Universidad Autónoma de

Baja California

Facultad de Ingeniería

Campus Ensenada

Diseño e implementación de herramientas para el análisis de vulnerabilidades de sistemas operativos de red

Tesis

Que para obtener el titulo de

Ingeniero en Computación

Presenta:

Christian Salomón Torres Morales

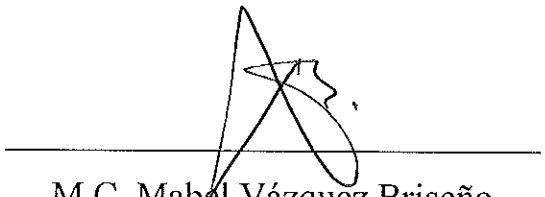
Ensenada, B.C. México, Diciembre del 2005

Declaratoria:

Por medio de la presente me responsabilizo de la autenticidad y originalidad del presente trabajo titulado:

Diseño e implementación de herramientas para el análisis de vulnerabilidades de sistemas operativos de red.

Asesor y director de Tesis



M.C. Mabel Vázquez Briseño

Ensenada B.C. México, Octubre del 2005

Miembros del Jurado del Examen Profesional que para obtener el titulo de Ingeniero en Computación presenta el C. Christian Salomón Torres Morales, con el trabajo titulado

Diseño e implementación de herramientas para el análisis de vulnerabilidades de sistemas operativos de red.

M.C. Sergio Omar Infante Prieto

M.C. Elitania Jiménez García

Ensenada B.C. México, Octubre del 2005

Agradecimientos

Deseo agradecer primeramente a Dios, a mis padres por el apoyo recibido, mis familiares por haber creído en mí, mis maestros por haber solucionado dudas y haberme exigido y motivado, mis amigos por haberme aguantado y haber estado en las buenas y las malas durante el transcurso de la carrera.

También deseo agradecer a los autores de los libros, artículos, etc. Por haber liberado la información contenida en estos, la cual me ayudo al entendimiento de algunas de mis dudas, y un agradecimiento a las personas que olvide mencionar.

Resumen

Este trabajo tiene como propósito concientizar a los administradores de redes de la importancia de mantener el sistema libre de servicios innecesarios, es decir, si un servicio no se usa, no debe estar ejecutándose, ya que cada puerto es un potencial agujero de seguridad y puede comprometer el sistema.

La herramienta desarrollada se encarga obtener toda la información posible de una máquina, es decir, conocer todos los puertos abiertos de la misma, tratar de obtener el servicio que está corriendo, y llegar a conocer la versión del mismo que está utilizando. Así el administrador sabrá exactamente, que información obtiene un atacante cuando hace un análisis de su sistema, y podrá tomar medidas en el asunto, ya sea cancelar el servicio, disfrazar la cadena identificativa del mismo, etc.

Esta herramienta, es básicamente un servidor que se encarga de visualizar los resultados obtenidos del análisis hecho por los clientes de la herramienta, estos hacen el análisis (si se requiere) de las máquinas remotas.

Cabe señalar que la herramienta esta diseñada para trabajar con un servidor corriendo Linux, las máquinas remotas pueden ser Windows o Linux, e idealmente estarán conectadas por medio de un enlace *Ethernet*, en una red de área local (LAN).

Contenido

I INTRODUCCIÓN
I.1 – Objetivo General 2
I.2 - Equipo Utilizado 2
I.3 – Organización de la tesis 3

II PROTOCOLOS DE COMUNICACIÓN
II.1 – Modelo OSI 4
II.2 - Capas del modelo ISO/OSI 4
II.3 - Protocolos de red 6
II.4 - TCP/IP 7
II.5 - Protocolos usados en el desarrollo de la herramienta 9
II.5.1 – TCP 9
II.5.2 - UDP 13
II.5.3 - ICMP 14

III SEGURIDAD EN LOS SISTEMAS DE CÓMPUTO
III.1 – Amenazas a la seguridad 16
III.2 – Los agujeros (o vulnerabilidades) 20

IV VULNERABILIDADES Y ATAQUE A LA SEGURIDAD
IV.1 - La escala de vulnerabilidades 22
IV.2 - Vulnerabilidades genéricas 24

V VULNERABILIDADES EN SISTEMAS OPERATIVOS DE RED
V.1 - Características de los sistemas operativos de red 29
V.2 - Las Vulnerabilidades del Instituto SANS 30
V.2.1 – Vulnerabilidades en sistemas UNIX-Linux 31
V.2.2 – Vulnerabilidades en Windows 42

VI DESARROLLO DE LA HERRAMIENTA PARA EL ANÁLISIS DE VULNERABILIDADES
VI.1 - 1ª etapa (Escáner de puertos TCP y UDP) 60
VI.1.1 - Escáner TCP 61
VI.1.2 - Escáner UDP 64
VI.2 - 2ª Etapa (Banners y Fingerprinting de Servicios de Red) 68
VI.2.1 – Banner (cadena identificativa) 68
VI.2.2 - Fingerprinting de servicios de red 71
VI.3 - 3ª Etapa (Directivas básicas de seguridad en el servidor) 75
VI.3.1 – Directivas locales 75

VII RESULTADOS
VII.1 – Ejemplo de un análisis local 79
VII.2 – Ejemplo de un análisis remoto 82

VIII – CONCLUSIONES 87

IX – REFERENCIAS 90

APÉNDICE A – MÁS DIRECTIVAS DE SEGURIDAD
A.1 – Directivas básicas locales 91
A.2 – Directivas de los servicios de red 101
A.3 – Directivas de administración 114

Lista de Figuras

FIGURA 1. CAPAS DE LA RED EN EL MODELO ISO/OSI. 5
FIGURA 2. MODELO DE RED QUE MUESTRA LOS PROCESOS PARES. 7
FIGURA 3. MODELO ISO/OSI Y LA PILA DE PROTOCOLOS TCP/IP. 9
FIGURA 4. ESTRUCTURA SIMPLE DE DATOS DEL *SOCKET* EN UNIX. 11
FIGURA 5. USO DE LOS *SOCKETS* CON UN PROTOCOLO ORIENTADO A CONEXIÓN. 13
FIGURA 6. USO DE LOS *SOCKETS* CON UN PROTOCOLO SIN CONEXIÓN. 14
FIGURA 7. DIAGRAMA DE BLOQUES DEL ANALIZADOR 59
FIGURA 8. COMPORTAMIENTO DEL CLIENTE Y SERVIDOR TCP. 60
FIGURA 9. COMPORTAMIENTO DEL CLIENTE Y SERVIDOR TCP EN CASO DE ERROR. 61
FIGURA 10. DIAGRAMA GENERAL DEL ESCÁNER TCP. 62
FIGURA 11. COMPORTAMIENTO DEL CLIENTE Y SERVIDOR UDP. 63
FIGURA 12. COMPORTAMIENTO DEL CLIENTE Y SERVIDOR UDP EN CASO DE ERROR. 64
FIGURA 13. DIAGRAMA GENERAL DEL ESCÁNER UDP 67
FIGURA 14. CADENA IDENTIFICATIVA (O *BANNER*) 69
FIGURA 15. TÉCNICA DEL SALUDO DE TRES PASOS. 69
FIGURA 16. COMPORTAMIENTO DE UN SERVICIO QUE REGRESA UN *BANNER*. 70
FIGURA 17. DIAGRAMA DE BLOQUES DEL *FINGERPRINTING* DE SERVICIOS. 72
FIGURA 18. DIAGRAMA DE BLOQUES DEL ANALIZADOR DE SERVICIOS. 74
FIGURA 19. CONFIGURACIÓN DE UN ANÁLISIS LOCAL. 79
FIGURA 20. RESULTADOS DE UN ANÁLISIS LOCAL. 80
FIGURA 21. DIRECTIVAS LOCALES DE UN ANÁLISIS LOCAL. 81
FIGURA 22. PANTALLA DE DIRECTIVAS DE SEGURIDAD. 82
FIGURA 23. ALTAS A CLIENTES. 83
FIGURA 24. RASTREO DE CLIENTES. 84
FIGURA 25. RESULTADOS DE LOS CLIENTES. 85
FIGURA 26. CLIENTE LINUX. 86
FIGURA 27. CLIENTE WINDOWS. 86

Lista de Tablas

TABLA I. PROTOCOLOS TCP/IP MÁS COMUNES. 8
TABLA II. VALORES VÁLIDOS DEL CAMPO TIPO DE MENSAJE DE ICMP. 15
TABLA III. PRINCIPALES VULNERABILIDADES ASOCIADAS A TCP/IP. 25

I – INTRODUCCIÓN

La seguridad de los sistemas operativos de red cobra cada vez más importancia en el mundo actual, ya que como se sabe, en estos tiempos la información es poder. Por supuesto que esto es preocupante, ya que si una persona es capaz de burlar la seguridad de una empresa importante que cuenta con personal encargado de la seguridad de sus datos, cualquier empresa puede ser victima de un ataque o intrusión de este tipo, esto puede suceder incluso sin importar el sistema operativo con que cuente su empresa. Ya que en el mercado se encuentran varios sistemas operativos de red, cada sistema funciona de una manera muy diferente, con distintas funcionalidades y características, que los hace diferentes, y por consiguiente unos mejores que otros.

Algunas de las vulnerabilidades más comunes son de servicios o demonios viejos, (un demonio *-daemon-* es un proceso que se esta ejecutando en segundo plano, en este caso sería el proceso que se encarga de abrir y manejar las conexiones del servicio) los cuales el administrador se olvido de actualizar, ó pueden ser el resultado de una mala configuración del sistema o servicio. Esto nos lleva al factor más importante; según mi punto de vista, el cual es el factor humano, esto engloba y enmarca a la vez la responsabilidad tanto del administrador del sistema así como los usuarios del mismo, ya que es importante una actualización por parte del administrador, de un servicio que tiene vulnerabilidades, como la responsabilidad de los usuario de poner una contraseña fuerte a su cuenta.

I.1 – Objetivo General

Realizar herramientas computacionales que permitan evaluar de manera práctica las vulnerabilidades en cuanto a seguridad que presentan los sistemas operativos de red más utilizados hoy en día (Windows y Linux). Estas herramientas tendrán como finalidad apoyar a administradores de red y usuarios en general para conocer y de ser posible reparar los puntos inseguros de su conexión de red.

I.2 - Equipo Utilizado

- 2 computadoras.
- Interfaz de red (en cada una).
- Mandrake Linux 9.1.
- Windows 2000 Advanced Server.
- Bibliotecas para sockets (para Linux).
- Bibliotecas para desarrollo GTK (para entorno visual en Linux).
- Bibliotecas para emulación POSIX (para Windows).

I.3 – Organización de la tesis

El primer capítulo de la tesis es la introducción, la cual nos da una base de que esperar de este trabajo y establece el marco de referencia.

El capítulo dos es una investigación y explicación de la base en la comunicación de las computadoras, esto es, explica en forma breve, los protocolos de comunicación más usados hoy en día (TCP/IP), así como su inicio e integración con el modelo ISO/OSI.

El tercer capítulo es una breve introducción y explicación del término "seguridad" en los sistemas de cómputo, así como los diferentes tipos de amenazas.

El capítulo cuatro es una explicación más amplia del termino "vulnerabilidades" y un repaso de las vulnerabilidades asociadas al protocolo TCP/IP.

El capítulo cinco investiga las vulnerabilidades de los sistemas operativos de red más usados hoy en día (Windows y Linux), también nos muestra las 10 vulnerabilidades más criticas de cada uno de estos sistemas, hecha por el Instituto SANS y otras organizaciones.

El capítulo seis explica el desarrollo paso a paso del diseño de la herramienta y algunas de las principales directivas de seguridad locales.

En el capítulo siete se muestra la forma en que trabaja la herramienta, así como los resultados obtenidos de la misma.

En el capítulo ocho se mencionan las conclusiones hechas por el autor y las aportaciones.

El último capítulo son las referencias consultadas por el autor.

II PROTOCOLOS DE COMUNICACIÓN

II.1 – Modelo OSI

[11] Entre 1977 y 1984 los profesionales en redes diseñaron un modelo llamado modelo de referencia para Interconexión de Sistemas Abiertos (OSI, por sus siglas en ingles). Como se sabe, en un **sistema abierto** el diseño o los aspectos del sistema no son patentados, es decir, podemos obtener todos los detalles sobre el sistema y utilizarlos con entera libertad para nuestros propios fines. En suma los sistemas abiertos proporcionan la documentación y conexiones que se pueden utilizar para crear programas que ocupen o extiendan el sistema.

II.2 - Capas del modelo ISO/OSI

El modelo ISO/OSI (Organización Internacional de Estándares/Interconexión de Sistemas Abiertos) utiliza capas para organizar una red en módulos funcionales bien definidos. Los diseñadores emplean descripciones de las capas del modelo para construir redes reales pero, de acuerdo al propósito de la red, el diseño puede modificar el número, nombre y función de las capas. Como resultado, una red construida a partir del modelo ISO/OSI puede tener variaciones significativas comparadas con él o con otras redes construidas con base en él.

En una red de capas, cada módulo (o capa) proporciona funcionalidad específica o servicios a sus capas adyacentes. Además, cada capa protege a las capas superiores de los detalles de la implementación de las inferiores. Dicho de otra manera, cada capa se preocupa por su interfase con la siguiente capa. El diseño en capas del modelo OSI se muestra en su forma general en la figura 1.

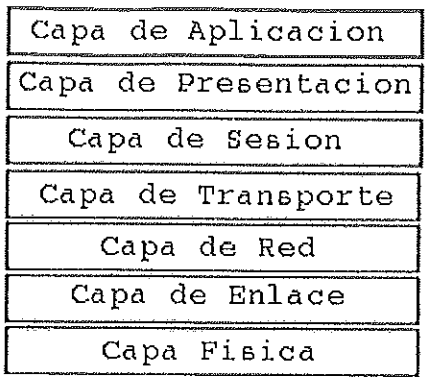


Figura 1. Capas de la red en el modelo ISO/OSI.

Capa física

La capa física trasmite datos a través de los canales de comunicación de la red. Incluye los elementos físicos (hardware) necesarios para cumplir esta función. Por lo tanto, las líneas de transmisión de la red (cables que conectan todas las computadoras a la red) son parte de esta capa. Los métodos de transmisión, incluyendo las señales de control y la sincronización, también forman parte de la capa física. Asimismo, esta capa incluye tecnologías de red (Ethernet, ARCNET y Token Ring) que definen parámetros para trasmitir datos.

Capa de enlace

La capa de enlace (o capa de enlace de datos) trasfiere datos directamente entre la capa física y la capa de red. La tarjeta de interfase de red representa el enlace de datos en la computadora. La función principal de la capa de enlace es evitar que la información dentro de la capa física se corrompa.

Capa de red

La capa de red determina la ruta o trayectoria que siguen los datos para llegar a su destino. Por eso debe manejar el tráfico, el congestionamiento y las tasas de transferencia (velocidad) a través de las líneas de transmisión, así como la corrupción de datos en la red.

Capa de trasporte

Así como la capa de red entrega paquetes de datos a través de la red, la capa de trasporte entrega (trasporta) datos dentro de un anfitrión. Luego de que la capa de red entrega los datos a la dirección correcta del anfitrión, la capa de trasporte los entrega a la aplicación correcta dentro del mismo anfitrión destino.

Capa de sesión

Como interfase entre el usuario y la red, la capa de sesión negocia las conexiones entre los procesos o aplicaciones entre diferentes anfitriones. Por lo tanto, maneja los detalles tales como nombres de cuentas, contraseñas y permisos de usuarios.

Capa de presentación

La capa de presentación consolida funciones comunes que la computadora debe usar varias veces durante las comunicaciones en red. Esta capa maneja detalles relacionados con la interfase de red e impresoras, monitores y formatos de archivo. En pocas palabras, la capa de presentación define como se presenta la red a sí misma ante su máquina y sus programas.

Capa de aplicación

La capa de aplicación contiene los detalles acerca de las aplicaciones de toda la red. Ejemplos de aplicaciones para la red son el correo electrónico y las bases de datos distribuidas. De hecho, todos los programas para usuarios de redes (los usuarios finales de la red) son parte de la capa de aplicación.

II.3 - Protocolos de red

Como puede observarse, las redes utilizan módulos funcionales llamados capas. Un protocolo es un conjunto de reglas y convenciones que utiliza una capa de red para comunicarse. En una red de capas, cada una de ellas emplea protocolos bien definidos para comunicarse con las que la rodean. Conceptualmente, cuando dos computadoras anfitrión hablan entre sí, las capas correspondientes en cada una sostienen también una

conversación. Podemos llamar "procesos pares" a las capas homólogas en diferentes anfitriones en la red, como puede verse en la figura 2.

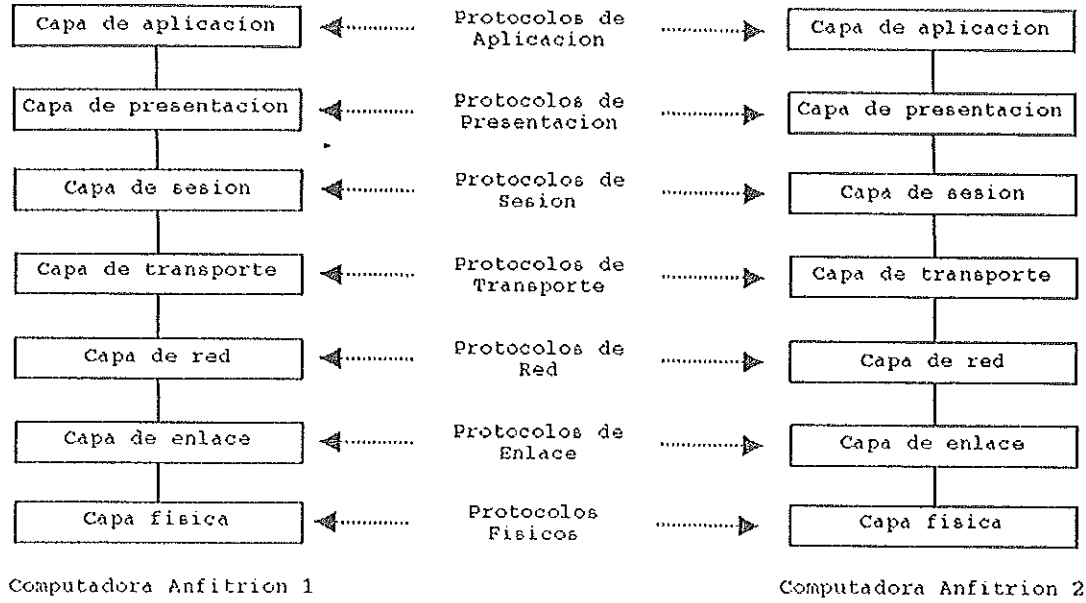


Figura 2. Modelo de red que muestra los procesos pares.

La comunicación entre procesos pares es una "comunicación virtual" (o llamado también "circuito virtual"). En realidad, toda comunicación entre dos computadoras anfitrión sucede en la capa inferior (física) de la red, donde existe la conexión real (física) en hardware.

II.4 - TCP/IP

TCP son las siglas de Protocolo de Control de Transporte (*Transport Control Protocol*), e IP de Protocolo Internet (*Internet Protocol*). Cuando se combinan ambas tenemos TCP/IP, siglas que representan algo más que dos protocolos. Internet se basa en una colección de protocolos llamada conjunto de protocolos TCP/IP. Un conjunto de protocolos es una colección de protocolos cooperativos y complementarios. El conjunto de protocolos TCP/IP incluye los protocolos de Control de Transporte e Internet, así como otros varios. Los más usuales se muestran en la tabla I. Todos ellos trabajan en conjunto para trasmitir información a través de Internet.

Protocolo	Propósito
IP	El Protocolo Internet es un protocolo de la capa de red que mueve la información entre computadoras anfitriones.
TCP	El Protocolo de Control de Trasporte es un protocolo de la capa de trasporte que mueve la información entre las aplicaciones.
UDP	El Protocolo de Datagrama de Usuario es otro protocolo de la capa de trasporte. UDP también mueve información entre las aplicaciones, pero es menos complejo (y menos confiable) que TCP.
ICMP	El Protocolo de Control de Mensajes de Internet lleva mensajes de error de la red y notifica otras condiciones que requieren atención del software de la red.

Tabla I. Protocolos TCP/IP más comunes.

Pila de protocolos TCP/IP

El modelo ISO/OSI representa a una red como una pila vertical de módulos o capas. Puesto que el modelo asocia al menos un protocolo con cada capa, podemos decir que apila protocolos uno arriba del otro. El termino pila de protocolos deriva de este concepto de redes como capas verticales y protocolos apilados.

El concepto pila de protocolos puede referirse a cualquier combinación de capas de la red y sus protocolos asociados. La pila de protocolos TCP/IP es solo una de las muchas que soporta el modelo en capas ISO/OSI. La figura 3 muestra una analogía entre la pila TCP/IP y el modelo IOS/OSI.

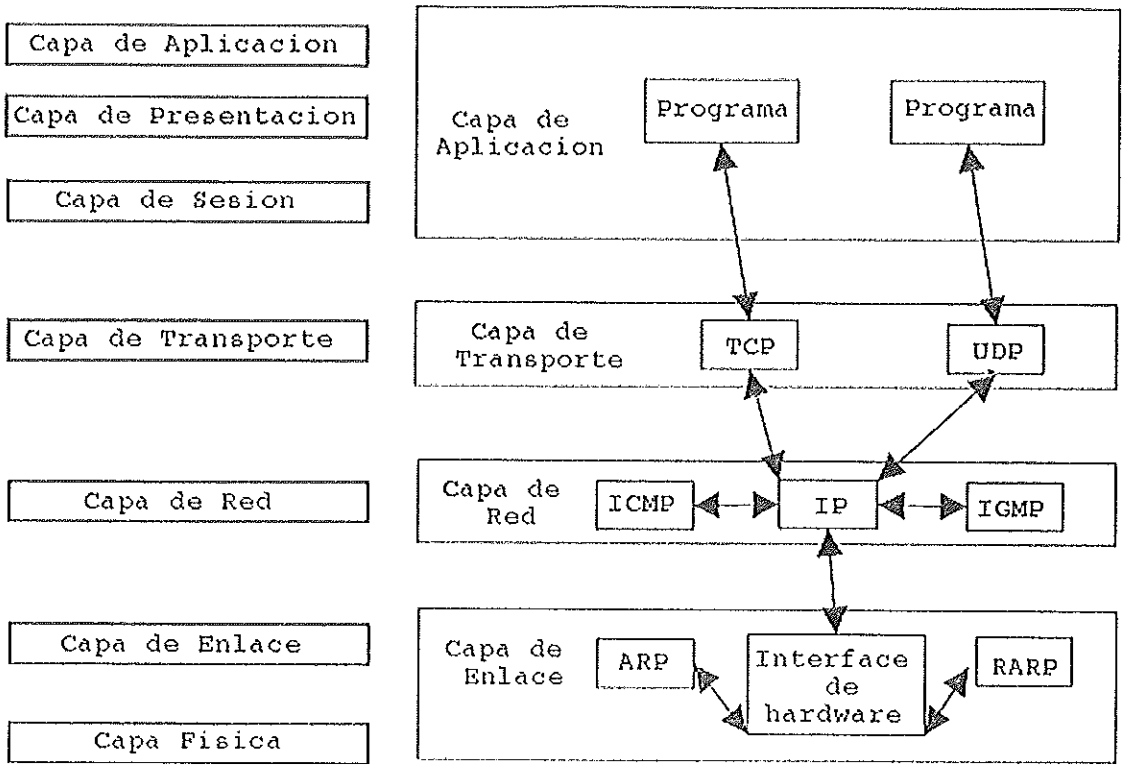


Figura 3. Modelo ISO/OSI y la pila de protocolos TCP/IP.

El conjunto de protocolos TCP/IP mueve información a través de la red y, como proporciona un conjunto de protocolos cooperativos, la información fluye de una capa a otra y de un protocolo al siguiente. Cuando los datos llegan al destino físico, fluyen hacia arriba, a través de la pila de protocolos, hacia la aplicación destino. Mientras los datos fluyen por la pila, los protocolos TCP/IP pueden dividirlos en piezas más pequeñas (o unidades).

II.5 Protocolos usados en el desarrollo de este trabajo

II.5.1 – TCP (Protocolo de Control de Transmisión)

El protocolo de control de transmisión provee un considerable número de servicios, pero más importante, provee un servicio orientado a conexión, que notifica a la capa superior que el datagrama enviado a través de la red fue recibido en su totalidad. De esta manera, TCP actúa como un protocolo de validación, al proveer comunicaciones confiables. Si un paquete es corrompido o perdido, TCP usualmente maneja la retransmisión.

Debido a que TCP es un protocolo orientado a conexión, es responsable de asegurar la transferencia de un paquete de la máquina fuente a la máquina destino (comunicación punto a punto), TCP debe recibir mensajes de comunicación de la máquina destino para el reconocimiento de la recepción del paquete. El término circuito virtual es usualmente utilizado para referirse a la comunicación entre dos máquinas usando puntos (o *endpoints* llamados en inglés), la mayoría son simples mensajes de reconocimiento (sean de confirmación, o recepción, o un código de alguna falla) y números de secuencia de paquetes.

Puertos y *Sockets*

Puertos

Todas las aplicaciones de nivel superior que usan ya sea UDP o TCP tienen un número de puerto que identifica a la aplicación. En teoría, los números de puerto pueden ser asignados en máquinas individuales, o como un administrador lo desee, pero algunas convenciones han sido adoptadas para permitir una mejor comunicación entre las implementaciones de TCP. Esto permite al número de puerto identificar el tipo de servicio que un sistema TCP está solicitando a otro. El número de los puertos puede cambiar, aunque esto puede causar dificultades. La mayoría de los sistemas mantienen un archivo con los números de puertos y su correspondiente servicio.

