard.conf`:

```
processor dp_alert
processor dp_log
processor dp_stream_stat
output log_acid_db: mysql, sensor_id 1, database snortdb, server lo-
calhost, user
snort, password xxxx, detail full
```

Con estas líneas le estamos indicando que use los plugins de entrada `dp_alert`, `dp_log` y `dp_stream_stat` y que escriba utilizando el plugin de salida de MySQL. Se pueden lanzar tantas instancias de Barnyard como sean necesarias. Es conveniente hacerlo cuando tenemos varios plugins de salida y una máquina multiprocesador, ya que así conseguimos paralelizar en el tiempo el proceso de captura de alertas.

Durante el funcionamiento de Barnyard se consiguió almacenar las alertas en MySQL pero no hubo rastro de paquetes IP. Pese a que la configuración parecía correcta, no llegó a funcionar. Se enviaron varios mensajes a la lista de distribución de Snort pero no se obtuvieron respuestas, así que es posible que debido a que Barnyard está todavía en una fase muy temprana de su desarrollo, no sea posible almacenar en la base de datos información detallada sobre los paquetes que producen alarmas.

3.2.5. Configuración de MySQL

Se ha instalado la versión 11.15 distribución 3.23.44 de MySQL a partir de su RPM. Para lanzarlo se utiliza el binario `safe_mysql` que viene incluido en el paquete como se indica a continuación[1]:

```
#safe_mysql --skip_networking --datadir=/home/mysql/
```

Hemos cambiado el directorio de la base de datos a `/home/mysql` por la forma en la que estaban distribuidas las particiones de disco: la partición que se monta en `/home` tiene un tamaño del 87 % del total del disco duro, mientras que el de la partición que se monta en `/var`, el directorio por defecto para almacenar la base de datos, es del 1.5 %, claramente insuficiente. El resto de espacio estaba dedicado a otras particiones del sistema.

[1] La opción `--skip_networking` tuvo que eliminarse más tarde al utilizar el módulo DBI de Perl, tal y como se cuenta en 3.4.3.

Para conectar con la base de datos, algunos clientes buscaban el socket en un lugar distinto al que el servidor MySQL lo abría. Para evitar esto, se editó el fichero de configuración `/etc/my.cnf` para que todos los clientes de MySQL encontraran el socket en el lugar en el que `mysqld` lo había abierto:

```
[client]
socket=/tmp/mysql.sock
[mysqld]
socket=/tmp/mysql.sock
```

Por alguna razón, aunque en el fichero de configuración ya le indicamos al cliente donde encontrar el socket que le conecta al servidor, hemos de crear la variable de entorno `MYSQL_UNIX_PORT=/tmp/mysql.sock` antes de lanzar el cliente.

En algunas ocasiones la máquina caía por fallo del suministro eléctrico. Para recuperar la base de datos de un estado inconsistente podemos ejecutar el siguiente comando dentro del directorio donde se encuentran las tablas que queremos restaurar:

```
# for f in *.MYI; do myisamchk -o $f; done
```

`myisamchk` es la herramienta de MySQL utilizada para reparar tablas corruptas. Con la opción `-o` le indicamos que repare, aunque no lo hará si la tabla no lo necesita.

3.2.6. Configuración de Apache+PHP

El servidor web Apache fue necesario para el programa en PHP SnortReport. Por defecto, el paquete RPM de Apache que viene con RedHat no lleva incorporado el módulo PHP, por lo que hay que recompilar el servidor web con este módulo de la siguiente forma:

1. Configuración de PHP
   - Descargar el módulo de `www.php.net/downloads.php`.
   - Descomprimir con `tar -zxf php-4.2.1.tar.gz`
   - `./configure --with-mysql --with-apche=../apche_1.3.x --with-gd`
   - Copiar `php.ini_dist` a `/usr/local/lib/php.ini`
   - Modificaciones en php.ini para que funcione la librería gráfica `gd` de la que hace uso en SnortReport.
     - Comentar: `#define HAVE_GD_GIF 1`
     - Añadir: `#define HAVE_GD_JPG 1`
   - `make && make install`
2. Configuración de Apache
   - Descargar de `www.apache.org/dist/httpd/`.
   - Descomprimir con `tar -zxf apache_1.3.26.tar.gz`
   - `./configure --activate-module=src/modules/php4/libphp4.a`
   - Descomentar las siguientes líneas en `http.conf`:

    - AddType php
    - AddType php-source
  - make && make install

Una vez instalado colocamos los scripts PHP de SnortReport en el directorio /usr/local/httpd/httpdocs. Es conveniente proteger el acceso a este directorio por contraseña. Esto lo podemos hacer añadiendo a httpd.conf lo siguiente:

```
<Directory "/usr/local/httpd/httpdocs/snortreport">
AllowOverride All
</Directory>
```

Además, tenemos que indicar donde se encuentran los ficheros de contraseñas. Esto lo hacemos con el fichero .httpaccess:

```
AuthUserFile /usr/local/httpd/httpdocs/snortreport/.htpasswd
AuthGroupFile /dev/null
AuthName "Snort Admin"
AuthType Basic

Options All
<Limit GET POST PUT>
require valid-user
</Limit>
```

Ahora basta con crear el usuario en el fichero de contraseñas de Apache con:

```
# htpasswd -c /usr/local/httpd/httpdocs/snortreport/.htpasswd snor-
tadm
New password:
Re-type new password:
Adding password for user snortadm
```

3.3. Implantación del IDS

Como vimos en el capítulo 2, existen varias posibles ubicaciones del sensor. En este proyecto se partió de una configuración sencilla, en la que el IDS monitorizaba un único host. Poco a poco, se fue complicando, aumentando el número de host monitorizados hasta abarcar todos los hosts de la Universidad de Valencia.

3.3.1. Sesión SPAN en un conmutador Catalyst 5500

Mediante sesiones SPAN se hizo el primer bloque de pruebas, que consistió en la monitorización de un servidor proxy y de una VLAN. Las pruebas se iniciaron a principios de noviembre de 2001 y terminaron en febrero de 2002, cuando recibimos la tarjeta ATM de RedIRIS.

3.3.1.1. Monitorización de un servidor proxy

La primera prueba de monitorización se realizó a principios de noviembre. Se hizo configurando una sesión SPAN con puerto origen el del servidor proxy *proxy2.uv.es* y destino la máquina Linux con Snort. Como la Universidad tiene cerrado el puerto 80 en el sentido de salida y obliga al tráfico web a salir por el proxy, se pensó que monitorizando este servidor se analizaba gran parte de tráfico que tenia la Universidad hacia Internet.

El servidor *proxy2.uv.es* tiene dos interfaces, uno interno a la Universidad y otro externo. Se eligió el interno porque era el interfaz por el que circulaba un mayor volumen de tráfico[2], que era lo que nos interesaba para medir el rendimiento del sistema. De esta forma estábamos analizando tanto las peticiones web que los clientes de la Universidad hacían hacia Internet como cualquier tráfico dirigido hacia el proxy desde el interior (pings, ataques a servicios abiertos, etc.), pero no el tráfico que se recibía del exterior.

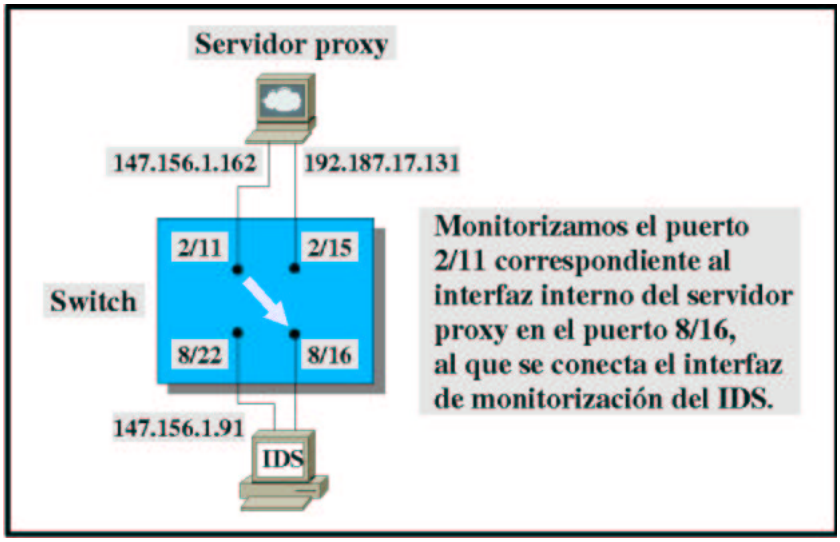

Figura 3.1: Monitorización del servidor proxy mediante una sesión SPAN.

En la figura 3.1vemos el esquema de la sesión SPAN con puerto origen 2/11 y destino 8/16 en un conmutador Catalyst 5500. Para establecer esta sesión se debe introducir el siguiente comando en el conmutador:

```
set span 2/11 8/16
```

Obsérvese que el puerto 8/16 no necesita tener asignada una dirección IP. El puerto 8/22 del conmutador corresponde a la interfaz de gestión del IDS, situado en la VLAN 20, la de servidores. Este puerto no desempeña ningún papel en la función de monitorización del tráfico.

[2]Puesto que *proxy2.uv.es* es un servidor proxy-caché, cuando recibe una petición interna por primera vez, envía esa petición hacia el exterior. Una vez le llega la respuesta, además de reenviarla al host que se la pidió, guarda una copia. Más tarde, cuando se produzcan peticiones a ese mismo objeto, si no se ha sobrepasado un tiempo en el que se considera que ha caducado la copia, se responde la petición desde la propia caché, con lo que estamos reduciendo el tráfico que sale al exterior.

3.3.1.2. Monitorización de una VLAN

A finales de noviembre se pasó a monitorizar una VLAN. Se empezó con la VLAN de servidores (VLAN 20), cuyos hosts vemos en la tabla 3.1:

Puerto	Host (uv.es)		Puerto	Host (ci.uv.es)
2/2	furgoneta		2/14	laos
2/3	tiberio		2/16	ardemuro
2/5	peque		2/17	proxy1
2/6	arcon		2/18	sereno
2/8	nebula.sib		2/19	sello
2/9	vpn		2/21	director
2/10	cdrom2.sib		2/23	bancuv
2/11	proxy2		8/3	cesar
2/12	cdrom.sib		8/15	movie
2/13	dkw		8/22	discreto

Cuadro 3.1: VLAN de servidores de la Universidad de Valencia.

Para establecer la sesión SPAN de la VLAN 20 en el puerto 8/16 el comando en el conmutador era:

```
set span 20 8/16
```

De esta forma estamos enviando una copia de todos los paquetes que salen y entran por todos los puertos que pertenecen a la VLAN 20 hacia el puerto 8/16. Este puerto no recibirá tráfico del IDS, por ser esta la configuración por defecto[3]. Además, incluirá todo el tráfico multicast y de spanning tree. Es posible que un paquete que entre y salga de la misma VLAN sea recibido dos veces.

En diciembre se pasó a monitorizar la VLAN del semisótano de farmacia (VLAN 32). En el edificio de farmacia hay dos VLANs, la propia de farmacia y la del departamento de informática, localizado en el sótano del edificio. La conexión entre el Catalyst 5500 situado en el servicio de informática y el Cisco 4912 situado en el edificio de farmacia es Gigabit y está configurada en modo trunk para permitir la conexión de las dos VLANs a través del router, una tarjeta de routing en el propio conmutador, tal y como se ve a la figura 3.2:

Dado que el interfaz del IDS era Fast Ethernet, el puerto destino de la sesión SPAN tenia que ser a 100Mbps, lo cual hace imposible que el puerto origen de la sesión sea Gigabit. Se optó por el montaje de la figura 3.2, en el que vemos principalmente tres cambios. Primero se cambió el puerto del Catalyst a uno de 100Mps en fibra (puerto 6/3) para permitir el SPAN hacia el puerto de monitorización del IDS. Segundo, se añadió un conmutador Cisco 3524 con interfaces a 100Mbps en cobre y Gigabit en fibra, y se conectó un Gigabit a la fibra que llegaba de farmacia. Puesto que este conmutador no tenía ningún interfaz a 100Mbps en fibra para conectar directamente con el Catalyst, se añadió un hub 10/100Mbps con interfaces en fibra y en cobre.

[3] Si queremos que este puerto pueda reenviar tráfico proveniente del IDS, deberemos añadir `inpkts enable` al final del comando span anterior.

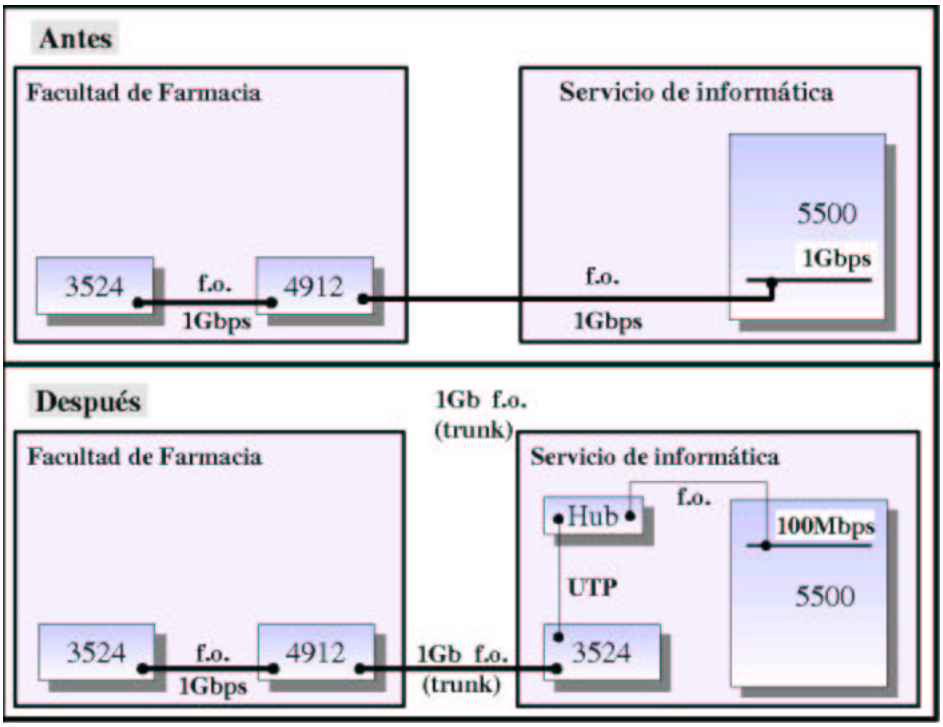

Figura 3.2: Cambios realizados en la infraestructura de red para la monitorización de la VLAN 32.

Pero, ¿por qué se eligió en el Catalyst un puerto 100Mbps en fibra y no se eligió uno en cobre para conectar directamente con el 3524?. Sencillamente porque este puerto tiene que soportar un enlace trunk, y estos enlaces no son soportados en puertos 100Mb en cobre, tan solo en puertos 1Gb y 100Mb de fibra.

Ahora ya podemos establecer una sesión SPAN entre el puerto 6/3 y el 8/16, pero filtrando sólo las tramas que perteneciesen a la VLAN 32. Esto se hace con el siguiente comando en el Catalyst:

```
set span 6/3 8/16 filter 32
```

Otra opción podría haber sido la de conectar directamente el IDS a una interfaz eléctrica del hub, con lo que nos habríamos evitado la configuración de la sesión SPAN. El problema es que al tratarse de un enlace trunk, todas las tramas vendrían con etiquetas 802.1Q, por lo que habría que configurar el IDS para que soportase este encapsulado y para que filtrase todos los paquetes que no pertenecen a la VLAN 32. Puesto que la opción de SPAN resultaba más sencilla, se optó por ésta.

Pero como ya dijimos en el capítulo 2 cuando hablábamos de las posibles ubicaciones de un sensor, el hub tiene el inconveniente de que limita el tráfico full-duplex a half-duplex. En este caso es todavía peor porque hemos pasado de tener un rendimiento de 1Gb FD a 100Mb HD, pero se pensó que éste no sería demasiado inconveniente para los usuarios del edificio de farmacia. Además, nos daba la ventaja de limitar el rendimiento al máximo tráfico que soporta la interfaz del IDS, por lo que si éste perdiese paquetes, esto no sería debido a la excesiva velocidad de la conexión con farmacia.

Ahora bien, ¿qué hubiese ocurrido si el 3524 hubiese tenido interfaces a 100Mbps en fibra?. No habría sido necesario el hub, por lo que habríamos tenido 100Mbps FD y esto podría haber sido perjudicial para el IDS. Pero existe la posibilidad de configurar los puertos del Catalyst y el 3524 en modo half-duplex, obteniendo el mismo resultado y eliminando un posible punto de fallo en la infraestructura.

3.3.2. PVC ATM multipunto

La solución adoptada para monitorizar todo el tráfico de entrada/salida fue la de establecer un PVC multipunto entre el router de RedIRIS y el de la Universidad. En un principio, este PVC estaba formado entre un router Cisco 7507 de RedIRIS (*EB-Valencia*) que forma parte del POP (Point of Presence) de RedIRIS en la Comunidad Valenciana y el router principal de la Universidad de Valencia (*gordius*), que consiste en una tarjeta RSM (Routing Switch Module) conectada en el Catalyst 5500 donde se realizaba el SPAN. En abril de 2002, RedIRIS añadió a su POP un router Juniper y hubo que cambiar la configuración en el IDS.

3.3.2.1. PVC multipunto entre EB-VALENCIA y Gordius

Vimos en la parte de análisis y diseño la imposibilidad de constituir circuitos multipunto bidireccionales y la necesidad de dividir cada PVC en dos, uno para cada sentido. También comentamos que si no configuramos adecuadamente los routers, podemos acabar con rutas asimétricas al asignar un PVC a cada subinterfaz (por lo tanto, IPs distintas), y esto es un problema para algunas funciones, por ejemplo para el protocolo de routing multicast, PIM-SM.

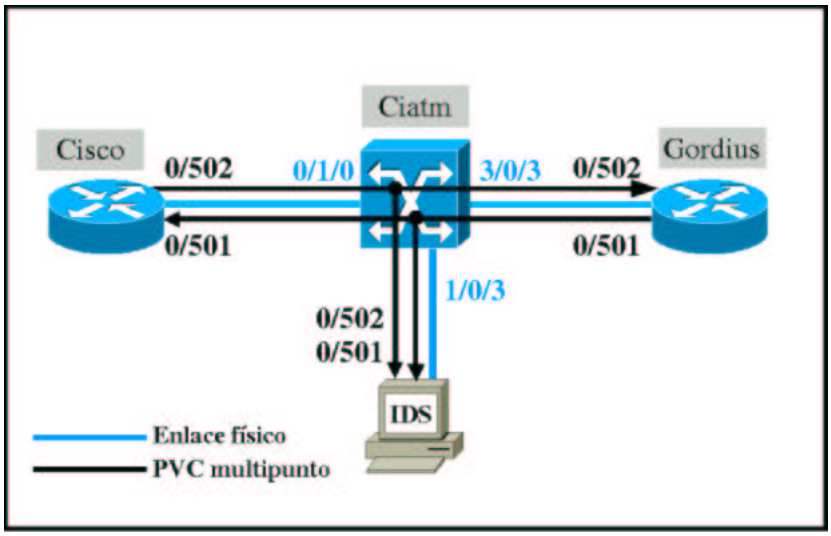

Figura 3.3: Maqueta formada por *cisco* y *gordius* para la configuración de un PVC multipunto.

Antes de modificar la red de producción se probó la configuración sobre una maqueta formada por *gordius* y *cisco*, un router secundario de la Universidad (ver figura 3.3). La configuración es la siguiente:

Configuración en *cisco:*

```
cisco(config)#int ATM0.36 multipoint
cisco(config-subif)#mtu 1500
cisco(config-subif)#bandwidth 45000
cisco(config-subif)#ip address 192.168.10.2 255.255.255.252
cisco(config-subif)#atm pvc 150 0 502 aal5mux ip 45000 45000 32
cisco(config-subif)#atm pvc 151 0 501 aal5mux ip 45000 45000 32
cisco(config-subif)#map-group ip-snort2
cisco(config-subif)#exit
cisco(config)#map-list ip-snort2
cisco(config-map-list)#ip 192.168.10.1 atm-vc 150 broadcast
```

Configuración en *gordius*:

```
gordius(config)#int ATM0/0.9 multipoint
gordius(config-subif)#mtu 1500
gordius(config-subif)#bandwidth 45000
gordius(config-subif)#ip address 192.168.10.1 255.255.255.252
gordius(config-subif)#atm pvc 100 0 501 aal5mux ip 45000 45000 32
gordius(config-subif)#atm pvc 110 0 502 aal5mux ip 45000 45000 32
gordius(config-subif)#map-group ip-snort2
gordius(config-if)#exit
gordius(config)#map-list ip-snort2
gordius(config-map-list)#ip 192.168.10.2 atm-vc 100 broadcast
```

Configuración en el conmutador ATM:

```
ciatm(config)#int atm 0/1/0
ciatm(config-if)#atm pvc 0 502 cast-type p2mp-root int atm3/0/3 0
502 cast-type p2mp-leaf
ciatm(config-if)#atm pvc 0 502 cast-type p2mp-root int atm1/0/3 0
502 cast-type p2mp-leaf
ciatm(config-if)#int atm 3/0/3
ciatm(config-if)#atm pvc 0 501 cast-type p2mp-root int atm0/1/0 0
501 cast-type p2mp-leaf
ciatm(config-if)#atm pvc 0 501 cast-type p2mp-root int atm1/0/3 0
501 cast-type p2mp-leaf
```

Veamos rápidamente como funciona esta configuración. Los comandos en *cisco* y en *gordius* son análogos. A la subinterfaz ATM, le asignamos un ancho de banda de 45Mb/s, que corresponderá con el caudal que asignemos al circuito CBR en la orden '`atm pvc`'. En cada parte asignamos una dirección IP perteneciente a la misma subred, y definimos el PVC con el siguiente comando:

```
Router(config-subif)#atm pvc map-list VPI VCI aal5mux ip 45000 45000
32
```

El número que ponemos en *map-list* es una referencia al map-list con el mismo número dentro del map-group del interfaz. Se utiliza para que el router pueda hacer una resolución de la dirección IP con el PVC: en el caso del router *cisco* tendrá una ruta que diga que para llegar a una red dada (o a cualquier red con una ruta por defecto) lo hará reenviando el tráfico a la IP 192.168.10.1. ¿Cómo sabe por qué PVC ha de llegar a esta IP?. El map-list del grupo ip-snort2 en el subinterfaz ATM0.36 le dice que ha de usar el PVC etiquetado con el map-list 150, por lo que usará el VPI.VCI 0.502. En el

otro router hacemos lo mismo, pero de forma cruzada, por lo que tendremos dos PVCs simplex y en el conmutador ya podremos establecer cada PVC como multipunto.

Por último con respecto a la configuración de los routers, comentar que el encapsulado utilizado es el aal5mux, que consiste en establecer un PVC por protocolo. El protocolo que circulará por el PVC sera IP, y lo hará con un caudal reservado de 45Mbps con un máximo de 45Mbps, esto es, un CBR de 45Mbps. El tamaño del buffer será de $32 \times 32 = 1024$ celdas ATM.

Con respecto al conmutador ATM, la interfaz que envía celdas se configura como raíz, y las interfaces que reciben celdas se configuran como hojas.

3.3.2.2. PVC multipunto entre Juniper y Gordius

Como ya hemos comentado, en abril de 2002 RedIRIS añadió un nuevo router (Juniper) al POP de Valencia. El PVC de la Universidad de Valencia pasó a terminar en dicho router, pero la configuración utilizada con el Cisco no funcionaba en el Juniper. Finalmente en mayo pudo encontar una forma equivalente de realizar la misma función, con lo que el IDS volvió a estar operativo. Se cambió la configuración de los PVCs en el IDS asignando los siguientes números de VPI.VCI:

- 0.225 : Tráfico no prioritario[4] Madrid-Valencia.
- 0.325: Tráfico no prioritario Valencia-Madrid.
- 0.226: Tráfico prioritario Madrid-Valencia.
- 0.326: Tráfico prioritario Valencia-Madrid.

La configuración del router Juniper quedó de la siguiente forma:

```
at-0/2/0 {
    mtu 1512;
    clocking external;
    sonet-options {
        payload-scrambler;
    }
    atm-options {
        vpi 1 maximum-vcs 255;
        vpi 3 maximum-vcs 255;
    }
    unit 0 {
        description " -- Acceso UV -- ";
        encapsulation atm-vc-mux;
        multipoint;
        family inet {
            mtu 1500;
            no-redirects;
            address 130.206.211.181/30 {
                multipoint-destination 130.206.211.182 vci 1.225;
            }
            address 10.0.1.1/30 {
                multipoint-destination 10.0.1.2 vci 1.226;
            }
        }
```

[4]La configuración actual de la Universidad de Valencia utiliza dos clases de PVCs: el de tráfico prioritario, destinado al tráfico generado por los servidores proxy, y el no prioritario, destinado al resto del tráfico.

```
        }
        unit 6 {
            description "-- Acceso PROXY Valencia --";
            encapsulation atm-vc-mux;
            multipoint;
            family inet {
                mtu 1500;
                no-redirects;
                address 130.206.211.189/30 {
                    multipoint-destination 130.206.211.190 vci 3.225;
                }
                address 10.0.1.5/30 {
                    multipoint-destination 10.0.1.6 vci 3.226;
                }
            }
        }
    }
```

3.4. Aplicaciones desarrolladas

3.4.1. 'IDS Alerts': Programa para la gestión de respuestas a incidencias

La aplicación 'IDS Alerts' está dividida en dos programas desarrollados en Perl: *RecvAlert.pl* y *SendAlert.pl*., tal y como se ve en los diagramas UML del apéndice A. Ambas aplicaciones son lanzadas por el demonio *cron* a las 4 a.m.

SendAlert.pl se encarga de leer la base de datos cada cierto tiempo (24 horas por defecto), filtrar las alertas que el administrador no considera importantes y construir un mail con esta información para enviárselo al administrador. La sintaxis de este mail es la siguiente:

```
[]   'ID de la firma'     'Texto de la firma'     (URL explicativa)
[]   Total_1     'IP Src_1'     Fecha_inicio_1     Fecha_fin_1
[]   Total_2     'IP Src_2'     Fecha_inicio_2     Fecha_fin_2
.       .             .               .                .
.       .             .               .                .
.       .             .               .                .
[]   Total_n     'IP Src_n'     Fecha_inicio_n     Fecha_fin_2
```

Para filtrar alertas que no deseamos que nos sean notificadas existe el fichero `.deselect`. La sintaxis para este fichero es la siguiente:

- Todo lo que empiece por '#' se considera comentario.
- Para filtrar todas las alertas generadas por un SID en particular lo pondremos en una línea:

  ```
  SID
  ```

- Para filtrar un rango de SIDs lo indicaremos así:

  ```
  SID1-SID2
  ```

- A estas dos formas de indicar el SID le podemos añadir las IPs involucradas en las firmas, seguidas por comas:

```
SID1-SID1 IP1
SID1 IP1,IP2,IP3
```

Veamos a modo de ejemplo un mail generado por SendAlert.pl:

```
Date: Sat, 29 Jun 2002 03:53:48 +0200
From: Emilio.J.Mira@uv.es
To: emial@alumni.uv.es
Subject: SECURITY ALERTS

 ##########################################################################
 #                                                                        #
 #  S E C U R I T Y   A L E R T S                                         #
 #                                                                        #
 #  Report from Fri Jun 28 04:02:14 2002 to Sat Jun 29 04:02:40 2002      #
 #                                                                        #
 ##########################################################################

 []       22  SCAN nmap TCP - [ http://online.securityfocus.com/bid/2250 ]
          []       25  193.144.127.9      2002-06-28 17:00:59   2002-06-29 03:58:27
          []        2  194.30.63.66       2002-06-28 23:34:58   2002-06-28 23:44:08
 []       32  WEB-IIS cmd.exe access
          []       29  63.168.13.199      2002-06-29 00:50:55   2002-06-29 00:52:06
          []       14  168.131.90.34      2002-06-29 03:34:23   2002-06-29 03:34:32
          []       14  218.31.103.37      2002-06-29 03:29:43   2002-06-29 03:30:02
          []       14  200.52.14.5        2002-06-28 23:31:39   2002-06-28 23:31:44
          []        5  80.224.226.210     2002-06-29 01:45:07   2002-06-29 01:45:08
          []        1  61.136.113.28      2002-06-28 23:47:25   2002-06-28 23:47:25
          []        1  218.0.217.203      2002-06-29 00:46:22   2002-06-29 00:46:22
          []        1  210.21.91.66       2002-06-28 23:44:34   2002-06-28 23:44:34
 []      254  WEB-CGI MachineInfo access
          []        1  216.239.46.128     2002-06-28 23:32:06   2002-06-28 23:32:06
 []      289  EXPLOIT LPRng overflow - [ http://online.securityfocus.com/bid/2250 ]
          []      127  62.174.32.154      2002-06-29 01:31:25   2002-06-29 01:39:32

 ##########################################################################
 #                                                                        #
 #                   E N D   O F   R E P O R T                            #
 #                                                                        #
 ##########################################################################
```

En este mail vemos las alertas producidas desde el 28 de junio a las 4,02h, hasta el 29 de junio a las 4,02h. Vemos dos tipos de líneas: una con información de la alerta y otra con información específica de las direcciones IP que han generado dicha alerta, que aparecen sangradas respecto a su línea de alerta. Para cada alerta, tenemos el identificador al que corresponde en la base de datos, el texto explicativo y, algunas veces, un URL en el que podemos encontrar más información sobre dicho ataque. Para las líneas sangradas tenemos información de cada dirección IP que ha generado la alerta, el número total de veces que la ha generado y la fecha de inicio y la de final.

Una vez el administrador ha recibido el mail debe rellenarlo y responderlo. Para rellenarlo añadirá un carácter entre los corchetes '[]' que aparecen en el mail de la siguiente forma:

- [C]: Avisar al CERT.
- [R]: Avisar al responsable de la red en la que se encuentra esta IP.
- [A]: Avisar tanto al CERT como al responsable.

Dependiendo de si se rellenan los corchetes de una línea de alerta o de IP, los resultados serán distintos. En el primer caso, el aviso enviado contendrá información sobre todas las IPs que lo generaron, mientras que en el segundo, es aviso solo contendrá aquellas líneas IP marcadas.

Por ejemplo, supongamos que el administrador ha recibido el mail anterior y quiere avisar al CERT sobre todas las pruebas de escaneo con *nmap* y sobre los ataques de exploits al demonio de impresión LPRng, y sólo quiere avisar de la existencia del gusano Nimda (generalmente visible por accesos a *cmd.exe*) al responsable de las redes que albergan las IPs `218.31.103.37` y `218.0.217.203`. Para ello respondería con este mail:

```
To      : Emilio.J.Mira@uv.es
Cc      :
Attchmnt:
Subject : Re: SECURITY ALERTS

On Sat, 29 Jun 2002 Emilio.J.Mira@uv.es wrote:

>  ##########################################################################
>  #                                                                        #
>  #  S E C U R I T Y   A L E R T S                                         #
>  #                                                                        #
>  #  Report from Fri Jun 28 04:02:14 2002 to Sat Jun 29 04:02:40 2002  #
>  #                                                                        #
>  ##########################################################################
>
>  [C]       22  SCAN nmap TCP - [ http://online.securityfocus.com/bid/2250 ]
>           []       25  193.144.127.9      2002-06-28 17:00:59   2002-06-29 03:58:27
>           []        2  194.30.63.66       2002-06-28 23:34:58   2002-06-28 23:44:08
>  []       32  WEB-IIS cmd.exe access
>           []       29  63.168.13.199      2002-06-29 00:50:55   2002-06-29 00:52:06
>           []       14  168.131.90.34      2002-06-29 03:34:23   2002-06-29 03:34:32
>           [R]       14  218.31.103.37      2002-06-29 03:29:43   2002-06-29 03:30:02
>           []       14  200.52.14.5        2002-06-28 23:31:39   2002-06-28 23:31:44
>           []        5  80.224.226.210     2002-06-29 01:45:07   2002-06-29 01:45:08
>           []        1  61.136.113.28      2002-06-28 23:47:25   2002-06-28 23:47:25
>           [R]        1  218.0.217.203      2002-06-29 00:46:22   2002-06-29 00:46:22
>           []        1  210.21.91.66       2002-06-28 23:44:34   2002-06-28 23:44:34
>  []      254  WEB-CGI MachineInfo access
>           []        1  216.239.46.128     2002-06-28 23:32:06   2002-06-28 23:32:06
>  [C]      289  EXPLOIT LPRng overflow - [ http://online.securityfocus.com/bid/2250 ]
>           []      127  62.174.32.154      2002-06-29 01:31:25   2002-06-29 01:39:32
>
>  ##########################################################################
>  #                                                                        #
>  #                  E N D   O F   R E P O R T                             #
>  #                                                                        #
>  ##########################################################################
```

La otra parte de la aplicación, RecvAlert.pl, lee las respuestas del administrador y las analiza buscando las líneas marcadas e interpretándolas. Envía un mail al CERT con la información que el administrador marcó y un mail por responsable encontrado mediante whois. Además, guarda un log de todos los mails enviados, y es capaz de procesar varios mails si el administrador ha estado un tiempo sin responder los que SendAlert.pl le envió.

Es conveniente ejecutar estas aplicaciones cuando se prevea que la carga del sistema sea mínima, por ejemplo durante la madrugada.

3.4.2. 'IDS Bench': Programa para la medición del rendimiento

La aplicación 'IDS Bench' la hemos desarrollado en Tcl/Tk, y colabora con un conjunto de scripts escritos en Bash. Se encarga de lanzar el Snort, tomar muestras cada cierto tiempo sobre parámetros de interés que nos pueden proporcionar información de la máquina que necesitamos para correr Snort en nuestra red, como ya comentamos en el capítulo 2, y por último mostrar gráficas de la información obtenida. Los parámetros de los que se hace el muestreo son los siguientes:

- Paquetes por segundo recibidos directamente del interfaz.
- Paquetes por segundo recibido o procesador por Snort.
- Paquetes por segundo descartados por Snort por falta de CPU.
- Porcentaje de CPU consumida por Snort.
- Tamaño en memoria de Snort.

La aplicación se divide en dos partes: la parte encargada de la tomas de muestras y la que lanza las gráficas. En la figura 3.4 vemos la ventana principal y como se distinguen las dos partes, 'Contador' y 'Plot'.

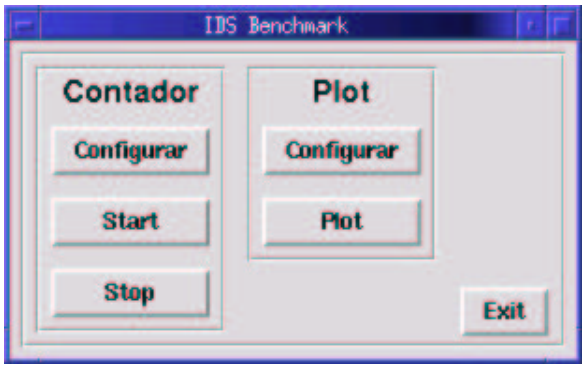

Figura 3.4: Ventana principal de 'IDS Benchmark'.

La ventana de configuración de los contadores aparece en la figura 3.5. En ella que se puede elegir la interfaz de la que se quiere hacer el muestreo, la velocidad de muestreo y las fuentes de las que queremos obtener cada parámetro, por si hubiese varias, como es el caso de los paquetes que se reciben en la interfaz. En el caso de la tarjeta ATM, existen dos formas de obtener estadísticas: desde los drivers de la tarjeta o desde el propio núcleo. Una vez configurados los contadores, lanzaríamos el Snort y los scripts de muestreo con el botón Start, y cuando estuviese listo lo pararíamos con Stop desde la ventana principal.

Para calcular los paquetes descartados restamos los paquetes procesados por Snort por unidad de tiempo de los recibidos por la interfaz. Podemos obtener los paquetes procesados por Snort enviándole la señal SIGUSR1 de Unix. Aunque existe la posibilidad de ver directamente de Snort el porcentaje de paquetes descartados, esto parece que no funciona bien. Se hicieron pruebas sobrecargando la máquina y priorizando otros procesos en momentos de carga máxima, y sospechosamente Snort seguía sin descartar paquetes. En cuanto se comparó el número de paquetes por segundo que entraban por la interfaz con los procesados por Snort se vió claramente que se estaban produciendo descartes.

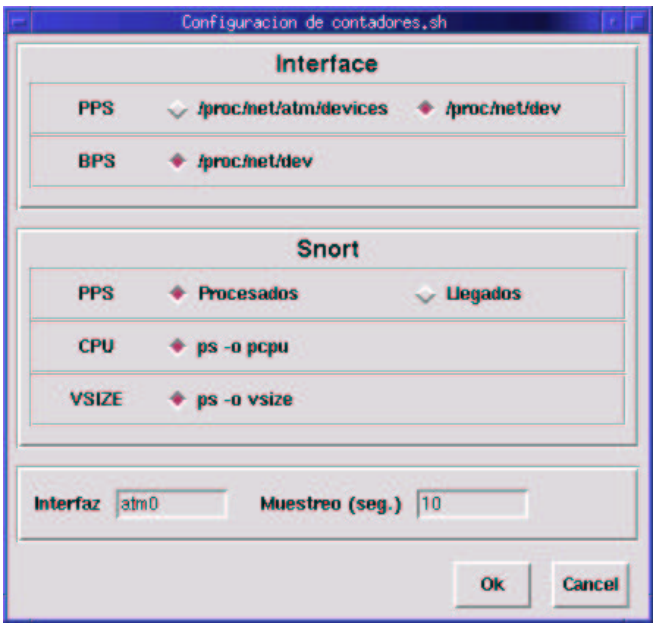

Figura 3.5: Ventana de configuración de los contadores en 'IDS Benchmark'.

La causa de esto es la forma que tiene Snort de tomar estadísticas. Lo hace por medio de la función `pcap_stats()` de la librería `pcap`. En la versión 0.6.2 de esta librería, que apareció en febrero de 2001 y que viene con RedHat 7.2, esta función devolvía 0 en los paquetes descartados al no haber una llamada al sistema para obtenerlos. En julio de 2001 se añadió código para obtener estas estadísticas, pero los resultados eran impredecibles hasta diciembre de 2001.

La solución de actualizar la librería llegó cuando el software se había desarrollado, con lo que el software se ha complicado, pero se ha conseguido que sea independiente a la librería.

Una vez tomadas las muestras, entraríamos en la configuración de Plot (figura 3.6). Seleccionaríamos los parámetros que deseamos que salgan en la gráfica y su escalado. La escala por defecto es la que se usa generalmente, pero el programa permite su modificación como opción avanzada. Además, nos permite realizar suavizado a nuestra gráfica de varias formas, con c-splines o mediante curvas de Bezier. Otra forma de obtener suavizado es aumentar el periodo de muestreo. Por último, el cuadro vista nos permite elegir la forma de representar la curva.

Por último, para visualizar pulsaremos el botón Plot.

'IDS Benchmark' hace uso de varios scripts y del programa gnuplot que deben encontrarse en alguno de los directorios de la variable de entorno PATH. En el caso de que no se encuentren generará el mensaje de error que corresponda.

3.4.3. Scripts de consulta a la base de datos.

En la fase de análisis se tomó la decisión de utilizar Snort Report para la interacción del usuario con la base de datos. Disponía de un interfaz gráfico bastante cómodo (ver figura 3.7) y parecía una buena herramienta.

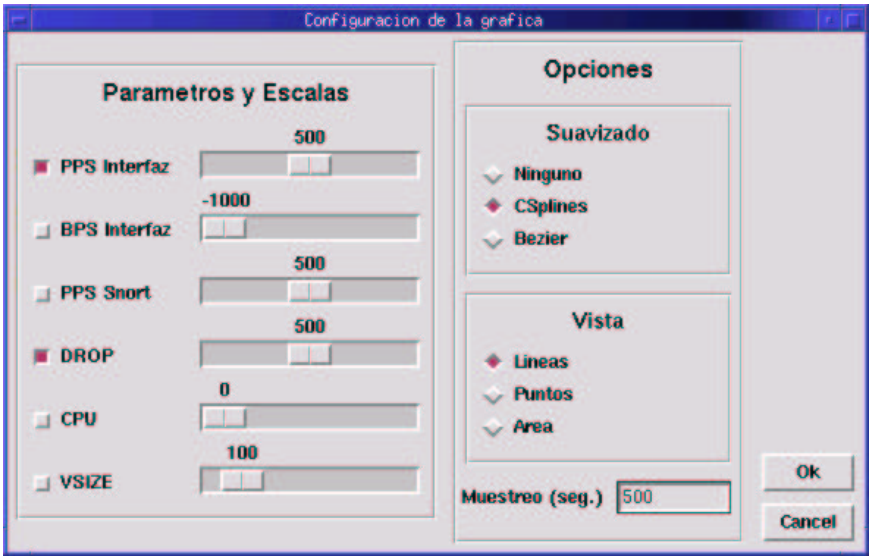

Figura 3.6: Ventana de configuración de la gráfica en 'IDS Benchmark'.

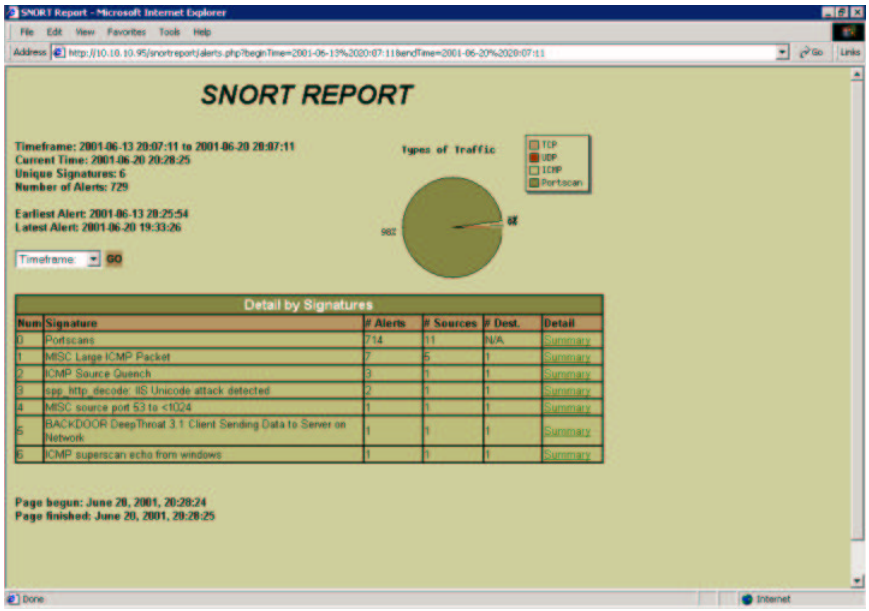

Figura 3.7: Pantalla de gestión de alertas de SnortRepor.

El problema con este programa era su lentitud cuando la base de datos crecía un poco, cosa por otra parte inevitable debido muchas veces al gran número de falsas alarmas. La lentitud se debía principalmente a dos razones:

- Las consultas SQL eran poco eficientes.
- La ejecución de los programas PHP a veces se realizaba sobre resultados excesivamente grandes de las consultas a la base de datos.

La primera decisión que se tomó fue la de optimizar al máximo las consultas SQL incluidas en Snort Report. Esto mejoró un poco los tiempos de respuesta, pero no fue suficiente; tras una petición vía un navegador web, el tiempo de espera era tal que el navegador cerraba la conexión por time-out. La solución final fue desistir del uso de Snort Report y crear un conjunto de scripts que sirviesen de interfaz a la base de datos. Esto descargaba la máquina, al no ser necesario el servidor web, lo que la dejaba en unas condiciones óptimas: 2 procesadores, uno para Snort y otro para MySQL.

Puesto que queríamos unos scripts lo más generales posibles (multiplataforma e independientes de la base de datos), se escribieron en Perl utilizando DBI (Database Interconnection). Para ganar en flexibilidad, la interacción con los scripts se hace en modo texto, lo que también mejora el rendimiento.

El módulo DBI de Perl conecta a la base de datos por un socket TCP, no Unix como lo hace Snort, por lo que hubo que eliminar la opción `--skip_networking` cuando lanzábamos el servidor MySQL.