Socket

Cada circuito de comunicación dentro y fuera de la capa de TCP es únicamente identificado por su combinación de dos números, que juntos son llamados *socket*. El *socket* está compuesto de la dirección IP de la máquina y el número de puerto usado por el *software* TCP. Tanto la máquina emisora como la máquina receptora tienen *sockets* asociados. Debido que la dirección IP es única a través de la Internet, y el número de puerto es único de la máquina individual, los números de *socket* son también únicos a través de Internet. Esto permite a un proceso establecer una comunicación con otro proceso a través de la red, basado completamente en el número de *socket*.

Interfaz de *sockets*

Una Interfaz de Programa de Aplicación (API, por sus siglas en inglés) es, simplemente, un grupo de funciones (una interfaz de *software*) que los programadores utilizan a fin de desarrollar programas de aplicación para un ambiente de cómputo específico. La interfaz de *sockets* es una API para redes TCP/IP, es decir, define una variedad de funciones o rutinas que permiten desarrollar aplicaciones para utilizarlas en redes TCP/IP. Los diseñadores de la interfaz de *sockets* originalmente la construyeron dentro del sistema operativo UNIX.

El Proceso del *socket*

Como se mencionó anteriormente un *socket* representa un extremo de la comunicación por red. Este también identifica a un registro en la tabla de descripción de forma similar a la manera en que un identificador de archivo reconoce un registro en la tabla de archivos descriptores, como se ve a continuación en la figura 4.

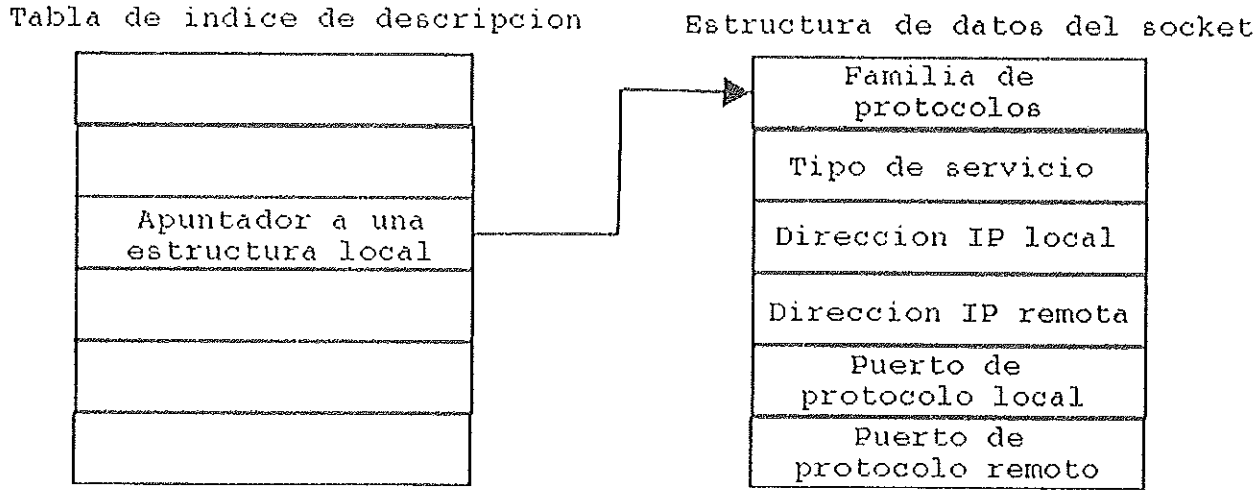


Figura 4. Estructura simple de datos del *socket* en UNIX.

El modelo de la interfase de *sockets* para las comunicaciones de red se refiere a la comunicación entre computadoras anfitrión o procesos como extremos. Cada conversación en la red incluye dos extremos: el anfitrión local y el remoto (o destino). La interfaz denomina a cada extremo de la conversación en la red: *socket*. La mayoría de las

aplicaciones utilizan el modelo cliente/servidor. Las comunicaciones de red en el modelo cliente/servidor también incluyen dos extremos. Sin embargo, el modelo cliente/servidor hace su distinción entre los extremos de la función que ejecutan. Por ejemplo, el modelo llama al extremo que inicia o solicita un servicio de la red proceso o programa cliente. Al extremo que responde a la solicitud del cliente lo denomina proceso o programa servidor.

Conexión de un socket

Los *sockets* proporcionan una abstracción que puede utilizar para representar y programar los extremos de una conversación en red. Un protocolo orientado a conexión establece un circuito virtual entre los extremos de la conexión, es decir, el enlace entre los dos extremos parece directo, punto a punto. En la capa de transporte TCP/IP, TCP establece un circuito virtual (mantiene la conexión abierta) intercambiando mensajes de confirmación entre los dos extremos. Como resultado, un programa cliente orientado a conexión en una red TCP/IP no se ocupa de la dirección local que utiliza el software de red para la transferencia de datos. En otras palabras, el programa cliente puede recibir datos en cualquier puerto de protocolo. Por lo tanto, en la mayoría de los casos un programa cliente orientado a conexión no especifica el número de puerto del protocolo local.

Un programa cliente orientado a conexión utiliza la función `connect` para configurar un *socket* de comunicación en red. En pocas palabras, la función `connect` almacena información en la estructura de datos del socket acerca de los extremos local y remoto. Esta función requiere la especificación del *socket*, una estructura de dirección que contenga información sobre el anfitrión remoto y la longitud de la estructura de dirección del *socket*. Esta función indica también una conexión directa con el anfitrión remoto. En la figura 5 se muestra un diagrama a bloques de los pasos de una comunicación orientada a conexión.

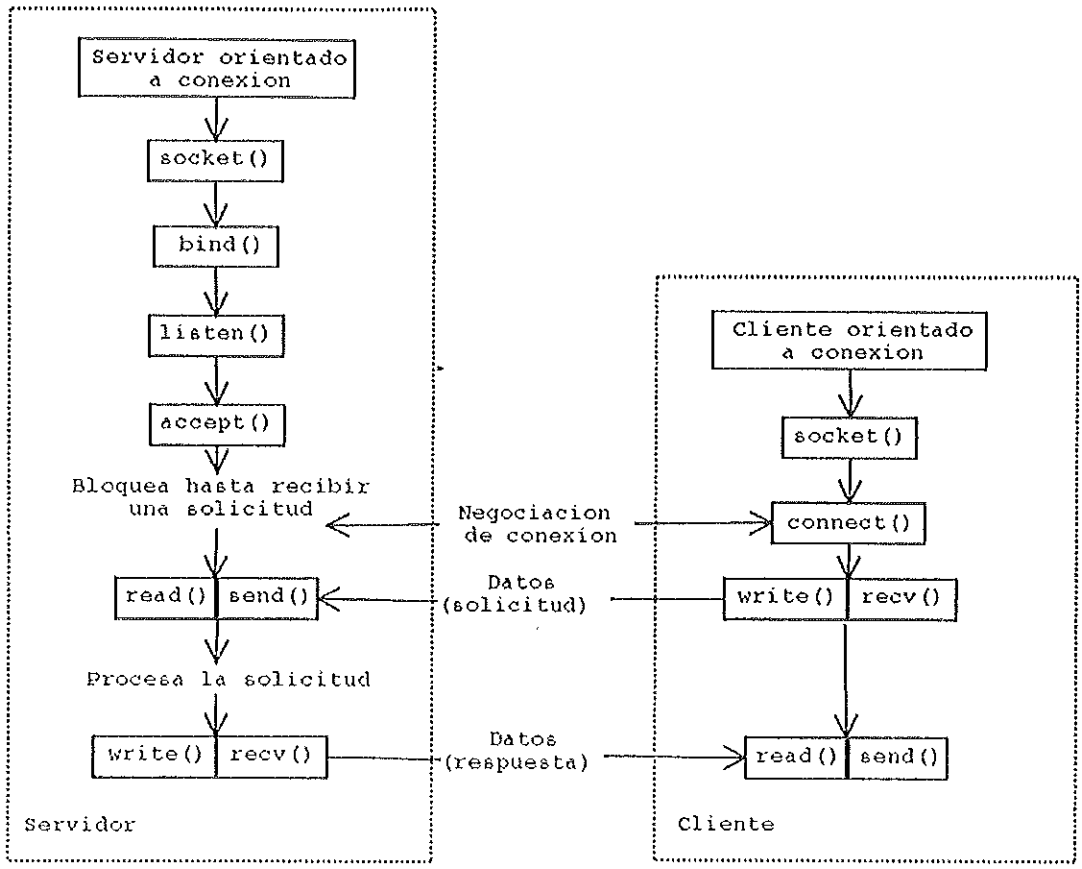


Figura 5. Uso de los *sockets* con un protocolo orientado a conexión.

II.5.2 - UDP (Protocolo de Datagrama de Usuario)

TCP es un protocolo orientado a conexión. Cuando se hace necesario un protocolo no orientado a conexión, entonces se usa UDP. UDP es utilizado tanto en el Protocolo de Transferencia de Archivos Trivial (TFTP) como en el Procedimiento de Llamadas Remotas (RPC). Las comunicaciones no orientadas a conexión no son confiables, significa que no hay ninguna indicación a la máquina emisora de que el mensaje fue recibido correctamente. Los protocolos no orientados a conexión tampoco ofrecen capacidad para la recuperación de errores –los cuales deben ser ignorados o corregidos por las capas de nivel superior-. UDP es mucho más simple que TCP como puede verse en el diagrama de bloques mostrado en la figura 6. Ya que interactúa con IP (u otros protocolos) sin las funciones del flujo de control o los mecanismos de recuperación de errores, actuando simplemente como un emisor y receptor de datagramas.

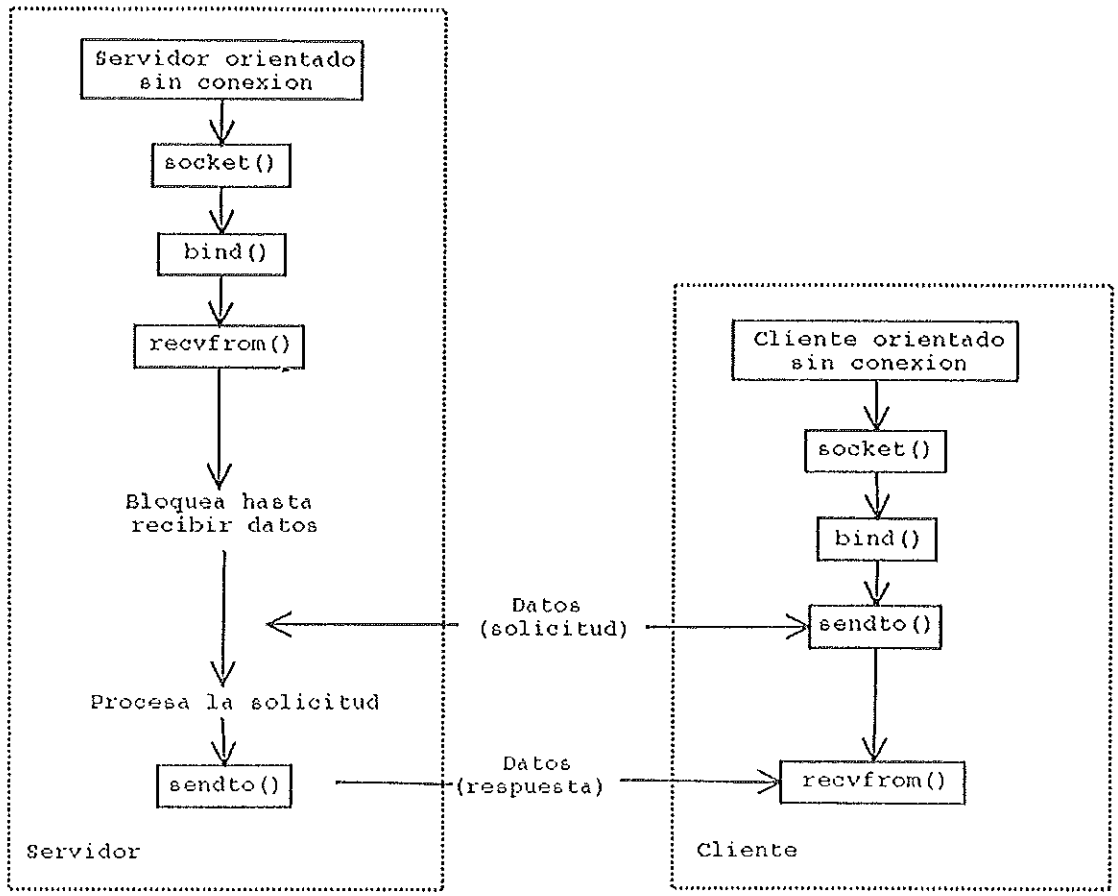


Figura 6. Uso de los *sockets* con un protocolo sin conexión

II.5.3 - ICMP (Protocolo de Control de Mensajes de Internet)

Internet depende de un servicio no confiable (pero efectivo) de entrega de datagramas: el Protocolo Internet (IP). IP no garantiza la entrega de paquetes de la red ni notifica una falla cuando el canal de comunicación se interrumpe o pierde paquetes de información. Sin embargo, bajo ciertas condiciones, el *software* de una red TCP/IP genera mensajes de error. El Protocolo Internet de Control de Mensajes (ICMP) transporta mensajes de error de la red e informa otras condiciones que requieren la atención del *software* de la red.

Como se sabe muchos problemas pueden ocurrirle a un mensaje cuando viaja por la red: el tiempo de vida del paquete puede expirar, datagramas fragmentados pueden no llegar intactos, etcétera. Es importante notificar a la máquina emisora que hay un error en el envió de su mensaje, así como el manejo de estos errores. El Protocolo de Control de

Mensajes de Internet (ICMP), se encarga de esta tarea. Los tipos de errores del paquete ICMP se pueden ver en la tabla II.

Valor	Descripción
0	Respuesta eco
3	Destino inalcanzable
4	Fuente apagado
5	Petición de direccionamiento
8	Petición eco
11	Tiempo de vida excedido
12	Problema en los parámetros
13	Petición de huella de tiempo
14	Respuesta de huella de tiempo
15	Petición de información (ya obsoleta)
16	Respuesta de información (ya obsoleta)
17	Petición de mascara de red
18	Respuesta de mascara de red

Tabla II. Valores validos del campo tipo de mensaje de ICMP.

Sockets raw (o básicos)

Un *socket* básico permite que sus programas "brinquen" la capa de transporte TCP/IP y tengan acceso a protocolos de nivel inferior, como ICMP. Cuando se usan este tipo de *sockets*, el programa debe hacer funciones de la capa de transporte, como el encapsulamiento de información, para la capa IP. Con el uso de los *sockets* básicos se abren los protocolos de gestión de mensajes (de error). Los protocolos de alto nivel como TCP y UDP, cierran la posibilidad de envió y recepción de paquetes ICMP, lo cual resulta muy útil.

III SEGURIDAD EN LOS SISTEMAS DE CÓMPUTO

Se puede entender como seguridad una característica de cualquier sistema, que nos indique que el mismo esta debidamente protegido y libre de todo peligro, daño y/o riesgo. Es decir, hablando de una forma más especifica, en sistemas operativos o redes de computadoras, es muy difícil conseguir o implementar el término seguridad, y en lugar de éste, se habla de **fiabilidad** la cual de define como la probabilidad de qué un sistema se comporte tal y como se espera de él.

A grandes rasgos se entiende que mantener un sistema seguro (o fiable) consiste básicamente en garantizar tres aspectos: **confidencialidad, integridad y disponibilidad.**

- ✓ La confidencialidad nos dice que los objetos de un sistema han de ser accedidos únicamente por elementos autorizados a ellos, y que esos elementos autorizados no van a compartir esa información a otras entidades.

- ✓ La integridad significa que los objetos sólo pueden ser modificados por elementos autorizados, y de una manera controlada.

- ✓ La disponibilidad indica que los objetos del sistema tienen que permanecer accesibles a elementos autorizados; es el contrario de la negación de servicio.

III.1 – Amenazas a la seguridad

Personales

Generalmente se tratará de intrusos que intentan conseguir el máximo nivel de privilegios posibles aprovechando algún o algunas de las vulnerabilidades que nuestro sistema operativo, o red, pudiera tener.

- Personal.

Las amenazas de seguridad de un sistema proveniente del personal de la propia organización no son tomadas muy en cuenta, ya que se presume un entorno de confianza que muy posiblemente en algunos casos no existe, por lo que se pasa por alto el hecho de que casi cualquier persona de la organización, incluso el personal ajeno a la infraestructura informática (secretaria, personal de seguridad, personal de limpieza y mantenimiento) puede comprometer la seguridad de los equipos y consecuentemente la red.

- Ex - empleados.

Otro gran grupo de personas potencialmente interesadas en atacar el sistema son los antiguos empleados del mismo, especialmente los que no abandonaron el entorno por voluntad propia, los que pasaron a la competencia. Generalmente se trata de personas descontentas con la organización que pueden aprovechar las debilidades del sistema que conocen perfectamente para dañarlo como venganza por un hecho que no consideraron justo.

- Curiosos.

Estos son los atacantes más habituales en sistemas de redes. Recordemos que los equipos están trabajando en entornos donde se forman a futuros profesionales de la informática y las telecomunicaciones, es decir gente que tiene interés por las nuevas tecnologías. También recordemos que las personas son curiosas por naturaleza; esta combinación produce una avalancha de estudiantes o personal intentando conseguir mayor privilegio del que tiene o intentando acceder a sistemas a los que oficialmente no tienen acceso. En la mayoría de los casos esto se hace simplemente para leer el correo de un amigo, enterarse de cuanto cobra un compañero, copiar un trabajo o comprobar que es posible romper la seguridad de un sistema concreto. En la mayoría de las situaciones se trata de ataques no destructivos.

- *Crackers*.

Los entornos de seguridad media son un objetivo típico de los intrusos, ya sea para fisgonear, para utilizarlas como enlace hacia otras redes o simplemente por diversión. Por un lado, son redes generalmente abiertas, y la seguridad no es un factor muy considerado en ellas. De esta forma un atacante sólo requiere utilizar un escáner de seguridad contra el dominio completo y luego atacar mediante un simple *exploit* los equipos que presenten vulnerabilidades; esto convierte a las redes en un objetivo fácil y apetecible para intrusos con cualquier nivel de conocimientos, desde los más novatos (y a veces más peligrosos) hasta los más expertos, que puede utilizar toda la red para probar nuevos ataques o como nodo intermedio en un ataque a otros organismos.

- Terroristas.

Persona que ataca al sistema simplemente por causar algún tipo de daño en él. Por ejemplo, alguien puede destruir los sistemas de archivos de un servidor que alberga páginas *Web*.

- Intrusos Remunerados.

Este es el grupo de atacantes de un sistema más peligroso, aunque por fortuna el menos habitual en redes normales; suele afectar más a las grandes empresas u organismos de defensa. Se trata de intrusos con gran experiencia en problemas de seguridad y un amplio conocimiento del sistema, que son pagados por una tercera persona, por lo generalmente para robar o simplemente para dañar la imagen de la entidad afectada. Esta tercera persona suele ser una empresa de la competencia o un organismo de inteligencia, es decir, una organización que puede permitirse el lujo de hacer un gran gasto en el ataque.

Lógicas

Bajo la etiqueta de "amenazas lógicas" encontramos todo tipo de programas que de una u otra forma pueden dañar un sistema, creados de forma intencionada para ello (*exploits*, troyanos, etc.) o simplemente por error (*bug*). Algunos tipos de amenazas lógicas son:

- Software Incorrecto.

Las amenazas más habituales provienen de errores cometidos de forma involuntaria por los programadores de sistemas o de aplicaciones. A estos errores de programación se les denomina *bugs*.

- Herramientas de Seguridad.

Cualquier herramienta de seguridad representa un arma de doble filo: de la misma forma que un administrador las utiliza para detectar y solucionar fallos en su sistema o en la subred completa, un potencial intruso las puede utilizar para detectar esos mismos fallos y aprovecharlos para atacar los equipos. Herramientas como Nessus, Nmap o Satan pasan de ser útiles a ser peligrosas cuando las utilizan *crackers* que buscan información sobre las vulnerabilidades de un anfitrión o una red completa.

- Bombas Lógicas.

Las bombas lógicas son partes de código de ciertos programas que permanecen sin realizar ninguna función hasta que son activadas; en ese punto, la función que realizan no es la original del programa anfitrión, sino que generalmente se trata de una acción perjudicial.

- Virus.

Un virus es una secuencia de código que se inserta en un archivo ejecutable, denominado huésped, de forma que cuando el archivo se ejecuta, el virus también lo hace, insertándose a si mismo en otros programas.

- Puertas Traseras.

Durante el desarrollo de aplicaciones grandes o de sistemas operativos es habitual entre los programadores insertar "atajos" en los sistemas habituales de autentificación del programa o del núcleo que se esta diseñando. A estos atajos se les denomina puertas traseras, y con ello se consigue mayor velocidad a la hora de detectar y depurar fallos. Algunos programadores pueden dejar estos atajos en las versiones definitivas de su software para facilitar un mantenimiento posterior, para garantizar su propio acceso, o

simplemente por descuido. La cuestión es que si un atacante descubre una de estas puertas traseras va a tener un acceso global a datos que no debería poder leer, lo que obviamente supone un grave peligro para la integridad de nuestro sistema.

- Canales Cubiertos.

Los canales cubiertos, o canales ocultos, son canales de comunicación que permiten a un proceso transferir información de forma que viole la política de seguridad del sistema, dicho de otra forma, un proceso trasmite información a otros, sean locales o remotos, que no están autorizados a leer dicha información.

- Gusanos.

Un gusano es un programa capaz de ejecutarse y propagarse por si mismo a través de redes, en ocasiones portando un virus o aprovechando *bugs* de los sistemas a los que se conecta para dañarlos. Al ser difíciles de programar su número no es muy elevado, pero el daño que pueden causar es muy grande: el mayor incidente de seguridad en Internet fue precisamente el famoso *Internet Worm*, un gusano que en 1988 causo pérdidas millonarias al infectar y detener más de 6000 máquinas conectadas a la red. Su principal peligro reside en que un gusano puede automatizar y ejecutar en unos segundos todos los pasos que seguiría un atacante humano para acceder a un sistema.

- Caballos de Troya.

Los troyanos o caballos de Troya son instrucciones escondidas en un programa de forma que parezca realizar tareas que un usuario espera de él, pero realmente ejecute funciones ocultas, generalmente en detrimento de la seguridad, sin el conocimiento del usuario.

- Programas Conejo o Bacterias.

Bajo este nombre se conoce a los programas que no hacen nada útil, sino que simplemente se dedican a reproducirse hasta que el número de copias acaba con los recursos del sistema (memoria, procesador, disco, etc.), produciendo una negación de servicio. Por sí mismos no hacen ningún daño, sino lo que realmente perjudica es el gran

número de copias suyas en el sistema, que en algunas situaciones pueden llegar a provocar la parada total de la máquina.

III.2 – Los agujeros (o vulnerabilidades)

El concepto de agujero u hoyo

Un agujero es cualquier característica de *hardware* o *software* que permite a un usuario no autorizado ganar acceso o incrementar su nivel de acceso o privilegios sin autorización. Un hoyo puede ser virtualmente cualquier cosa. Por ejemplo, algunas peculiaridades de *hardware* o *software* comúnmente conocido por todos los usuarios califica como agujero. Una de estas peculiaridades, probablemente la más conocida, es que las contraseñas de CMOS compatibles con IBM se pierden cuando la batería CMOS es cortada, deshabilitada o desconectada. Incluso la habilidad de iniciar el sistema en modo simple-usuario en una estación de trabajo puede ser clasificada como un agujero. Esto es porque permitiría a un usuario malicioso entrar en modo interactivo de comandos, y podría tener parcialmente el control de la máquina.

Así que un agujero u hoyo no es nada más que alguna forma de vulnerabilidad de un sistema. Cualquier plataforma tiene hoyos, así sea en *hardware* o *software*. No hay plataforma absolutamente segura.

IV VULNERABILIDADES Y ATAQUE A LA SEGURIDAD

IV.1 - La escala de vulnerabilidades

Hay diferentes tipos de hoyos o vulnerabilidades

- Vulnerabilidades que permiten negación de servicio.
- Vulnerabilidades que permiten a usuarios locales con privilegios limitados el incremento de estos sin autorización.
- Vulnerabilidades que permiten accesos de afuera (en una máquina remoto) sin autorización a la red de trabajo.

Estos tipos de vulnerabilidades y ataques pueden ser clasificados de acuerdo al peligro que ellos representan al sistema victima. Algunos representan peligro significativo que puede destruir al objetivo; otros son menos serios, calificados solo como molestos.

Vulnerabilidades que permiten negación de servicio

Las vulnerabilidades que permiten negación de servicio están en la categoría C de la escala de vulnerabilidades, y son los de más baja prioridad. Estos ataques son casi siempre basados en el sistema operativo. Es decir, estos hoyos existen dentro de las **particiones de red del sistema operativo**. Cuando este hoyo existe, debe ser corregido por los autores del software o por algún parche del vendedor.

Para redes o sitios grandes, un ataque de negación de servicio es de significado limitado. Puede ser solamente molesto y no más. Aunque un ataque de negación de servicio distribuido, si podría poner en peligro el sitio o la red, por muy grande que sea. Los sitios pequeños, de cualquier manera podrían sufrir con un ataque de negación de servicio. Esto es especialmente por que estos sitios sólo poseen una máquina, por lo tanto un solo servidor de correo que da servicio a todo el sitio, por ejemplo.

Vulnerabilidades que permiten a usuarios locales acceso sin autorización.

Aún más arriba en la jerarquía de vulnerabilidades, en la clase B, están esos hoyos que permiten a usuarios locales incrementar su nivel de acceso. Este tipo de hoyos es encontrado típicamente dentro de aplicaciones en su propia plataforma.