3.4.3.1. Ranking.pl

Este programa muestra el ranking de ataques ordenado por el número de apariciones en la base de datos. Tiene opciones que añaden información al listado, pero que ralentizan a su vez la búsqueda. Las opciones son:

-T : añade a la salida la cantidad total de alertas en la BD.
-S : muestra las direcciones IP de origen involucradas en cada firma.
-D : muestra las IPs direcciones IP de destino involucradas en cada firma.
-t <tiempo>{m,h,d,M} : muestra las alertas producidas en el ultimo lapso de <tiempo>.
-B <db>: selección de la base de datos a utilizar.
-P <puerto>: puerto para conectarse a la base de datos.
-U <usuario>: usuario de la base de datos.
-A <password>: password de la base de datos.
-d : debug (muestra consultas realizadas a la BD).
-h : muestra ayuda

Veamos un ejemplo. La orden:

```
$ Ranking.pl -SD -t 5m
```

muestra las alertas producidas en los últimos 5 minutos con información añadida que explicaremos a continuación. La salida que produce es la siguiente:

```
[emilio@discreto scripts]$ Ranking.pl -SD -t 5m
Mostrando ultimos 5 minutos
COUNT   SIG_ID  NUM_SRC NUM_DST SIG_NAME (REF)
-------------------------------------------------------------------------
120     15      3172    13764   ICMP Destination Unreachable (Communication Administratively Pro-
hibited)
37      10      39556   135     WEB-MISC count.cgi access (bugtraq 128)
27      28      13396   750     WEB-IIS scripts access
27      26      62424   36147   SCAN Proxy attempt (url help.undernet.org/proxyscan/)
19      19      7491    816     MISC Large ICMP Packet (arachnids 246)
15      77      177     15778   ICMP PING NMAP (arachnids 162)
12      29      1925    100     WEB-MISC http directory traversal (arachnids 297)
10      32      1831    848     WEB-IIS cmd.exe access
7       5       468     14      WEB-CGI php access (bugtraq 2250)
```

```
7       6       35668   65536   SCAN Proxy attempt
3       21      110     3433    WEB-MISC 403 Forbidden
2       22      90      426     SCAN nmap TCP (arachnids 28)
2       81      66      71      X11 outgoing (arachnids 126)
1       75      151     12      WEB-MISC backup access
1       39      1671    773     WEB-IIS CodeRed v2 root.exe access (url www.cert.org/advisories/CA-
2001-19.html)
1       30      1394    68      WEB-IIS iissamples access
1       64      49      126     TELNET login incorrect (arachnids 127)
1       117     22      23      WEB-IIS .cnf access
```

Donde:

- COUNT: Número de alertas de este tipo.
- SIG_ID: Identificador que usa esta firma en la base de datos.
- NUM_SRC: Número de IPs distintas que aparecen como origen en esta firma (opción -S).
- NUM_DST: Número de IPs distintas que aparecen como destino en esta firma (opción -D).
- SIG_NAME: Nombre de la firma.
- REF: URL en la que se puede encontrar información más detallada sobre el ataque.

Con esta herramienta podemos distinguir ataques de falsas alarmas. Por ejemplo, cuando una alerta que aparece muchas veces es producida por muchas IP origen distintas y va hacia pocas IPs destino, es posible que se trate de una falsa alarma. En el caso contrario, con muy pocas IPs origen y muchas IPs destino, se suele tratar de barridos de host con un servicio habilitado probando un exploit.

3.4.3.2. shsig.pl

shsig.pl muestra la información disponible de una determinada alerta dada en la línea de comando. El uso es el siguiente:

Uso: shsig.pl -s <SigId>[Opciones]
Opciones:
-s: identificador de la firma a buscar.
-n : resuelve direcciones IP.
-L <limite>: limita el numero de líneas de salida a <limite>.
-S : muestra el número total de ocurrencias de la IP como origen de un ataque.
-D : muestra el número total de ocurrencias de la IP como destino de un ataque.
-T : muestra el intervalo de tiempo que recoge los ataques.
-B <db>: selección de la base de datos a utilizar.
-P <puerto>: puerto para conectarse a la base de datos.
-U <usuario>: usuario de la base de datos.
-A <password>: password de la base de datos.
-d : Debug.
-h : muestra la ayuda.

Después de ver el ranking podemos averiguar más cosas sobre una alerta dada, por ejemplo, una respuesta de ataque que devuelva el identificador de usuario `root`. Lo haríamos de esta forma:

```
[emilio@discreto scripts]$ shsig.pl -s 80 -SDTn

## ATTACK RESPONSES id check returned root ## 80 ##

COUNT   TOT_SRC TOT_DST TIME_INI               TIME_FIN               SRC
-------------------------------------------------------------------------------------
8       111     0       2002-06-25 14:22:13    2002-06-25 16:13:31    zeppo.rediris.es
8       8       0       2002-06-10 21:42:17    2002-06-22 06:07:19    irc.ITDNet.net
6       13      0       2002-06-09 13:52:45    2002-06-24 22:32:38    irc-1.stealth.net
5       505     535     2002-06-18 11:49:59    2002-06-29 15:56:37    sibserv.ci.uv.es
3       3       0       2002-06-20 12:29:19    2002-06-21 13:05:17    linux.tu-varna.acad.bg
3       3       41      2002-06-07 20:33:49    2002-06-13 14:35:40    dune.ci.uv.es

COUNT   TOT_SRC TOT_DST TIME_INI               TIME_FIN               DST
-------------------------------------------------------------------------------------
13      5       3660    2002-06-09 23:08:32    2002-06-22 06:07:19    intras12.intras.uv.es
11      0       5728    2002-06-13 16:18:45    2002-06-25 14:38:31    pastilla.uv.es
5       5       900     2002-06-20 10:47:07    2002-06-24 22:32:38    garfi.informat.uv.es
4       5       3073    2002-05-24 23:12:23    2002-06-18 21:07:22    sello.ci.uv.es
4       0       6157    2002-06-18 20:02:09    2002-06-26 11:26:45    yxorp.upv.es
3       0       3824    2002-06-30 11:33:31    2002-06-30 11:33:57    capsula.uv.es
```

Al igual que en Ranking.pl, el resultado está ordenado por el número de veces que aparece una IP involucrada en ese tipo de ataque, y separado por ’origen’ y ’destino’. También vemos el número total de veces que la IP aparece en la base de datos como origen y también como destino, lo cual nos puede ayudar a identificar un supuesto atacante, al aparecer muchas veces como origen y pocas como destino. Por último nos muestra la primera y la última vez que la IP aparece en esa firma.

3.4.3.3. ship.pl

ship.pl muestras la información de la base de datos correspondiente a una dirección IP en particular:

Uso: ship.pl -s <IPSrc_Dot>[Opciones]

Opciones:

-t <tiempo>{m,h,d,M} : muestra las alertas producidas en el ultimo lapso de <tiempo>.

-L <limite>: muestra <limite>lineas como máximo

-D : muestra detalles de los paquetes

-n : muestra nombres en lugar de direcciones IP si es posible

-B <db>: selección de la base de datos a utilizar.

-P <puerto>: puerto para conectarse a la base de datos.

-U <usuario>: usuario de la base de datos.

-A <password>: password de la base de datos.

-d : debug (muestra consultas ralizadas a la BD).

-h : muestra esta ayuda

Veamos un ejemplo:

```
[emilio@discreto scripts]$ ship.pl -s 206.252.192.195 -n
SOURCE
NUM     SIG_ID  SIG_NAME
------------------------
6       80      ATTACK RESPONSES id check returned root
6       149     DOS MSDTC attempt
1       27      SHELLCODE x86 NOOP

DESTINATION
```

```
NUM     SIG_ID  SIG_NAME
------------------------
```

Vemos como la IP 206.252.192.195 ha generado alertas de respuesta de ataque con identificador root, es decir, que ha conseguido ser superusuario en una máquina interna, ataques de denegación de servicio MSDTC, y códigos de operación NOOP para arquitectura x86 en conexiones tipo texto (telnet, ftp canal de comandos).

3.4.3.4. shsigip.pl

Este programa muestra la información disponible en la base de datos de una determinada alerta con origen o destino una determinada dirección IP. Su uso es:

Uso: shsigip.pl -s <id_firma>-S <IP>[Opciones]

Opciones:

-n : muestra nombres en lugar de direcciones IP si es posible.

-L <limite>: muestra <limite>lineas como máximo.

-B <db>: selección de la base de datos a utilizar.

-P <puerto>: puerto para conectarse a la base de datos.

-U <usuario>: usuario de la base de datos.

-A <password>: password de la base de datos.

-d : debug (muestra consultas ralizadas a la BD).

-h : muestra esta ayuda.

Esta herramienta nos da mas información sobre una firma y una IP, mostrándonos todos los paquetes que han sido almacenados:

```
[emilio@discreto scripts]$ shsigip.pl -s 36 -S 61.171.34.39 -n
## SMTP RCPT TO overflow ## 36 ##
SOURCE
-----------------

  *  2002-06-13 12:48:56 61.171.34.39 ->sello.ci.uv.es
       [IP] Ver:4 HLen:5 Tos:16 Len:2475 ID:0 Flags:0x0 Offset:0 TTL:240 Proto:6 Cksum:0
       [TCP] SPort:3394, DPort:25 Seq:872367402 Ack:229738184 Offset:5 Res:0 Flags:0x18 Win:11520
Cksum:0 Urp:0 Res:0


  *  2002-06-13 13:49:07 61.171.34.39 ->sello.ci.uv.es
       [IP] Ver:4 HLen:5 Tos:16 Len:2495 ID:0 Flags:0x0 Offset:0 TTL:240 Proto:6 Cksum:0
       [TCP] SPort:3975, DPort:25 Seq:383725294 Ack:2257763838 Offset:5 Res:0 Flags:0x18 Win:11520
Cksum:0 Urp:0 Res:0


  *  2002-06-13 14:15:37 61.171.34.39 ->sello.ci.uv.es
       [IP] Ver:4 HLen:5 Tos:16 Len:2493 ID:0 Flags:0x0 Offset:0 TTL:240 Proto:6 Cksum:0
       [TCP] SPort:3911, DPort:25 Seq:2053445322 Ack:3088515567 Offset:5 Res:0 Flags:0x18 Win:11520
Cksum:0 Urp:0 Res:0


DESTINATION
-----------------
```

En este caso vemos información más detallada sobre los ataques de RCPT TO overflow al servidor SMTP de la Universidad que vienen de 61.171.34.39.

3.4.4. Scripts de configuración

Se ha intentado que la máquina funcione con la máxima autonomía posible para reducir el tiempo que el administrador debe dedicarle. Para ello, se han creado una serie de scripts que liberan al administrador de algunas tareas. Los listados de los scripts se encuentran en el apéndice B y se explican a continuación:

- /etc/logrotate.d/snort: 'logrotate' es una aplicación creada por Erik Troan y Preston Brown, de RedHat, que rota los logs automáticamente cada cierto tiempo y los comprime. Cuando un fichero de logs ha sido rotado un número determinado de veces, se elimina. En nuestro caso, además de los logs del sistema, hemos rotado los generados por Snort y les hemos dado un tiempo de vida de 15 días. Esto no quiere decir que en 15 días desaparezcan las alertas generadas, ya que éstas se guardan a su vez en la base de datos.

- Inicialización del sistema: en el script de inicialización se incluye la configuración del firewall interno, la configuración del interfaz ATM, la ejecución de la base de datos MySQL y la de Snort.

- runsnort.sh: es un envolvente de Snort. Comprueba cada cierto tiempo que Snort sigue en ejecución y lo vuelve a lanzar en caso de que haya caído. Monitoriza toda su actividad en el fichero `/var/log/runsnort`.

3.5. Pruebas controladas de ataques

Además de las medidas de rendimiento se han realizado pruebas controladas de ataques reales sobre una máquina de la Universidad de Valencia. La máquina en cuestión, *elizabeth.uv.es*, de la cual somos administradores, se encuentra ubicada en el Servicio de Informática de la Universidad de Valencia, corre el sistema operativo RH7.2, y no es vulnerable a ninguno de los ataques que lanzaremos.

3.5.1. Ataques estándar

Vamos a lanzar cuatro ataques consistentes en programas que explotan vulnerabilidades conocidas en los siguientes servicios:

- Apache 1.3.20 en OpenBSD x86

- wu-ftpd <= 2.6.0 en Linux x86

- rpc.rstatd en Solaris 2.5

Estos exploits son bien conocidos y Snort ya dispone de firmas para ellos, por lo que tendrían que ser detectados cuando los lancemos siempre y cuando la carga no sea tan grande que se descarten los paquetes de los ataques.

La forma de lanzarlos va a ser la siguiente: desde una máquina externa conectada las 24 horas a Internet por medio de conexión ADSL se lanzarán los ataques cada 5 minutos a lo largo de varios días. Luego se comprobará que los ataques se encuentran perfectamente recogidos en la base de datos.

3.5.2. Ataques enmascarados

También se han planificado un conjunto de pruebas anti-IDS para ver el comportamiento de Snort ante este tipo de ataques. Las pruebas se dividen en dos grupos:

- Modificaciones sintácitcas y no semánticas del patrón de búsqueda en la firma de Snort
- Evasión e inserción.

En el primer conjunto de pruebas lo que haremos básicamente será variar la sintaxis del ataque por otra sintaxis que signifique lo mismo en la máquina destino. Si Snort es lo suficientemente robusto sabrá que ambas sintaxis corresponden al mismo ataque y lo reconocerá como tal. Por ejemplo, la regla que detecta cuando el fichero `/etc/passwd` está siendo consultado por petición web busca la cadena '/etc/passwd' en una conexión dirigida al puerto 80. Si nuestra consulta en lugar de ser:

```
GET /etc/passwd
```

es:

```
GET%65%74%63/%70%61%73%73%77%64
```

el servidor web generará el mismo resultado, pero si el IDS no conoce esta sintaxis, no será capaz de reconocer el ataque.

En el segundo tipo de pruebas, mediante el programa Fragroute[5] y algunas de las técnicas de evasión documentadas en [20] y [25], vamos a lanzar ataques exploit pero esta vez enmascarados. Las técnicas de evasión que emplearemos serán estas:

- Fragmentar los paquetes IP: útil para conocer si el IDS reensambla los fragmentos IP.
- Desordenar los fragmentos IP enviados: si la pila IP del IDS no reordena los fragmentos, será incapaz de detectar nuestros ataques.
- Expiración del time-out de los fragmentos: esta prueba consiste en hacer expirar el time-out de los fragmentos en Snort sin que expire en el host atacado. Su éxito depende por lo tanto del tiempo en el que el IDS mantiene los fragmentos en memoria antes de descartar el paquete. En la mayoría de los sistemas el time-out expira en 60 segundos. Si el IDS no mantiene el fragmento durante los 60 segundos será vulnerable a un ataque de evasión.
- Solapar fragmentos IP de forma que se favorezca el antiguo o el nuevo: dependiendo del sistema operativo atacado, el solapamiento se realiza favoreciendo el nuevo fragmento o el antiguo. Si el IDS lo realiza de una forma y el sistema atacado de otra, podemos enviar el ataque de forma que pase desapercibido al IDS.

Fragroute es una aplicación diseñada para probar IDSs. Se llama de la siguiente forma:

```
# fragroute <host>
```

[5] Se puede descargar Fragroute de `www.monkey.org/~dugsong/fragroute`. Actualmente tan solo soporta interfaces Ethernet, ya que no se basa en la librería `pcap`, sino en `dnet`.

donde el host es el host al que va dirigido el tráfico que fragroute va a interceptar y a modificar. Fragroute toma la configuración del fichero `/usr/local/etc/fragroute.conf`, donde le decimos qué hacer con los paquetes. Por ejemplo, supongamos que queremos fragmentar todo el tráfico dirigido a host Windows en fragmentos de 8 bytes para que se solapen favoreciendo los datos antiguos, reordenándolos aleatoriamente y mostrándolos en salida estándar. Para ello, en fragroute.conf pondríamos:

```
ip_frag 8 old
order random
print
```

Bastaría con lanzar fragroute como superusuario y veríamos en la salida estándar las modificaciones hechas al tráfico que salga de nuestra máquina hacia el host objetivo.

Capítulo 4

Resultados

4.1. Resultados de las pruebas de rendimiento

En este apartado veremos los resultados al medir el rendimiento de Snort mediante la aplicación desarrollada 'IDS Benchmark'. Las pruebas se han realizado durante junio y julio de 2002 con el IDS monitorizando la red completa de la Universidad, y se han utilizado dos configuraciones del sistema distintas. La primera ha sido ejecutar Snort de la forma más sencilla, mostrando la salida de las alertas y los logs en un fichero binario, y en la segunda configuración se le ha añadido la base de datos MySQL como back-end.

4.1.1. Snort

En la primera prueba, realizada durante la tercera semana de junio de 2002, se ha lanzado el proceso Snort con el plugin tcpdump, que escribe en un fichero binario (flag -b) las alertas y los logs producidos en formato tcpdump. Además hemos lanzado Snort con el flag -A fast, lo que indica que las alertas deben ser lo más breves posibles. Según los diseñadores de Snort este es el modo de funcionamiento que mejor rendimiento ofrece.

En la figura 4.1podemos comparar el tráfico entrante con el tráfico procesado por Snort.

En el eje X tenemos el tiempo medido en horas, y en el Y los paquetes por segundo. La gŕafica representa una semana y podemos ver claramente los picos correspondientes a los días y los valles a las noches. Vemos también que por la noche no hay ningún exceso de carga, pero que durante el día Snort pierde tráfico, y esta pérdida no es regular, no podemos fijar un máximo de carga en pps para el sistema. Esto quizá lo veamos mejor en la figura 4.2, que compara el tráfico entrante y el descartado.

Por ejemplo, para las horas 100 y 120 (días 5 y 6) en las que entró un volumen de tráfico similar, Snort no tuvo el mismo comportamiento. Esto es así porque el comportamiento de un IDS es dinámico en función del tipo de tráfico que entra y del conjunto de reglas. Como vimos en el capítulo anterior, en Snort existen varios conjuntos de reglas y algunos producen más carga computacional que otros, en especial el conjunto *shellcode.rules*. Dependiendo del tipo y contenido del tráfico entrante, Snort tendrá más o menos trabajo. Es más, no solo depende del tráfico externo, sino también del propio estado de Snort. Cuando la memoria reservada para reconstruir el flujo TCP de las conexiones se llena, Snort elimina de memoria los nodos que no va a utilizar de forma masiva. Esto es una operación costosa y puede llevar al descarte de paquetes.

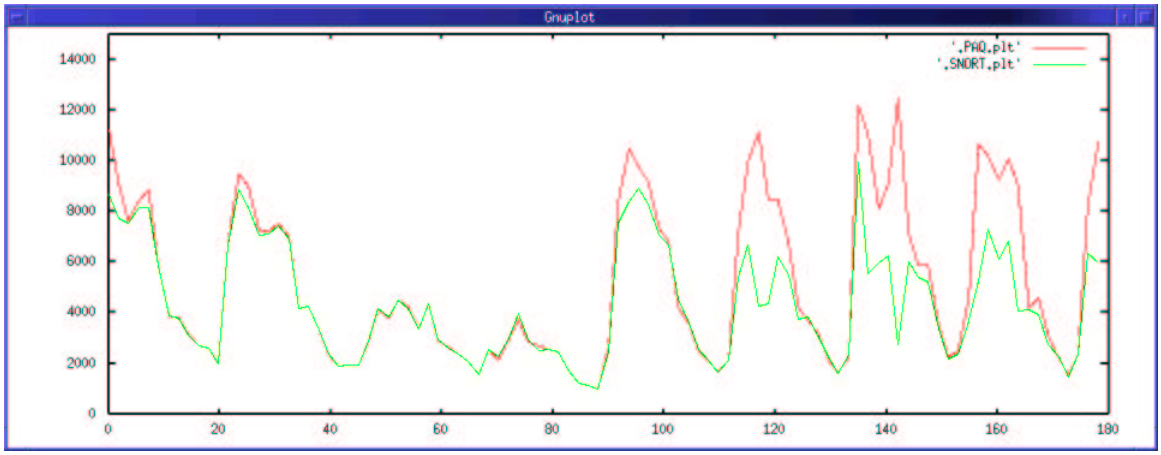

Figura 4.1: Gráfica comparativa entre el tráfico entrante y el procesador por Snort.

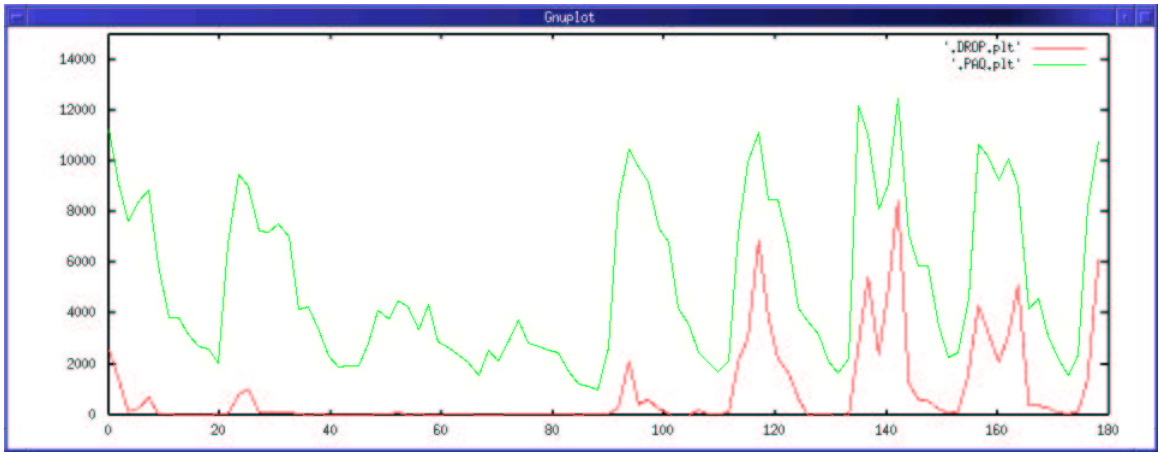

Figura 4.2: Gráfica comparativa entre el tráfico entrante y el descartado en Snort.

Otro dato importante es la cantidad de memoria física que consume el proceso: un exceso de paginación conlleva un descarte masivo de paquetes se puede confundir con falta de CPU. La figura 4.3 nos muestra el tamaño del proceso Snort en memoria a lo largo de toda la semana:

El eje Y muestra el tamaño total de memoria virtual de proceso. Vemos como el tamaño medio del programa es de unos 16MB. Aparece una caída en el centro que puede ser provocada por lo que comentábamos antes sobre al eliminación de nodos innecesarios de memoria. Para ver si esto corresponde con los momentos de descarte, veamos la gráfica de memoria en relación con los paquetes descartados:

En la figura 4.4vemos como el tamaño del proceso es estable mientras que no se descartan muchos paquetes, y variable cuando el descarte aumenta. Además, vemos como coincide la bajada del tamaño del programa con un incremento en el número de paquetes descartados, por lo que podría ser causa del borrado de nodos innecesarios del que habíamos hablado.

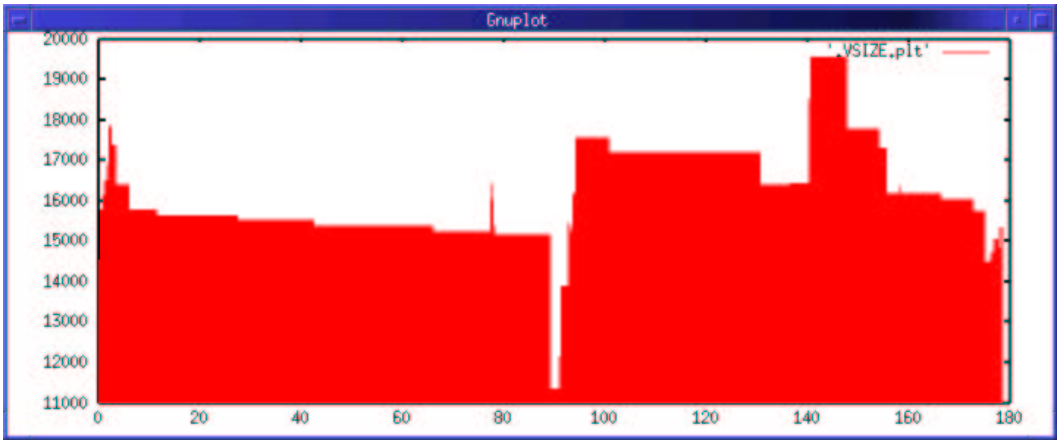

Figura 4.3: Gráfica del tamaño del proceso en memoria en Snort.

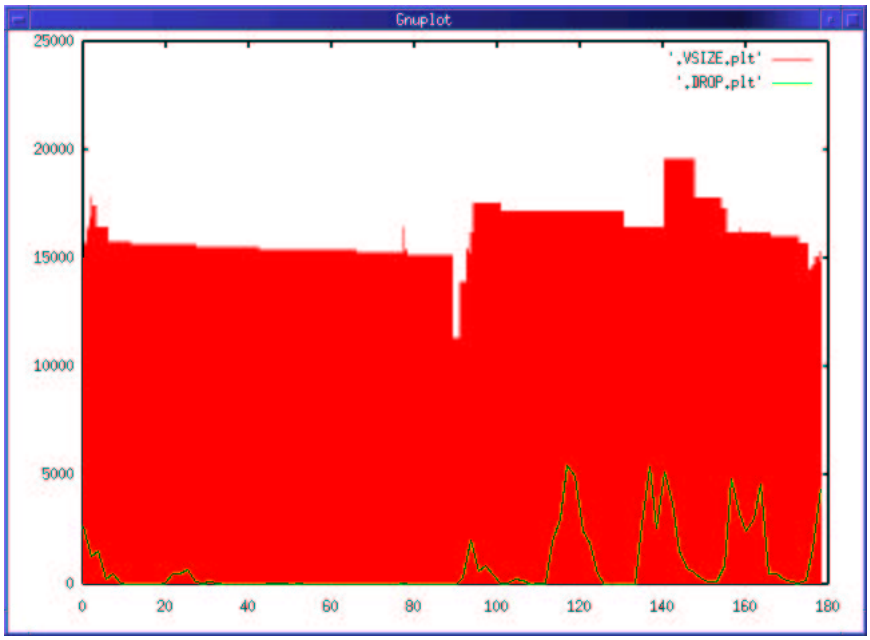

Figura 4.4: Gráfica comparativa entre el tamaño del proceso en memoria con respecto los paquetes descartados.

4.1.2. Snort+MySQL

En esta prueba, realizada durante la cuarta semana de junio de 2002, le hemos añadido a Snort una base de datos para almacenar las alertas que se producen. En teoría, al tener que convertir sus paquetes en inserciones a una base de datos SQL, Snort estará ocupado durante más tiempo en otras tareas que no son la de procesar paquetes, por lo que el descarte será mayor. En la figura 4.5vemos la comparación entre tráfico entrante y procesado:

Si echamos un vistazo a la misma gráfica pero de la prueba anterior, parece que ahora la diferencia entre los paquetes entrantes y los procesados sea menor, pero en teoría ahora la diferencia tendría que

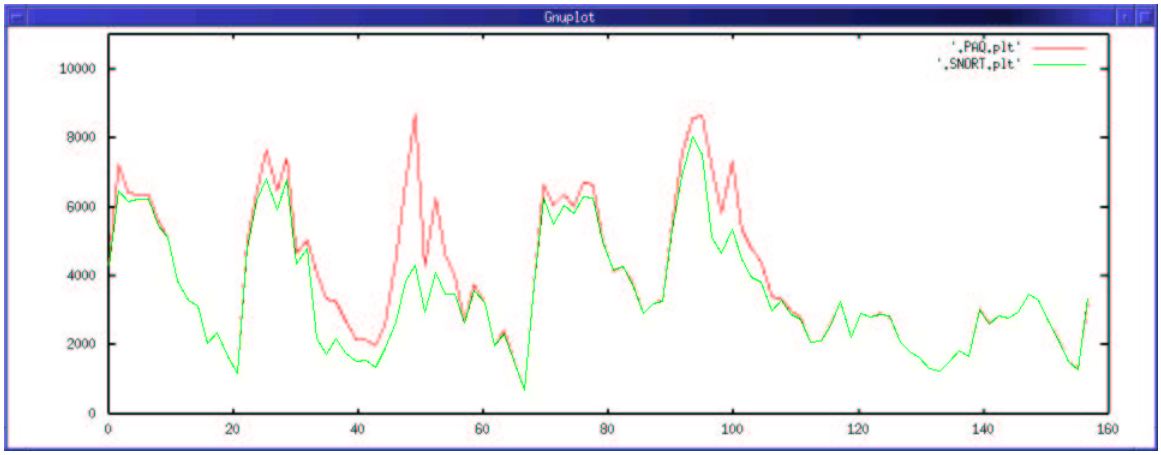

Figura 4.5: Gráfica comparativa entre el tráfico entrante y el procesador por Snort en Snort+MySQL.

ser mayor. Es más, si miramos en la figura 4.6, veremos que aparentemente ahora hay un número menor de paquetes descartados:

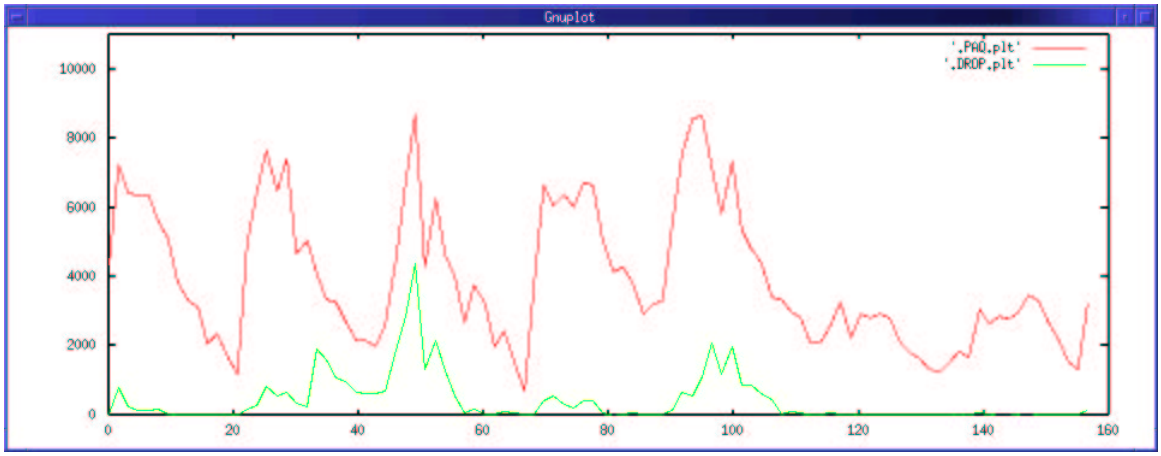

Figura 4.6: Gráfica comparativa entre el tráfico entrante y el descartado en Snort+MySQL.

La explicación es la siguiente: durante la semana en la que se utilizó Snort+MySQL el tráfico fue inferior al de la semana anterior en la que no se usó MySQL. Vemos como durante la prueba de Snort el tráfico llegó a los 13,000 pps y en la prueba de Snort+MySQL no alcanzó los 9,000 pps, por lo que aunque la configuración Snort sea más rápida, al tener más tráfico que procesar es perfectamente posible que descarte más paquetes en porcentaje. Como el criterio de evaluación del rendimiento se basa en la carga y ésta no es igual en ambas pruebas, el cálculo del número medio de paquetes por segundo procesados nos puede dar una idea del rendimiento de cada configuración (ver tabla 4.1).

En la segunda semana (Snort+MySQL) se ha procesado más tráfico en términos relativos porque ha llegado menos, pero se ha procesado menos en términos absolutos. Como cabía esperar, la configuración con Snort solo es más rápida que con Snort+MySQL.

Por último veamos si existe alguna relación entre el tamaño en memoria del programa y los paque-

	Snort	Snort+MySQL
Tráfico procesado	84.07 %	89.33 %
Media de PPS recibidos	5060.45	4014.29
Media de PPSs procesados	4254.12	3586.01

Cuadro 4.1: Comparación del rendimiento en Snort y Snort+MySQL.

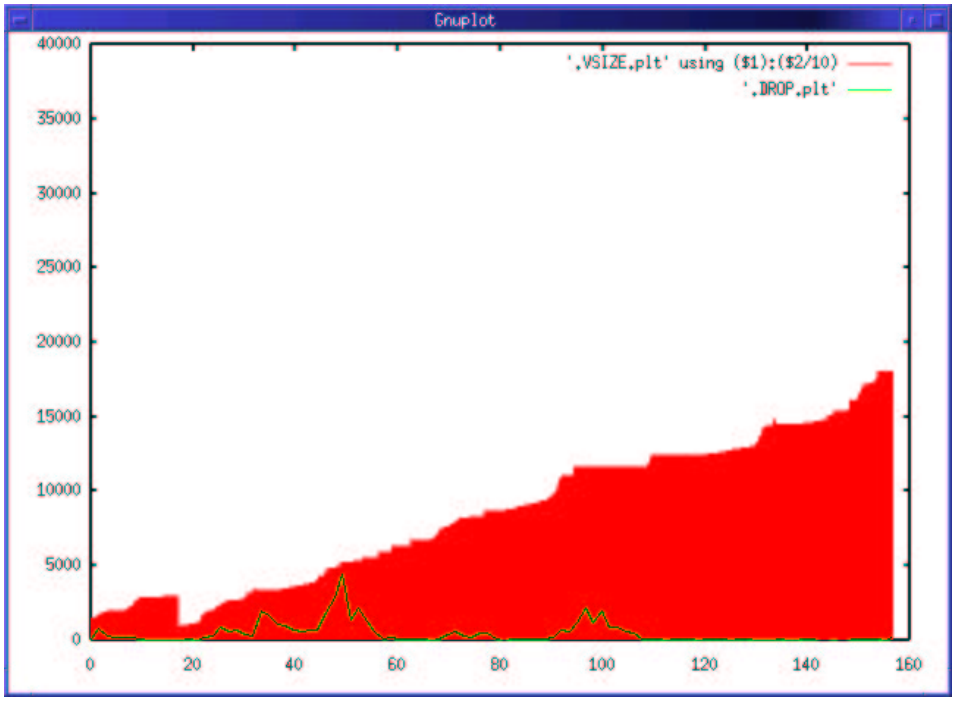

Figura 4.7: Gráfica comparativa entre el tamaño del proceso en memoria con respecto a los paquetes descartados en Snort+MySQL.

tes descartados. En la figura 4.7 no se aprecia la variación de tamaño del programa cuando aumenta el número de paquetes descartados, ni tampoco la estabilidad del tamaño cuando no hay descarte, por lo que no podemos asegurar las afirmaciones del apartado anterior. El estudio del comportamiento de Snort es un tema complejo que debiera ser tratado en profundidad quizá en otro proyecto.

4.2. Resultados de la simulación de ataques

4.2.1. Ataques estándar

La simulación de ataques se hizo utilizando la configuración de Snort con MySQL como back-end. Durante cuatro días se lanzaron cada cinco minutos tres ataques de exploit distintos. Los resultados de estas pruebas se muestran en la tabla 4.2. En esta tabla vemos los días en los que se lanzaron los ataques, los ataques realizados y las horas a las que tuvieron lugar los fallos de detección. En la columna de la derecha vemos el porcentaje de falsos negativos de un total de 1032 pruebas por exploit lanzado.

Ataque	Jueves	Viernes	Sábado	Domingo	Falsos negativos %
wu-ftpd *exploit*	·	12:40h, 12:47h	·	·	0.2 %
rpc.rstatd *exploit*	10:58h, 17:50h	8:56h	·	·	0.3 %
Apache *DoS*	·	·	·	·	0 %

Cuadro 4.2: Resultados de la simulación de ataques.

Vemos como es en jueves y viernes cuando se producen falsos negativos. Esto lo corroboran los resultados del apartado anterior, donde veíamos que era de lunes a viernes cuando se producían descartes de paquetes.

4.2.2. Ataques enmascarados

Se han utilizado dos técnicas para enmascarar los ataques, tal y como comentamos en el capítulo anterior: modificaciones sintácticas y evasión e inserción, que veremos a continuación.

4.2.2.1. Modificaciones sintácticas del patrón de búsqueda

En las peticiones al servidor web, probamos lo siguiente:

```
GET /etc/passwd
```

lo que dio como resultado esta alerta:

```
WEB-MISC /etc/passwd
```

La regla que la genera se encuentra en el fichero `web-misc.rules` y busca en el contenido de las peticiones al puerto 80 la cadena '/etc/passwd'. Para ver si Snort reconoce los caracteres % de HTTP probamos:

```
GET /%65%74%63/%70%61%73%73%77%64
```

equivalente a la anterior para el servidor web, y Snort acertó y generó la misma alerta. Snort resuelve por tanto los caracteres % de HTTP. Pero cuando probamos:

```
GET /etc/./passwd
```

Snort no fue capaz de detectar el ataque.

Snort dispone de un preprocesador llamado `http_decode` que se encarga de traducir las cadenas con formatos distintos al ASCII al formato ASCII, y es la cadena traducida la que pasa al motor de búsqueda. Este preprocesador funciona bien con caracteres %, pero como hemos comprobado no traduce la ruta de acceso a un fichero tal y como lo haría un sistema operativo.

4.2.2.2. Evasión e inserción

Para estas pruebas se ha creado una regla nueva que genera una alarma llamada “SENDMAIL mi Exploit” cuando detecta la cadena “mi_exploit” en una conexion dirigida al puerto 25. La regla es la siguiente:

```
alert tcp $EXTERNAL_NET any <>$HOME_NET 25 (msg:"SENDMAIL mi Exploit";
flags:A+; content:"mi_exploit"; sid:3379;)
```

Esta vez el ataque lo lanzaremos desde dentro a una máquina externa a la Universidad. El ataque consistirá en hacer una conexión al puerto 25 y teclear la cadena “mi_exploit”. Trataremos de evitar que el IDS detecte el ataque con algunas de las técnicas documentadas en [20] y [25].

Las pruebas de evasión e inserción que se utilizaron fueron las siguientes:

Fragmentar los paquetes IP.

Con esta prueba queríamos comprobar que la pila IP de Snort era capaz de reensamblar fragmenos y detectar así ataques fragmentados. Para lanzar el ataque, la confiugración de Fragroute empleada fue la siguiente:

```
ip_frag 8
print
```

Con esto le indicamos que debe fragmentar los paquetes IP que reciba a un tamaño de 8 bytes más la cabecera, excepto el primer paquete, que contiene la cabecera a nivel de transporte.

La traza del ataque vista con *tcpdump* `www.tcpdump.org` que se obtuvo fue:

```
   10:10:12.143971 147.156.2.19.32793 >217.127.34.166.25: P ack 75 win 5840 <nop,nop,timestamp
2623837 90438187>(frag 23837:32@0+) [tos 0x10]
   0x0000   4510 0034 5d1d 2000 3f06 6cc2 939c 0213        E..4]...?.l.....
   0x0010   d97f 22a6 8019 0019 bbb6 5bfb 9431 105d        ..".......[..1.]
   0x0020   8018 16d0 6050 0000 0101 080a 0028 095d        ....‘P.......(.]
   0x0030   0563 fa2b                                      .c.+
   10:10:12.143971 147.156.2.19 >217.127.34.166: (frag 23837:8@32+) [tos 0x10]
   0x0000   4510 001c 5d1d 2004 3f06 6cd6 939c 0213        E...]...?.l.....
   0x0010   d97f 22a6 6d69 5f65 7870 6c6f                  ..".mi_explo
   10:10:12.143971 147.156.2.19 >217.127.34.166: (frag 23837:4@40) [tos 0x10]
   0x0000   4510 0018 5d1d 0005 3f06 8cd9 939c 0213        E...]...?.......
   0x0010   d97f 22a6 6974 0d0a                           ..".it..
```

donde vemos tres fragmentos correspondientes a un solo paquete IP. El primero contiene la cabecera del segmento TCP, el segundo contiene la cadena “mi_explo” y el tercero la cadena “it”.

El ataque fue detectado por Snort con dos alertas:

```
"SENDMAIL mi Exploit"
"MISC Tiny Fragments"
```

La primera alerta corresponde a la regla que creamos, y la segunda viene dentro del conjunto de reglas de Snort y aparace cuando se ha detectado un fragmento cuyo payload es menor a 25 bytes.

Desordenar los paquetes IP.

Para que Fragroute desordene los fragmentos creados, usaremos la siguiente configuración:

```
ip_frag 8
order random
print
```

El orden es importante, y vemos que cuando ejecutamos Fragroute, éste produce la siguiente salida:

```
fragroute: ip_frag ->order ->print
```

Es decir, primero fragmenta en payloads de 8 bytes, luego desordena y por último muestra por pantalla.

El ataque generó este tráfico:

```
   10:14:20.303971 147.156.2.19.32793 >217.127.34.166.25: P ack 53 win 5840 <nop,nop,timestamp
2648648 90462668>(frag 23850:32@0+) [tos 0x10]
   0x0000   4510 0034 5d2a 2000 3f06 6cb5 939c 0213        E..4]*..?.l.....
   0x0010   d97f 22a6 8019 0019 bbb6 5c37 9431 10df        .."......\7.1..
   0x0020   8018 16d0 9f05 0000 0101 080a 0028 6a48        .............(jH
   0x0030   0564 59cc                                      .dY.
   10:14:20.303971 147.156.2.19 >217.127.34.166: (frag 23850:4@40) [tos 0x10]
   0x0000   4510 0018 5d2a 0005 3f06 8ccc 939c 0213        E...]*..?.......
   0x0010   d97f 22a6 6974 0d0a                            .".it..
   10:14:20.303971 147.156.2.19 >217.127.34.166: (frag 23850:8@32+) [tos 0x10]
   0x0000   4510 001c 5d2a 2004 3f06 6cc9 939c 0213        E...]*..?.l.....
   0x0010   d97f 22a6 6d69 5f65 7870 6c6f                  .".mi_explo
```

Al igual que antes vemos dos tres framentos, esta vez en distinto orden.

Snort detectó el ataque con las mismas alertas que antes: "SENDMAIL mi Exploit" y "MISC Tiny Fragments".

Expiración del time-out de los fragmentos

Vamos a configurar el último fragmento enviado para que sea enviado tras 59 segundos. La configuración de Fragroute es la siguiente:

```
ip_frag 8
delay last 59000
print
```

El ataque es detectado, puesto que Snort tambien espera 60 segundos para descartar los fragmentos. En Snort el encargado de los fragmento es el preprocesador frag2. En él el puede configurar el time-out de los fragmentos y la cantidad de memoria que va a dedicar al reensamblado de paquetes.

Solapar fragmentos IP.

El último ataque que probaremos será el de solapamiento de fragmentos IP favoreciendo los fragmentos nuevos. Para ello, el fichero fragroute.conf contendrá los siguiente:

```
ip_frag 8 new
print
```

Esta vez, haremos la cadena más larga para que aparezca un mayor número de fragmentos y facilitar la ocultación. La cadena será "vamos_a_mi_exploit_enganyarlo".

Una vez lanzado el ataque, el tráfico generado es el siguiente:

```
   10:44:12.503971 147.156.2.19.32803 >217.127.34.166.25: P ack 101 win 5840 <nop,nop,timestamp
2827824 90628412>(frag 30180:32@0+) [tos 0x10]
   0x0000   4510 0034 75e4 2000 3f06 53fb 939c 0213        E..4u...?.S.....
   0x0010   d97f 22a6 8023 0019 3231 9277 0ac8 5559        .."..#..21.w..UY
   0x0020   8018 16d0 b423 0000 0101 080a 002b 2630        .....#.......+&0
   0x0030   0566 e13c                                      .f.<
   10:44:12.503971 147.156.2.19 >217.127.34.166: (frag 30180:8@32+) [tos 0x10]
   0x0000   4510 001c 75e4 2004 3f06 540f 939c 0213        E...u...?.T.....
   0x0010   d97f 22a6 7661 6d6f 735f 615f                  .."vamos_a_
   10:44:12.503971 147.156.2.19 >217.127.34.166: (frag 30180:8@48+) [tos 0x10]
   0x0000   4510 001c 75e4 2006 3f06 540d 939c 0213        E...u...?.T.....
   0x0010   d97f 22a6 464c 7457 7651 3500                  .."FLtWvQ5.
   10:44:12.503971 147.156.2.19 >217.127.34.166: (frag 30180:7@56) [tos 0x10]
   0x0000   4510 001b 75e4 0007 3f06 740d 939c 0213        E...u...?.t.....
   0x0010   d97f 22a6 7961 726c 6f0d 0a                    .."yarlo..
   10:44:12.503971 147.156.2.19 >217.127.34.166: (frag 30180:16@40+) [tos 0x10]
   0x0000   4510 0024 75e4 2005 3f06 5406 939c 0213        E..$u...?.T.....
   0x0010   d97f 22a6 6d69 5f65 7870 6c6f 6974 5f65        .."mi_exploit_e
   0x0020   6e67 616e                                      ngan
```

Vemos que el primer fragmento contiene la cabecera TCP y los 4 fragmentos restantes la cadena con el ataque y algo más. Si colocamos la información que contienen en forma de tabla (ver tabla 4.4) comprobamos que el contenido de los fragmentos se solapa.