Un usuario local es alguien que tiene una cuenta en una máquina de la red objetivo. Este usuario puede ganar acceso como súper usuario (*root* o administrador, dependiendo del sistema) gracias a una vulnerabilidad que puede ser común.

Este tipo de hoyos representa un serio problema por una razón: si un usuario local maneja exitosamente un *exploit* en determinada vulnerabilidad, el administrador puede nunca descubrirla. Además tener acceso para subir su nivel de privilegios, es mucho más peligroso en las manos de un usuario local que uno remoto. Esto es porque un usuario local puede emplear utilidades básicas del sistema para aprender más acerca de la red local. Estas utilidades revelan mucho más que cualquier escáner desde afuera. Por lo tanto, un usuario local con un pequeño incremento de acceso puede explotar este acceso a un nivel más grande.

La única comodidad real respecto a este tipo de vulnerabilidad es que hay más oportunidad de identificar al atacante, particularmente si es inexperto. Si el administrador del sistema esta corriendo utilidades de bitácoras, el atacante necesitara mucha experiencia para escapar de la detección. La mayoría de vulnerabilidades u hoyos clase B surgen de algún defecto dentro de una aplicación. Estos pueden ser justamente algunos errores comunes de programación que pueden llevar a explotar dichos hoyos. Estos serían las llamados sobrecargas de *buffer*. Estos pueden ser eliminados por minuciosas técnicas de programación.

Vulnerabilidades que permiten a usuarios remotos acceso sin autorización

Este tipo de vulnerabilidades pertenecen a la clase A, las cuales son las más amenazantes de todas y no sorpresivamente, muchas de ellas son provocadas por una pobre administración del sistema o una mala configuración. Los fabricantes raramente no ponen atención en estos hoyos que permiten a usuarios remotos acceso no autorizado.

Naturalmente, estos hoyos provocan un peligro significativo al sistema desde fuentes externas. En muchos casos, si el administrador del sistema esta corriendo solo bitácoras mínimas, estos ataques pueden no ser grabados o guardados. Esto hace más difícil la identificación del intruso.

IV.2 - Vulnerabilidades Genéricas

Estas vulnerabilidades pretenden describir las debilidades y los métodos más comunes que se utilizan para perpetrar ataques a la seguridad de la familia de protocolos TCP/IP, como confidencialidad, integridad y disponibilidad de la información.

Los ataques pueden estar motivados por diversos objetivos, incluyendo fraude, extorsión, robo de información confidencial, venganza, acceso no autorizado a un sistema, anulación de un servicio o simplemente el desafío de penetrar un sistema. Estos pueden provenir principalmente de dos fuentes

- Usuarios autentificados, al menos a parte de la red, como por ejemplo empleados internos o colaboradores externos con acceso a sistemas dentro de la red de la empresa. También denominados *insiders*.
- Atacantes externos a la ubicación física de la organización, accediendo remotamente. También denominados *outsiders*.

Los métodos de ataque descritos se han divido en categorías que pueden estar relacionadas entre sí, ya que el uso de un método permite o facilita el uso de otros, en ocasiones, complementarios. Un ejemplo de ataque podría ser la realización del análisis

de un sistema, mediante *fingerprinting*, tras el cual es posible explotar una vulnerabilidad como una sobrecarga de *buffer* de un servicio TCP/IP, enviando paquetes que parecen válidos mediante *IP spoofing*. En numerosas ocasiones se emplea el término en inglés para nombrar la vulnerabilidad, ya que es como se conoce comúnmente como se apreciará en la tabla III.

Footprinting **(impresión de huella)**	*Footprinting*, es el arte o técnica de extraer toda la información posible de la red objetivo del ataque.
Fingerprinting **(impresión digital)**	*Fingerprinting*, es la obtención de la huella identificativa del sistema operativo con respecto a la pila TCP/IP. Esta información junto con la versión del servicio o servidor facilitara la búsqueda de vulnerabilidades asociadas al mismo.
Sniffing **(olfatear)**	El *sniffer* almacenara en una bitácora todo el tráfico que circule por la red, ya sea destinado o generado por el propio sistema o desde/hacia cualquiera de los sistemas existentes en el entorno de red compartido (segmento *Ethernet*).
Eavesdropping **(escuchando a escondidas)**	El *eavesdropping* es una variante del *sniffing* caracterizada porque únicamente contempla la adquisición o intercepción en forma pasiva del trafico que circula por la red, es decir sin modificar el contenido de la misma.
Snooping **(husmeando)**	Esta es otra variante del *sniffing* basada en el almacenamiento de la información obtenida en el ordenador del atacante. En este caso, tampoco se modifica la información incluida en la transmisión.
IP spoofing **(usurpación de IP)**	El *spoofing* como tal, se basa en actuar en nombre de otro usuario tal y como si se fuese el mismo. En TCP/IP, se basa en la generación de paquetes IP con una dirección origen falsa.
SMTP Spoofing y Spamming **(usurpación y**	En un nivel superior, concretamente a nivel de aplicación, en el protocolo SMTP (puerto TCP 25) es posible usurpar la dirección fuente de un correo electrónico, enviando por tanto mensajes en nombre de otra persona. Es así porque el protocolo no lleva a cabo ningún

envío masivo de SMTP)	mecanismo de autentificación cuando se realiza la conexión TCP al puerto asociado. El *spamming* consiste en el envió masivo de un mensaje de correo a muchos usuarios destino, pudiendo llegar a saturarse los servidores de correo.
DoS: Denial of Service **(negación de servicio)**	Un ataque de éste tipo se centra en sobrepasar los límites de recursos establecidos para un servicio determinado. Este tipo de ataques no supone ningún peligro para la seguridad de las máquinas, ya que no modifica los contenidos de la información, ni permite obtener información sensible. Simplemente se persigue entorpecer el acceso de los usuarios a los servicios de un sistema.
Net Flood **(inundación de red)**	El objetivo de este ataque es degradar la capacidad de conexión a la red de un sistema, saturando sus enlaces de comunicaciones. Si el enlace de una organización es de 34 Mb. Y un atacante dispone de un enlace de 155 Mb. Prácticamente la totalidad del tráfico pertenecerá al atacante.
Smurf **(emisión de archivos)**	Dentro del concepto de *Net Flood*, existe una técnica que se aprovecha de las características de *broadcast* de las redes: el *smurf*. Previamente denominado *nuking*. La dirección lógica del *broadcast*, es decir, aquella que representa a todas las máquinas de una red es usada en vez de consultar uno por uno a todos los sistemas existentes.
TCP Syn Flood **(inundación de paquetes syn TCP)**	Dentro de los ataques DoS, existe uno asociado directamente al protocolo TCP, y consiste en el envió masivo de paquetes de establecimiento de conexión (SYN) contra un sistema. La recepción de estas solicitudes provoca que el sistema destino, reserve cierta cantidad de memoria (*buffer*) para almacenar las estructuras de datos asociadas a cada una de las nuevas conexiones en curso.
Connection Flood **(inundación de conexión)**	Los servicios TCP orientados a conexión, que son la mayoría (Telnet, FTP, HTTP, SMTP, etc.) tienen un límite máximo de conexiones simultáneas soportadas; cuando este limite se alcanza, cualquier conexión nueva es rechazada.
SMTP flood **(inundación**	Mediante el envió masivo de mensajes de correo electrónico a grandes listas de usuarios de forma continua, se provoca la saturación de los

de SMTP)	servidores de correo destino intermedio.
DDos (Negación de servicio distribuido)	Una variante más potente de los ataques DoS, son los DDoS, o Ataques de Negación de Servicio Distribuido, que se basan en realizar ataques DoS de forma masiva a un mismo objetivo desde diferentes localizaciones en la red, de forma que la potencia de ataque sea mucho mayor.
Ping of Death (*ping* de la muerte)	Conocido como el *ping* de la muerte, este ataque se basa en enviar un paquete de "*ping*" de un tamaño muy grande. Teniendo en cuenta el tamaño máximo de paquete en TCP/IP de 64 Kbytes, la implementación de la pila TCP/IP asigna un *buffer* en memoria de este tamaño. En el caso de que la información sea mayor, el *buffer* puede desbordarse.
Land	Este ataque permite bloquear un sistema, mediante el envió de un paquete SYN cuya dirección IP fuente y destino es la misma. Existe una variación de este ataque, basada en que los puertos origen y destino son iguales.
Session Hijaking (secuestro de sesión)	Esta técnica pretende mostrar la posibilidad de apoderarse de una sesión ya establecida. El *TCP hijaking* puede realizarse en entornos de red de difusión (*broadcast*), basado en introducir paquetes en medio de una transmisión como si provinieran del dispositivo original (*IP spoofing*). Este tipo de ataques también se conoce como "*Man in the middle attack*".
Winnuke	Este ataque afecta a los sistemas que utilizan el protocolo NetBIOS, típicamente en el sistema Windows. El envío de un paquete urgente (bit URG=1), conocido como paquete "*Out of Band*" (OOB) da lugar al envío de datagramas UDP a estos puertos (135, 445), que al intentar ser enviados a las capas superiores, pueden provocar que el sistema destino se "cuelgue" o que disminuya su rendimiento de forma notable.
Teardrop (lagrima)	El ataque *teardrop* se basa en el envío de fragmentos de paquetes en lugar de paquetes completos. Algunas implementaciones de la pila TCP/IP no son capaces de reconstruir paquetes con fragmentos cuyos

	bytes se superponen. El resultado es que el sistema destino puede bloquearse. Existen dos versiones de este ataque: *teardrop* y *teardrop2*. La variación se basa en la inclusión de la bandera de urgencia (URG) en la cabecera TCP de los fragmentos.
Finger Bomb **(bomba *finger*)**	Existe otro tipo de ataque DoS que permite forzar al sistema destino a un consumo elevado de CPU realizando una petición *finger* recursiva. Asimismo, se dispone de *scripts* como *kaput* que hacen uso de esta vulnerabilidad.
Buffer Overflows **(sobrecarga de *buffer*)**	Los desbordamientos de un *buffer*, son mencionados como el último tipo de vulnerabilidad dentro de las asociadas a las comunicaciones por TCP/IP, ya que podría considerarse más una vulnerabilidad del sistema que de la red. Técnicamente, este método se basa en la posibilidad de escribir información sobrepasando los límites de un arreglo, consiguiendo así corromper la pila de ejecución, sobrescribiendo el valor retorno de la función, y causando que el flujo de ejecución continúe en una dirección arbitraria.

Tabla III. Principales vulnerabilidades asociadas a TCP/IP.

V VULNERABILIDADES EN SISTEMAS OPERATIVOS DE RED

V.1 - Características de los sistemas operativos de red

Los sistemas operativos de red se definen como aquellos que tiene la capacidad de interactuar con sistemas operativos en otras computadoras por un medio de transmisión, con el objeto de intercambiar información, transferir archivos, ejecutar comandos remotos etc. El punto crucial de estos sistemas es que el usuario debe conocer la sintaxis de un conjunto de comandos o llamadas al sistema para ejecutar estas operaciones, además de la ubicación de los recursos que desee acceder.

Los primeros sistemas operativos de red ofrecían algunas utilidades de gestión de archivos de seguridad simples. La demanda de los usuarios se ha incrementado de forma que los modernos sistemas operativos de red ofrecen amplias variedad de servicios. Estos son algunos de ellos.

- Adaptadores y cables de red
- Nomenclatura global
- Servicios de archivos y directorios
- Sistema tolerantes a fallos
- Optimización de acceso al disco (*Disk Caching*)
- Sistema de Control de Transacciones (TTS, *Transation Tracking System*)
- Seguridad en la conexión
- Puentes y enrutadores (*Bridges* y *Routers*)
- Pasarelas (*Gateways*)
- Servidores Especiales
- Herramientas *software* de administración

V.2 - Las Vulnerabilidades del Instituto SANS

El Instituto SANS

SANS (*SysAdmin, Audit., Network, Security*) es el más confiable y por mucho la fuente más grande de información para la capacitación y certificación de seguridad informática en el mundo. Además desarrolla, mantiene y pone a disponibilidad, sin costo, la más grande colección de documentos de investigación acerca de seguridad en la información.

Las 20 vulnerabilidades del Instituto SANS

Los gusanos y otros ciber-ataques exitosos son posibles principalmente por vulnerabilidades en un pequeño número de servicios en común de los sistemas operativos. Los atacantes son oportunistas. Toman la ruta más fácil y conveniente y explotan la falla más conocida con la herramienta de ataque más efectiva y disponible. Ellos cuentan con organizaciones que no arreglan el problema, y frecuentemente atacan indiscriminadamente, escaneando en la Internet algún sistema vulnerable. Los grupos de gusanos destructivos y fáciles de construir, como el *Blaster*, *Slammer* y *Code Red*, pueden ser rastreados directamente a la explotación de una vulnerabilidad no parchada.

Hace cuatro años, el Instituto SANS y el Centro de Protección de la Infraestructura Nacional (NIPC, por sus siglas en ingles) del FBI liberaron un documento resumiendo las Diez Vulnerabilidades de Internet Más Criticas en Seguridad (*Ten Most Critical Internet Security Vulnerabilities*). Miles de organizaciones han usado esta lista, que ha sido continuada y ampliada 3 años después, para priorizar sus esfuerzos y puedan cerrarse primero las vulnerabilidades más peligrosos. Los servicios vulnerables que guiaron a los gusanos *Blaster*, *Slammer* y *Code Red*, así como el gusano *NIMDA*, están en la lista.

La lista de las 20 (10 vulnerabilidades de Windows y 10 de UNIX-Linux) es una concienzuda lista de vulnerabilidades que requieren un remedio inmediato. Es el resultado de un proceso que unió docenas de líderes expertos en seguridad. Se unieron las agencias del gobierno más expertas en seguridad del Reino Unido, Estados Unidos y

Singapur; los vendedores lideres de *software* de seguridad y firmas consultoras; los mejores programadores de seguridad de las universidades; muchas otras organizaciones de usuarios; y el Instituto SANS.

La lista esta dividida en las vulnerabilidades más criticas de los sistemas UNIX-Linux y el sistema Windows [2], la cual se muestra a continuación.

V.2.1 – Vulnerabilidades en UNIX-Linux

1. Sistema de Nombres de Dominios (DNS)

El paquete de Nombres de Dominio de Internet de Berkeley (BIND) es la implementación más ampliamente usada del Servicio de Nombres de Dominio (DNS), un sistema crítico que permite la conversión de nombres de anfitrión en direcciones IP registradas.

Desde que existen muchos servidores de DNS alrededor de Internet, estos se han convertido en un blanco popular para los atacantes. Aún sabiendo que estos servidores son críticos para cualquier organización, estudios han mostrado un número excesivo de servidores antiguos, mal configurados y/o vulnerables que todavía se encuentran en circulación. Un número de factores han contribuidos a esta condición. Los encargados de ellos son administradores que no están al tanto de las actualizaciones de seguridad, sistemas que están corriendo el demonio BIND (llamado "`routed`") innecesariamente y malos archivos de configuración. Cualquiera de estos puede causar una negación de servicio, sobrecargas de *buffer* o *cache* DNS alterado.

Como protegerse

Aplicar todos los parches y actualizaciones recientes del servicio; aplicar reglas apropiadas en el cortafuego para cualquier servidor DNS que se encuentre dentro de la red y no requiera ser accesible desde Internet; configurar el servidor para asegurar las trasferencias entre un servidor primario y uno secundario usando criptografía; enjaular y

configurar el servidor para que trabaje con privilegios mínimos usando un directorio "enjaulado" (*chroot*); deshabilitar la recursividad y las colas alcanzables para defenderse del envenenamiento del *cache* DNS.

2. Servidor Web

El tráfico de HTTP es por mucho el uso más común del público de Internet. Apache ha sido y continuará siendo el servidor *Web* más popular en Internet, por lo cual merece un mayor escrutinio en consideración a sus asuntos de seguridad. Estos incluyen vulnerabilidades no en el mismo servidor, sino en los módulos agregados, *scripts* CGI de prueba/defecto/ejemplo, *bugs* de PHP y otros varios vectores de ataque.

Mientras muchos de estos vectores existan, la principal vulnerabilidad del servidor *Web* de UNIX es el resultado de un sistema mal configurado al tiempo de la instalación o sin mantenimiento regular. El resultado podría ser desde una Negación de Servicio, una mutilación (*defacement*) del sitio *Web*, un acceso completo al servidor (*root*) por parte del atacante a cualquier ataque dentro de estas categorías.

Varios desarrolladores y proyectos de Código Abierto proveen buenas prácticas de configuración y mantienen actualizaciones de seguridad para sus productos, y es vital que cualquier administrador de un sitio *Web* este atento a estas actualizaciones. Es importante saber que la mayoría de los servidores *Web* son comprometidos vía *exploits* públicos que son bien conocidos, los cuales toman ventaja de las vulnerabilidades parchadas o manejadas ya desde hace algún tiempo por los desarrolladores.

Como protegerse

Asegurarse que el servidor está corriendo la versión más actual que este parchada; deshabilitar funcionalidades no necesarias como acceso CGI, soporte para PHP, módulo de SSL y módulo de *proxy*; asegurar el contenido de los directorios de *scripts* como por ejemplo CGIs; asegurarse que PHP este corriendo en modo seguro y deshabilitar los parámetros que muestran información sobre PHP en las cabeceras de HTTP; algunos

módulos pueden ayudar a Apache en su seguridad (*mod_security*); auditar los *scripts* buscando vulnerabilidades (hay herramientas especializadas en ello); considerar correr Apache en un ambiente enjaulado (*chroot*); no correr el servidor con privilegios de *root*, crear un usuario especial con los mínimos privilegios y correrlo con esa cuenta; limitar la información del sistema que es expuesta.

3. Autentificación en UNIX

Contraseñas, frases y/o códigos de seguridad son usados virtualmente en cada interacción entre el usuario y los sistemas de información. La mayoría de las formas de autentificación, así como la protección de datos y archivos, confían fuertemente en contraseñas dadas por los usuarios o vendedores. Además, debido a que el acceso propiamente identificado es frecuentemente no guardado en bitácoras, o si el acceso no parece despertar sospechas, una contraseña comprometida es una oportunidad a explorar un sistema sin virtualmente ser detectado. Un atacante en posesión de una contraseña de usuario podrá tener acceso completo a cualquier recurso disponible para el mismo y estará significativamente cerca de acceder a otras cuentas, máquinas cercanas e incluso obtener acceso de *root* en el sistema. A pesar de esta amenaza, cuentas con niveles de usuario y administrador con contraseñas pobres o sin ellas son todavía muy comunes. Además, las organizaciones con una política bien desarrollada que fuercen la utilización de buenas contraseñas son muy escasas. Las vulnerabilidades más comunes en contraseñas son:

- Cuentas de usuarios con contraseñas débiles o sin contraseñas.
- Cuentas de usuarios con contraseñas ampliamente conocidas o con contraseñas desplegadas abiertamente.
- El sistema o las aplicaciones crean cuentas con niveles de administrador con contraseñas conocidas ampliamente, débiles o sin contraseñas.
- *Hash*, formula matemática, de algoritmos débiles o bien conocidos y/o contraseñas "*hasheadas*" que son guardadas con débil seguridad y pueden ser vistas por cualquier usuario.

Como protegerse

La mejor manera de protegerse es con una política bien desarrollada de contraseñas que incluya: instrucciones detalladas para que los usuarios creen contraseñas fuertes; reglas explicitas para que los usuarios aseguren sus contraseñas recordándoles cambiarlas regularmente y asegurarlas; adecuar un proceso para que el personal, reemplace las contraseñas por defecto/débiles/inseguras/ o las bien conocidas y bloquee las cuentas inactivas o dadas de baja, revise regularmente si las contraseñas son fuertes, borre las cuentas de usuarios y administrador que se crean por defecto y revise regularmente las bitácoras de acceso y autentificación del sistema. En general debe asegurarse que las contraseñas sean fuertes, proteger las contraseñas, control hermético de cuentas, encriptar las sesiones, asegurar y proteger la cuenta del *root*, revisar las cuentas genéricas y auditar regularmente el sistema.

4. Sistema de Versión Concurrente

El sistema de control de versiones provee herramientas para el manejo de diferentes versiones de documentos o código fuente, y facilita a múltiples usuarios trabajar concurrentemente con el mismo grupo de archivos. Este sistema es esencial para el manejo de cualquier proyecto de desarrollo de *software*, o para los documentos corporativos y legales que no proveen una solución central de almacenamiento, pero permite el restablecimiento de diferentes versiones.

El Sistema de Versiones Concurrentes (CVS) es el sistema más popular de control de código fuente que se usa en ambientes UNIX/Linux hoy en día. Muchos proyectos de código abierto permiten el acceso anónimo a los depósitos de CVS. Este depósito puede ser configurado para acceso remoto vía el protocolo del servidor que corre en el puerto TCP 2401 por defecto. Un servidor configurado de esta manera contiene las siguientes vulnerabilidades:

- Una sobrecarga de *buffer* basado en la pila puede ser activado por unas líneas de entrada especialmente preparadas. Un atacante puede explotar la sobrecarga de *buffer* para ejecutar código arbitrario en el servidor CVS. Es importante notar que

cualquier depósito configurado para acceso anónimo es potencialmente vulnerable.

- Vulnerabilidades en las implementaciones de otros comandos y funciones pueden ser explotados por un atacante autentificado para causar una negación de servicio en el servidor CVS, o la ejecución de código arbitrario en el mismo servidor. Algunas de estas vulnerabilidades también pueden ser explotadas por un usuario anónimo.

Subversión es otro sistema de control de versiones muy popular en Linux. El depósito de Subversión puede ser accedido vía el protocolo "svn" que corre en el puerto TCP 3690 por defecto. El servidor contiene las siguientes vulnerabilidades:

- Una sobrecarga de *buffer* basado en la pila puede ser explotado por un atacante no autentificado para ejecutar código arbitrario.
- Una sobrecarga de *buffer* basado en la pila puede ser activado por un comando (*svn get-dated-rev*) especialmente preparado. Si el servidor esta configurado para acceso anónimo, un atacante no autentificado puede explotar código arbitrario en el servidor.

Si un atacante obtiene acceso, podría no sólo infectar los archivos fuentes con *bugs* y puertas traseras, cuando el *software* fuera distribuido, representaría un gran número de sistemas comprometidos, incluso podría ser usado para incriminar a un empleado real en actividades ilegales por medio de la usurpación de identidad.

Como protegerse

Asegurarse que el *software* del servidor este actualizado con el parche más reciente; configurar los servidores para que el acceso remoto se haga por medio de un servicio encriptado; si los depósitos son accedidos desde dentro de la red de la empresa u organización bloquear los puertos por defecto del perímetro de la red; tratar de tener el servidor en un sistema dedicado y configurarlo para que el acceso anónimo sirva para sólo lectura.

5. Servicio de transporte de correos

El correo electrónico es una de las aplicaciones más extendidas en Internet, que utiliza al Protocolo de Transporte Simple de Correo (SMTP) el cual es uno de los protocolos más viejos de TCP/IP. Los Agentes de Transporte de Correo (MTA, por sus siglas en inglés) son los servidores responsables de traer el correo del emisor al receptor requerido, usualmente a través de SMTP, el cual puede ser encriptado en puertos inseguros si el emisor y receptor soportan el servicio. Sendmail es el programa que manda, recibe y remite la mayoría de los procesos de correo en sistemas UNIX y Linux. Sendmail es el MTA más popular. A través de los años la barrera de asuntos de seguridad relacionados con la complejidad de configuración de ésta herramienta se ha incrementado considerablemente.

No es de sorprender, que al darle un uso extendido al correo electrónico, el mismo este bajo un constante ataque de virus, gusanos y atacantes personales. Mientras muchos ataques se enfocan en los clientes de correo más usados comúnmente, los MTA son un blanco común de ataques. La mayoría de las vulnerabilidades actuales en contra de estos servidores caben en una de las siguientes categorías:

- Ataques en contra de sistemas no parchados, incluyendo sobrecargas de *buffer*.
- Abuso de retransmisores abiertos, herramienta favorita de los *spammers*.
- Explotación de otros, mala configuración de retransmisores, ataque a las bases de datos de usuarios para *spam* o ingeniería social.

Como protegerse

Decidir si realmente se necesita un MTA, y si se necesita que sea accesible a Internet; deshabilitar el *software* de servidor de correo si el sistema no está específicamente diseñado y autorizado a funcionar como un servidor de correo, y usar una política de cortafuegos; aplicar los parches y las actualizaciones más recientes de la aplicación; tener un MTA separado para el correo interno y otro para el correo externo; limitar el nivel de privilegios con que correrá el MTA y usarlo enjaulado (*chroot*) si es posible; leer la documentación del mismo y estar atento a las actualizaciones.

6. Protocolo de Manejo Simple de Red

El Protocolo de Manejo Simple de Red (SNMP) es usado extensivamente para configuración y monitoreo remoto en casi todos los tipos de dispositivos modernos habilitados de TCP/IP. Mientras SNMP es ubicuo en la distribución a través de las plataformas de red, es más usado como método para la configuración y manejo de dispositivos como impresoras, enrutadores, interruptores, puentes y puntos de acceso inalámbrico, y para proveer entradas para el servicio de monitoreo de red.

La comunicación de SNMP consiste en diferentes tipos de intercambio de mensajes entre las estaciones de manejo del protocolo y los dispositivos de red ejecutados, que son comúnmente conocidos como agentes de *software*. Tanto la manera de manejar estos mensajes como el mecanismo de identificación detrás de su manejo son vulnerabilidades significativamente explotables.

El mecanismo inherente de autentificación de los viejos sistemas SNMP posee una vulnerabilidad significativa. Las versiones 1 y 2 de SNMP usan una "cadena comunitaria" sin encriptar como su único medio de autentificación. Esta cadena no encriptada es una falla de seguridad, Ya que en la mayoría de los dispositivos SNMP la cadena es "*public*", y para el manejo de información más sensible la cadena es "*private*". Los atacantes usan esta vulnerabilidad para reconfigurar y apagar los dispositivos remotamente. El husmeo de tráfico SNMP puede revelar gran información acerca de la estructura de la red así como los sistemas y dispositivos que son parte de la misma.

La mayoría de los desarrolladores habilitan por defecto la versión 1 de SNMP, y muchos otros no ofrecen productos con capacidad de uso para la versión 3, la cual tiene un modelo de seguridad más desarrollado y más configurable que las versiones anteriores.

Como protegerse

Deshabilitar SNMP si no es requerido en el sistema; usar la versión 3 del SNMP; asegurarse de correr la versión más actual con sus respectivos parches; bloquear el trafico SNMP (puertos TCP y UDP 161 y 162) en los puntos de ingreso a la red a menos que sea

absolutamente necesario manejar los dispositivos externamente (entonces filtrarlo); emplear controles de acceso basados en anfitrión en los agentes SNMP; cambiar la cadena comunitaria usando una política de contraseñas fuertes.

7. Capa abierta de *sockets* seguros (SSL)

La biblioteca de código abierto OpenSSL es un paquete popular que agrega seguridad de encriptación a las aplicaciones que se comunican con otras redes. Aún cuando Apache es la aplicación más conocida que soporta SSL (HTTPS: conectado al puerto TCP 443), muchos otros programas o servicios han sido modificados para usar la seguridad de OpenSSL. Como por ejemplo POP3, IMAP, SMTP y LDAP.

Desde que la librería OpenSSL ha sido integrada en algunas aplicaciones, cualquier vulnerabilidad en la librería puede ser explotada a través de esta aplicación. Por ejemplo, algunos *exploits* pueden comprometer servidores ejecutando Apache, que usa ciertas versiones de la biblioteca, y estos pueden ser modificados fácilmente para comprometer aplicaciones que usen la misma biblioteca por ejemplo: Sendmail, CUPS, OpenLDAP, etc.

Algunas vulnerabilidades pueden ser explotadas remotamente para ejecutar código arbitrario en el nivel de privilegio que se encuentre la aplicación que usa la biblioteca OpenSSL. En muchos casos como Sendmail, una explotación exitosa puede obtener privilegios de *root*.

Como protegerse

Actualizar el sistema a la versión más reciente de OpenSSL, si la biblioteca estaba preinstalada con el sistema, instalar el parche más reciente; si es posible, considerar filtros de IP y de red u otras herramientas de cortafuego para el ingreso de conexiones al servidor que usa las bibliotecas.

8. Mala Configuración de los Servicios en Demanda NIS/NFS

El Sistema de Archivos de Red (NFS, por sus siglas en inglés) y el Servicio de Información de Red (NIS, por sus siglas en inglés) son dos servicios importantes usados en redes y sistemas UNIX. NFS es un servicio originalmente creado por Sun Microsystems que fue diseñado para compartir archivos entre sistemas UNIX a través de la red. NIS es también un paquete de servicios que trabaja como un servicio de base de datos que provee información de localización, llamado *Maps*, hacia otros servicios de red como NFS. Los ejemplos más comunes de *Maps* son los archivos de contraseñas y grupos que son usados para centralizar la información.

Los problemas de seguridad con ambos servicios, están representados por los continuos problemas descubiertos a través de los años (sobrecargas de *buffer*, negaciones de servicio y autentificaciones débiles), que los hacen blancos frecuentes de ataques. Además de los servicios no parchados que están ampliamente desplegados. Los riesgos más elevados pueden ser representados por las malas configuraciones de NFS y NIS que fácilmente permitirán explotar y acceder local y remotamente a agujeros de seguridad.

La pobre autentificación ofrecida por las peticiones NIS permite a los usuarios desplegar los valores de la base de datos NIS por medio de aplicaciones como `ypcat` y `getent`, y restablecer el archivo de contraseñas por medio de aplicaciones como *map*. El mismo tipo de problemas ocurre con NFS que confía implícitamente en el identificador del usuario (UID) y el identificador del grupo (GID) que el cliente NFS presenta al servidor, y dependiendo de la configuración del mismo, este podría permitir a cualquier usuario el montar y explorar el sistema de archivos remoto.

Como protegerse

En cada cliente listar explícitamente la lista de servidores NIS posibles; asegurar que el servidor responda sólo a peticiones en puertos privilegiados; incluir los anfitriones confiables en la lista de anfitriones de los procesos `ypserv` y `ypxfrd`; en los clientes NIS agregar la entrada `+:*:0:0:::` en el archivo de contraseñas; considerar el uso de

NIS a través de un protocolo seguro como SSH (o usar LDAP en vez de NIS); usar direcciones numéricas o nombres completos de dominio en vez de alias en la lista de máquinas permitidas; usar el archivo /etc/exports y sus directivas para restringir el acceso al sistema de archivos NFS; revisar la configuración (se puede usar la herramienta NFSBug); repasar las políticas del cortafuegos y asegurarse de bloquear todos los puertos innecesarios, como podría ser el puerto TCP UDP 111 (*portmap*) y el puerto TCP UDP 2049 (*Rpc.nfsd*), permitir el acceso a los servidores NIS y NFS solo de clientes autorizados; considerar el uso de NFS a través de un protocolo seguro (SSH); aplicar todos los parches y actualizaciones de las versiones más recientes; deshabilitar los demonios NFS y NIS si el sistema no esta diseñado y configurado específicamente para ser un servidor de estos sistemas.

9. Bases de datos

Las bases de datos son elementos de sistemas electrónicos como: comercios, finanzas, bancos, etc., que contienen información crítica, de compañeros, empleados, clientes, entre otros. Aún con la importancia que requiere la integridad y confidencialidad de los datos, los Sistemas Manejadores de Bases de Datos (DBMS, por sus siglas en inglés) típicamente no han sido sujetos al mismo nivel de seguridad que las redes o los sistemas operativos. Los DBMS son colecciones de programas que guardan modifican y extraen la información de la base de datos.

La integridad y confidencialidad de los datos puede ser comprometida por varios factores, incluyendo la complejidad de la implementación, uso de contraseñas inseguras, pobres configuraciones, códigos de aplicaciones pobremente escritos, etc. Los DBMS guardan todo tipo de información: gubernamental, registros médicos, datos financieros sensibles, números de tarjeta de crédito y muchos más.

Las bases de datos son aplicaciones extremadamente complejas y, la mayoría de las veces es difícil configurarlas y asegurarlas adecuadamente. Las aplicaciones de bases de datos (como MySQL, Oracle y PostgreSQL) incluyen muchas características como: sistemas de auditorias de contraseñas, cuentas de usuarios, comandos propios, lenguajes y *scripts*

únicos, protocolos de red, parches, etc. Sin embargo, muchos administradores manejan las bases de datos usando sólo la parte básica y no consideran las funciones más complejas de la misma. Como resultado se tienen vulnerabilidades y configuraciones pobres que no son revisadas, ni detectadas. La mayoría de las bases de datos tienen un amplio repertorio de características que pueden ser explotadas logrando comprometer la confidencialidad, disponibilidad e integridad de los datos contenidos en ella.

Todos los sistemas de bases de datos relacionales modernos son direccionados por un puerto TCP (Oracle: 1521, MySQL: 3306, PostgreSQL: 5432), lo que significa que cualquier persona con una herramienta para procesar peticiones puede intentar conectarse con la base de datos directamente, traspasando los mecanismos de seguridad del sistema operativo. La mayoría de las aplicaciones de bases de datos tienen cuentas por defecto y contraseñas ampliamente conocidas, que proveen varios niveles de acceso a las tablas y los recursos de la base de datos. Hoy en día las bases de datos están íntimamente ligadas a aplicaciones, como paginas *Web*. Si esta aplicación está pobremente escrita y/o configurada, puede permitir a un atacante conducir un ataque del código SQL o explotar alguna vulnerabilidad de la base de datos.

Como protegerse

Primero es esencial el asegurar que la aplicación de bases de datos esta debidamente parchada con la versión más actual; usar el mínimo de privilegios posible; borrar o cambiar las cuentas y contraseñas por defecto del sistema de bases de datos; usar los procedimientos ya cargados cuando sea posible; deshabilitar o borrar los procedimientos ya cargados que no son necesarios; establecer límites máximos para el tamaño de los campos en las formas; validar todos los datos en el lado del servidor (tamaño, formato y tipo).

10.*Kernel*

El componente central de los sistemas operativos es el *kernel*. El *kernel* es responsable de un número de interacciones de bajo nivel entre el sistema operativo y el *hardware*, la memoria, la comunicación entre procesos, el sistema de archivos y muchos otros

servicios. Como el *kernel* tiene accesos privilegiados en todos los aspectos del sistema, una vulnerabilidad a nivel de *kernel* puede ser devastadora. Los riesgos de vulnerabilidades del *kernel* incluyen negaciones de servicios, ejecución de código arbitrario con privilegios de sistema, acceso no restringido al sistema de archivos o acceso a nivel de *root*. Muchas vulnerabilidades son explotadas remotamente, y son especialmente peligrosas cuando la vía del ataque es provista por un servicio de Internet. En algunos casos, enviando un paquete ICMP mal ensamblado, el *kernel* puede quedar atrapado en un ciclo, consumiendo todos los recursos del CPU y dejando inútil la máquina, causando una negación de servicio. La afinación propia del *kernel*, no sólo protege el sistema de ataques, sino que también mejorara el rendimiento del sistema.

Como protegerse

Primero debemos afinar, o entonar, los recursos del sistema restringiendo los ataques de negación de servicio y sobrecargas de *buffer*; fortalecer la configuración de las características de red evitando los ataques por medio de la misma; es recomendable que todas las modificaciones sean revisadas totalmente antes que la implementación sea puesta en un ambiente hostil y de producción.

V.2.2 – Vulnerabilidades en Windows

11. Servidores W*eb* y servicios

Las instalaciones por defecto de varios servidores del Protocolo de Transferencia de Hipertexto (HTTP) y sus componentes adicionales para el servicio de peticiones, como el flujo de datos multimedia de plataformas Windows hacia Internet han probado ser vulnerables a un número de ataques serios a través del tiempo. El impacto de estas vulnerabilidades puede incluir:

- Negaciones de servicio.
- Exposición de archivos sensibles.
- Ejecución de comandos arbitrarios en el servidor.

- Servidores comprometidos completamente.

El Servicio de Información de Internet (IIS) usa un "disparador" de programación conocido como ISAPI (Interfase de Programa de Aplicación del Servidor de Internet) para asociar archivos que tengan ciertas extensiones con DLLs (conocidos como filtros ISAPI). Preprocesadores como ColdFusion y PHP usan ISAPI. IIS incluye muchos filtros ISAPI para manejar funciones como el ASP (Páginas de Servidor Activo), servicios W*eb* .Net e impresoras compartidas basadas en el *Web*. Muchos filtros ISAPI instalados con el IIS por defecto no son requeridos en la mayoría de las instalaciones, y muchos de estos filtros son explotables. Ejemplos de programas maliciosos que usan este tipo de mecanismo de propagación incluyen el bien conocido gusano *Code Red* y *Code Red 2*. Como muchos servidores de Web, IIS incluye muestras de aplicación que fueron diseñadas para demostrar la funcionalidad del servidor Web. Estas aplicaciones no fueron diseñadas para operar en forma segura en un ambiente de producción. Algunas muestras de aplicaciones de IIS han permitido inspección remota y sobre escritura de archivos arbitrarios así como acceso remoto a otra información sensible del servidor, incluyendo la contraseña del administrador.

Como protegerse

Actualizarse a la versión más reciente del IIS, la cual ofrece un incremento dramático de seguridad. Parchar el servidor en la instalación es necesario, pero no suficiente. Cuando sea descubierta una nueva vulnerabilidad debe ser rápidamente parchada.

12. Servicios de Estación de Trabajo (*Workstation*)

Los servicios de estación de trabajo de Windows son responsables del procesamiento de las peticiones del usuario para acceder a recursos como archivos e impresoras. El servicio determina si el recurso reside en el sistema local o en una red compartida, y encamina la petición del usuario apropiadamente. Las funciones del manejo de red provistas por el servicio pueden ser invocadas por cualquiera de los siguientes mecanismos:

- Llamadas a través del protocolo de Servidor de Bloques de Mensajes (SMB).
- Llamadas directas a través de un puerto UDP. (> 1024).
- Llamadas directas a través de un puerto TCP. (> 1024).

Los servicios contienen un desbordamiento de *buffer* basado en la pila que puede ser activado por una llamada preparada. El problema se origina por que los parámetros son pasados a la función ingresada sin analizar los límites. Este desbordamiento puede ser explotado por un atacante no autentificado y ejecutar código arbitrario en la máquina Windows vulnerable con privilegios del sistema ("*system*"). El atacante puede obtener control total de la máquina comprometida.

Como protegerse

Asegurarse que el sistema Windows tenga los parches de seguridad más actuales y tenga los paquetes de servicios (*service packs)* instalados. Bloquear los puertos 139 (TCP) y 445 (TCP) en el perímetro de la red. Abrir solo los puertos necesarios (TCP) mayores de 1024 en el perímetro de la red. Usar un filtro TCP/IP. Si el sistema no pertenece a un ambiente de una red Windows, los servicios de estación de trabajo pueden ser deshabilitados, teniendo precaución en los efectos que esto puede tener con otras aplicaciones y con la funcionalidad del sistema.

13. Servicios de Acceso Remoto de Windows

La familia de Plataformas Operativas de Windows soporta una variedad de métodos y tecnologías para redes. Hay soporte nativo para la mayoría de la industria estándar de protocolos de red y funcionalidades hechas y diseñadas para muchos métodos y técnicas de redes específicas de Microsoft. Entradas comunes para la explotación incluyen redes compartidas, sesiones anónimas, acceso de registro remoto y llamadas de procedimientos remotos, los cuales se describen a continuación.

Redes compartidas

Windows provee la habilidad de compartir archivos o carpetas a través de una red con otros anfitriones por medio de redes compartidas de Windows. Los mecanismos principales de esta habilidad son el protocolo de Servidor de Bloques de Mensaje (SMB) y el Sistema de Archivos Comunes de Internet (CIFS). Estos protocolos permiten a un anfitrión manipular archivos remotos como si fueran locales. Aunque esta es una herramienta poderosa y útil de Windows, una configuración inapropiada de la red compartida puede exponer archivos críticos del sistema o puede proveer un mecanismo para que usuarios o programas dañinos puedan tomar un control total del anfitrión. Muchos usuarios de computadoras abren su sistema a ataques sin saberlo, cuando tratan de hacerlo más conveniente para sus compañeros e investigadores externos haciendo sus dispositivos libres para escribir y borrar a usuarios de la red. Cuando esto se realiza adecuadamente, el riesgo de comprometer el sistema puede ser mitigado.

Sesiones anónimas – *(Anonymous Logon)*

Una sesión anónima es una sesión establecida sin credenciales (nombre de usuario y contraseña en blanco). Las sesiones anónimas pueden ser usadas para desplegar información acerca de usuarios, grupos, recursos compartidos y políticas de contraseñas. Los servicios de Microsoft Windows NT corriendo con la cuenta del Sistema Local en la computadora local se comunican con otros servicios a través de la red para el establecimiento de sesiones anónimas. Windows 2000 y los servicios más actuales corriendo con la cuenta del Sistema Local en la computadora local usan esta cuenta para autentificación de otros servidores.

Acceso Remoto al Registro – *Remote Registry Access*

Microsoft Windows 9x, Windows CE, Windows NT, Windows 2000, Windows ME y Windows XP emplean una base de datos central jerárquica, conocida como el Registro, para manejo de *software*, configuraciones de dispositivos y características de usuarios. Permisos incorrectos o ambientes de seguridad pueden permitir acceso remoto de registros. Los atacantes pueden explotar esta característica para comprometer el sistema o formar las bases para la asociación de permisos de archivos ya adaptados para permitir código malicioso.

Llamadas a Procedimientos Remotos - *Remote Procedure Calls*

Todas las versiones de los sistemas operativos de Microsoft (Windows NT 4.0, 2000, XP y 2003) proveen un mecanismo de comunicación entre procesos, que permite a un programa corriendo en un anfitrión ejecutar código en un anfitrión remoto. Tres vulnerabilidades han sido publicadas que permitirán a un atacante correr código arbitrario en anfitriones susceptibles con privilegios del Sistema Local. Una de estas vulnerabilidades fue explotada por los gusanos *Blaster/MSblast/LovScan y Nachi/Welchia*. Hay también otras vulnerabilidades que permitirían a atacantes realizar ataques de Negación de Servicio en contra de componentes RPC.

Como protegerse

Microsoft trata las vulnerabilidades de seguridad en sus Paquetes de Servicios (*Service Packs*), reparaciones (*hotfixes*) y aplicaciones. Es extremadamente importante tener el paquete más actual instalado en el sistema. Si el sistema no necesita proveer servidores de archivos ni de impresión y no necesita ser administrado remotamente (la mayoría de las estaciones de trabajo caben en esta categoría), el servicio `server` puede ser deshabilitado. Si el servicio antes mencionado debe estar habilitado necesariamente es recomendado seguir los siguientes pasos:

1. Enumerar todos los recursos compartidos (*shares*) por defecto que están escondidos en el sistema.
2. Borrar estos recursos por defecto (en caso de que no se ocupen).
3. Para asegurarse que no se vuelvan a cargar estos recursos debemos modificar el registro de Windows (recordar hacer un respaldo antes de modificarlo).

El acceso al registro de Windows debe ser restringido. Debemos deshabilitar o restringir la funcionalidad de las Llamadas a Procedimientos Remotos (RPC). Pero debemos tener en cuenta que al hacer esto se puede causar que algunos servicios de Windows no funcionen correctamente, por lo tanto se debe hacer con mucho cuidado y analizar el sistema en un ambiente controlado, antes de colocarlo en un ambiente de trabajo. Además

podemos bloquear los puertos asociados a RPC en el perímetro de la red (TCP: 135, 139, 445 y 593; UDP: 135, 137, 138 y 445).

14. Servidor de SQL de Microsoft (MSSQL)

El MSSQL contiene varias vulnerabilidades serias que permiten a atacantes remotos obtener información sensible, alterar el contenido de la base de datos, comprometer el servidor SQL, y, en algunas configuraciones, comprometer el servidor anfitrión.

Las vulnerabilidades del MSSQL son publicadas y están continuamente bajo ataque. Dos gusanos de MSSQL en Mayo del 2002 y Enero del 2003 explotaron varias fallas conocidas del servidor. Los anfitriones comprometidos por estos gusanos generaron un nivel de daños en el tráfico de la red de trabajo cuando fueron preparados para escanear otros servidores vulnerables.

SQL Snake

La rutina del *exploit* depende de la cuenta por defecto del administrador, o llamada cuenta "*sa*", teniendo una contraseña nula. Es esencial analizar la configuración y restricción de cualquier sistema para asegurar que todas las cuentas del sistema tengan una contraseña de protección, o estén completamente desactivadas si no se usarán. La cuenta "*sa*", debe tener una contraseña compleja, difícil de adivinar aunque no sea usada para ejecutar la implementación SQL/MSDE.

SQL Slammer

Los puertos 1433 y 1434, que son los puertos por defecto del servidor y monitor de MSSQL, han sido regularmente registrados como dos de los puertos más escaneados frecuentemente. La rutina de este *exploit* esta basada en una sobrecarga de *buffer* en el Servicio de Resolución de Servidor SQL. Esta sobrecarga de *buffer* es direccionada, y la seguridad del anfitrión es comprometida cuando el gusano manda paquetes de ataque alterados al puerto UDP 1434 del sistema vulnerable elegido. Si la máquina corre los servicios de SQL que son sujetos a esta sobrecarga de *buffer* de la pila y recibe paquetes de esta naturaleza, usualmente resultará en un servidor y un sistema de seguridad totalmente comprometido. El medio de defensa más eficaz es el parchado del servicio,

prácticas pro-activas de configuración del sistema y el filtrado del puerto UDP 1434. El Motor del Escritorio del Servidor de Microsoft 2000 (MSDE 2000, por sus siglas en inglés) puede ser empleado como un "Servidor Ligero de SQL". Muchos administradores de sistemas no se dan cuenta que su sistema esta corriendo el MSDE 2000 y tienen una versión del servidor de SQL instalada. El MSDE 2000 se instala como parte de los siguientes productos de Microsoft: SQL/MSDE Server 2000 (ediciones *Developer*, *Standard* y *Enterprise*), Visual Studio .NET (ediciones *Architect*, *Developer* y *Proffesional*), ASP.NET *Web Matrix Tool*, Office XP, Access 2002 y Visual Fox Pro 7.0/8.0.

Como protegerse

Deshabilitar el servicio del Monitor SQL/MSDE en el puerto UDP 1434 (esto puede interferir con los servicios de administración remota y respaldos); aplicar los *Service Packs* más recientes para el SQL/MSDE y/o el MSDE 2000; aplicar el parche acumulativo correspondiente al paquete de servicios anterior; aplicar los parches individuales correspondientes; habilitar el Servidor de Autentificación SQL (*Server Authentication Logging*); asegurar el servidor del sistema a nivel de red; minimizar los privilegios del Servidor MSSQL/MSDE y del Agente SQL/MSDE.

15. Identificación en Windows

Contraseñas, parafrases y códigos de seguridad son usados virtualmente en cada interacción entre los usuarios y los sistemas de información. La mayoría de las formas de autentificación de usuario, así como archivos y datos protegidos, dependen de contraseñas provistas por el usuario. Ya que el acceso apropiadamente autentificado es frecuentemente no registrado, o incluso cuando el registro no parece sospechoso, una contraseña comprometida es una oportunidad para explorar el sistema virtualmente desde adentro sin ser detectado. Un atacante tendrá acceso completo a cualquier recurso disponible para el usuario, y estará significativamente cerca de ser capaz de acceder a otras cuentas, máquinas cercanas, y tal vez hasta privilegios de administrador. A pesar de esta amenaza, las cuentas sin o con malas contraseñas son extremadamente comunes, y

las organizaciones con buenas políticas de contraseñas son escasas. Las vulnerabilidades más comunes de las contraseñas son:

- Las cuentas de usuarios tienen una contraseña débil o no cuentan con una.
- Indiferentes a una contraseña fuerte, los usuarios fallan en protegerse.
- El sistema operativo o aplicaciones terceras crean cuentas con contraseñas débiles o sin ellas.
- En muchas aplicaciones (tanto de software propietario como de Cogido Abierto), los algoritmos de contraseñas ya son conocidos y estos *hash* frecuentemente son guardados donde pueden ser accedidos por usuarios regulares.

Microsoft Windows no guarda o trasmite contraseñas en texto claro –usa un *hash* de la contraseña para autentificarse-. Un *hash* es el resultado obtenido de aplicarle una función matemática (el algoritmo *hash*) a una cantidad arbitraria de datos. Windows cuenta con tres algoritmos de autentificación: LM (menos seguro, más compatible); NTLM y NTLMv2 (más seguro, menos compatible). Windows guarda las contraseñas localmente usando el algoritmo LM por defecto en las versiones NT, 2000 y XP. Como LM tiene una encriptación más débil que NTLM y NTLMv2, las contraseñas pueden ser descifradas en un periodo relativamente corto de tiempo. Incluso las contraseñas consideradas fuertes pueden ser descifradas por fuerza bruta en menos de una semana con el hardware actual.

Las debilidades del *hash* LM se derivan de:

- Las contraseñas son truncadas a 14 caracteres.
- Las contraseñas son rellenadas con espacios para completar los 14 caracteres.
- Las contraseñas son convertidas a solo mayúsculas.
- Las contraseñas son divididas en dos bloques de 7 caracteres.

Este proceso de hash significa que el atacante solo necesita completar la tarea trivial de romper dos bloques de siete caracteres, que estarán solo en mayúsculas para obtener acceso al sistema. En vista de que la complejidad de romper algoritmos hash incrementa geométricamente con el largo del *hash*, cada bloque de siete caracteres es al menos un orden de magnitud simple para un ataque de fuerza bruta que será una combinación de un bloque de catorce caracteres. En vista de que las cadenas o bloques son de exactamente siete caracteres (incluyendo espacios) y completamente en mayúsculas, un ataque de tipo diccionario se simplifica. El método hash de LM por consiguiente esta completamente bajo el nivel de las políticas de buenas contraseñas.

Como protegerse

La forma más apropiada de defensa en contra de las contraseñas débiles es una buena política de contraseñas, que incluya instrucciones para crear buenos hábitos en la selección de las mismas. Como por ejemplo: Asegurarse que las contraseñas sean consistentemente fuertes (tamaño mínimo de contraseñas, vencimiento, etc.); indicar a los usuarios que nunca le digan su contraseña a otra persona; prevenir que las bases de datos de las contraseñas y los algoritmos *hash* no puedan ser copiados; control hermético de las cuentas del sistema; mantener la política de contraseñas fuertes; deshabilitar autentificación de *hash* LM a través de la red; prevenir que la base de datos del *hash* sea guardada.

16. Navegadores *Web*

El navegador es el medio por el cual las computadoras accedan al W*eb* en los sistemas Microsoft Windows. El navegador dominante hasta la fecha es el Explorador de Internet (IE) de Microsoft, que es el navegador por defecto en las plataformas Windows. Los problemas principales en el IE 6.0 son:

1. Mayor número de vulnerabilidades a través de los últimos años en comparación con los otros navegadores. 153 vulnerabilidades desde Abril del 2001.