Fragmento																				
32-40																				
48-56									F	L	t	W	v	Q	5					
56-63																	y	a	r	l	o
40-56	m	i	_	e	x	p	l	o	i	t	_	e	n	g	a	n					

Cuadro 4.4: Fragmentos del 3 al 5 del ataque de inserción de solapamiento.

Cuando esto ocurre Snort favorece los fragmentos antiguos, por lo que su reconstrucción del flujo TCP devolverá la cadena “vamos_a_mi_exploFLtWvQ5nyarlo”, que al no contener la firma “mi_exploit” no generará ninguna alarma. Si la máquina víctima favorece los nuevos, su reconstrucción del flujo será “vamos_a_mi_exploit_enganyarlo”.

4.3. Caso práctio de análisis de ataques

Como truco para facilitar el análisis hemos utilizado los logs que ha generado Netfilter sobre las conexiones entrantes al IDS. Como en teoría el IDS tiene que ser desconocido por el resto del mundo, hemos hecho una búsqueda en la base de datos generada por Snort de las direcciones IP que hemos encontrado como origen en nuestro log de Netfilter, por lo que en principio estamos investigando directamente sobre presuntas IP atacantes.

4.3.1. IDS monitorizando una VLAN

Los ejemplos que veremos a continuación son casos reales de alarmas generadas por el IDS monitorizando la VLAN de informática, y fueron tomados durante enero de 2002. Los filtros de Snort han sido modificados para que ignoren los eventos producidos por escaneadores de red, servidores Web, proxy-cache Web y DNS. Algunas reglas también han sido eliminadas al provocar demasiados falsos positivos ('SCAN Proxy attempt', 'BAD TRAFFIC bad frag bits', 'ICMP Destination Unreachable (Port Unreachable)', 'ICMP Echo Replay (Undefined Code!)'). Por supuesto, antes de eliminar una regla hay que comprobar que la causa de que genere tantas alarmas está justificada.

Existen multitud de entornos gráficos para la visualización de las alertas generadas por Snort, entre los más destacados ACID, IDScenter, Snort Report y Snort Monitor, todos disponibles en la web de Snort [22]. Estos entornos funcionan con PHP, así que se hace necesaria la instalación de un servidor web dentro del IDS, lo cual le da más carga y le resta seguridad. Para evitar esto, la forma en la que accederemos a las alertas generadas por Snort almacenadas en MySQL será por medio de unos scripts en Perl. Estos scripts nos permiten obtener información de una forma más flexible y rápida que los bonitos pero costosos programas PHP.

El script principal es *ranking*, que nos muestra para cada alerta lo siguiente:

- `COUNT`: el número total de veces que aparece en la BD.
- `SIG_ID`: el identificador con el que aparece.
- `NUM_SRC`: el número de direcciones IP origen distintas de esa alarma.
- `NUM_DST`: el número de direcciones IP destino distintas de esa alarma.
- `SIG_NAME`: breve mensaje explicando la causa que provocó la alarma
- `REF`: URL o referencia en una base de datos donde se encuentra una explicación más detallada sobre el ataque.

La salida es la siguiente:

```
COUNT   SIG_ID  NUM_SRC NUM_DST SIG_NAME (REF)
-------------------------------------------------------------------------------------------
585889  17660   981     301     WEB-IIS cmd.exe access
222337  1       130     44      RPC portmap request rstatd (arachnids 10)
77690   12926   4       5       NETBIOS nimda .eml (url www.datafellows.com/v-descs/nimda.shtml)
68499   17661   1025    316     WEB-IIS CodeRed v2 root.exe access
65128   17666   9       4       WEB-IIS multiple decode attempt (cve CAN-2001-0333)
62721   17665   50      10      WEB-IIS scripts access
57057   17662   939     261     WEB-FRONTPAGE /_vti_bin/ access
51818   17689   1       1       EXPLOIT ssh CRC32 overflow NOOP (bugtraq 2347)
34651   44      351     221     MISC Large ICMP Packet (arachnids 246)
16720   275     47      47      INFO msn chat access
4297    131     124     191     SHELLCODE x86 NOOP (arachnids 181)
4114    17672   1       1       NETBIOS nimda .nws (url www.datafellows.com/v-descs/nimda.shtml)
4080    1054    18      27      INFO Possible IRC Access
2807    320     60      17      RPC portmap request mountd (arachnids 13)
1923    4646    162     12      spp_stream4: STEALTH ACTIVITY (Vecna scan) detection
1782    259     206     61      INFO FTP anonymous FTP
966     12      157     2       WEB-MISC count.cgi access (bugtraq 550) (cve CVE-1999-0021)
712     108     55      2       SMTP RCPT TO overflow (cve CAN-2001-0260) (bugtraq 2283)
604     1048    55      12      EXPLOIT ssh CRC32 overflow filler (bugtraq 2347)
```

```
543     716     21      28      SCAN nmap TCP (arachnids 28)
380     17686   3       269     spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection
341     1950    26      77      FTP Bad login
309     58      9       60      TELNET Bad Login
254     28      10      160     INFO Napster Client Data
187     200     5       43      Virus - Possible scr Worm
168     306     43      102     DNS named version attempt (arachnids 278)
164     119     34      50      SHELLCODE x86 setuid 0 (arachnids 436)
142     11823   24      11      SHELLCODE x86 unicode NOOP
55      330     4       40      Virus - Possible pif Worm
45      2169    3       3       BAD TRAFFIC same SRC/DST
41      3852    7       1       DNS zone transfer (arachnids 212)
33      4687    4       5       spp_stream4: STEALTH ACTIVITY (FIN scan) detection
30      10670   1       17      Virus - Possible NAVIDAD Worm
26      167     11      1       TCP SYN received
16      8204    3       7       TELNET Bad Login
16      329     3       5       Virus - Possible MyRomeo Worm
16      3981    2       4       Virus - Possible PrettyPark Trojan (MCAFEE 10175)
15      3087    9       7       BAD TRAFFIC udp port 0 traffic
14      667     8       5       spp_stream4: STEALTH ACTIVITY (NULL scan) detection
14      5507    3       11      RPC EXPLOIT statdx (arachnids 442)
14      17687   2       7       FTP CWD / - possible warez site
14      5931    6       4       BAD TRAFFIC tcp port 0 traffic
13      17674   1       1       MISC Tiny Fragments
12      6114    8       4       SCAN FIN (arachnids 27)
12      331     10      8       SHELLCODE x86 setgid 0 (arachnids 284)
9       17659   5       4       spp_stream4: STEALTH ACTIVITY (nmap XMAS scan) detection
8       17658   4       3       spp_stream4: NMAP FINGERPRINT (stateful) detection
8       17673   7       4       WEB-MISC apache DOS attempt
8       1436    2       2       INFO Napster Client Data
6       5503    2       6       FTP EXPLOIT format string (arachnids 453)
3       17726   1       1       WEB-CGI finger access (arachnids 221) (cve CVE-1999-0612)
3       17695   1       1       EXPLOIT ssh CRC32 overflow (bugtraq 2347) (cve CVE-2001-0144)
3       17677   3       3       SCAN nmap fingerprint attempt (arachnids 05)
3       17685   2       2       ATTACK RESPONSES id check returned root
2       1998    1       1       PORN free XXX
2       17667   2       2       SCAN XMAS (arachnids 144)
2       7979    2       2       SCAN NULL (arachnids 4)
2       3653    1       1       DOS Winnuke attack (bugtraq 2010) (cve CVE-1999-0153)
2       17700   1       1       RSERVICES rsh root (arachnids 389)
1       4292    1       1       Virus - Possible NAIL Worm (MCAFEE 10109)
1       17716   1       1       WEB-MISC BigBrother access
1       17698   1       1       FTP CWD ...
1       17761   1       1       WEB-ATTACKS cc command attempt
1       17693   1       1       SCAN ident version (arachnids 303)
1       17691   1       1       SCAN NMAP XMAS (arachnids 30)
1       17707   1       1       WEB-MISC cat%20 access (cve CVE-1999-0039) (bugtraq 374)
1       17765   1       1       spp_stream4: STEALTH ACTIVITY (Full XMAS scan) detection
```

Para ver por qué se han producido las alertas es de gran ayuda echar un vistazo a la regla que la define dentro de Snort (ficheros de configuración `*.rules`). No haremos un análisis detallado de cada alarma, tan solo veremos algunas:

585889 17660   981   301   WEB-IIS cmd.exe access

Esta regla salta cuando en el contenido de una sesión TCP aparece la cadena `cmd.exe` y el destino es un puerto 80.

Snort también nos permite configurarlo de forma que este tipo de reglas (WEB-IIS) solo salten cuando el destino sea uno de nuestros servidores web, por medio de una variable `$HTTP_SEVERS` que habremos rellenado con todos nuestros servidores web. Esto ralentizará menos el IDS puesto

que la regla está más acotada (cuantos menos grados de libertad tenga una regla, más rápida se ejecutará). El problema es que en una organización no siempre es posible llevar la cuenta de todas las máquinas que hay como servidores web (por ejemplo, muchos usuarios instalan en su PC de escritorio Windows 2000 Server sin saber que el IIS estará activo por defecto), por lo que el mejor valor para $HTTP_SERVERS es ANY.

El encontrar cmd.exe en una conexión TCP dirigida al puerto 80 de un servidor web suele significar que el gusano *Nimda* está intentando colarse en ese servidor. El gusano solo compromete sistemas que ejecutan IIS 4.0 e IIS 5.0 en los sistemas operativos Windows NT y Windows 2000.

El gusano envía su código como un requerimiento HTTP. El requerimiento HTTP ejecuta un exploit sobre una vulnerabilidad conocida del IIS, lo cual permite que el gusano se ejecute en el sistema víctima, depositando 3 archivos:

- readme.eml
- sample.eml
- desktop.eml

Dentro de los logs de los equipos que intenta atacar dicho gusano y propagarse se pueden apreciar diversos patrones de comportamiento e intentos de búsqueda para atacar y así seguir propagándose.

```
132.248.XXX.XXX - - [18/Sep/2001:10:16:47 -0400] "GET /scripts/root.exe?/c+dir HTTP/1.0" 404 287
"-" "-"
132.248.XXX.XXX - - [18/Sep/2001:10:16:48 -0400] "GET /MSADC/root.exe?/c+dir HTTP/1.0" 404 285
"-" "-"
132.248.XXX.XXX - - [18/Sep/2001:10:16:48 -0400] "GET /c/winnt/system32/cmd.exe?/c+dir HTTP/1.0"
404 295 "-" "-"
132.248.XXX.XXX - - [18/Sep/2001:10:16:49 -0400] "GET /d/winnt/system32/cmd.exe?/c+dir HTTP/1.0"
404 295 "-" "-"
132.248.XXX.XXX - - [18/Sep/2001:10:16:49 -0400] "GET
/scripts/..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 309 "-" "-"
132.248.XXX.XXX - - [18/Sep/2001:10:16:50 -0400] "GET
/_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 404 326 "-" "-
"
132.248.XXX.XXX - - [18/Sep/2001:10:16:50 -0400] "GET
/_mem_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir
```

El que esta alarma haya saltado casi 600,000 veces significa que este virus está aun muy vivo en nuestra red y/o en Internet. Veamos con el comando *shsig* los 10 primeros orígenes y destinos de estos ataques:

```
## WEB-IIS cmd.exe access ## 17660 ##

COUNT   TOT_SRC TOT_DST TIME_INI               TIME_FIN               SRC
-------------------------------------------------------------------------------------------
280996  415694  698     2001-12-27 00:57:14    2002-01-08 16:35:53    pericles.adeit.uv.es
164308  237721  1083    2001-12-22 04:24:27    2001-12-25 22:25:24    212.146.128.223
53263   76123   417     2001-12-19 10:05:53    2001-12-21 10:38:05    auladeit40.adeit.uv.es
22781   29592   181     2001-12-19 12:49:53    2002-01-07 21:41:03    contab21.admeco.uv.es
19661   25454   147     2002-01-07 15:58:16    2002-01-08 13:10:59    tornasol.electron.uv.es
11957   15948   185     2001-12-19 10:23:22    2002-01-07 09:11:05    galadriel.fbiolo.uv.es
9569    12109   129     2002-01-07 14:53:13    2002-01-09 12:16:22    servuniversia.dise.uv.es
3482    5342    0       2001-12-28 03:23:33    2001-12-28 03:26:22    147.46.67.224
1863    2208    232     2001-12-24 12:52:37    2002-01-08 20:34:13    visionarc.irobot.uv.es
442     526     2       2001-12-20 14:11:15    2001-12-26 07:14:32    c2.sll.se
```

```
COUNT   TOT_SRC TOT_DST TIME_INI               TIME_FIN               DST
-------------------------------------------------------------------------------------------
324331  10      529371  2001-12-19 10:31:01    2002-01-11 04:50:49    vpnserver.ci.uv.es
9285    4659    10924   2001-12-19 10:10:16    2002-01-12 16:58:21    capone.informat.uv.es
6617    9612    7699    2001-12-19 10:32:10    2002-01-12 06:07:18    catafar4b1.ci.uv.es
6063    9944    7074    2001-12-19 10:27:53    2002-01-11 16:31:26    catafar2a.ci.uv.es
5917    9635    6881    2001-12-19 11:21:48    2002-01-11 04:47:18    catafar4a.ci.uv.es
5585    120     6822    2001-12-19 10:54:00    2002-01-09 11:10:53    suelos.bioveg.uv.es
5421    9979    6319    2001-12-19 10:10:46    2002-01-11 04:47:19    catafar4a1.ci.uv.es
5374    895     6381    2001-12-19 10:22:14    2002-01-10 17:09:26    teruca.informat.uv.es
5331    280     59125   2001-12-19 10:44:46    2002-01-12 11:43:04    tapec.informat.uv.es
5320    24221   6220    2001-12-19 10:15:44    2002-01-09 13:10:51    nexus.informat.uv.es
```

La mayoría de estos orígenes (recordamos que solo aparecen listados los 10 primero) pertenecen a nuestra red, y serán casi con toda seguridad debidos a que el gusano ha infectado esas máquinas.

Es ahora cuando se podrían poner en marcha los dos mecanismos de respuesta vistos en la introducción:

- Activa: en el momento en el que detectamos que una máquina intenta infectar a otras, el IDS podría terminar la conexión en ambos lados mediante un segmento TCP RST. Esto lo hace falseando su dirección IP (que puede no tener) para engañar a ambas partes. A esta técnica también se le conoce como *TCP hijacking* y para que funcione debemos asegurarnos de que las interfaces de los routers por los que pasamos no tengan filtros anti-spoofing.
- Pasiva: el IDS podría estar configurado para buscar en una BD la dirección de correo del responsable y avisarle de que su máquina está infectada.

Por último vemos que el servidor VPN (vpnserver.ci.uv.es) aparece como víctima más frecuente del gusano Nimda, aunque es un poco extraño que el número de ataques recibidos sea tan alto.

51818 17689 1 1 EXPLOIT ssh CRC32 overflow NOOP (bugtraq 2347)

Esta es una alarma provocada por una cadena hexadecimal (`90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90`) sobre el puerto 22 de una conexión TCP que apareció en un programa liberado en Internet que explotaba una vulnerabilidad en un servidor SSH. Esta ha sido la firma elegida para detectar un ataque de este tipo.

```
## EXPLOIT ssh CRC32 overflow NOOP ## 17689 ##

COUNT   TOT_SRC TOT_DST TIME_INI               TIME_FIN               SRC
-------------------------------------------------------------------------------------------
51818   51818   0       2001-12-31 13:38:04    2001-12-31 13:58:42    remote02.sn.com

COUNT   TOT_SRC TOT_DST TIME_INI               TIME_FIN               DST
-------------------------------------------------------------------------------------------
51818   281     59150   2001-12-31 13:38:04    2001-12-31 13:58:42    tapec.informat.uv.es
```

Una vez ejecutado *shsig*, vemos que no tiene mucho sentido este comportamiento tratándose de un exploit. Normalmente, el atacante lanzará el exploit un par de veces para ver si tiene éxito, y de ser así entrará como `root`. En caso contrario, lo intentaría con otra máquina que escuche en ese puerto.

Lo que puede estar ocurriendo es que, al tratarse de una comunicación encriptada, esta cadena aparezca en dicha comunicación, y el IDS de un falso positivo pensando que ha detectado un ataque.

16720 275 47 47 INFO msn chat access

Aunque esto no es ningún ataque, dada la política de seguridad de la organización este tipo de actividades puede que no estén permitidas.

164 119 34 50 SHELLCODE x86 setuid 0 (arachnids 436)

Es muy posible que esta regla de muchos falsos positivos, pero algunas veces también acertará y nos pondrá en la pista de algún ataque.

Esta regla busca el código hexadecimal correspondiente a la llamada al sistema `'setuid 0'` en una arquitectura `x86`, esto es `'b017 cd80'`. Esta llamada es ejecutada normalmente por los `shellcodes` de la mayoría de los exploits, pero su firma es demasiado corta como para asegurar que cada vez que la encontremos en una conexión TCP sea debido a un intento de penetración en un sistema, por lo que provocará muchas falsas alarmas.

Además, al ser un IDS interno que monitoriza una VLAN, podremos tener ataques incluso desde los servidores proxy, como vemos a continuación:

```
## SHELLCODE x86 setuid 0 ## 119 ##

COUNT   TOT_SRC TOT_DST TIME_INI               TIME_FIN               SRC
-------------------------------------------------------------------------------------------
75      8521    738     2001-12-03 12:14:03    2002-01-08 20:40:01    proxy2.ci.uv.es
50      10173   32928   2001-12-03 17:27:14    2002-01-07 09:36:32    proxy1.ci.uv.es
2       16      17      2002-01-09 16:43:58    2002-01-09 16:43:58    vicky.irobot.uv.es
2       2       0       2002-01-10 11:17:36    2002-01-10 11:17:39    dsl170.boi.micron.net
2       2208    232     2001-12-17 12:03:12    2001-12-17 12:09:01    visionarc.irobot.uv.es
2       2       0       2002-01-09 13:08:59    2002-01-09 13:09:03    24.65.99.127
1       1       0       2001-12-18 12:42:59    2001-12-18 12:42:59    pD951E322.dip.t-dialin.net

COUNT   TOT_SRC TOT_DST TIME_INI               TIME_FIN               DST
-------------------------------------------------------------------------------------------
29      1544    1425    2001-12-03 13:48:22    2002-01-09 13:09:03    chusta.ffarma.uv.es
15      998     3099    2001-12-03 13:52:43    2002-01-09 13:28:39    bigelnota.farmatec.uv.es
15      92      3121    2001-12-03 14:42:03    2001-12-12 14:53:11    eulogiov.microb.uv.es
15      950     792     2001-12-03 17:27:14    2002-01-10 11:17:39    adrfarma2.ffarma.uv.es
11      176     85      2001-12-03 19:22:50    2001-12-20 17:00:44    jurasic.farmac.uv.es
6       0       95      2001-12-03 12:14:03    2001-12-12 21:07:57    vgcasabo.farmatec.uv.es
6       7283    2199    2001-12-14 20:49:30    2002-01-12 03:31:13    linea900.farmatec.uv.es
6       284     1466    2001-12-03 18:25:44    2001-12-21 07:03:11    tcrom.farmatec.uv.es
4       179     577     2001-12-03 13:52:09    2001-12-27 11:46:15    secfarma.farmac.uv.es
4       69      134     2001-12-14 15:43:39    2001-12-14 15:49:19    godzilla.ffarma.uv.es
```

Pero también vemos que hay otras IPs de fuera que han hecho saltar la alarma, podemos prestarle más atención a esas. Por ejemplo, vamos a fijarnos en `h24-65-99-127.ed.shawcable.net`

Con el comando *shsigip* podemos concentrarnos en una firma y una IP. Obtenemos lo siguiente:

```
## SHELLCODE x86 setuid 0 ## 119 ##
SOURCE
-----------------

2002-01-09 13:08:59 h24-65-99-127.ed.shawcable.net ->chusta.ffarma.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:1500 ID:48966 Flags:0x0 Offset:0 TTL:111 Proto:6 Cksum:10860
[TCP] SPort:1214, DPort:1097 Seq:113522033 Ack:113078 Offset:5 Res:0 Flags:0x10 Win:8437
Cksum:35362 Urp:0 Res:0

2002-01-09 13:09:03 h24-65-99-127.ed.shawcable.net ->chusta.ffarma.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:1500 ID:24647 Flags:0x0 Offset:0 TTL:111 Proto:6 Cksum:35179
```

```
[TCP] SPort:1214, DPort:1097 Seq:113522033 Ack:113078 Offset:5 Res:0 Flags:0x10 Win:8437
Cksum:35362 Urp:0 Res:0
```

Podemos ver la fecha, la hora, y el tipo de paquete que se envió. Era un segmento TCP desde el puerto 1214 dirigido al 1097. No parece que hayan servidores importantes en este último puerto, así que puede tratarse de un falso positivo.

41 3852 7 1 DNS zone transfer (arachnids 212)

Las transferencias de zona solo deberían tener como origen servidores DNS secundarios. Cuando aparece una transferencia de zona iniciada desde fuera de la organización, es muy probable que alguien esté reuniendo información para llevar a cabo un ataque sobre nuestra red.