2. Mayor tiempo para parchar las vulnerabilidades conocidas del IE. Los usuarios deben esperar hasta 6 meses a partir del tiempo que la vulnerabilidad es divulgada hasta que Microsoft pone en circulación el parche.
3. A los controles ActiveX y Active Scripting por si solos no se les ha encontrado una fuente abierta para alguna explotación, pero pueden ser usados para brincar la seguridad de los constructores de seguridad del navegador y tener un impacto potencial una vez, en el sistema local.
4. Mayor número de vulnerabilidades sin parchar. 34.
5. Vulnerabilidades de *Spyware/Adware*. Esto afecta todos los navegadores y sistemas que facilitan el acceso y uso de los recursos del W*eb*.
6. Integración del IE en el sistema operativo. Lo cual hace al sistema más vulnerable a la explotación.

Todos los navegadores tienen vulnerabilidades críticas si no son mantenidas al día con los parches actuales en circulación. Un diseñador malicioso del *Web* puede crear páginas *Web* para explotar estas vulnerabilidades en el contexto de usuario del IE mientras se encuentra buscando estas páginas. Estas vulnerabilidades pueden ser clasificadas en muchas categorías incluyendo páginas *Web* o la interfase *spoofing* de Windows, el control de vulnerabilidades de ActiveX, vulnerabilidades del Active Scripting, malas interpretaciones y sobrecarga de *buffer* de tipo MIME y *Content*. Las consecuencias pueden incluir revelación de *cookies*, archivos locales o datos, ejecución de programas locales, descarga y ejecución de código arbitrario, o un completo dominio del sistema vulnerable.

Como protegerse

Para configurar las características de seguridad del navegador primero debemos actualizar el sistema operativo con los paquetes de servicio más recientes. La mayoría de las vulnerabilidades son explotadas a través del Active Scripting o ActiveX. Por lo tanto sería una buena opción desactivar ambos controles, aunque debemos tomar en cuenta que deshabilitando el Active Scripting puede causar que algunos sitios W*eb* no trabajen apropiadamente. Los Java *applets* tienen típicamente más capacidades que los *scripts* y es

una buena práctica el asegurarse que las operaciones del sistema y de mantenimiento toman los privilegios de una cuenta que no sea la cuenta del administrador. Debemos asegurarnos también que sitios "inseguros", no estén en la zona de sitios "seguros" ni en la zona de Intranet local, ya que estas zonas tienen características más débiles de seguridad.

17. Archivos Compartidos Punto a Punto de Windows (P2P)

Los programas de compartimiento de archivos punto a punto (*peer to peer* –P2P-) son usados por una base de usuarios que crece rápidamente. Estas aplicaciones son usadas para la descarga y distribución de diferentes tipos de datos, como música, video, graficas, texto, código fuente e información propietaria por mencionar algunos. Las aplicaciones P2P tienen un número de usos legítimos, incluyendo la distribución de binarios de Código Abierto/GPL, imágenes ISO de distribuciones de Linux, creaciones de artistas independientes y hasta medios comerciales como avances de películas y demostraciones de juegos. Con la experiencia de los problemas legales de Napster, la mayoría de las programas P2P ahora operan a través de redes o clientes distribuidos, compartiendo archivos, directorios o incluso discos duros llenos de datos. Los clientes participan descargando archivos de otros usuarios, haciendo sus datos disponibles a otros usuarios, y en algunos modelos funcionando como súper-nodos que pueden coordinar búsquedas para múltiples usuarios.

Existe un número de vulnerabilidades cuando se usan aplicaciones P2P. Pueden ser reconocidas en tres tipos: vulnerabilidades técnicas, que son aquellas que pueden ser explotadas remotamente; vulnerabilidades sociales, que son aquellas que son explotadas al alterar o enmascarar contenido binario de una solicitud; y vulnerabilidades legales, que son aquellas que pueden resultar de la infracción de las marcas registradas o del material censurable.

Como se mencionó, las vulnerabilidades técnicas son aquellas que pueden ser explotadas remotamente y resultan simplemente de un usuario descargando, instalando y corriendo uno de estos programas, el rango puede ser desde negación de servicio hasta acceso de

arbitrario de archivos, y debe ser tomada muy seriamente. Una preocupación seria son los problemas de privacidad y confidencialidad que puede causar. Muchas de estas aplicaciones incluyen componentes *spyware* o *adware*, que pueden consumir hasta más ancho de banda que los hábitos de un usuario del Web. Una pobre configuración del cliente P2P puede proveer acceso sin autentificación hacia una red entera por el compartimiento de los manejadores de mapeo a través de la aplicación P2P. Hay poco para no restringir en el tipo de archivos de datos que pueden ser compartidos. Comprometimiento de información confidencial, de propiedad intelectual y de otros datos.

Las vulnerabilidades sociales existen cuando un archivo malicioso o previamente infectado o alterado es parecido a un archivo deseado por otro usuario. Pueden resultar virus, caballos de Troya, gusanos, etc. Las victimas de dichos ataques son usualmente los usuarios menos técnicos, que darán doble *clic* a un archivo sin enterarse que la extensión o el icono no esta asociado normalmente al tipo de dato, o que puede ser engañado lanzándose hacia un archivo ejecutable.

Las vulnerabilidades legales deben ser tomadas seriamente entre el usuario corporativo y el usuario casero. El contenido disponible a través de las aplicaciones P2P incluyen música, películas y archivos de aplicación propietarios.

Como protegerse

Debemos asegurarnos de tener una política en contra de la descarga de material propietario; una política de uso aceptable de la conexión a Internet; un escaneo regular del material guardado, ya sea en la red o en las estaciones de trabajo, para analizar material no autorizado; A los usuarios regulares no se les debe permitir instalar aplicaciones, específicamente P2P; considerar usar un servidor *proxy* para controlar el acceso a Internet; filtrar la salida de paquetes a cualquier puerto que no sea de propósito oficial, aunque las aplicaciones P2P pueden modificar el puerto de egreso e ingreso al número 80; monitorear la red buscando trafico P2P; utilizar software antivirus y actualizarlo al más reciente posible.

18. Falta de protección en el LSAS

El Servicio de Windows del Subsistema de Autoridad de la Seguridad Local (LSAS, por sus siglas en inglés) de las plataformas 2000, 2003, etc. Contienen una sobrecarga de *buffer* crítica que si es explotada puede llevar comprometer seriamente un sistema. Este ataque puede ser consumado remota y anónimamente a través de RPC de un sistema Windows 2000 y XP no parchado, aunque requiere privilegios de administrador para ser efectivo.

El LSAS juega un importante rol en la autentificación del sistema y en la funcionalidad del Active Directory. Es aquí en la interfase del proceso con el Active Directory que la función de ingreso de la librería LSASRV.dll puede ser sobrecargada con una cadena excesivamente larga. Potencialmente esta vulnerabilidad puede llevar a comprometer completamente el sistema.

Es fuertemente recomendable que el administrador de la red no solo aplique el parche para la vulnerabilidad, sino que implemente todos los controles de acceso a los puntos del ingreso a la red para detener los abusos basados en las llamadas RPC de Windows.

Como protegerse

Bloquear los puertos con un cortafuego, aplicar los parches correspondientes, habilitar el filtrado avanzado de TCP/IP.

19. Clientes de correo

El servicio Outlook de Microsoft, parte de la suite de Office de Microsoft, es un manejador personal de información y un cliente de correo electrónico para Microsoft. Mientras que principalmente es una aplicación de correo electrónico, también tiene un calendario, y manejo de tareas. Cuando se usa en conjunto con el Microsoft Exchange Server, el Outlook de Microsoft puede proveer funcionalidades adicionales y de mayor

valor, como es soporte de múltiples usuarios, asistente para coordinar citas y provee calendarios compartidos y bandejas de correo.

El Outlook Express (OE), es un cliente básico de correo electrónico y manejador de contactos, ha sido empaquetado con el Internet Explorer desde las primeras versiones, siendo una parte integral de todo los sistemas Windows desde la versión 95. Por la integración de productos como el Internet Explorer y el Outlook Express en otras líneas de productos, incluyendo el Office, BackOffice y el sistema operativo Windows, tecnologías comunes y código puede ser usado a través de la plataforma. Desafortunadamente, esta práctica también introduce puntos simples de fallas y aumenta el impacto de una sencilla vulnerabilidad que podría tener.

Una de las metas de Microsoft ha sido el desarrollo de una solución de un útil e intuitivo manejador de correo electrónico. Desafortunadamente, las características de la automatización ya embebidas con la disparidad de los controles de seguridad creados (frecuentemente despreciados por los usuarios finales). Esto ha guiado a la explotación, dando paso a los virus de correo electrónico, gusanos, código malicioso para comprometer el sistema local, y muchas otras formas de ataque.

Las amenazas potenciales de seguridad de los clientes de correo incluyen:

- Infección de virus y gusanos en la computadora. El código malicioso que se propaga a través de los agregados o de los *scripts* embebidos en el cuerpo del mensaje.
- *Spam*. Correo electrónico comercial no solicitado.
- *Web* guiada. Validación de direcciones de correo activadas con la apertura del mensaje por el recipiente.

Las versiones recientes del Outlook y el OE pueden ser exitosamente protegidas por los usuarios de las amenazas mencionadas, si son adecuadamente configuradas.

Como protegerse

Bloquear la recepción de agregados con extensión `.com`, `.exe`, `.vbs`, etc. En vez de enviar archivos ejecutables como agregados, podemos usar una herramienta de compresión de archivos, o una ruta alternativa para el envió del mismo (como FTP o SCP); configurar en el Outlook las directivas a) No filtrado automático (el *Spam* no es filtrado adecuadamente). b) Filtrado bajo (mueve la mayoría del correo basura a la carpeta de reciclaje y prácticamente tiene muy pocos falsos positivos). c) Filtrado alto (filtrado agresivo, casi todo el correo basura es mandado a la carpeta de reciclaje, pero es recomendable revisar la carpeta regularmente de la presencia de correo legitimo identificado como *Spam*). d) Solo listas seguras (solo se entregara correo de remitentes o dominios presentes en la Lista de Remitentes Seguros o la Lista de Recipientes Seguros. Esta es la característica más eficiente, aunque se requiere esfuerzo para llenar las listas con todas las direcciones legitimas (estas directivas solo funcionan con la versiones mas reciente del Outlook); leer todos los mensajes de correo en texto plano; no descargar dibujos u otro contenido HTML del correo automáticamente; no abrir agregados inesperados, en caso necesario de salvarlos, hacerlos en una partición diferente al resto de los datos personales; instalar un *software* antivirus y mantenerlo actualizado; mantener actualizados los clientes de correo y sus respectivos parches.

20. Mensajero instantáneo

La tecnología de los mensajeros instantáneos ha madurado en los últimos años. Mientras que aplicaciones terceras de la Mensajeria Instantánea aún tienen un largo despliegue en la misma, hay una tendencia creciente de integrar la funcionalidad de los mensajeros en el mismo sistema operativo, lo cual pose potencialmente una amenaza de seguridad directa a las organizaciones que tienen un aceptable política de uso o un sistema de operación segura que niega el uso de esta tecnología. Además el descubrimiento de vulnerabilidades en estos programas posee un riesgo significante a organizaciones que carecen de las medidas técnicas, el staff de seguridad o las capacidades para mitigar la amenaza creciente de esta tecnología embebida.

Las capacidades que los mensajeros de hoy en día (Yahoo, ICQ, Trillian, etc.) traen al escritorio son de amplio rango y proveen a los usuarios la habilidad de revisar su correo remotamente, *chatear* con voz en tiempo real, mejorar la comunicación en video y enviar y compartir archivos desde y a través de la comunicación de texto simple. Además de haber una tendencia creciente de mensajeros "multi-red" que proveen al usuario de una interfase centralizada para enviar mensajes a redes y protocolos (Trillian, etc.).

Vulnerabilidades explotables remotamente en estos programas o a sus dependencias asociadas son una amenaza creciente de la integridad y seguridad de las redes, directamente proporcional a su rápida integración y despliegue en los sistemas Windows. Los escenarios de ataque de las vulnerabilidades de los Mensajeros Instantáneos son ampliamente variados, y pueden venir en la forma de una sobrecarga de *buffer* ejecutada remotamente (basadas en RPC, malformación de paquetes), ataques basados en URI/ligas maliciosas, vulnerabilidades en la transferencias de archivos y *exploits* del ActiveX.

Las vulnerabilidades de estos programas provienen típicamente de las siguientes categorías:

- Controles ActiveX antiguos. Sobrecargas de *buffer* de DLLs, etc.
- Problemas de Implementación de URI. Ejecución de códigos maliciosos.
- Varias sobrecargas de *buffer* que resultan de transferencias de archivos. Validaciones de archivos, etc.

Estas aplicaciones no solo crean vulnerabilidades basadas en red a los sistemas, sino también riesgos de pérdidas de propiedad intelectual, pérdida potencial de confidencialidad y amenazas a la pérdida de producción de los empleados. Mientras que el mitigar las debilidades explotables remotamente en estos programas es de importancia extrema, la necesidad de una aceptable política de uso y el filtrado de tráfico es también de importancia máxima para asegurarse de evitar los problemas que el Mensajero Instantáneo puede introducir a la red.

Como protegerse

Asegurarse que cualquier *software* de mensajeria este al día con los parches de seguridad; configurar el sistema de detección de intrusos para alertar en caso de una transferencia de archivos que sea usada por uno de estos programas; en caso de poderse, bloquear los puertos desde los cuales se comunican los mensajeros; bloquear el acceso a páginas W*eb* que contengan ligas como "`aim:`", "`ymsgr:`"; bloquear el acceso a páginas W*eb* que invocan controles ActiveX asociados con cualquier problema del mensajero.

El propósito de este capítulo es que el usuario, pueda apoyarse para conocer si su sistema puede ser comprometido por medio de alguna de estas vulnerabilidades, basado en la herramienta y en la investigación hecha. Ademas de mostrarle al usuario algunos consejos de cómo protegerse de las mismas.

VI DESARROLLO DE LA HERRAMIENTA PARA EL ANÁLISIS DE VULNERABILIDADES

La plataforma seleccionada para el desarrollo de la herramienta fue Linux, más específicamente Mandrake Linux 9.1, como se conoce Linux es un sistema abierto, el cual tiene una herramienta muy poderosa para el desarrollo de aplicaciones: el lenguaje C. Es por este hecho que la herramienta se programó en Linux, este lenguaje tiene un conjunto de bibliotecas para el manejo de puertos (en este caso sería la interfaz de *sockets* –ya mencionada anteriormente-). Para facilitar el trabajo el analizador se dividió en tres etapas como se observa en la figura 7.

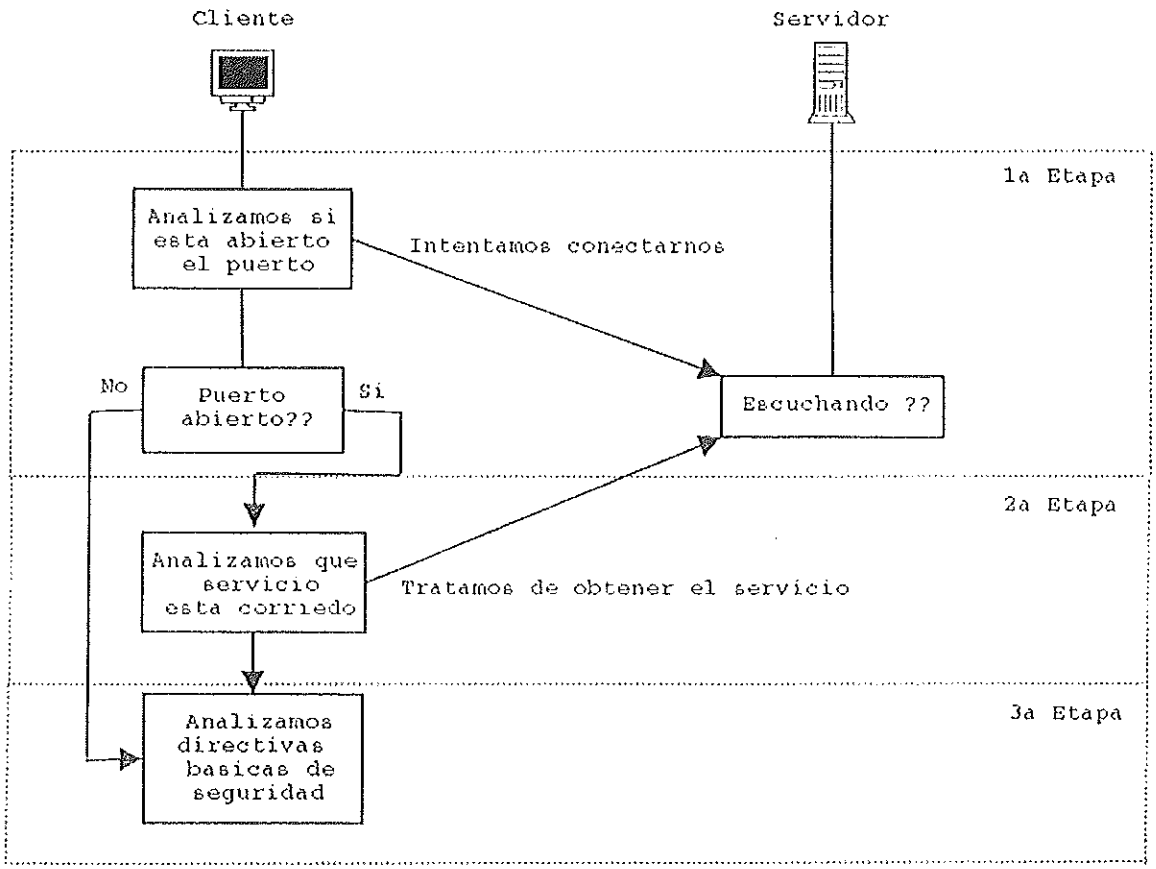


Figura 7. Diagrama de bloques del analizador.

VI.1 - 1ª etapa (Escáner de puertos TCP y UDP)

Lo primero que debemos saber acerca de nuestro sistema es conocer que puertos están abiertos en el mismo, ya que cada puerto que tenga contacto con el exterior es un potencial hoyo de seguridad. Por lo tanto la primera etapa se enfoca en el diseño de un escáner de puertos (ya sean TCP o UDP).

Cliente/Servidor TCP

Para familiarizarnos con la programación de *sockets* inicialmente se creó un cliente de conexión de tipo TCP y su respectivo servidor, este cliente lo único que hace es solicitar una petición de conexión al servidor de conexiones TCP a un puerto determinado (en este caso el puerto 9999), el servidor espera la conexión por el mismo puerto como se observa en la figura 8, después imprime los datos que se le envían en el paquete (*payload*).

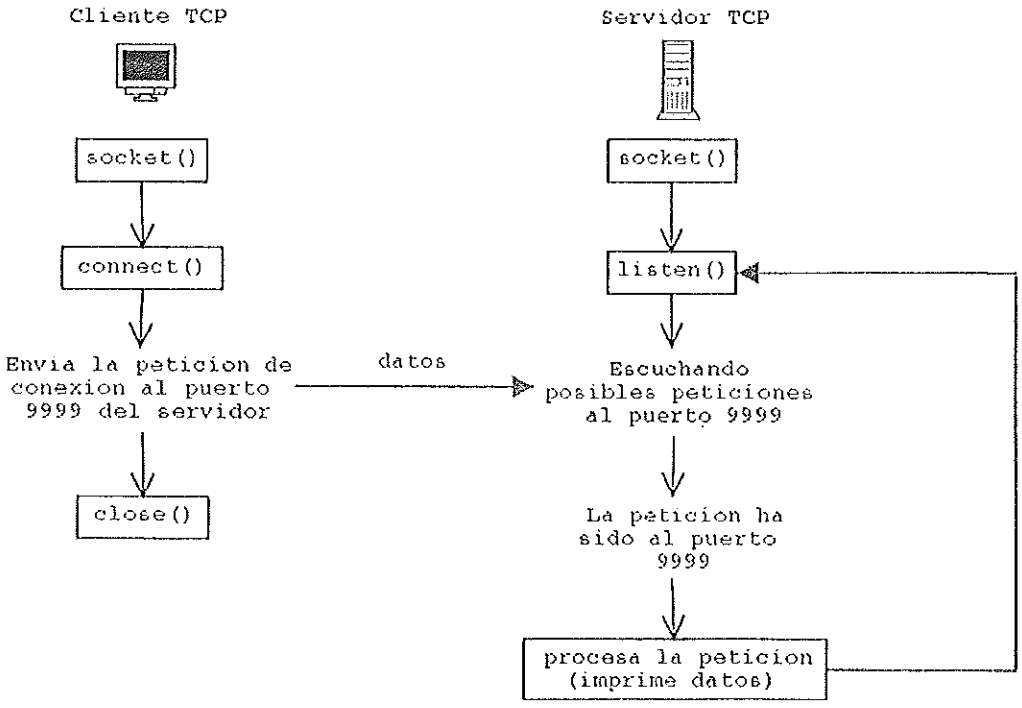


Figura 8. Comportamiento del cliente y servidor TCP.

Aquí se observó que cuando el cliente solicitaba una conexión al servidor, si el servidor estaba esperando la conexión en el puerto especificado, no había ningún problema, la conexión se llevaba a cabo y el servidor imprimía los datos del paquete, en cambio si el cliente solicitaba una conexión a un puerto donde el servidor no estuviera escuchando o esperando la conexión, el cliente no podía conectarse y generaba un código de error, como se ve en la figura 9.

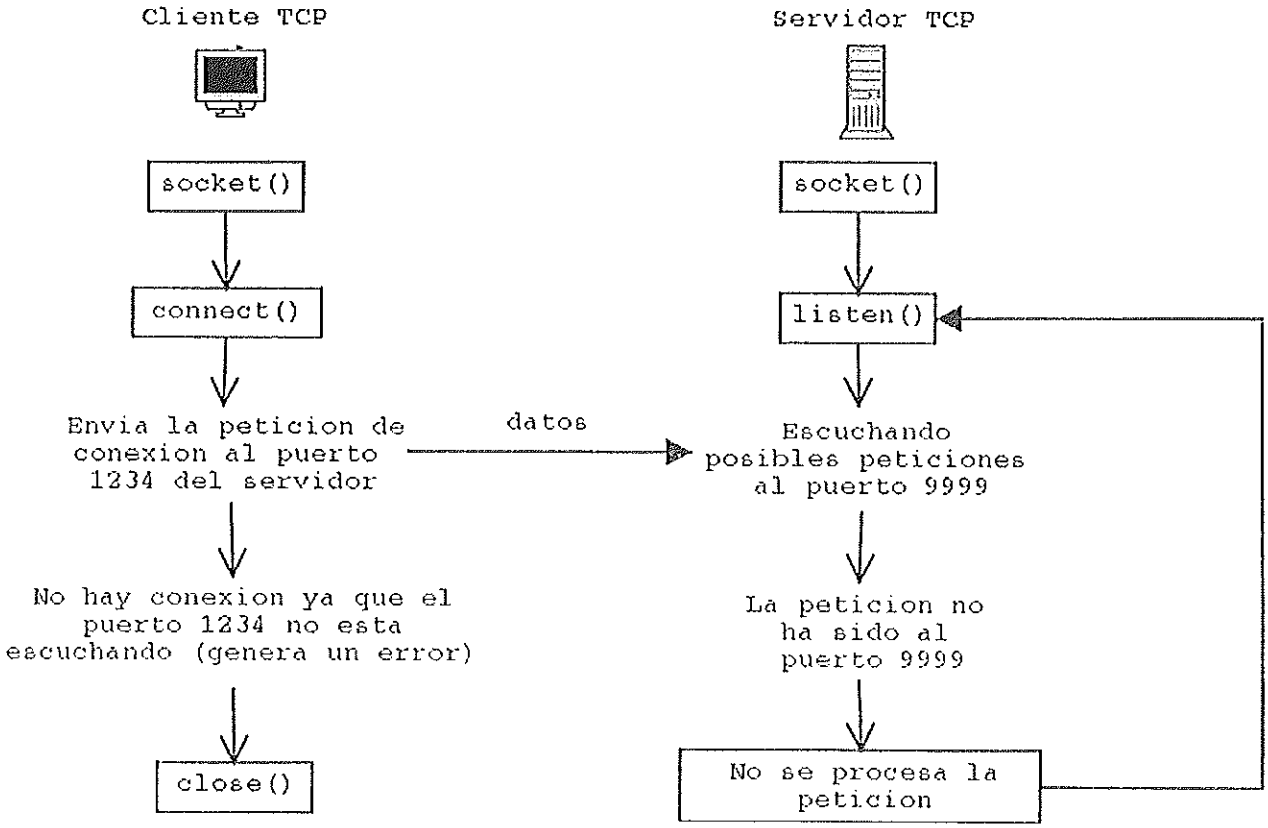


Figura 9. Comportamiento del cliente y servidor TCP en caso de error.

VI.1.1 - Escáner TCP

El comportamiento mencionado anteriormente resultó muy útil, ya que con las correspondientes modificaciones se pudo diseñar un cliente que busca en un anfitrión, y se asegurará puerto por puerto, cual de ellos está abierto. Solo fue cuestión de agregar un ciclo común para que vaya revisando desde un puerto determinado hasta otro, el resultado puede verse en la figura 10.