En este caso, `qfgate.quifis.uv.es` es un servidor DNS secundario de la UV que está mal configurado y acepta peticiones de transferencia de zona hacia máquinas que no son DNS de la UV.

```
## DNS zone transfer ## 3852 ##

COUNT   TOT_SRC TOT_DST TIME_INI               TIME_FIN               SRC
-------------------------------------------------------------------------------------
25      2775    8658    2001-12-21 09:34:58    2001-12-21 09:48:10    slabii.informat.uv.es
4       4       0       2001-12-11 12:51:37    2001-12-27 16:04:24    hostcount.ripe.net
2       8       2       2001-12-05 00:48:23    2001-12-05 00:48:23    194.85.9.53
2       2       0       2001-12-20 11:11:54    2001-12-20 11:11:54    acfpp06.acfp.upv.es
2       2       0       2001-12-29 01:03:35    2001-12-29 01:03:35    ineco.nic.es
2       2       0       2002-01-02 18:26:57    2002-01-02 18:26:57    208.5.183.250
2       2       0       2002-01-12 07:20:45    2002-01-12 07:20:45    ariston.netcraft.com

COUNT   TOT_SRC TOT_DST TIME_INI               TIME_FIN               DST
-------------------------------------------------------------------------------------
43      2389    174130  2001-12-05 00:48:23    2002-01-12 07:20:45    qfgate.quifis.uv.es
```

Lo que podríamos hacer ahora es, con el comando *shsip*, ver si alguna de estas IPs ha estado involucrada en otros ataques. Al mirar sobre `194.85.9.53` obtenemos lo siguiente:

```
SOURCE
NUM     SIG_ID  SIG_NAME
------------------------
5       5503    FTP EXPLOIT format string

2001-12-08 08:11:48 194.85.9.53 ->apotkpr.ffarma.uv.es
[IP] Ver:4 HLen:5 Tos:32 Len:64 ID:2978 Flags:0 Offset:0 TTL:47 Proto:6 Cksum:54106
[TCP] SPort:3086, DPort:21 Seq:2147483647 Ack:2147483647 Offset:5 Res:0 Flags:0x18 Win:32120 Ck-
sum:63768 Urp:0 Res:0

2001-12-08 09:33:37 194.85.9.53 ->aficio2.informat.uv.es
[IP] Ver:4 HLen:5 Tos:32 Len:76 ID:38242 Flags:0 Offset:0 TTL:47 Proto:6 Cksum:17912
[TCP] SPort:2184, DPort:21 Seq:618981048 Ack:2110736262 Offset:8 Res:0 Flags:0x18 Win:32120 Ck-
sum:36706 Urp:0 Res:0

2001-12-08 09:34:25 194.85.9.53 ->nexus.informat.uv.es
[IP] Ver:4 HLen:5 Tos:32 Len:76 ID:38628 Flags:0 Offset:0 TTL:47 Proto:6 Cksum:17462
[TCP] SPort:2254, DPort:21 Seq:669818701 Ack:1998022812 Offset:8 Res:0 Flags:0x18 Win:32120 Ck-
sum:7593 Urp:0 Res:0

2001-12-08 09:34:58 194.85.9.53 ->teruca.informat.uv.es
[IP] Ver:4 HLen:5 Tos:32 Len:76 ID:38828 Flags:0 Offset:0 TTL:47 Proto:6 Cksum:17216
[TCP] SPort:2304, DPort:21 Seq:697739214 Ack:2147483647 Offset:8 Res:0 Flags:0x18 Win:32120 Ck-
sum:14328 Urp:0 Res:0

2001-12-08 11:37:09 194.85.9.53 ->capone.informat.uv.es
```

```
[IP] Ver:4 HLen:5 Tos:32 Len:76 ID:12685 Flags:0 Offset:0 TTL:47 Proto:6 Cksum:43730
[TCP] SPort:4158, DPort:21 Seq:2147483647 Ack:2147483647 Offset:8 Res:0 Flags:0x18 Win:32120 Ck-
sum:30413 Urp:0 Res:0

2       3852    DNS zone transfer

2001-12-05 00:48:23 194.85.9.53 ->qfgate.quifis.uv.es
[IP] Ver:4 HLen:5 Tos:32 Len:75 ID:52910 Flags:0 Offset:0 TTL:47 Proto:6 Cksum:41847
[TCP] SPort:3657, DPort:53 Seq:2147483647 Ack:1718339697 Offset:8 Res:0 Flags:0x18 Win:32120 Ck-
sum:52791 Urp:0 Res:0

2001-12-05 00:48:23 194.85.9.53 ->qfgate.quifis.uv.es
[IP] Ver:4 HLen:5 Tos:32 Len:75 ID:52910 Flags:0 Offset:0 TTL:46 Proto:6 Cksum:42103
[TCP] SPort:3657, DPort:53 Seq:2147483647 Ack:1718339697 Offset:8 Res:0 Flags:0x18 Win:32120 Ck-
sum:52791 Urp:0 Res:0

1       44      MISC Large ICMP Packet

2001-12-08 09:34:08 194.85.9.53 ->flop.informat.uv.es
[IP] Ver:4 HLen:5 Tos:32 Len:1500 ID:38478 Flags:0 Offset:0 TTL:238 Proto:1 Cksum:49236
[ICMP] Type:0, Code:0 Cksum:34386 ID:39612 Seq:57072

DESTINATION
NUM     SIG_ID  SIG_NAME
------------------------
2       1950    FTP Bad login

2001-12-08 09:33:45 wooster.informat.uv.es ->194.85.9.53
[IP] Ver:4 HLen:5 Tos:16 Len:74 ID:55970 Flags:0 Offset:0 TTL:64 Proto:6 Cksum:61379
[TCP] SPort:21, DPort:2191 Seq:1424374098 Ack:623887842 Offset:8 Res:0 Flags:0x18 Win:32120 Ck-
sum:11079 Urp:0 Res:0

2001-12-08 09:34:19 tapec.informat.uv.es ->194.85.9.53
[IP] Ver:4 HLen:5 Tos:16 Len:74 ID:65510 Flags:0 Offset:0 TTL:64 Proto:6 Cksum:51802
[TCP] SPort:21, DPort:2232 Seq:991799545 Ack:652061349 Offset:8 Res:0 Flags:0x18 Win:32120 Ck-
sum:33904 Urp:0 Res:0
```

Esta IP se ha hecho con la lista de IPs-nombres de la UV desde un DNS secundario, y a los 3 días ha lanzado exploits FTP sobre algunas máquinas. También ha intentado entrar en algunos servidores FTP con un login incorrecto.

No podemos saber si los ataques exploit tuvieron éxito, por lo que deberíamos informar al administrador de estas máquinas de que hubo un intento de penetrar en ellas.

Si tenemos curiosidad, podríamos tratar de averiguar de donde es esta IP, aunque debemos saber que el atacante puede estar manipulando su IP origen. Para ello, podemos consultar las BBDD Whois existentes, que son las siguientes:

- Europa: `www.ripe.net`
- América: `www.arin.net`
- Asia y pacífico: `www.apnic.net`

Haciendo una búsqueda de la IP `194.85.9.53` en RIPE, obtenemos los siguiente:

```
inetnum: 194.85.8.0 - 194.85.9.255
netname: SIOBC
descr: Shemyakin-Ovchinnikov Institute of Bioorganic Chemistry
```

```
descr: Russian Academy of Sciences
descr: Moscow
country: RU
admin-c: IVM11-RIPE
tech-c: DEN2-RIPE
rev-srv: wowa.siobc.ras.ru
rev-srv: nss.ras.ru
status: ASSIGNED PI
notify: noc@nc.ras.ru
notify: root@wowa.siobc.ras.ru
mnt-by: AS3058-MNT
changed: eugene@nc.ras.ru 20010103
source: RIPE
```

Existen herramientas gráficas que nos permiten ver la ruta geográfica que atravesamos para llegar a una IP destino. Un ejemplo puede ser VisualRoute, de Visualware (`www.visualware.com/visualroute/index.html`). Probamos esta aplicación para la IP `194.85.9.53` y el resultado lo podemos ver en la figura 4.8:

Figura 4.8: Recorrido del ataque realizado desde Moscú.

Vemos que como punto inicial en España aparece 'El Puerto de Santa Maria'. Esto es así porque el Applet en Java que se descarga en nuestro navegador está configurado para que el punto de origen sea éste, puesto que el servidor VisualRoute al que hemos accedido se encuentra ahí.

Después de haber detectado el ataque y haber obtenido información del atacante, podríamos dar cuenta de lo ocurrido a nuestro CERT. En el caso de la UV avisaríamos a IRIS-CERT (de RedIRIS), miembro del FIRST (Forum of Incident Response and Security Teams, `www.first.org`) desde 1997.

Sobre qué hacer cuando detectamos una ataque existen obras completas. Ver [14].

14 5507 3 11 RPC EXPLOIT statdx (arachnids 442)

rpc.statd es un servidor RPC que implementa el protocolo NSM (Network Status Monitor). Se utiliza para monitorizar clientes y servidores NFS. En versiones antiguas este servidor tiene una vulnerabilidad que puede ser explotada remotamente. Veamos las alarmas que ha generado el IDS con respecto a esta firma:

```
## RPC EXPLOIT statdx ## 5507 ##
COUNT   TOT_SRC TOT_DST TIME_INI                TIME_FIN                SRC
------------------------------------------------------------------------
7       22      0       2001-12-08 08:39:45     2001-12-08 08:42:01     61.182.50.241
6       20      0       2001-12-26 07:53:22     2001-12-26 07:53:40     caetano.empresa.net
1       7       0       2001-12-26 09:27:31     2001-12-26 09:27:31     65.107.106.51

COUNT   TOT_SRC TOT_DST TIME_INI                TIME_FIN                DST
------------------------------------------------------------------------
2       365     340     2001-12-08 08:39:48     2001-12-26 07:53:26     bugs.informat.uv.es
2       39      45      2001-12-08 08:39:45     2001-12-26 07:53:22     iris.quiorg.uv.es
2       2389    174287  2001-12-08 08:41:12     2001-12-08 08:41:12     qfgate.quifis.uv.es
1       6       33      2001-12-26 09:27:31     2001-12-26 09:27:31     crean.red.uv.es
1       0       22      2001-12-26 07:53:40     2001-12-26 07:53:40     wild.red.uv.es
1       84      4647    2001-12-26 07:53:27     2001-12-26 07:53:27     marvin.informat.uv.es
1       1326    5619    2001-12-26 07:53:27     2001-12-26 07:53:27     taz.informat.uv.es
1       44      7158    2001-12-26 07:53:27     2001-12-26 07:53:27     tro.informat.uv.es
1       4199    22      2001-12-08 08:39:50     2001-12-08 08:39:50     crac.informat.uv.es
1       0       13      2001-12-08 08:42:00     2001-12-08 08:42:00     hussey.red.uv.es
```

La IP `61.182.50.241` ha generado 7 alarmas en un periodo de algo más de 2 minutos y aparece como origen de 22 alarmas en total. Vamos a ver estas alarmas con un poco más de detalle:

```
SOURCE
NUM     SIG_ID  SIG_NAME
-----------------------
14      1       RPC portmap request rstatd
7       5507    RPC EXPLOIT statdx
1       44      MISC Large ICMP Packet

DESTINATION
NUM     SIG_ID  SIG_NAME
-----------------------
```

Vemos como además de lanzar exploits, ha emitido peticiones al servidor `rpc.statd`, seguramente para comprobar si se encontraba activo. Veamos con más detalle estas alarmas y nos fijaremos en las IPs atacadas y en el orden temporal de los ataques:

```
SOURCE
NUM     SIG_ID  SIG_NAME
-----------------------
14      1       RPC portmap request rstatd

2001-12-08 08:39:44 61.182.50.241 ->iris.quiorg.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:84 ID:21911 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:10116
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:39:47 61.182.50.241 ->slabii.informat.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:84 ID:22917 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:8114
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:39:48 61.182.50.241 ->sdisco.informat.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:84 ID:22942 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:8088
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:39:48 61.182.50.241 ->bugs.informat.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:84 ID:22950 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:8065
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:39:49 61.182.50.241 ->slopez.informat.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:84 ID:22965 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:8049
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:39:49 61.182.50.241 ->bunny.informat.uv.es
```

```
[IP] Ver:4 HLen:5 Tos:0 Len:84 ID:22968 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:8044
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:39:49 61.182.50.241 ->crac.informat.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:84 ID:22971 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:7833
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:41:12 61.182.50.241 ->qfgate.quifis.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:84 ID:54632 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:14765
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:41:12 61.182.50.241 ->qfgate.quifis.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:84 ID:54632 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:14765
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:41:22 61.182.50.241 ->neron.ci.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:84 ID:57704 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:9185
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:41:22 61.182.50.241 ->neron.ci.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:84 ID:57704 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:9185
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:42:00 61.182.50.241 ->hussey.red.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:84 ID:7566 Flags:0 Offset:0 TTL:44 Proto:17 Cksum:46393
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:42:01 61.182.50.241 ->macklin.red.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:84 ID:7578 Flags:0 Offset:0 TTL:44 Proto:17 Cksum:46148
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:42:23 61.182.50.241 ->efcanon.ecofin.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:84 ID:15554 Flags:0 Offset:0 TTL:44 Proto:17 Cksum:30839
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

7       5507    RPC EXPLOIT statdx

2001-12-08 08:39:45 61.182.50.241 ->iris.quiorg.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:1104 ID:21912 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:9095
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:39:48 61.182.50.241 ->bugs.informat.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:1104 ID:22962 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:7033
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:39:50 61.182.50.241 ->crac.informat.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:1104 ID:22973 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:6811
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:41:12 61.182.50.241 ->qfgate.quifis.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:1104 ID:54670 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:13707
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:41:12 61.182.50.241 ->qfgate.quifis.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:1104 ID:54670 Flags:0 Offset:0 TTL:45 Proto:17 Cksum:13707
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:42:00 61.182.50.241 ->hussey.red.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:1104 ID:7569 Flags:0 Offset:0 TTL:44 Proto:17 Cksum:45370
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

2001-12-08 08:42:01 61.182.50.241 ->macklin.red.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:1104 ID:7581 Flags:0 Offset:0 TTL:44 Proto:17 Cksum:45125
[UDP] Sport:0 Dport:0 Len:0 Cksum:0

1       44      MISC Large ICMP Packet
```

```
2001-12-08 08:39:47 61.182.50.241 ->flop.informat.uv.es
[IP] Ver:4 HLen:5 Tos:0 Len:1500 ID:22928 Flags:0 Offset:0 TTL:236 Proto:1 Cksum:23062
[ICMP] Type:0, Code:0 Cksum:34386 ID:39612 Seq:57072
```

Efectivamente se han producido ataques a máquinas que estaban ejecutando este servidor RPC. Vemos como antes de ejecutar el exploit, se ha comprobado que la IP objetivo estaba ejecutando el `rpc.statd`. También vemos que esto ha sido ejecutado de forma automática (mediante un script, por ejemplo), puesto que la temporización entre los ataques es a veces de 1 segundo y entre las consultas a veces menor.

Es posible que este ataque compruebe primero que la estación tiene servicios RPC corriendo, y de ser así pase entonces a buscar el `rpc.statd`. Esa puede ser la causa por la que el tiempo transcurrido entre consultas sea un poco desigual.

A modo de curiosidad vemos que esta IP pertenece a algún lugar de China (ver figura 4.9).

Figura 4.9: Recorrido del ataque realizado desde Pekin.

Es posible que el mismo programa que uso la IP `61.182.50.241` fuese usado también por `caetano.empresa.net` y por `65.107.106.51`, puesto que los patrones de alerta son los mismos:

- `caetano.empresa.net`

```
DESTINATION
NUM     SIG_ID  SIG_NAME
------------------------

caetano.empresa.net
SOURCE
NUM     SIG_ID  SIG_NAME
------------------------
13      1       RPC portmap request rstatd
6       5507    RPC EXPLOIT statdx
1       44      MISC Large ICMP Packet
```

```
DESTINATION
NUM     SIG_ID  SIG_NAME
------------------------
```

- 65.107.106.51

```
SOURCE
NUM     SIG_ID  SIG_NAME
------------------------
5       1       RPC portmap request rstatd
1       44      MISC Large ICMP Packet
1       5507    RPC EXPLOIT statdx

DESTINATION
NUM     SIG_ID  SIG_NAME
------------------------
```

6 5503 2 6 FTP EXPLOIT format string (arachnids 453)

Este ataque lo habíamos visto antes, pero lo derivamos de una petición de transferencia de zona a un DNS secundario de nuestra red. Buscando de forma más general en esta firma encontramos dos IPs, la de Rusia y otra nueva, la `134.208.23.118`:

```
FTP EXPLOIT format string ## 5503 ##

COUNT   TOT_SRC TOT_DST TIME_INI               TIME_FIN               SRC
-----------------------------------------------------------------------------------------
5       8       2       2001-12-08 08:11:48    2001-12-08 11:37:09    194.85.9.53
1       3       0       2002-01-04 11:22:35    2002-01-04 11:22:35    134.208.23.118

COUNT   TOT_SRC TOT_DST TIME_INI               TIME_FIN               DST
-----------------------------------------------------------------------------------------
1       2329    10      2001-12-08 08:11:48    2001-12-08 08:11:48    apotkpr.ffarma.uv.es
1       19      5255    2001-12-08 09:33:37    2001-12-08 09:33:37    aficio2.informat.uv.es
1       24221   6222    2001-12-08 09:34:25    2001-12-08 09:34:25    nexus.informat.uv.es
1       895     6382    2001-12-08 09:34:58    2001-12-08 09:34:58    teruca.informat.uv.es
1       4659    11059   2001-12-08 11:37:09    2001-12-08 11:37:09    capone.informat.uv.es
1       27      1038    2002-01-04 11:22:35    2002-01-04 11:22:35    luciano.informat.uv.es
```

Esta nueva IP aparece una sola vez dentro de esta firma, pero en total aparece 3 veces como origen de ataques. Miremos en que ataques está involucrada:

```
SOURCE
NUM     SIG_ID  SIG_NAME
------------------------
2       44      MISC Large ICMP Packet

2002-01-04 11:22:26 134.208.23.118 ->dmcm.farmac.uv.es
[IP] Ver:4 HLen:5 Tos:32 Len:1500 ID:55161 Flags:0 Offset:0 TTL:235 Proto:1 Cksum:46188
[ICMP] Type:0, Code:0 Cksum:34386 ID:39612 Seq:57072

2002-01-04 11:22:32 134.208.23.118 ->flop.informat.uv.es
[IP] Ver:4 HLen:5 Tos:32 Len:1500 ID:58015 Flags:0 Offset:0 TTL:235 Proto:1 Cksum:42055
[ICMP] Type:0, Code:0 Cksum:34386 ID:39612 Seq:57072

1       5503    FTP EXPLOIT format string

2002-01-04 11:22:35 134.208.23.118 ->luciano.informat.uv.es
[IP] Ver:4 HLen:5 Tos:32 Len:76 ID:58038 Flags:0 Offset:0 TTL:44 Proto:6 Cksum:10465
```

```
[TCP] SPort:2865, DPort:21 Seq:2147483647 Ack:975696626 Offset:8 Res:0 Flags:0x18 Win:32120 Ck-
sum:15047 Urp:0 Res:0


DESTINATION
NUM     SIG_ID  SIG_NAME
------------------------
```

Las dos primeras alertas vienen dadas por reglas que detectan paquetes sospechosos, como por ejemplo un paquete ICMP de mas de 800 bytes de carga útil. No está muy claro a qué se deben estos paquetes y deberían ser objeto de un estudio posterior, puesto que son paquetes `ICMP Echo reply` que van de `134.208.23.118` hacia la IP atacada. Vemos también que aparecen varios segundos antes del ataque exploit al servidor FTP.

De nuevo, utilizamos VisualRoute para ver de donde proviene el ataque y vemos que viene de Taiwan (figura 4.10).

Figura 4.10: Recorrido del ataque realizado desde Taiwan.

```
9       17659   5       4       spp_stream4: STEALTH ACTIVITY (XMAS scan) detectio
33      4687    4       5       spp_stream4: STEALTH ACTIVITY (FIN scan) detection
8       17658   4       3       spp_stream4: NMAP FINGERPRINT (stateful) detection
```

Para finalizar el caso práctico veremos algunos intentos de recogida de información, que aunque en sí no son ataques, sí que son pruebas que indican que alguien está curioseando en nuestra red no sabemos con qué intenciones.

Muchos escaneadores de puertos (*nmap* por excelencia - `www.insecure.org/nmap/`) utilizan técnicas muy variadas para intentar pasar desapercibidos ante firewalls o IDSs. Las técnicas más utilizadas son las siguientes:

- escaneo FIN, XMAS, NULL: estos escaneos activan distintos flags en un inicio de conexión para ver si el puerto en cuestión está abierto. FIN usa un segmento FIN como prueba, XMAS (como si de un árbol de Navidad se tratase) usa FIN, URG y PUSH. NULL no usa ninguno. Según el RFC 793, cuando estos segmentos inician una conexión deberían ser respondidos con RST si el puerto destino esta cerrado e ignorados si estaba abierto, por lo que ya tenemos una forma de escanear un puerto si utilizar el flag SYN.

- Fragmentar las cabeceras TCP: la idea es fragmentar el paquete IP que contiene el primer segmento TCP de forma que la cabecera TCP se encuentre en varios fragmentos. Esto hace que los filtros de paquetes, IDSs y firewalls no sean capaces de detectar estos escaneos si en su software no se ha implementado el reensamblado de paquetes.

Además, también es posible conocer el sistema operativo de una máquina remota viendo la forma en la que responden a peticiones TCP extrañas. Esta técnica se conoce como 'identificación vía huella dactilar TCP/IP' o *TCP/IP fingerprint.*

Vemos como Snort (y en particular su preprocesador de flujo TCP *spp_stream4*), es capaz de detectar estos intentos de recogida de información.