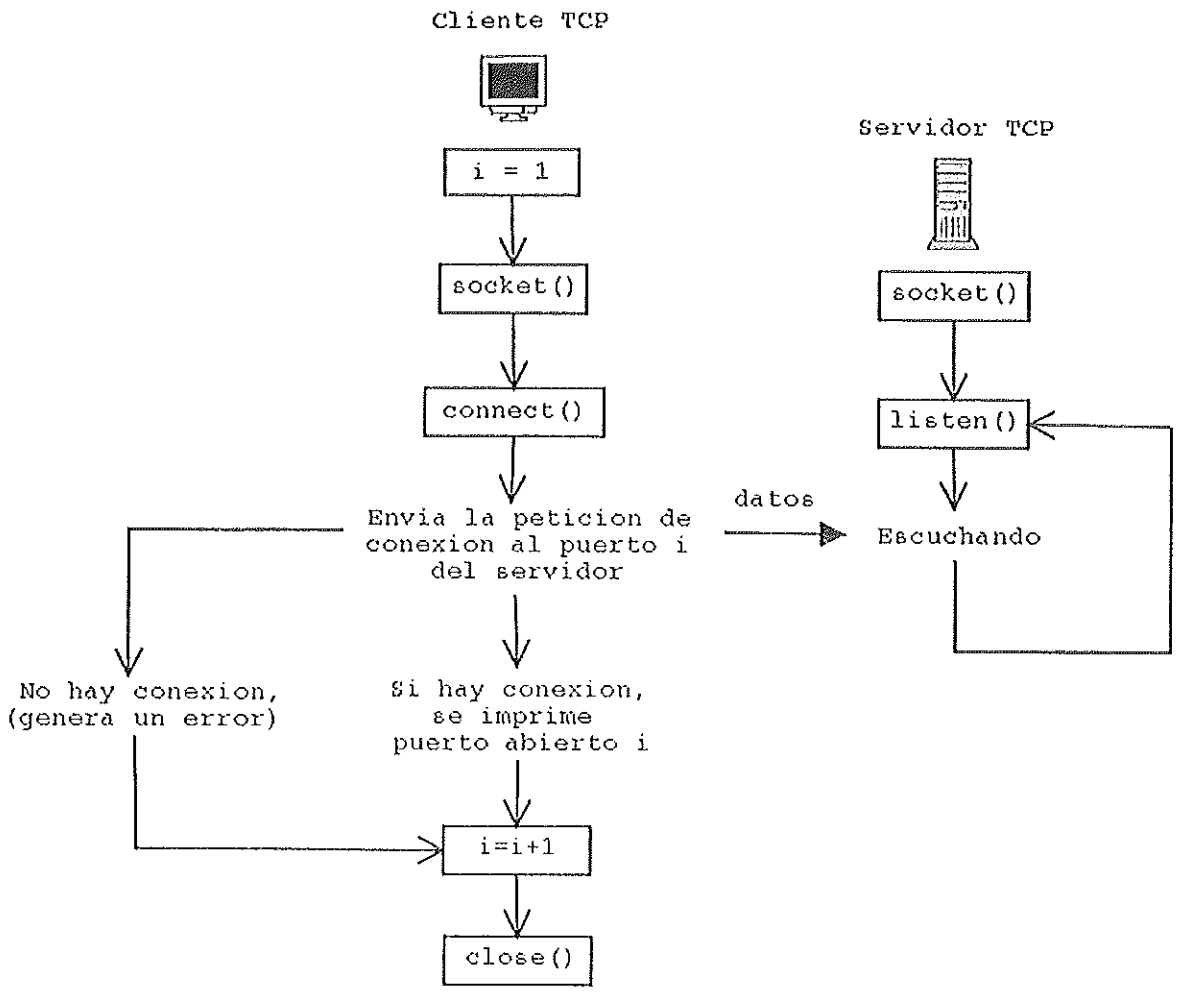


Figura 10. Diagrama general del escáner TCP.

Cliente/Servidor UDP

En un principio (igual que en TCP), se creó un cliente UDP que envía un paquete a un determinado puerto –como se observa en la figura 11-(en este caso el puerto 9999).El servidor UDP después de recibir el paquete lo único que hace es imprimir el contenido útil del paquete (*payload*).

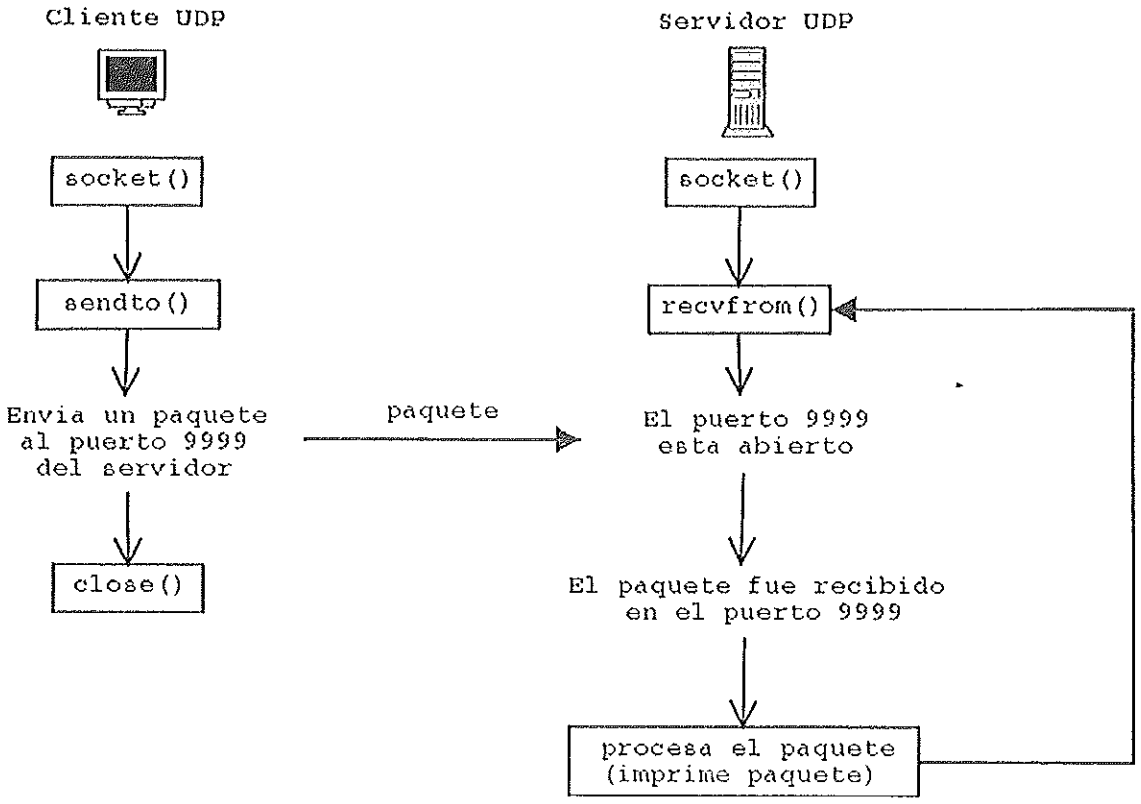


Figura 11. Comportamiento del cliente y servidor UDP.

En este caso se observó, que si el cliente envía un paquete al puerto que el servidor tiene abierto, este recibe el paquete sin ningún problema y lo imprime. Ahora, en el caso de que el cliente envíe el paquete a un puerto que el servidor no tenga abierto, el servidor regresará un paquete ICMP de destino inalcanzable (código 3), como puede observarse en la figura 12.

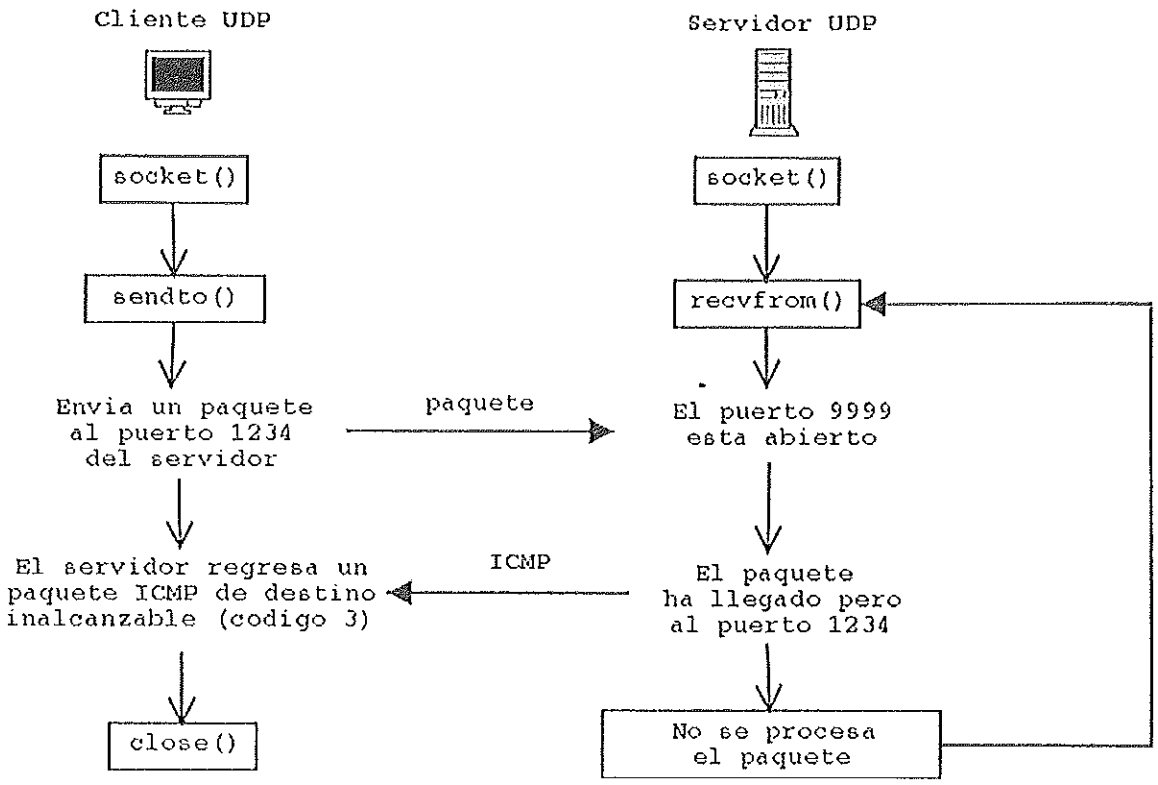


Figura 12. Comportamiento del cliente y servidor UDP en caso de error.

VI.1.2 - Escáner UDP

De igual manera que en el diseño del escáner TCP, lo que se hizo fue modificar el cliente UDP, para que el mismo nos mostrara si el paquete fue recibido o desechado. Aquí estarían unos de los problemas del escáner UDP, ya que es un protocolo no orientado a conexión.

Problemas del escáner UDP

Problema 1: puerto abierto

Al momento de diseñar el escáner de UDP, nos topamos con un problema; ya que UDP es un protocolo no orientado a conexión, no hay manera de saber si el paquete de prueba que le enviamos fue recibido o desechado al estar el puerto abierto o cerrado respectivamente. Así que se tuvo que echar mano del protocolo ICMP. Este protocolo, como fue mencionado anteriormente, se ocupa del manejo de errores de la capa de red del modelo ISO/OSI. La solución de este problema es crear dos *sockets* del lado del cliente,

uno de ellos se utilizará sólo para enviar el paquete de prueba UDP, el otro nos servirá para atrapar la respuesta del servidor después de enviarle el paquete UDP (en caso de que hubiera una respuesta).

Aquí es donde entraría el protocolo ICMP, ya que si le enviamos un paquete UDP a un puerto que el servidor tiene cerrado, el protocolo ICMP nos regresaría un mensaje de error. Así que en teoría solo es cuestión de dejar esperando el *socket* a una respuesta posible del servidor; si el servidor regresa el paquete ICMP de error, indicaría que el puerto está cerrado. La forma de dejar esperando el socket para la recepción del paquete ICMP, es utilizando la función `recvfrom`, así cuando llegue el paquete, esta función lo recibirá y se sabrá que el puerto estaba cerrado. La forma de usar el protocolo ICMP, siendo de nivel de red, fue con el uso de los llamados *sockets* básicos, vistos anteriormente, así que el *socket* de recepción utilizado en la función `recvfrom` fue de tipo básico y se usó el protocolo ICMP.

Pero, ¿Que pasaría si el puerto estuviera abierto y en consecuencia el servidor no envía el paquete de ICMP de error? .

Problema 2: Llegada del paquete ICMP

Como se menciona anteriormente, si el paquete ICMP llega, la función `recvfrom` lo recibe y la aplicación continúa su curso. Pero hay un problema, la función `recvfrom` es un función bloqueante, es decir, cuando la función es ejecutada, el flujo del programa queda suspendido esperando la respuesta de la función, esto significa que el programa se queda esperando el paquete ICMP hasta que llegue (en caso de que el servidor tenga cerrado el puerto).

Si el servidor tiene el puerto abierto, la función `recvfrom` se quedaría esperando un paquete que no llegaría, y entonces el programa queda suspendido temporalmente.

Para solucionar este problema se tuvo que echar mano de los llamados "*sockets* no bloqueantes", estos *sockets* lo que hacen (o no hacen) es quedarse esperando hasta que la función (en este caso `recvfrom`) reciba un paquete que esta esperando.

Por lo tanto, lo ideal es: crear el *socket*, hacerlo (o ponerlo) en modo no bloqueante; y así al momento de llamar la función `recvfrom`, esta se ejecutaría y, si llega el paquete ICMP, lo recibirá, así sabemos que el puerto esta cerrado y la aplicación seguirá su flujo. En caso de que el paquete no llegue; previamente llamaríamos a la funcion `recvfrom`, la pondríamos en modo no bloqueante, y así, al no recibir ningún paquete, la función simplemente se "cerrara" y el flujo del programa continuara.

Al momento de llamar la función, la cual está en modo no bloqueante, si esta no recibe ningún paquete en el mismo instante de mandarla llamar, se cierra y nos indica que el puerto está abierto. Aún si el puerto esta cerrado y manda el paquete ICMP, este no alcanza a llegar al programa cliente y la función `recvfrom` piensa que no mando nada, y por lo tanto indica que el puerto está abierto, esto recibe el nombre de **falso positivo**.

Problema 3: En espera del paquete ICMP

El lenguaje C, tiene una gran cantidad de bibliotecas, una de estas tiene una función que nos permite dejar durmiendo (*sleeping*) nuestra aplicación por algunos segundos (incluso nanosegundos). Así, usando esta función se puede lograr que la función `recvfrom` espere un tiempo al paquete (potencial) ICMP, y si este no llega en determinado tiempo, la función se "cerrara" y el flujo del programa seguirá su curso sin problema, y de esta forma estaremos evitando los **falsos positivos**.

Solución general

La solución general en resumen es:

1. Primero usar los *sockets* básicos para recibir el paquete ICMP que devolverá el servidor si el puerto esta cerrado.
2. Poner el *socket* antes mencionado en modo no bloqueante.

3. Dejar un transcurso breve de tiempo (pueden ser unos pocos nanosegundos) el *socket* no bloqueante en modo "durmiendo", para dar oportunidad al paquete ICMP de llegar (en caso de que lo haga)

Ahora solo nos resta hacer un ciclo (como en el caso del escáner TCP) para que el programa revise puerto por puerto, cual de ellos esta abierto en el servidor, como se observa en la figura 13.

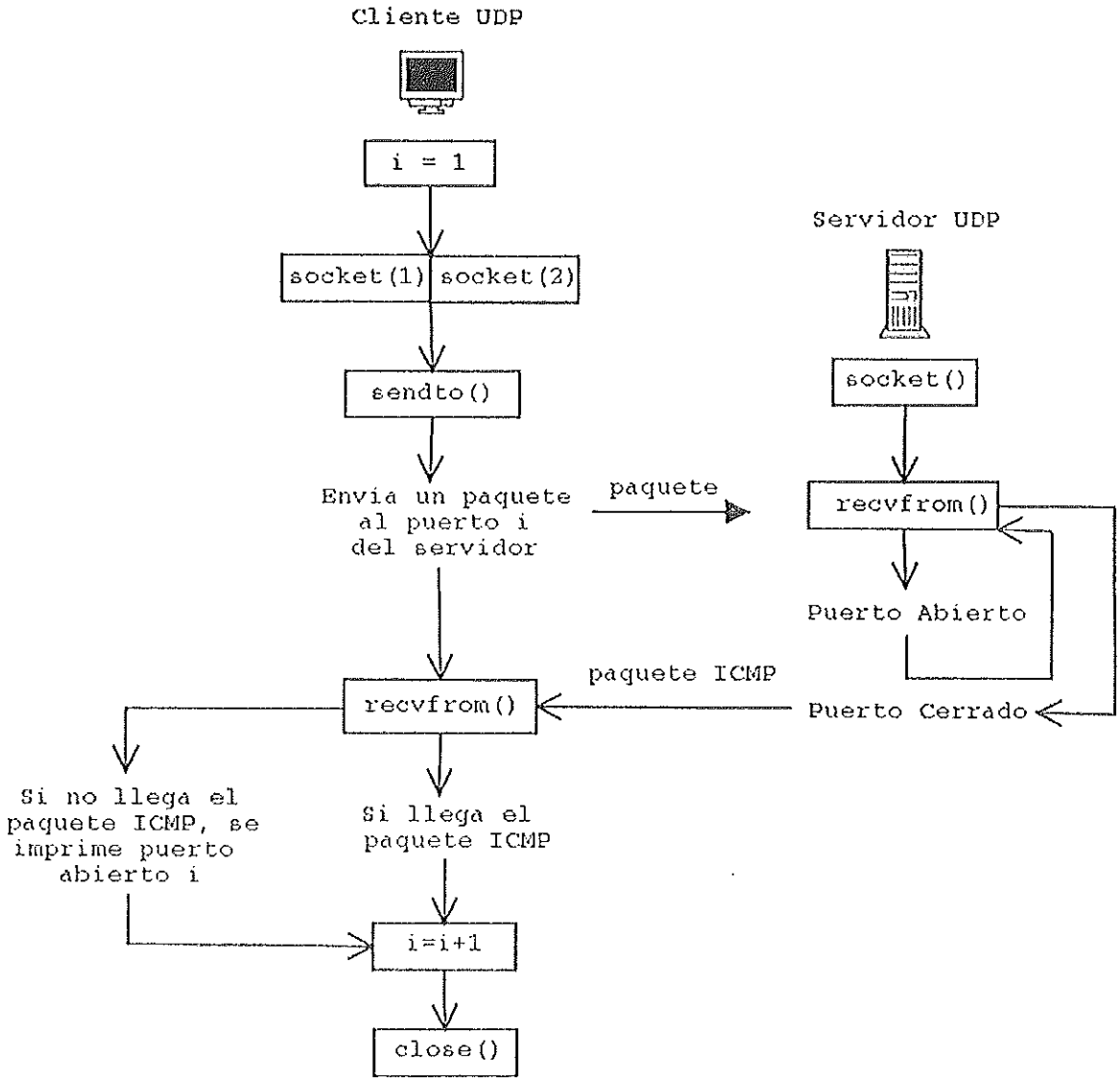


Figura 13. Diagrama general del escáner UDP.

Después de realizar esta acción, se creó una biblioteca para cada escáner la cual se unió en el proyecto de una manera estructurada.

VI.2 - 2ª Etapa (*Banners* y *Fingerprinting* de Servicios de Red)

Una de las características de los sistemas operativos de red, es la posibilidad de ofrecer un determinado número de servicios disponibles, como por ejemplo: servidor de paginas Web, servidor de bases de datos, servidor de archivos, etc., cada uno de estos servicios tiene –como cualquier *software*- errores o *bugs*, dependiendo de la versión del mismo, son los *bugs* correspondientes.

Como se ha mencionado anteriormente, un atacante primero tratara de conocer la versión del sistema operativo de su objetivo, después de esto, se centrara en conocer que servicios tiene disponibles (o sea, que puertos tiene abiertos –UDP ó TCP-), sabiendo que puerto esta abierto, tratara de descubrir que servicio es el que está corriendo, ya sea un servidor Telnet en el puerto 23, un servidor de archivos en el puerto 21, etc. El paso siguiente sería conocer la versión del servicio (ProFTP 1.2.9 en el puerto 21 por ejemplo), así al momento de conocer exactamente que servicio y que versión del mismo esta corriendo en determinado puerto, el atacante tiene más posibilidades de brincar la seguridad del mismo, ya sea con un *exploit* para determinada versión o alguna contraseña por defecto del servicio, por ejemplo.

Servicios de red

Un servicio de red es un *software* o aplicación, que nos brinda la oportunidad de interactuar con la aplicación que se encuentra en la máquina remota (servidor), ya sea para la transferencia de archivos (FTP), ejecución de una terminal remota (Telnet o SSH), visualización de páginas Web (Apache), etc.

VI.2.1 - *Banner* (cadena identificativa)

Es una cadena de texto la cual nos da información acerca del servicio que esta corriendo una máquina (en este caso un servidor), por ejemplo la figura 14 es la cadena identificativa de un servicio de red.

```
SSH-1.99-OpenSSH_3.5p1
```

Figura 14. Cadena identificativa (o *banner*).

La forma de obtener un *banner* de algún servicio de red es usando la técnica del protocolo TCP, la cual es llamada *three way shaking,* que significa "saludo de tres pasos". Como se mencionó anteriormente el protocolo TCP, es un protocolo orientado a conexión, esto quiere decir que antes de empezar a recibir y/o enviar datos a través de este protocolo, debe asegurarse de que el otro extremo de la conexión este debidamente conectado y sincronizado, para esto se utiliza la técnica del "saludo de tres pasos", la cual se muestra en la figura 15.

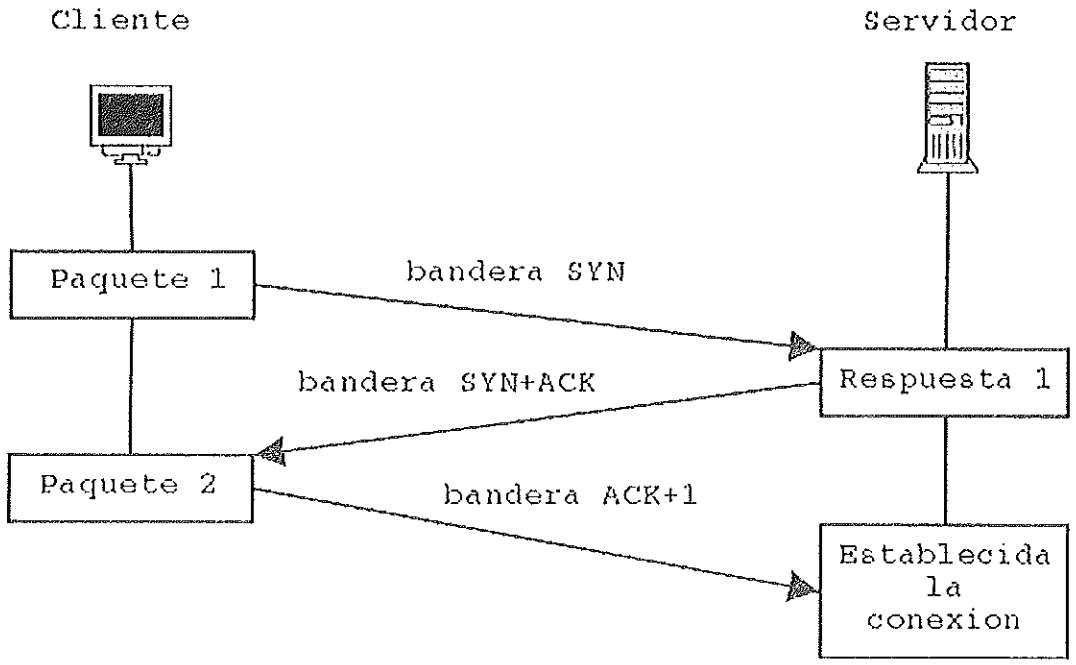


Figura 15. Técnica del saludo de tres pasos.

Como se puede observar en la figura, el cliente (o emisor) primero envía un paquete con la bandera *SYN* activada, el servidor le responderá con un paquete con las banderas *SYN* y *ACK* activadas, esto es, el número de sincronización será el resultado de la suma de las banderas *SYN* + *ACK*, después el cliente le regresará el paquete con el número de sincronización más uno, esto indica que la transmisión fue establecida. Este

comportamiento ya está establecido dentro del protocolo TCP; siempre y cuando el servidor este listo para establecer la conexión por determinado puerto.

De la misma manera que el comportamiento anterior ya está establecido en el protocolo TCP. Algunos servicios de red al momento de conectarnos a ellos (para iniciar una comunicación), estos nos envían una cadena identificativa (como se vio en la figura 14), en la cual nos dice que versión esta "corriendo". En la figura 16 se muestra un diagrama a bloques del comportamiento de estos servicios.

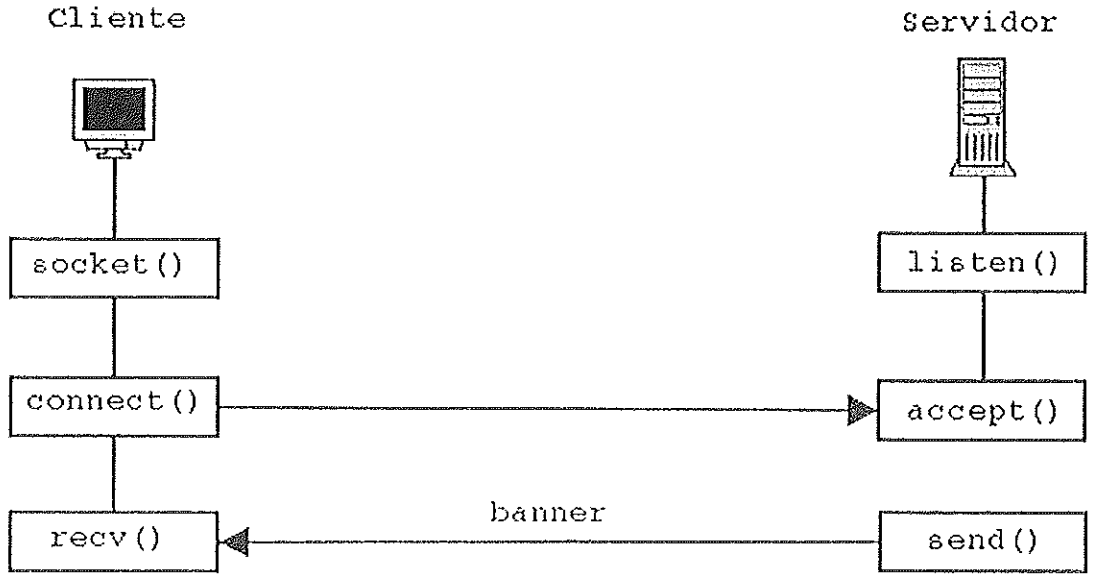


Figura 16. Comportamiento de un servicio que regresa un *banner*.