4.3.2. Ranking de ataques

El ranking que vemos a continuación corresponde a los ataques detectados desde el 25 de mayo de 2002 hasta el 20 de junio de 2002, cuando el IDS estaba monitorizando el enlace ATM:

```
COUNT   SIG_ID  NUM_SRC NUM_DST SIG_NAME (REF)
--------------------------------------------------------------------------------
1141929 26      58464   36069   SCAN Proxy attempt (url help.undernet.org/proxyscan/)
702180  6       35126   65535   SCAN Proxy attempt
481451  126     288     63      spp_stream4: STEALTH ACTIVITY (NULL scan) detection
416496  32      1645    842     WEB-IIS cmd.exe access
410590  15      3106    13071   ICMP Destination Unreachable
                                (Communication Administratively Prohibited)
278514  60      68      104     BAD TRAFFIC bad frag bits
213950  24      35046   32      INFO - Possible Squid Scan
157820  50      33      65024   ICMP superscan echo
121700  45      23      18      FTP wu-ftp file completion attempt [ (bugtraq 3581)
120938  10      36816   131     WEB-MISC count.cgi access (bugtraq 128)
72562   33      77      12      ICMP PING speedera
71871   28      12746   732     WEB-IIS scripts access
56318   19      7148    799     MISC Large ICMP Packet (arachnids 246)
28911   59      147     19      WEB-MISC whisker HEAD with large datagram (url
                                www.wiretrip.net/rfp/pages/whitepapers/whiskerids.html)
27508   21      109     3046    WEB-MISC 403 Forbidden
26264   77      160     15662   ICMP PING NMAP (arachnids 162)
20275   5       454     14      WEB-CGI php access (bugtraq 2250)
18958   37      610     355     ICMP Source Quench
15887   30      1325    65      WEB-IIS iissamples access
15351   56      11298   936     WEB-IIS ISAPI .ida attempt (arachnids 552)
12479   47      1570    56      NETBIOS SMB C access (arachnids 339)
10431   31      885     71      WEB-IIS view source via translate header (arachnids 305)
9584    107     26      709     WEB-IIS msdac access
8545    12      3789    45      WEB-FRONTPAGE _vti_rpc access (bugtraq 2144)
8059    29      1824    95      WEB-MISC http directory traversal (arachnids 297)
7863    7       3785    44      WEB-IIS _vti_inf access
7213    39      1491    756     WEB-IIS CodeRed v2 root.exe access
                                (url www.cert.org/advisories/CA-2001-19.html)
6163    124     10      10      MISC Tiny Fragments
5935    54      3344    66      WEB-CGI calendar access
```

5535	238	285	19	spp_stream4: STEALTH ACTIVITY (Vecna scan) detection
4938	22	86	420	SCAN nmap TCP (arachnids 28)
4813	149	31	28	DOS MSDTC attempt (bugtraq 4006)
4627	46	19	15	FTP wu-ftp file completion attempt { (bugtraq 3581)
3926	65	372	173	WEB-CGI scriptalias access (cve CVE-1999-0236)
3813	127	14	8	WEB-MISC whisker space splice attack (arachnids 296)
3545	42	260	20	WEB-MISC long basic authorization string (bugtraq 3230)
3060	8	1082	230	WEB-CGI formmail access (bugtraq 1187)
3010	120	29	18	ICMP Destination Unreachable (Communication with Destination Network is Administratively Prohibited)
2642	90	272	857	spp_stream4: STEALTH ACTIVITY (SYN FIN scan) detection
2323	17	955	64	WEB-FRONTPAGE /_vti_bin/ access
1933	41	189	18	WEB-FRONTPAGE author.exe access
1906	362	254	6	DDOS shaft synflood (arachnids 253)
1398	298	3	3	MISC Large UDP Packet (arachnids 247)
1375	44	353	5	WEB-IIS ASP contents view (bugtraq 1864)
1314	132	5	4	ICMP PING CyberKit 2.2 Windows (arachnids 154)
1275	293	256	8	spp_stream4: STEALTH ACTIVITY (Full XMAS scan) detection
1268	66	22	26	ICMP Destination Unreachable (Communication with Destination Host is Administratively Prohibited)
1112	43	106	48	WEB-IIS multiple decode attempt (cve CAN-2001-0333)
1009	192	17	35	WEB-MISC /etc/passwd
970	76	10	12	ICMP redirect host (arachnids 135)
955	40	503	6	WEB-IIS webdav file lock attempt (bugtraq 2736)
941	143	258	15	spp_stream4: STEALTH ACTIVITY (nmap XMAS scan) detection
913	227	266	28	spp_stream4: STEALTH ACTIVITY (FIN scan) detection
908	75	136	12	WEB-MISC backup access
881	72	46	38	X11 xopen (arachnids 395)
815	51	123	38	ICMP L3retriever Ping (arachnids 311)
803	38	346	10	WEB-IIS asp-dot attempt
792	104	185	12	WEB-MISC http directory traversal (arachnids 298)
786	62	290	157	WEB-ATTACKS id command attempt
766	48	246	149	WEB-MISC apache DOS attempt
756	363	246	10	spp_stream4: STEALTH ACTIVITY (SAPU scan) detection
741	69	129	736	DNS named version attempt (arachnids 278)
721	4	13	3	INFO FTP anonymous login attempt
609	105	6	3	NETBIOS nimda .eml (url www.datafellows.com/v-descs/nimda.shtml)
603	58	396	40	WEB-IIS fpcount access (bugtraq 2252)
566	81	63	68	X11 outgoing (arachnids 126)
563	2	37	27	INFO MSN chat access
544	241	12	9	MS-SQL xp_cmdshell - program execution
507	35	33	47	BAD TRAFFIC tcp port 0 traffic
491	228	13	358	RPC portmap listing (arachnids 429)
464	82	86	9	WEB-MISC intranet access
457	36	119	3	SMTP RCPT TO overflow (cve CAN-2001-0260)
436	64	47	120	TELNET login incorrect (arachnids 127)
375	74	30	216	WEB-FRONTPAGE shtml.exe access (cve CAN-2000-0413)
375	133	311	211	WEB-IIS ISAPI .ida access (arachnids 552)
346	414	1	1	DOS IGMP dos attack
340	67	62	3	WEB-MISC /....
324	53	44	29	WEB-IIS .... access (bugtraq 2218)
315	1	36	22	ICMP Destination Unreachable (Port Unreachable)
312	113	23	54	WEB-IIS Overflow-htr access
310	63	192	18	WEB-CGI finger access (arachnids 221)
281	100	21	10	WEB-FRONTPAGE fourdots request (bugtraq 989)
229	70	79	101	WEB-CGI redirect access (bugtraq 1179)
223	101	10	58	WEB-IIS Unicode2.pl script (File permission canonicalization)
213	83	37	13	WEB-IIS admin access
191	85	44	20	WEB-IIS ISAPI .idq access (arachnids 553)
190	148	6	2	DNS zone transfer (arachnids 212)
186	116	47	4	WEB-MISC ICQ Webfront HTTP DOS
176	135	14	89	NETBIOS Samba clientaccess (arachnids 341)
173	145	13	4	MISC source port 53 to <1024 (arachnids 07)
166	115	22	40	WEB-CGI perl.exe access (arachnids 219)

153	87	60	33	WEB-CGI wrap access (bugtraq 373)
153	91	1	2	WEB-MISC L3retriever HTTP Probe (arachnids 310)
143	49	63	22	WEB-FRONTPAGE shtml.dll access (arachnids 292)
141	121	41	20	WEB-IIS /scripts/samples/ access
138	384	2	2	WEB-MISC SGI InfoSearch fname access (bugtraq 1031)
135	73	93	10	WEB-IIS fpcount attempt (bugtraq 2252)
130	168	17	34	WEB-CGI webgais access (arachnids 472)
123	365	9	2	X11 MITcookie (arachnids 396)
110	140	16	46	WEB-CGI wayboard access (bugtraq 2370)
101	139	15	44	WEB-MISC Admin_files access
99	289	1	1	EXPLOIT LPRng overflow (bugtraq 1712)
98	163	13	13	WEB-CGI phf access (bugtraq 629)
97	147	16	34	WEB-CGI dcboard.cgi access
95	179	20	35	WEB-CGI rguest.exe access (cve CAN-1999-0467)
95	357	2	2	FTP site exec (bugtraq 2241)
92	156	18	34	WEB-CGI faxsurvey access (cve CVE-1999-0262)
91	173	14	11	WEB-IIS /msadc/samples/ access
90	155	15	11	WEB-COLDFUSION expeval access (bugtraq 550)
90	158	18	34	WEB-CGI htmlscript access (bugtraq 2001)
88	160	18	34	WEB-MISC /cgi-bin/jj attempt (bugtraq 2002)
85	364	3	3	NETBIOS nimda RICHED20.DLL (url www.datafellows.com/v-descs/nimda.shtml)
85	177	18	34	WEB-CGI aglimpse access (cve CVE-1999-0147)
84	112	19	34	WEB-CGI AnyForm2 access (bugtraq 719)
82	79	66	6	WEB-ATTACKS mail command attempt
82	181	18	34	WEB-CGI campas access (cve CVE-1999-0146)
78	52	18	12	WEB-IIS SAM Attempt
77	152	15	33	WEB-CGI mmstdod.cgi access
77	184	16	12	WEB-COLDFUSION snippets attempt (bugtraq 550)
77	99	3	13	TFTP GET Admin.dll (url www.cert.org/advisories/CA-2001-26.html)
76	233	15	33	WEB-MISC get32.exe access (bugtraq 1485)
72	61	36	3	WEB-IIS encoding access (arachnids 200)
69	207	10	12	WEB-MISC Phorum code access (arachnids 207)
66	151	20	10	WEB-MISC nc.exe attempt
61	350	1	1	RPC portmap request sadmind (arachnids 20)
57	130	16	50	BAD TRAFFIC ip reserved bit set
57	387	5	1	WEB-ATTACKS ps command attempt
57	117	21	23	WEB-IIS .cnf access
56	176	14	14	WEB-CGI wguest.exe access (cve CAN-1999-0467)
56	80	20	20	ATTACK RESPONSES id check returned root
55	11	6	4	ICMP Echo Reply
54	230	10	30	WEB-CGI pals-cgi access (cve CAN-2001-0216)
54	125	37	5	WEB-IIS global-asa access
54	229	10	30	WEB-CGI commerce.cgi access
51	232	9	29	WEB-CGI webspirs directory traversal attempt (bugtraq 2362)
51	169	11	13	WEB-CGI websendmail access (cve CVE-1999-0196)
50	231	8	28	WEB-CGI Amaya templates sendtemp.pl directory traversal attempt (bugtraq 2504)
50	3	10	10	ICMP PING
49	89	15	15	WEB-MISC .htaccess access
48	95	3	17	NETBIOS NT NULL session (bugtraq 1163)
47	170	18	15	WEB-CGI www-sql access
47	174	14	12	WEB-IIS jet vba access (bugtraq 286)
46	201	9	12	WEB-MISC shopping cart access access
45	200	9	13	WEB-MISC mall log order access
44	221	6	6	WEB-MISC VirusWall FtpSave access (bugtraq 2808)
43	164	14	14	WEB-IIS search97.vts access (bugtraq 162)
42	108	5	19	ICMP webtrends scanner (arachnids 307)
42	98	26	10	WEB-CGI bash access (cve CAN-1999-0509)
41	208	12	10	WEB-COLDFUSION exprcalc access (cve CVE-1999-0455)
40	366	4	5	EXPLOIT CDE dtspcd exploit attempt (cve CAN-2001-0803)
40	191	16	13	WEB-COLDFUSION getfile.cfm access (bugtraq 229)
40	114	12	14	WEB-CGI test-cgi access (cve CVE-1999-0070)
40	213	14	11	WEB-CGI win-c-sample.exe access (bugtraq 2078)
39	118	6	10	ICMP redirect net (arachnids 199)

39	157	14	16	WEB-MISC handler access (bugtraq 380)
39	190	16	13	WEB-COLDFUSION exampleapp access
39	223	7	11	WEB-MISC store.cgi access
39	209	12	16	WEB-MISC ultraboard access
38	134	8	4	WEB-MISC cat%20 access (cve CVE-1999-0039)
38	199	4	8	WEB-CGI lastlines.cgi access (bugtraq 3755)
38	311	8	7	WEB-MISC PageService access (bugtraq 1063)
38	106	3	3	NETBIOS nimda .nws (url www.datafellows.com/v-descs/nimda.shtml)
38	202	14	15	WEB-MISC order.log access
38	159	17	13	WEB-COLDFUSION startstop DOS access (bugtraq 247)
37	103	12	19	RPC portmap request mountd (arachnids 13)
37	122	29	5	WEB-CGI bnbform.cgi access (cve CVE-1999-0937)
37	154	15	11	WEB-COLDFUSION application.cfm access (bugtraq 1021)
37	94	33	3	SMTP chameleon overflow (bugtraq 2387)
36	161	12	14	WEB-CGI man.sh access
36	162	12	13	WEB-CGI nph-test-cgi access (arachnids 224)
35	166	11	13	WEB-CGI view-source access (cve CVE-1999-0174)
35	204	11	13	WEB-MISC apache source.asp file access (bugtraq 1457)
34	137	17	16	WEB-IIS msadc/msadcs.dll access (cve CVE-1999-1011)
34	165	11	14	WEB-CGI testcounter.pl access
33	182	16	12	WEB-COLDFUSION evaluate.cfm access (bugtraq 550)
33	167	11	14	WEB-MISC webdist.cgi access (bugtraq 374)
33	153	8	11	WEB-CGI webdriver access (arachnids 473)
33	188	15	11	WEB-COLDFUSION beaninfo access (bugtraq 550)
33	245	11	7	WEB-CGI uploader.exe access (cve CVE-1999-0177)
32	183	16	12	WEB-COLDFUSION fileexists.cfm access (bugtraq 550)
32	111	14	14	WEB-CGI webspirs access (bugtraq 2362)
32	416	1	1	DDOS mstream client to handler (cve CAN-2000-0138)
31	186	15	11	WEB-COLDFUSION parks access (bugtraq 550)
31	171	15	10	WEB-COLDFUSION administrator access
31	110	17	8	WEB-CGI ksh access (cve CAN-1999-0509)
30	138	12	12	WEB-FRONTPAGE rad fp30reg.dll access (arachnids 555)
30	172	14	11	WEB-IIS site server config access (bugtraq 256)
30	205	11	13	WEB-CGI whoisraw access (arachnids 466)
30	96	1	6	MS-SQL/SMB xp_cmdshell program execution
29	109	12	13	WEB-MISC htgrep access
29	210	14	11	WEB-IIS uploadn.asp access
29	226	11	13	WEB-CGI bb-hist.sh access (bugtraq 142)
29	211	10	13	WEB-CGI wais.p access
29	180	15	11	WEB-CGI args.bat access
28	102	17	1	WEB-FRONTPAGE posting
28	185	14	10	WEB-COLDFUSION cfmlsyntaxcheck.cfm access (bugtraq 550)
28	217	14	11	WEB-FRONTPAGE service.pwd (bugtraq 1205)
28	218	14	11	WEB-FRONTPAGE users.pwd access
28	187	14	10	WEB-COLDFUSION exampleapp application.cfm (bugtraq 1021)
28	332	8	8	WEB-IIS .asp access (bugtraq 149)
28	175	14	12	WEB-MISC queryhit.htm access
28	197	5	5	WEB-CGI agora.cgi access (bugtraq 3976)
27	216	14	11	WEB-FRONTPAGE authors.pwd access
27	189	15	11	WEB-COLDFUSION addcontent.cfm access
27	243	2	4	ATTACK RESPONSES http dir listing
26	178	7	7	WEB-MISC viewcode access
26	131	6	14	WEB-IIS File permission canonicalization
25	23	12	9	ICMP Time-To-Live Exceeded in Transit
25	136	19	5	WEB-MISC ws_ftp.ini access (cve CAN-1999-1078)
25	123	13	12	SCAN FIN (arachnids 27)
25	292	5	5	SCAN myscan (arachnids 439)
24	307	8	6	WEB-CGI visadmin.exe access (bugtraq 1808)
24	212	8	11	WEB-MISC Talentsoft Web+ exploit attempt (bugtraq 1725)
24	68	17	4	FTP passwd retreval attempt (arachnids 213)
23	214	12	9	WEB-CGI wwwboard passwd access (arachnids 463)
23	215	10	10	WEB-CGI files.pl access
23	395	1	23	DNS named iquery attempt (arachnids 277)
23	220	8	11	WEB-CGI cvsweb.cgi access (cve CVE-2000-0670)
23	224	9	12	WEB-MISC Poll-it access (cve CAN-2000-0590)

23	193	7	11	WEB-MISC webcart access
23	196	9	10	WEB-MISC admin.php access (bugtraq 3361)
22	198	5	8	WEB-CGI zml.cgi attempt (bugtraq 3759)
22	234	4	8	TELNET access (arachnids 08)
22	20	6	7	ICMP Destination Unreachable (Undefined Code!)
21	349	7	4	DDOS mstream client to handler (arachnids 111)
21	206	10	10	WEB-MISC Trend Micro OfficeScan access (bugtraq 1057)
21	146	16	4	WEB-ATTACKS rm command attempt
21	194	6	9	WEB-CGI tcsh access (cve CAN-1999-0509)
21	322	8	8	WEB-IIS perl access
20	347	8	6	WEB-FRONTPAGE dvwssr.dll access (bugtraq 1108)
20	141	8	3	WEB-MISC cd..
19	219	7	7	WEB-MISC sml3com access (bugtraq 2721)
19	195	5	8	WEB-CGI rksh access (cve CAN-1999-0509)
18	312	8	7	WEB-IIS srchadm access
18	251	5	9	WEB-ATTACKS /etc/shadow access
18	335	9	7	WEB-MISC Domino catalog.ns access
18	16	5	5	ICMP Destination Unreachable (Host Unreachable)
18	290	1	2	EXPLOIT redhat 7.0 lprd overflow
17	262	9	8	WEB-MISC wwwboard.pl access (bugtraq 649)
17	252	4	8	WEB-MISC Armada Style Master Index directory traversal
17	324	8	7	WEB-IIS MSProxy access
16	246	5	6	WEB-MISC Lotus Domino directory traversal (cve CVE-2001-0009)
16	55	10	8	WEB-IIS ISAPI .printer access (cve CAN-2001-0241)
16	281	9	7	WEB-MISC adminlogin access
16	225	6	6	WEB-MISC VirusWall FtpSaveCSP access
15	310	5	8	WEB-MISC Domino domcfg.nsf access
15	296	4	4	DDOS shaft client to handler (arachnids 254)
15	203	7	7	MISC PCCS mysql database admin tool (arachnids 300)
15	222	6	6	WEB-MISC VirusWall FtpSaveCVP access
15	256	4	4	WEB-CGI LWGate access
14	330	8	7	WEB-MISC .wwwacl access
14	142	7	7	spp_stream4: NMAP FINGERPRINT (stateful) detection
14	305	8	7	WEB-IIS CGImail.exe access (cve CAN-2000-0726)
14	309	7	5	WEB-MISC showcode access
13	284	5	2	WEB-CGI rsh access (cve CAN-1999-0509)
13	144	3	2	TELNET SGI telnetd format bug (arachnids 304)
13	321	8	7	WEB-FRONTPAGE form_results access
13	306	9	7	WEB-IIS newdsn.exe access (bugtraq 1818)
12	327	9	7	WEB-IIS getdrvs.exe access
12	376	5	3	WEB-COLDFUSION cfcache.map access (bugtraq 917)
12	297	8	4	WEB-ATTACKS netcat command attempt
12	254	8	6	WEB-CGI MachineInfo access
12	355	3	3	RPC portmap request ttdbserv (cve CVE-1999-0003)
11	326	8	7	WEB-CGI ppdscgi.exe access (bugtraq 491)
11	378	6	4	WEB-MISC AuthChangeUrl access
11	396	1	11	RPC EXPLOIT statdx (arachnids 442)
11	318	8	7	WEB-FRONTPAGE orders.txt access
11	304	8	6	WEB-FRONTPAGE administrators.pwd (bugtraq 1205)
10	358	2	2	BAD TRAFFIC udp port 0 traffic
10	314	8	6	WEB-FRONTPAGE registrations.txt access
10	283	8	6	WEB-CGI csh access (cve CAN-1999-0509)
10	236	3	3	SCAN nmap fingerprint attempt (arachnids 05)
10	316	8	6	WEB-FRONTPAGE register.txt access
10	128	5	4	WEB-MISC whisker tab splice attack (arachnids 415)
10	338	8	7	WEB-MISC Ecommerce import.txt access
10	18	1	1	ICMP PING *NIX
10	34	7	4	SCAN SYN FIN (arachnids 198)
10	244	4	1	WEB-ATTACKS /usr/bin/perl execution attempt
10	341	8	6	WEB-FRONTPAGE fpadmin.htm access
9	345	8	6	WEB-CGI snorkerz.cmd access
9	267	7	7	WEB-MISC guestbook.pl access (bugtraq 776)
9	268	5	5	WEB-CGI edit.pl access
9	398	2	2	WEB-MISC netscape dir index wp (bugtraq 1063)
9	336	6	8	WEB-MISC Domino names.nsf access
9	240	1	1	FTP EXPLOIT x86 linux overflow (bugtraq 113)

9	353	5	5	WEB-MISC musicat access
9	259	5	5	WEB-CGI perlshop.cgi access
9	276	5	5	WEB-CGI info2www access (bugtraq 1995)
9	261	5	5	WEB-CGI rwwwshell.pl access
9	277	5	6	WEB-CGI NPH-publish access
8	278	5	5	WEB-CGI classifieds.cgi access (bugtraq 2020)
8	279	5	5	WEB-CGI survey.cgi access (bugtraq 1817)
8	248	5	8	WEB-IIS File permission canonicalization
8	280	5	5	WEB-CGI environ.cgi access
8	377	5	5	WEB-MISC ICQ webserver DOS (cve CVE-1999-0474)
8	282	5	5	WEB-CGI maillist.pl access
8	269	5	5	WEB-CGI dumpenv.pl access
8	129	6	2	WEB-ATTACKS cc command attempt
8	273	5	6	WEB-CGI glimpse access (bugtraq 2026)
8	323	7	5	WEB-COLDFUSION cfappman access (bugtraq 550)
7	263	5	6	WEB-MISC cachemgr.cgi access
7	328	6	4	WEB-CGI w3-msql access (bugtraq 591)
7	266	5	5	WEB-MISC responder.cgi access
7	303	4	4	WEB-CGI AT-admin.cgi access
7	257	5	5	WEB-CGI day5datacopier.cgi access
7	242	2	6	MS-SQL sa login failed
7	84	3	3	FTP CWD ...
6	361	3	2	WEB-MISC ROXEN directory list attempt (bugtraq 1510)
6	9	1	1	ICMP PING BSDtype (arachnids 152)
6	379	4	2	WEB-IIS *.idc attempt (bugtraq 1448)
6	300	2	2	FTP serv-u directory transversal (bugtraq 2025)
6	253	4	4	WEB-CGI pfdisplay.cgi access (bugtraq 64)
6	285	5	3	WEB-CGI zsh access (cve CAN-1999-0509)
6	270	5	5	WEB-MISC Phorum admin access (bugtraq 2271)
6	255	4	4	WEB-CGI flexform access
6	352	2	2	WEB-MISC Phorecast remote code execution attempt (bugtraq 3388)
6	258	5	5	WEB-MISC plusmail access
6	274	4	4	WEB-CGI filemail access
6	291	6	3	WEB-MISC webdav search access (arachnids 474)
6	372	3	4	WEB-MISC .htpasswd access
5	71	4	3	BAD TRAFFIC data in TCP SYN packet (url www.cert.org/incident_notes/IN-99-07.html)
5	265	5	5	WEB-MISC axs.cgi access
5	57	3	4	WEB-IIS ISAPI .idq attempt (arachnids 553)
5	313	3	3	WEB-FRONTPAGE service.stp access
5	27	2	5	SHELLCODE x86 NOOP
5	348	4	4	WEB-MISC piranha passwd.php3 access (bugtraq 1149)
5	380	1	1	TFTP Write (cve CVE-1999-0183)
5	317	3	3	WEB-FRONTPAGE register.htm access
5	333	4	4	WEB-MISC Domino domlog.nsf access
5	286	4	4	WEB-MISC mlog.phtml access (bugtraq 713)
5	334	4	4	WEB-MISC Domino log.nsf access
5	320	3	3	WEB-FRONTPAGE form_results.htm access
5	371	3	4	WEB-MISC netscape admin passwd (bugtraq 1579)
5	260	4	4	WEB-CGI upload.pl access
4	359	2	2	FINGER root query (arachnids 376)
4	247	3	4	WEB-IIS File permission canonicalization
4	360	1	1	DNS EXPLOIT named
4	264	4	4	WEB-MISC ax-admin.cgi access
4	88	4	2	WEB-ATTACKS /bin/ps command attempt
4	14	1	1	INFO Possible IRC Access
4	351	1	2	ICMP traceroute ipopts (arachnids 238)
4	271	4	4	WEB-MISC Sojourn access (bugtraq 1052)
4	368	3	4	WEB-MISC SmartWin CyberOffice Shopping Cart access (bugtraq 1734)
4	272	4	4	WEB-MISC dfire.cgi access
4	370	3	3	WEB-CGI day5datanotifier.cgi access
4	386	3	3	WEB-MISC net attempt
3	342	3	3	WEB-FRONTPAGE contents.htm access
3	390	1	1	WEB-MISC /~root

```
3       150     1       1       MISC Invalid PCAnywhere Login
3       393     2       2       FINGER 0 query (arachnids 378)
3       329     3       3       WEB-MISC bigconf.cgi access
3       235     1       1       WEB-FRONTPAGE rad overflow attempt (arachnids 555)
3       315     3       3       WEB-FRONTPAGE registrations.htm access
3       397     1       1       FINGER null request (arachnids 377)
3       367     3       3       WEB-IIS .bat? access (bugtraq 2023)
3       383     2       2       WEB-MISC ftp.pl access (bugtraq 1471)
3       319     3       3       WEB-FRONTPAGE orders.htm access
3       337     3       3       WEB-MISC Ecommerce import.txt access
3       369     2       3       WEB-CGI tstisapi.dll access
3       354     2       2       WEB-MISC ftp attempt
3       356     1       3       FTP EXPLOIT wu-ftpd 2.6.0 site exec format string overflow
                                Linux (bugtraq 1387)
3       308     3       3       WEB-COLDFUSION mainframeset access (bugtraq 550)
3       325     3       3       WEB-CGI archie access
2       406     1       1       WEB-CGI view-source directory traversal (cve CVE-1999-0174)
2       294     2       2       FTP EXPLOIT stat overflow (url
                                labs.defcom.com/adv/2001/def-2001-31.txt)
2       86      1       2       EXPLOIT ssh CRC32 overflow (bugtraq 2347)
2       343     2       2       WEB-CGI anform2 access (cve CVE-1999-0066)
2       375     1       1       SCAN NULL (arachnids 4)
2       295     2       2       WEB-IIS _mem_bin access
2       344     2       2       WEB-CGI w2tvars.pm access
2       392     1       2       WEB-MISC weblogic view source attempt (bugtraq 2527)
2       25      1       1       SHELLCODE x86 NOOP (arachnids 181)
2       346     2       2       WEB-MISC rpm_query access (cve CVE-2000-0192)
2       299     2       1       SMTP exchange mime DOS
2       331     2       2       WEB-IIS srch.htm access
2       237     1       1       WEB-FRONTPAGE rad overflow attempt (cve CAN-2001-0341)
2       13      1       1       ICMP Fragment Reassembly Time Exceeded
2       301     2       2       WEB-IIS scripts-browse access
2       399     1       1       WEB-FRONTPAGE cfgwiz.exe access
2       239     2       2       FTP EXPLOIT aix overflow (bugtraq 679)
2       287     1       1       WEB-MISC mylog.phtml access (bugtraq 713)
2       400     1       1       WEB-FRONTPAGE fpremadm.exe access
2       288     2       2       SCAN XMAS (arachnids 144)
2       401     1       1       WEB-FRONTPAGE fpsrvadm.exe access
2       339     2       2       WEB-MISC Ecommerce checks.txt access
2       275     2       2       WEB-MISC filemail access
2       340     2       2       WEB-MISC Ecommerce check.txt access
2       388     1       1       WEB-CGI wwwadmin.pl access
2       373     2       2       WEB-MISC technote main.cgi file directory traversal attempt
                                (cve CVE-2001-0075)
2       389     1       1       WEB-CGI sendform.cgi access
1       374     1       1       MISC ramen worm outgoing (arachnids 461)
1       119     1       1       SCAN NMAP XMAS (arachnids 30)
1       391     1       1       WEB-CGI post-query access
1       407     1       1       WEB-MISC eXtropia webstore directory traversal (bugtraq 1774)
1       408     1       1       WEB-CGI webplus directory trasversal (arachnids 471)
1       409     1       1       WEB-MISC webplus access (cve CVE-2000-1005)
1       249     1       1       SCAN cybercop os probe (arachnids 146)
1       394     1       1       FINGER redirection attempt (arachnids 251)
1       410     1       1       WEB-MISC windmail access (cve CAN-2000-0242)
1       250     1       1       WEB-MISC prefix-get //
1       411     1       1       WEB-MISC ministats admin access
1       412     1       1       WEB-CGI webspeed access (arachnids 467)
1       92      1       1       WEB-MISC .nsconfig access
1       381     1       1       WEB-MISC sadmind worm access (url
                                www.cert.org/advisories/CA-2001-11.html)
1       413     1       1       WEB-MISC search.vts access
1       93      1       1       DDOS - TFN client command LE (arachnids 183)
1       382     1       1       WEB-ATTACKS mail command attempt
1       302     1       1       WEB-IIS perl-browse20 attempt
1       78      1       1       FTP CWD ~root (cve CVE-1999-0082)
1       415     1       1       ICMP PING WhatsupGold Windows (arachnids 168)
```

```
1       97      1       1       MS-SQL/SMB sa login failed
1       385     1       1       DDOS TFN Probe (arachnids 443)
1       402     1       1       WEB-CGI admin.pl access
1       403     1       1       WEB-MISC BigBrother access
1       404     1       1       WEB-MISC moreover shopping cart directory traversal
                                (bugtraq 1762)
1       405     1       1       WEB-MISC architext_query.pl access
_________
Total: 4560476
```

De entre el total de 4,560,476 alarmas, la mayoría fueron generadas por reglas que no son consideradas como ataques directos, como por ejemplo pings, peticiones a servicios no muy frecuentes (finger), actividad de NetBIOS, actividad de software p2p, etc.

Del resto de alertas haremos dos distinciones:

- Alertas de tráfico sospechoso: son indicios de que algo está ocurriendo, ya sea porque el tráfico sea anormal o porque se acceden a servicios o ficheros que se consideran delicados.
  - accesos http a scripts utilizados para la puesta a punto de servidores web (whoisraw, test-cgi, win-c-sample.exe).
  - accesos externos a servicios delicados (mountd, nfsd).
  - respuestas a ataques (identificador de usuario root como consecuencia del comando 'id', listado de directorios por http.
  - tráfico TCP/IP corrupto (puerto TCP o UDP 0, fragmentos muy pequeños, ICMP anómalos).
  - fallos de login en telnet o FTP.
- Alertas de ataques: ataques cláramente recogidos en las reglas de Snort, pero de naturaleza diversa.
  - ataques no intencionados por medio de gusanos.
  - recogida de información por escaneos.
  - ataques directos (ataques web, DoS y DDoS y Exploits).

Veamos en aproximadamente un mes cual ha sido el balance en cuanto a esta calsificación:

- Alertas de tráfico sospechoso: 89.96 % (4,560,476 en total).
- Alertas de ataques:10.04 % (458,177 en total).

El reparto de las alertas de ataques lo podemos ver en la tabla 4.5.
Por último, podemos ver el reparto de los ataques directos en la tabla 4.6.

Los ataques más peligrosos pueden ser quizá los exploits y en nuestro caso, afortunadamente son también los menos frecuentes. Pero vemos que en aproximadamente un mes se han producido 190 alertas, lo que da unos 6 ataques considerados peligrosos diarios.

Esta información puede ser utilizada por los administradores para evaluar el riesgo que tiene actualmente la Universidad, y elaborar así una política de seguridad completa, que podría ser más restrictiva cuanto mayor fuese el riesgo.

Tipo de ataque	Porcentaje	Total
Gusanos	92.62 %	424,403
Escaneos	6.85 %	31,402
Ataques web, DoS y DDoS, Exploits	0.53 %	2,372

Cuadro 4.5: Reparto de las alertas de ataques.

Tipo de ataque directo	Porcentaje	Total
Ataques web (ejecución de scripts con cadenas ofensivas)	43.54 %	1,033
DoS y DDoS (DoS en Apache e IGMP, DDoS mstream, shaft y TFN)	46.44 %	1,149
Exploit (LPRng, CDE, DNS, SSH, AIX, statdx)	8.02 %	190

Cuadro 4.6: Reparto de las alertas de ataques directos.

4.4. Trabajo futuro

Muchas han sido las cosas que se podrían haber hecho con el sistema, pero el proyecto tiene un límite. Quizá, como trabajo futuro para próximos proyecto, sería interesante lo siguiente:

- Desarrollo de una interfaz gráfica al sistema en general, que permita:
  - realizar consultas a la base de datos por medio de los scripts de consulta desarrollados.
  - la manipulación de la base de datos para eliminar falsas alarmas y reducir así su tamaño.
  - configurar Snort desde el propio interfaz
- Estudio de la variación en la carga de Snort en función del tráfico entrante y del conjunto de firmas: nos puede ayudar a conocer con más exactitud el comportamiento dinámico que tiene Snort, y adecuar su configuración a las necesidades de la organización.
- Programa inteligente que active o desactive reglas en función de la carga: se puede hacer una clasificación del conjunto de reglas y dependiendo de la carga y del descarte de paquetes que se está produciendo en Snort, cargar las más prioritarios o cargarlas todas.

Capítulo 5

Conclusiones

Como resultado del proyecto hemos obtenido las siguientes conclusiones:

1. Se ha implantado exitosamente un sistema de detección de intrusos en la Universidad de Valencia, lo cual permite a los administradores conocer cuando está siendo atacada la red y aporta información valiosa para determitar la naturaleza de los ataques.

2. Se han desarrollado aplicaciones para la consulta de las alertas generadas por el IDS y para su aviso al CERT y a los responsables de las redes origen de los ataques. También se ha desarrollado una herramienta para medir el rendimiento de la máquina que permite a los administradores estimar la potencia de la máquina necesaria para desarrollar la función de IDS dependiendo de la cantidad de tráfico a monitorizar.

3. Se han redactado varias políticas de seguridad, de las cuales los administradores de la Universidad de Valencia podrán elegir la que crean más apropiada dependiendo del riesgo que exista.

4. Se han hecho pruebas de localización del IDS colocándolo en varios puntos de la red, para finalmente dejarlo instalado en el punto de acceso a Internet monitorizando todo el tráfico intercambiado entre la Universidad de Valencia y el exterior.

5. Se han realizado pruebas de rendimiento que muestran que el tráfico actual de la Universidad de Valencia está en el límite del la capacidad de proceso de una CPU actual, y puesto que el tráfico de la Universidad crece continuamente, en un futuro no muy lejano será necesaria la utilización de repartidores de carga para IDSs o la continua actualización del hardware para que el IDS no descarte demasiados paquetes y no nos proporcione una falsa sensación de seguridad.

6. La Universidad recibe una media de seis ataques tipo exploit (considerados como los más peligrosos) cada día. Muchos de estos ataques se podrían haber evitado si la Universidad dispusiera de un cortafuegos que filtrara el tráfico sospechoso.

7. El IDS necesita de monitorización continua por parte de los responsables de seguridad para avisar a los administradores de las máquinas atacadas de la posibilidad de una instrusión en éstas, ya que no es capaz de determinar por sí mismo si el ataque ha tenido éxito o no.

8. Snort es un software IDS de gran aceptación, potente y gratuito, multiplataforma y de código abierto. Su mayor inconveniente es que se encuentra en un desarrollo todavía temprano y muchas de las características avanzadas de que disponen los IDSs comerciales no se encuentran todavía implementadas en Snort.

9. Es posible ocultar ataques al IDS enviando el tráfico de forma especial. Este problema no aparece solo en Snort, sino que está presente en muchos de los IDSs comerciales y es tema de investigación en la actualidad.

Apéndice A

Diagramas UML del sistema

Diagrama de clases del sistema

Diagrama de secuencia de IDSAlerts

Diagrama de secuencia de IDSBenchmark

Presupuesto

Apéndice B

Ficheros de configuración

Snort

snort.conf

```
    var HOME_NET 147.156.0.0/16
    var EXTERNAL_NET any
    var SMTP $HOME_NET
    var HTTP_SERVERS $HOME_NET
    var SQL_SERVERS $HOME_NET
    var DNS_SERVERS $HOME_NET
    var RULE_PATH ./
    var SHELLCODE_PORTS !80
    var HTTP_PORTS 80
    var ORACLE_PORTS 1521

    preprocessor frag2
    preprocessor stream4: detect_scans, disable_evasion_alerts
    preprocessor stream4_reassemble: both
    preprocessor http_decode: 80 -cginull
    preprocessor rpc_decode: 111 32771
    preprocessor bo
    preprocessor telnet_decode
    #preprocessor portscan-ignorehosts: 0.0.0.0
    #preprocessor spade-adapt: 20 2 0.5
    #preprocessor arpspoof
    #preprocessor arpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00
    output database: log, mysql, user=snort password=******* dbname=snort_rend 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
    #output alert_unified: filename snort.alert, limit 128
    #output log_unified: filename snort.log, limit 128
    #output trap_snmp: alert, 7, inform -v 2c -p 162 myTrapListener myCommunity
    #output trap_snmp: alert, 7, trap -v 3 -p 162 -u snortUser -l authPriv -a SHA -A SnortAu-
thPassword -x DES -X SnortPrivPassword myTrapListener
    #output trap_snmp: alert, 7, inform -v 3 -p 162 -u snortUser -l authPriv -a SHA -A SnortAu-
thPassword -x DES -X SnortPrivPassword myTrapListener

    include classification.config
    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/smtp.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-attacks.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/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/local.rules
```

barnyard.conf

```
#config localtime
config hostname: snorthost
config interface: atm0
config filter: not port 22
processor dp_alert
processor dp_log
processor dp_stream_stat
#output alert_fast
#output log_dump
#output alert_syslog
#output alert_acid_db: mysql, sensor_id 1, database snort_rend, server localhost, user snort,
password Hola1234, detail full
output log_acid_db: mysql, sensor_id 1, database snort_rend, server localhost, user snort,
password Hola1234, detail full
```

runsnort.sh

```
#!/bin/bash
INT=atm0
FLAGS=" -b -k none"
LOGDIR=/home/snort/logatm/
LOGFILE=snortatm.log
CONFIG=/root/snort/snort.conf
SNORT=/usr/bin/snort
$SNORT -i $INT $FLAGS -l $LOGDIR -L $LOGFILE -c $CONFIG -D
echo "$(date): Snort lanzado" >>/var/log/runsnort;
while true; do
ps -C snort >/dev/null;
if [ $? -eq 1 ]; then
echo "$(date): Snort ha muerto, lanzandolo de nuevo" >>/var/log/runsnort;
```

```
    $SNORT -i $INT $FLAGS -l $LOGDIR -L $LOGFILE -c $CONFIG -D;
    fi
    sleep 10;
    done
```

Netfilter

```
    iptables -t filter -A INPUT -j LOG -p tcp --syn -i eth0
    iptables -t filter -A INPUT -j REJECT --reject-with tcp-reset -p tcp --syn -i eth0 --dport
! ssh
    iptables -t filter -A INPUT -j DROP -i atm0 -d discreto.uv.es
```

Cron

SendAlert

```
    #!/bin/bash
    export MYSQL_UNIX_PORT=/tmp/mysql.sock
    /home/emilio/proyecto/scripts/SendAlert.pl
```

RecvAlert

```
    #!/bin/bash
    export MYSQL_UNIX_PORT=/tmp/mysql.sock
    /home/emilio/proyecto/scripts/RecvAlert.pl
```

Logrotate

```
    /home/snort/logrend/snortrend.log {
    notifempty
    daily
    rotate 14
    missingok
    compress
    postrotate
    endscript
    }
    /home/snort/logrend/portscan.log {
    notifempty
    daily
    rotate 14
    missingok
    compress
    postrotate
    endscript
    }
    /home/snort/logrend/alert {
    notifempty
    daily
    rotate 14
    missingok
    compress
    postrotate
    kill -HUP $(ps -o pid -C snort --no-headers)
```

```
    endscript
    }
```

InitSystem.sh

```
    #!/bin/bash
    ################################## Netfilter ###################################
    echo "Configurando Firewall interno"
    iptables -t filter -A INPUT -j LOG -p tcp --syn -i eth0
    iptables -t filter -A INPUT -j REJECT --reject-with tcp-reset -p tcp --syn -i eth0 --dport
! ssh
    iptables -t filter -A INPUT -j DROP -i atm0 -d discreto.uv.es
    ################################ Interfaz ATM ##################################
    echo "Levantando interfaz ATM"
    atmarpd -b
    atmarp -c atm0
    ifconfig atm0 192.168.1.1 up
    atmarp -s 192.168.1.2 0.225 null
    atmarp -s 192.168.1.3 0.325 null
    atmarp -s 192.168.1.4 0.226 null
    atmarp -s 192.168.1.5 0.326 null
    sleep 1s
    ################################## MySQL #######################################
    echo "Lanzando MySQL"
    export MYSQL_UNIX_PORT=/tmp/mysql.sock
    safe_mysqld --datadir=/home/mysql &
    sleep 1s
    ################################## Snort #######################################
    echo "Lanzando Snort"
    /usr/local/bin/snort -i atm0 -l /home/snort/logatm/ -L snortatm.log -c /root/snort/snort.conf
-D
```

Bibliografía

[1] **Anónimo:** “Linux. Máxima seguridad”. Prentice Hall. 2000.

[2] **Bace, R., Mell, P.:** “Intrusion Detection Systems”. NIST Special Publication on Intrusion Detection System. (`http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf`).

[3] **Brandenburg Department of Science: “**The Intrusion Detection System AID”, `http://www-rnks.informatik.tu-cottbus.de/~sobirey/aid.e.html`

[4] **Chapman, D.B., Zwicky, E.D.:** “Building Internet Firewalls, 2nd Ed.”. O’Reilly & Associates, Inc. 1999.

[5] **Cisco Systems, Inc.:** “Cisco - Cisco Intrusion Detection”, `http://www.cisco.com/warp/public/cc/pd/sqsw/sqidsz/`, 2002.

[6] **Cisco Systems, Inc.:** “Configuring SPAN and RSPAN”**,** `http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/sw_6_3/confg_gd/span.htm`**,** 2002.

[7] **Enterasys Networks, Inc.: “**Intrusion Detection Solutions”, `http://www.enterasys.com/ids/`, 2002.

[8] **Garfinkel, S., Spafford, G.:** “Practical UNIX & Internet security, 2nd Ed.”. O’Reilly & Associates, Inc. 1997.

[9] **Garfinkel, S., Spafford, G.:** “Web security and commerce”. O’Reilly & Associates, Inc. 1997.

[10] **Guttman, B., Bagwill, R.:** “Internet Security Policy: A technical guide”, NIST Specail Publication, Julio de 1999.

[11] **IETF:** “IDWG - Intrusion Detection Working Group”, `http://www.ietf.org/html.charters/idwg-charter.html`

[12] **Internet Security Systems, Inc.:** “RealSecure® Network Protection”, `http://www.iss.net/products_services/enterprise_protection/rsnetwork/index.php`, 2002.

[13] **Kaufman, C., Perlman, R., Speciner, M.:** “Network security. Private communication in a public world”. Prentice Hall. 1995.

[14] **Kenneth R. van Wyk, Richard F.:** “Incident response”. O’Reilly. Agosto 2001.

[15] **Marconi:** “FORE PCA-200E”, http://www.marconi.com/html/solutions/forerunner200enics/technicalspecifications.html

[16] **Mediavilla, M.:** “Seguridad en UNIX”. RA-MA. 1998.

[17] **Mourani, G.:** “Securing and Optimizing Linux”.www.tldp.org/LDP/solrhe/Securing-Optimizing-Linux-RH-Edition-v1.3/. 2000.

[18] **Net Optics, Inc.:** “Fiber Tap singlemode and multimode Gigabit”, http://www.netoptics.com/11.html, 2002.

[19] **Northcutt, S., Novak, J.:** “Detección de intrusos, 2ª Ed.”. Prentice Hall. 2001.

[20] **Ptacek, T.H., Newsham, T.N.:** “Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection”. Secure Networks, Inc. 1998. (http://secinf.net/info/ids/idspaper/idspaper.html).

[21] **Purdue University:** “AAFID - Autonomous Agents for Intrusion Detection”, http://www.cerias.purdue.edu/homes/aafid

[22] **Roesch, M.:** “Snort - The Open Source Intrusion Detection System”, http://www.snort.org

[23] **Sobiery, M.:** “Michael Sobirey’s Intrusion Detection Systems page”, http://www-rnks.informatik.tu-cottbus.de/~sobirey/ids.html

[24] **Sourceforge:** “ATM on Linux”, http://linux-atm.sourceforge.net/. 2002.

[25] **Timm, K.:** “IDS Evasion Techniques and Tactics”.http://online.securityfocus.com/infocus/1577. 2002.

[26] **University of California, Davis - Computer Security Laboratory:** “Graph-based Intrusion Detection System”, http://seclab.cs.ucdavis.edu/arpa/grids/welcome.html

[27] **USC Information Sciences Intitute:** “Common Intrusion Detection Framework**”,** http://www.isi.edu/gost/cidf/, Septiembre 1999