Hay que tener en cuenta que no todos los servicio de red envían una cadena identificativa cuando nos conectamos a ellos, por lo tanto se vuelve más complicado el averiguar que servicio esta corriendo en el puerto, debemos tener en cuenta además que la cadena identificativa puede ser cambiada por el administrador del sistema (lo cual es una buena práctica) para así poder engañar a un intruso que desee comprometer la seguridad del sistema.

Para tratar de averiguar que servicio esta corriendo el puerto, aún si el mismo no despliega ninguna cadena identificativa se usara lo que es llamado ***fingerprinting* de servicios** que se será explicado a continuación.

VI.2.2 - *Fingerprinting* de servicios de red

El llamado *fingerprinting* es una técnica que obtiene la huella identificativa de un servicio o de un sistema operativo (como se menciono en el capitulo 2), en este caso se basa en obtener la huella del servicio que este corriendo en determinado puerto.

Funcionamiento del *fingerprinting* de servicios

Cada servicio que esta corriendo en un anfitrión tiene su diferente protocolo (o conjunto de reglas) para comunicarse con el cliente, por ejemplo el Servidor de Nombre de Dominio (DNS) no utiliza el mismo protocolo para resolver una petición que el Protocolo de Transferencia de Archivos (FTP).

Como cada servicio tiene su protocolo lo que se hace es enviar una cadena específica para el servicio en forma binaria, al momento de recibirla el servicio, nos regresará una respuesta específica. Así analizando la respuesta que nos regreso el servidor y si es la que esperábamos sabemos qué servicio es el que puede estar ejecutando. La figura 17 muestra este procedimiento.

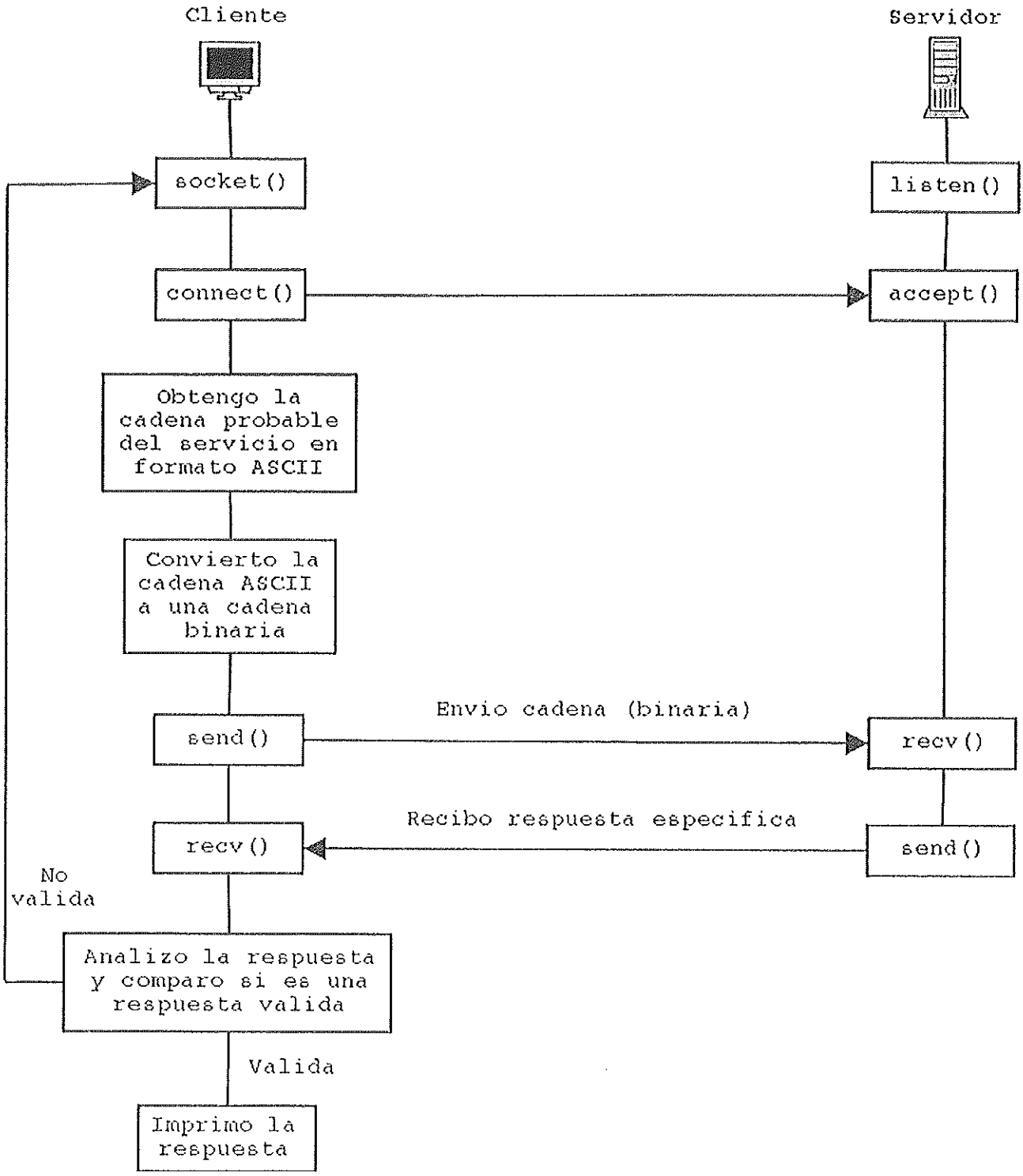


Figura 17. Diagrama de bloques del *fingerprinting* de servicios.

Si al momento de analizar la respuesta nos encontramos con una respuesta especifica, sabremos qué servicio es el que está corriendo, en caso contrario seguiríamos probando el

puerto con diferentes cadenas, en caso de que ninguna respuesta sea la esperada, entonces solo nos quedaría revisar si el puerto se encuentra en el archivo /etc/services. El cual nos indica el puerto estándar para cada servicio. Aunque esto no indica que ese servicio forzosamente este ejecutándose en determinado puerto.

La forma de implementar las dos técnicas mencionadas anteriormente y la revisión del archivo /etc/services, se puede observar en la figura 18.

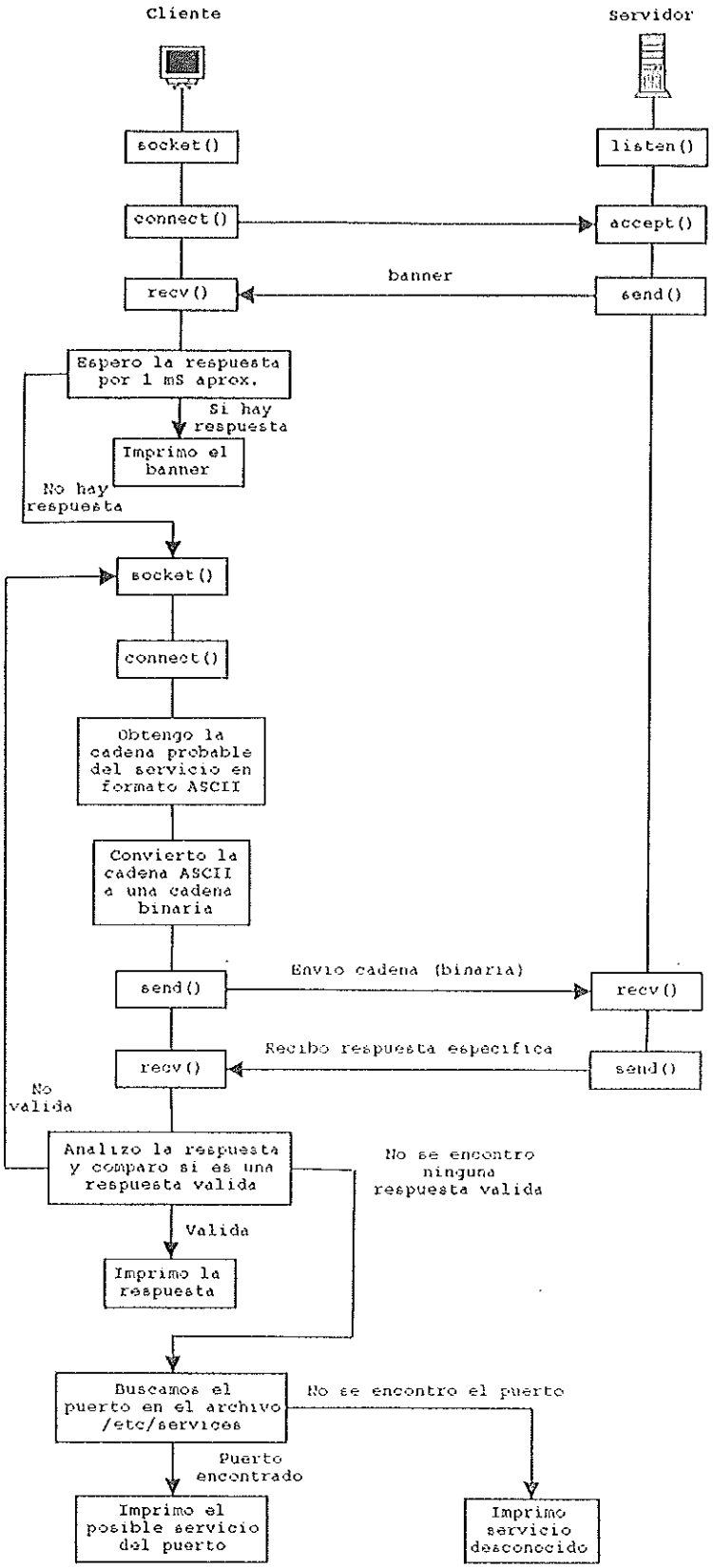


Figura 18. Diagrama de bloques del analizador de servicios.

VI.3 - 3ª Etapa (Directivas básicas de seguridad en el servidor)

Debemos tomar en cuenta que, aunque los servicios que se estén ejecutando estén configurados correctamente, no serviría de mucho si el servidor en sí, no esta bien asegurado. Por ejemplo si tenemos cuentas en nuestro servidor sin contraseña, o cuentas del mismo con contraseñas por defecto. Por lo tanto no debemos olvidarnos de alguna de las directivas más básicas para asegurar nuestro servidor, ya que un sistema es tan seguro como su elemento más inseguro.

VI.3.1 – Directivas básicas locales (para sistemas Linux)

Cuentas especiales y cuentas sin contraseña

Muchos clones de UNIX se instalan con cuentas consideradas "del sistema", es decir, que no corresponden a ningún usuario concreto sino que existen por cuestiones de compatibilidad o para la correcta ejecución de algunos programas. Algunas de estas cuentas no tienen contraseña, o tienen una conocida por todo el mundo, por lo que representan una grave amenaza a la seguridad: hemos de deshabilitarlas para evitar que alguien pueda conectarse a nuestro equipo mediante ellas. Algunos ejemplos de este tipo de cuentas son *guest*, *demo*, *uucp*, *games*, *4DGifts* o *lp*. Para deshabilitar una cuenta, en UNIX no tenemos mas que insertar un carácter de asterisco (*) en el campo *passwd* en la línea correspondiente del archivo de contraseñas (generalmente /etc/passwd o /etc/shadow). De esta forma, una entrada como:

```
toni:7atzxSJlPVVaQ:1001:10:Toni
Villalon:/export/home/toni:/bin/sh
```

Pasara a convertirse en

```
toni:*7atzxSJlPVVaQ:1001:10:Toni
Villalon:/export/home/toni:/bin/sh
```

Debemos deshabilitar todas las cuentas especiales del sistema que no necesitemos, así como los grupos innecesarios del mismo.

```
userdel    nombre_usuario    p.e.  userdel  uucp
groupdel   nombre_grupo      p.e.  groupdel games
```

Debemos asegurarnos que todas las cuentas del sistema tengan contraseña, para esto existe el siguiente *script* (o guión):

```
# awk -F: '$2 == "" { print $1, "has no password!" }'
/etc/shadow
```

Además debemos conocer todas las cuentas con acceso de superusuario al sistema;

```
$ awk -F: '$3 == 0 { print $1, "is a superuser!" }'
/etc/passwd
```

Si deseamos prevenir que el servidor responda a peticiones eco (*ping request*) podemos hacerlo de dos maneras:

1. Ejecutando el siguiente comando:
   `echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all`
   Y después agregándolo al archivo /etc/rc.d/rc.local para que cada vez que se reinicie el sistema se ejecute el comando. Para desactivar esta directiva bastaría con ejecutarlo de la siguiente manera:
   `echo 0 > /proc/sys/net/ipv4/icmp_echo_ignore_all`

2. Otra forma seria solamente editando el archivo /etc/sysctl.conf y cambiando la siguiente línea:
   `net.ipv4.icmp_echo_ignore_all=0`
   Por
   `net.ipv4.icmp_echo_ignore_all=1`

Y ejecutando el comando sysctl –p para así grabar los cambios hechos en el archivo anterior.

Si deseamos que el sistema no responda a peticiones hechas por el *broadcast*:

Solamente editamos el archivo /etc/sysctl.conf y cambiando la siguiente línea:

```
net.ipv4.icmp_echo_ignore_broadcast=0
```

Por

```
net.ipv4.icmp_echo_ignore_broadcast=1
```

Y ejecutando el comando sysctl –p para así grabar los cambios hechos en el archivo anterior.

La gestión de paquetes *icmp redirect* también puede presentar ciertos riesgos para nuestra seguridad. Si en /proc/sys/net/ipv4/conf/*/accept_redirects indicamos un valor diferente de 0 estamos diciéndole al núcleo de Linux que haga caso de este tipo de paquetes que en principio nos puede enviar un *ruteador*; su valor por defecto (0, desactivado) es el correcto. Análogamente, en /proc/sys/net/ipv4/conf/*/send_redirects permitimos la emisión de estos paquetes desde nuestra máquina escribiendo un valor diferente de 0; como solo un enrutador deberá enviar estos paquetes, la opción más segura es especificar un 0 para este parámetro (su valor por defecto es 1). Una opción intermedia entre bloquear todos los paquetes *icmp redirect* y permitirlas puede ser el escribir en el archivo secure_redirects un valor diferente de 0, logrando que se acepten este tipo de paquetes pero solo desde la lista de *gateways* validos definida en /etc/gateways.

Hablando del protocolo IP, uno de los parámetros que más nos va a interesar es la habilitación o negación del *IP Forwarding* en el núcleo de Linux; como hemos dicho antes, el sistema de filtrado de paquetes solo funciona cuando está opción esta habilitada, lo que se consigue con el comando:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Sin embargo, si no utilizamos las facilidades de *firewalling* del núcleo de Linux esta opción ha de estar deshabilitada (introduciríamos un *0* en lugar de un *1* en el archivo anterior), ya que de lo contrario corremos el peligro de que nuestra máquina se convierta en un *ruteador*.

Protocolos de Enrutamiento

El enrutamiento y los protocolos de enrutamiento pueden ocasionar varios problemas; si un atacante puede enviar un paquete enrutado en la dirección fuente a nuestra red, también podrá interceptar la respuesta y engañar al anfitrión, al creer que se está comunicando con un "anfitrión confiable". Para deshabilitar el enrutamiento de dirección fuente en el servidor debemos escribir el siguiente comando:

```
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
    echo 0 > $f
done
```

Para que no tengamos que escribir el comando cada vez que reiniciemos el sistema, podemos agregar el comando al archivo /etc/rc.d/rc.local.

Habilitando la protección TCP SYN *cookie*

Un ataque *SYN attack* es una ataque de Negación de Servicio (DoS) que consume todos los recursos de la máquina, forzando un reinicio del sistema. Para habilitar la protección contra este ataque debemos teclear el siguiente comando:

```
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

Para no escribir el comando cada vez que reiniciemos el sistema, agregaremos el comando al archivo /etc/rc.d/rc.local.

VII – RESULTADOS

Ahora solo nos queda probar la herramienta, tanto en un análisis remoto, como en un análisis local, para esto se mostrara en forma sencilla y breve como configurar la herramienta, además de mostrar los resultados del análisis hecho en ambos casos.

VII.1 – Ejemplo de un análisis local

El primer paso para configurar la herramienta sería establecer el rango de puertos que serán analizados y que tipo de escaneo será, TCP, UDP y/o Servicios. En la figura 19 se muestra la pantalla del sistema correspondiente a la configuración.

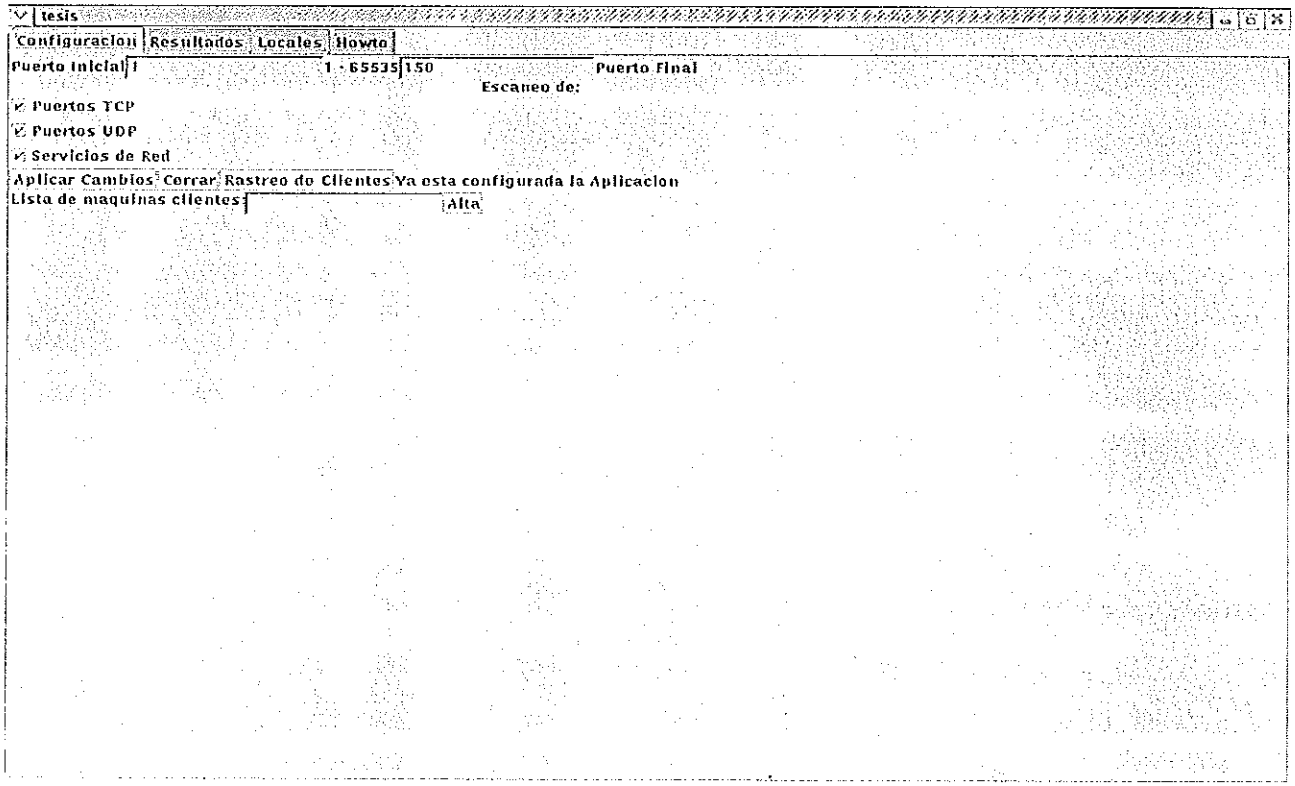


Figura 19. Configuración de un análisis local.

Como puede observarse en la figura 19, para este caso, el análisis será de los puertos 1 al 150, y se buscarán puertos TCP y UDP abiertos, además de tratar de obtener el servicio y si es posible la versión del mismo que esté corriendo en el puerto.

Ahora solo resta presionar el botón de escaneo y esperar a que finalice el análisis, después se observarán los resultados como se muestran en la figura 20.

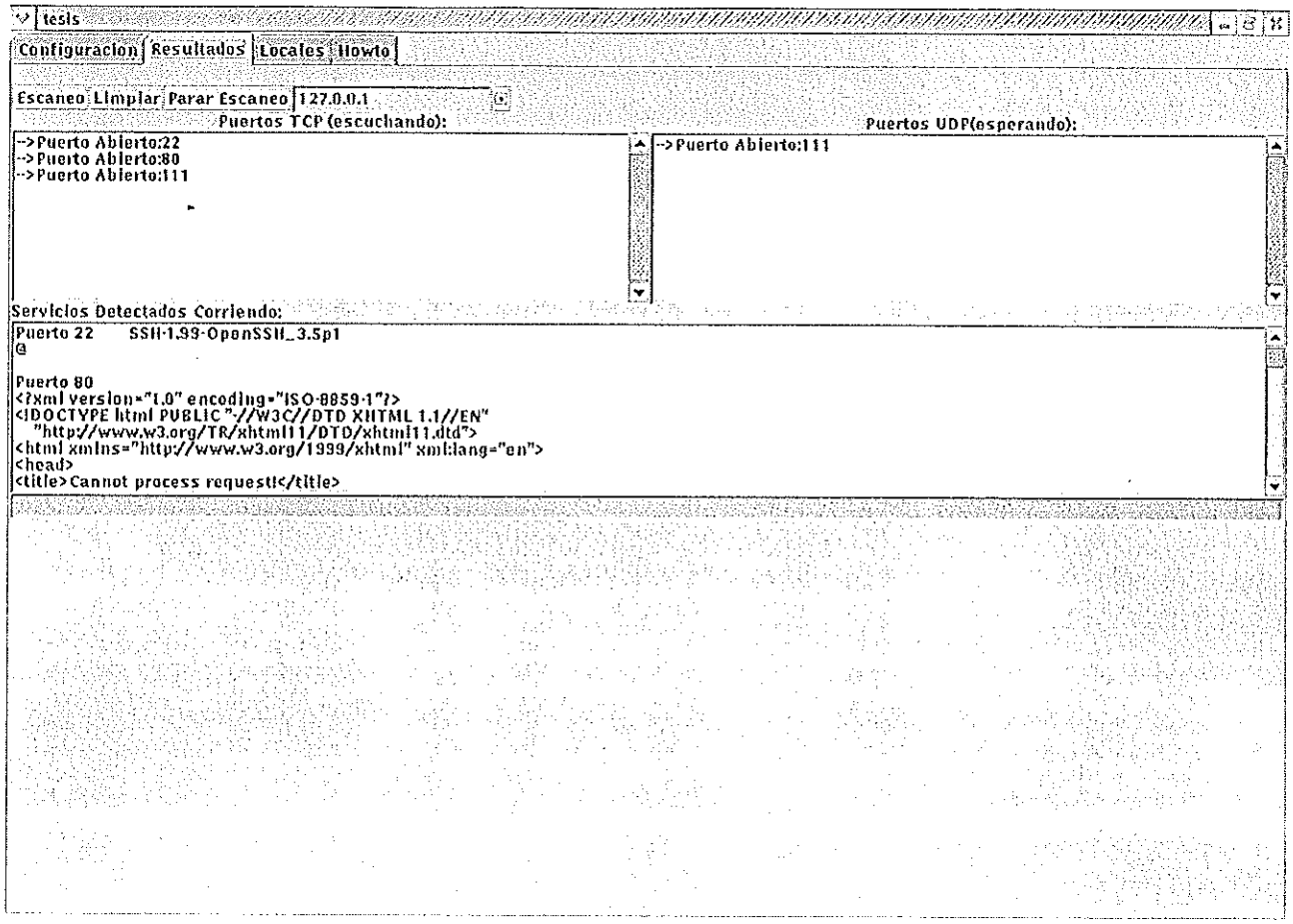


Figura 20. Resultados de un análisis local.

Como puede observarse, el servidor en este caso, tiene abiertos los puertos 22, 80 y 111. Se puede notar que además se pudo obtener la versión del servicio que esta corriendo en el puerto 22. También se obtuvo una respuesta en el puerto 80, aunque no se observa en la figura, en el puerto 111 no se obtuvo ninguna respuesta, así que lo que se hizo fue solo poner el servicio probable que podría estar, analizando el número de puerto en el archivo /etc/services.

La herramienta también hace un pequeño análisis de algunas directivas locales del servidor (por ejemplo, buscar cuentas sin contraseñas), como se muestra en la figura 21.

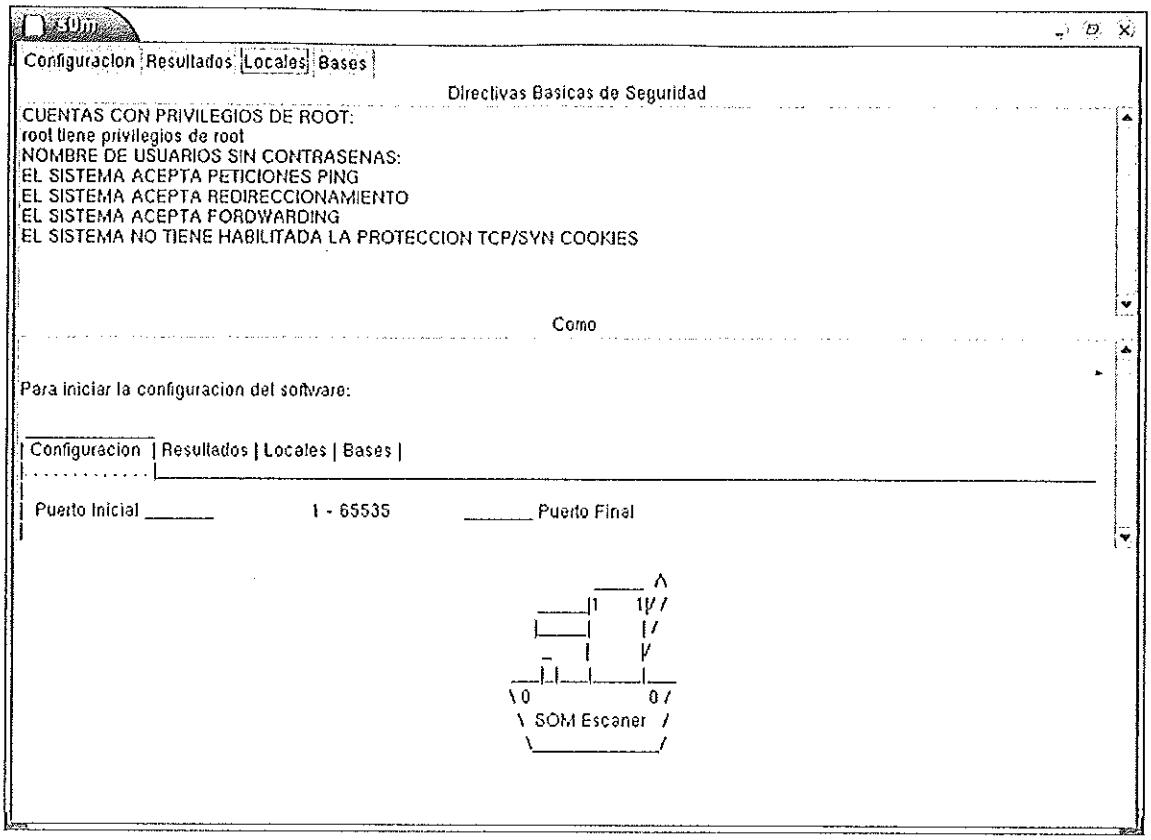


Figura 21. Directivas locales de un análisis local.

Como puede observarse se muestran las cuentas del sistema con privilegios de *root*, además de mostrar algunas directivas de redes importantes también nos muestra las cuentas (si es que las hay) del sistema que no tienen contraseña, así como un pequeño manual de configuración de la herramienta y su emblema correspondiente.

Para finalizar, la siguiente pantalla (figura 22) nos muestra algunas directivas de seguridad para algunos de los servicios más usados en los sistemas de hoy en día, estas directivas también se pueden observar en el Apéndice A de este trabajo.

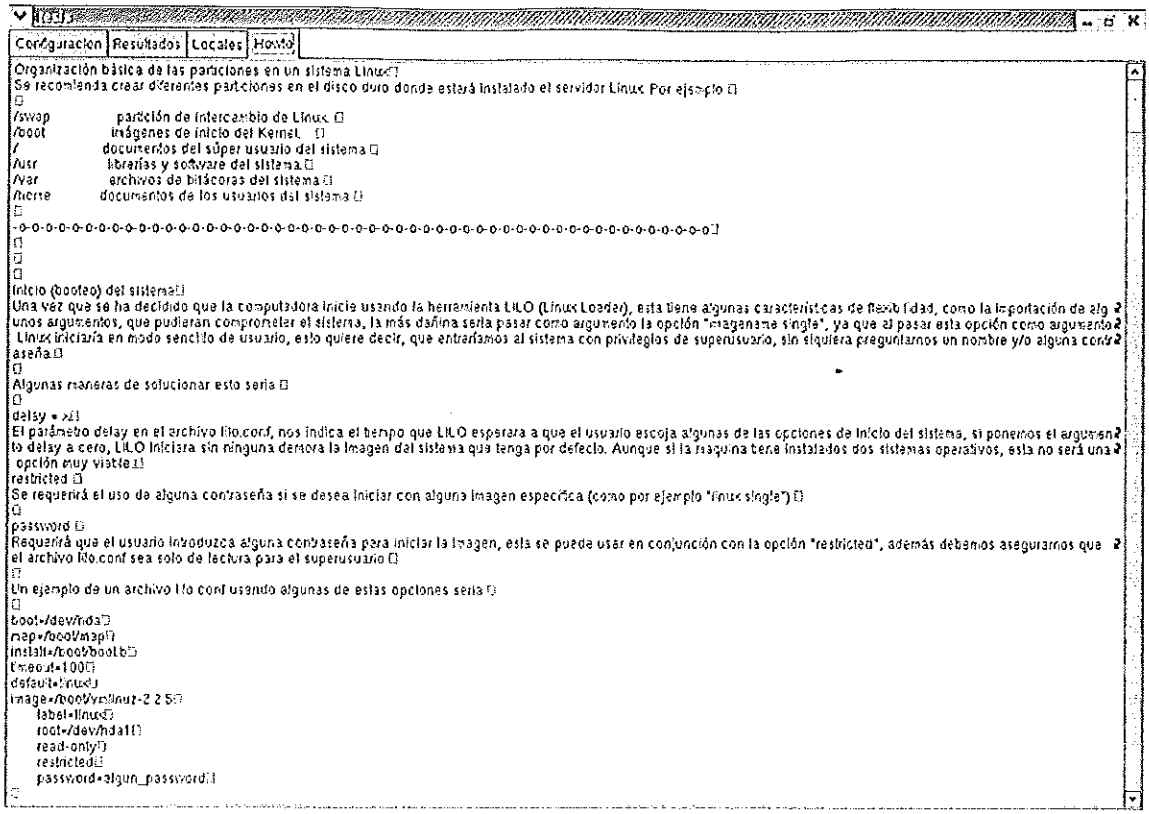


Figura 22. Pantalla de directivas de seguridad.

VII.2 Ejemplo de un análisis remoto

Para analizar los resultados del desarrollo de la herramienta, se hará un análisis de un servidor con un sistema Windows (Advanced Server 2000) instalado y uno con un sistema Linux (Mandrake 9.1); en ambos casos el análisis se hará con un cliente de la herramienta, especialmente diseñado para ello. El servidor (y pantalla de resultados) estará corriendo en otra distribución Linux (Yellow Dog 3.1).

Primero, -como en el caso anterior- se establecerá el rango de puertos que serán analizados, además del tipo de análisis que se llevará a cabo, ahora, como el análisis se hará en otros dos clientes, debemos dar de alta a los mismos, esto se hace ingresando su dirección IP a la caja de texto (como se observa en la figura 23) y presionando el botón de "alta".

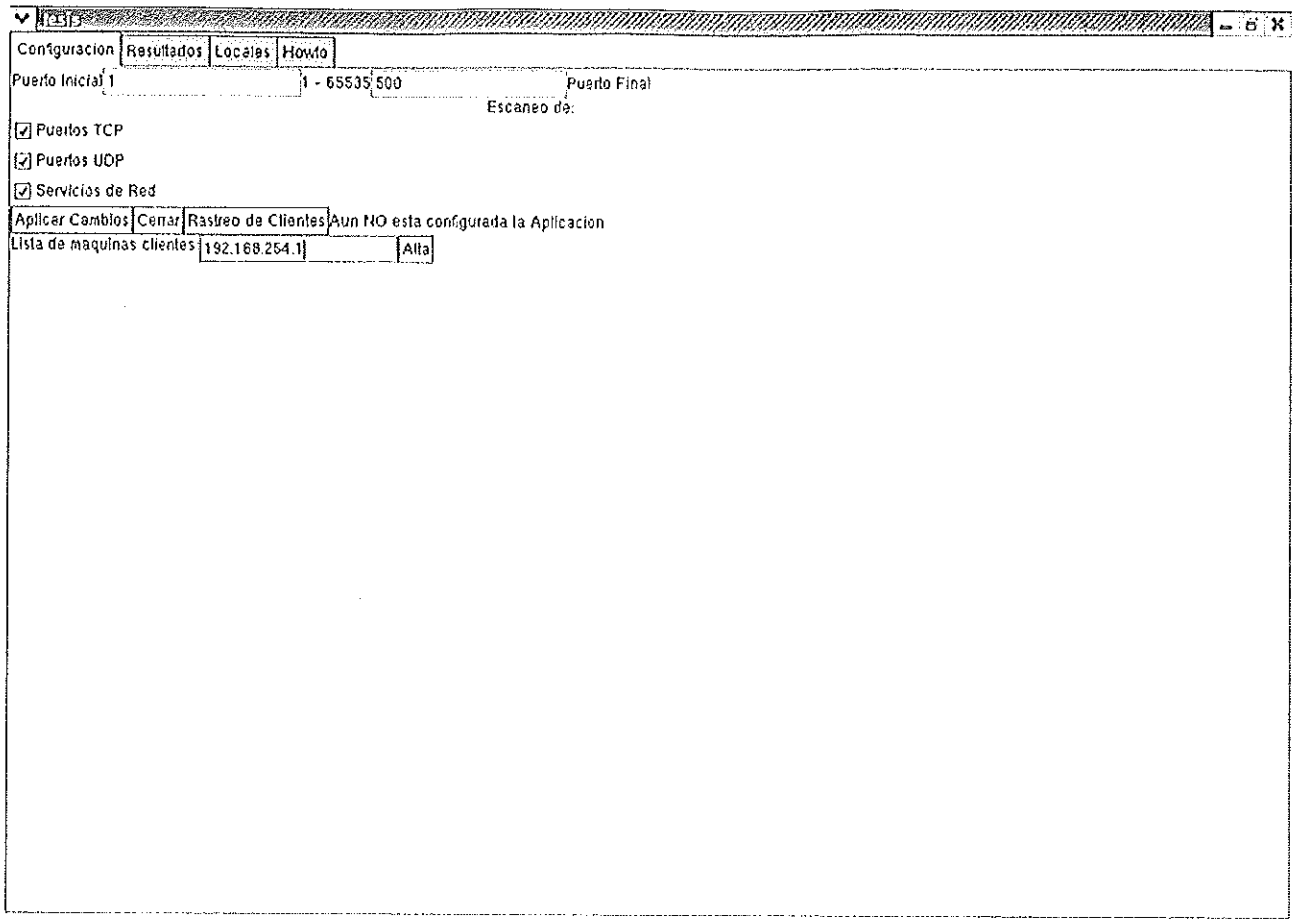


Figura 23. Altas a clientes.

Cuando se termina de ingresar las máquinas remotas que serán analizadas debemos conocer si en las direcciones ingresadas esta corriendo un cliente de la aplicación, para saber esto basta con presionar el botón "rastreo de clientes", y a continuación se mostrarán las diferentes direcciones IP que tienen corriendo clientes, así se podrá verificar, si todos los números IP ingresados están corriendo el cliente satisfactoriamente. Esto se puede observar en la figura 24.

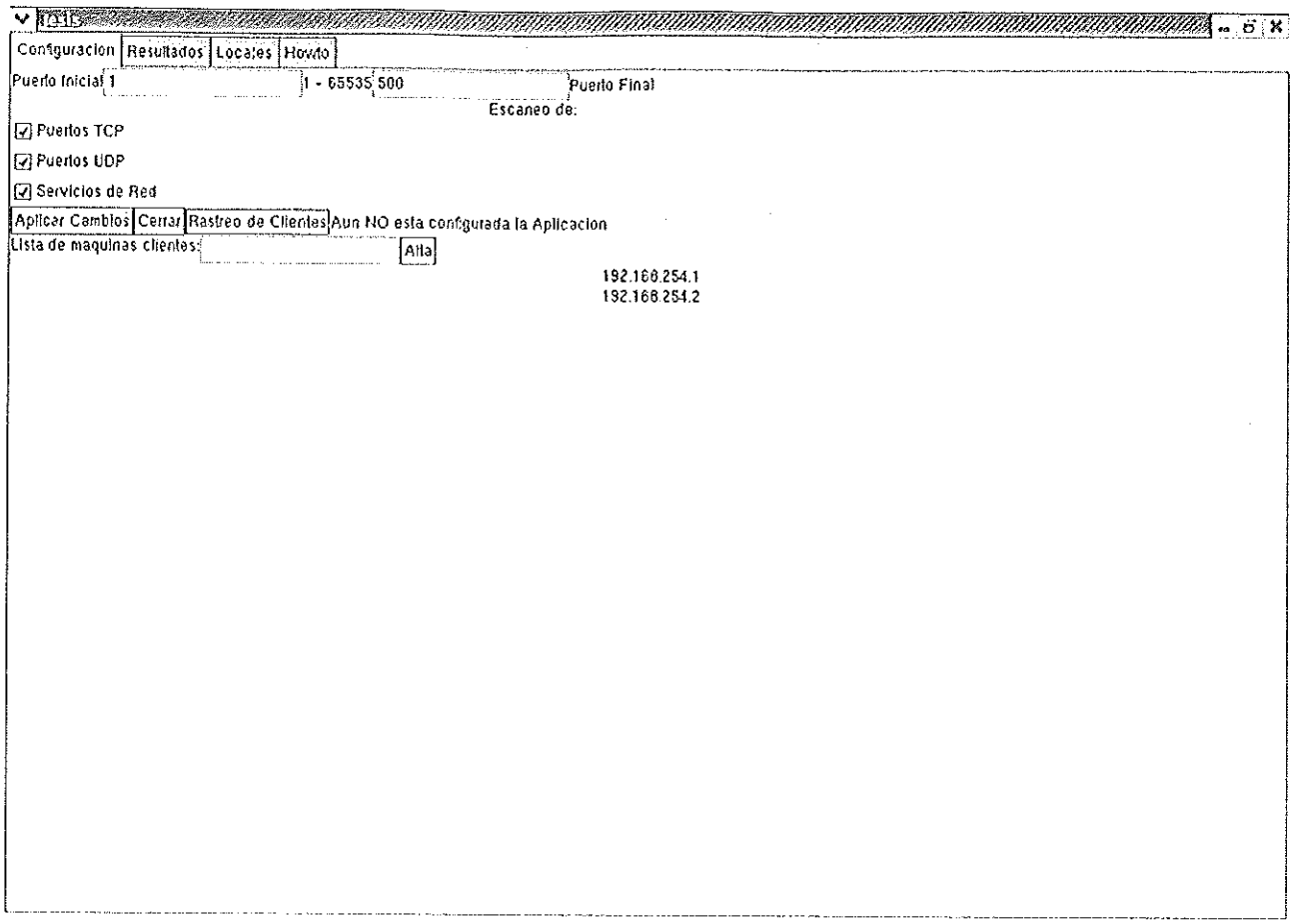


Figura 24. Rastreo de clientes.

Para este ejemplo, la dirección 192.168.254.1 corresponde al cliente ejecutando el sistema Linux y la dirección 192.168.254.2 al cliente ejecutando Windows.

Ahora en la pantalla de resultados podremos observar que en la casilla superior, se encuentran diferentes direcciones IP, cada vez que seleccionemos alguna, se muestra en pantalla los resultados obtenidos del análisis del la dirección IP seleccionada, como se muestra en la figura 25.

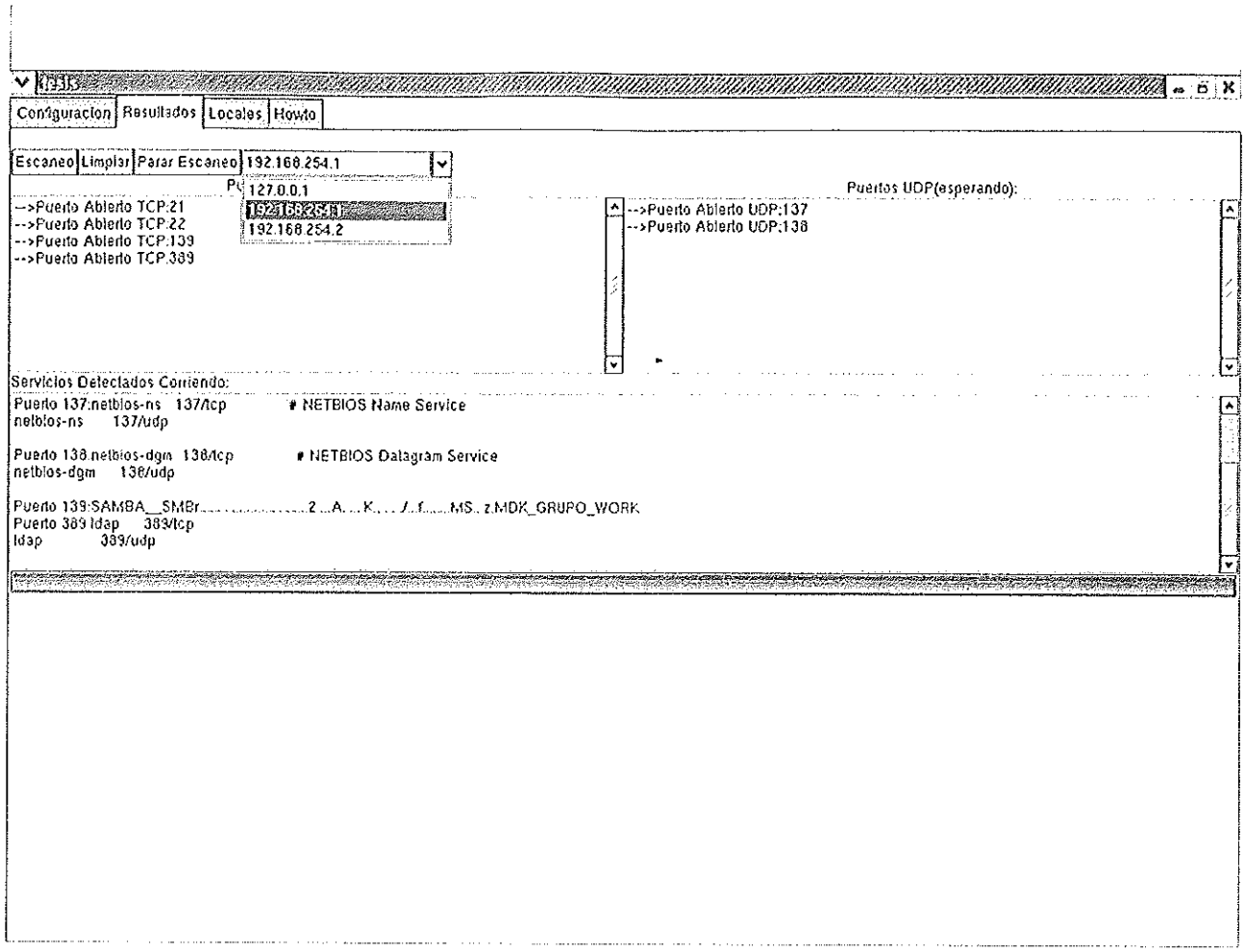


Figura 25. Resultados de los clientes.

Al momento de seleccionar la dirección IP, también podemos observar los resultados de sus directivas locales, observando la pantalla de locales, la cual nos mostrará información de algunas directivas locales básicas, dependiendo de la dirección IP seleccionado en la casilla de la pantalla de resultados, como se observo en la figura 21.

La forma de ejecutar o activar el cliente en las máquinas remotas es muy simple, basta con escribir en el directorio donde se encuentre el programa ejecutable, el nombre del mismo. Después de configurar el servidor de la herramienta bastara con presionar el botón "escaneo", como se mencionó anteriormente. En las figuras 26 y 27 se muestra la salida de los clientes (Windows y Linux) para una mejor comprensión.

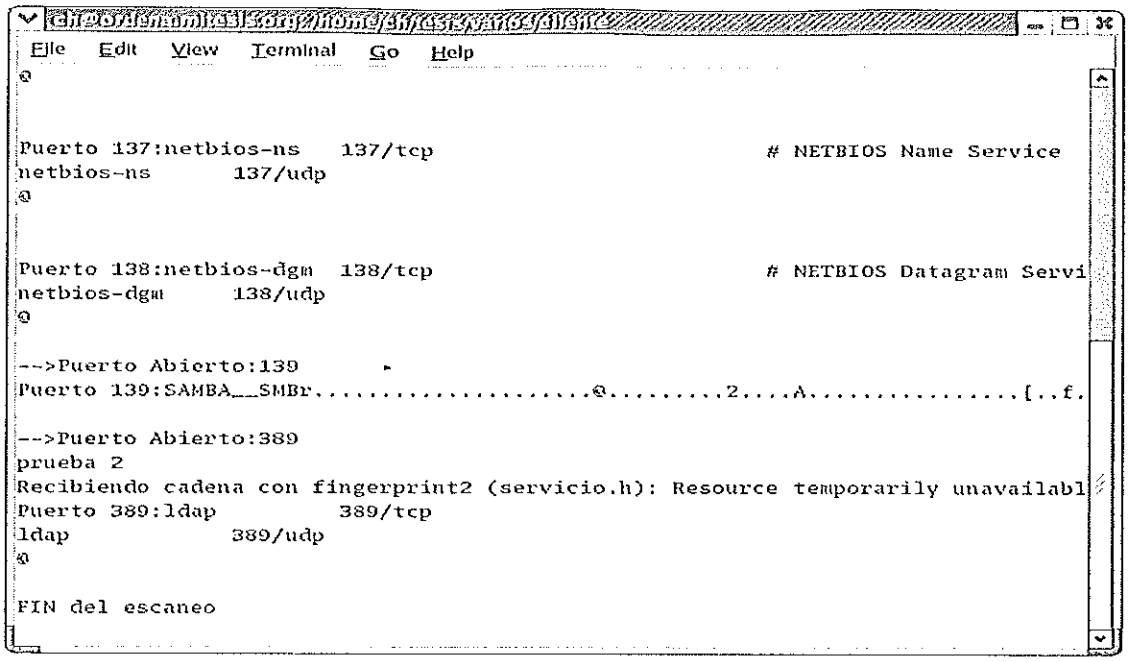


Figura 26. Cliente Linux.

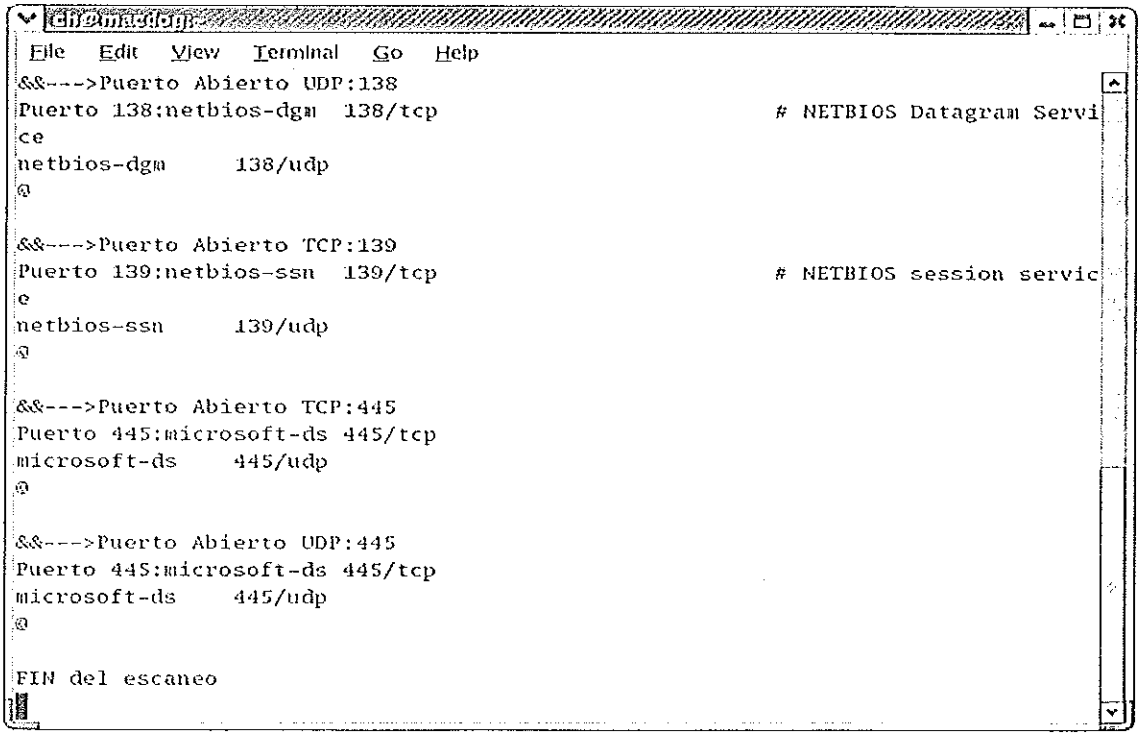


Figura 27. Cliente Windows.

VII – CONCLUSIONES Y APORTACIONES

VII.1 - Conclusiones

Al llegar a la conclusión de este trabajo nos damos cuenta que tan importante es la seguridad dentro de una red, incluso de una sola máquina, ya que los atacantes comprometiendo una sola máquina pueden llegar a tener acceso a toda la red mediante ataques a algunas de las vulnerabilidades ya mencionadas.

Hoy en día toda la información concerniente e importante para nosotros ya no se encuentra en papeles, documentos, archiveros, etc., sino que toda se encuentra digitalizada y guardada en máquinas (o servidores), que en manos de un administrador no muy preocupado por la seguridad puede estar en un peligro latente, ya que cualquier persona con conocimientos (aún cuando estos sean limitados) técnicos podría poner en riesgo nuestra privacidad o incluso dañar nuestra integridad.

Por eso este trabajo trató de dar a conocer algunos de los ataques y vulnerabilidades más comunes que podría tener una red, para así tratar de concienciar a las personas de lo importante que es el tema de la seguridad en las redes y computadores actuales.

Desarrollo en Linux

De una forma más específica la herramienta desarrollada cumple con las expectativas planteadas, ya que hace un análisis de la red de los sistemas, analizando que puertos –ya sean TCP o UDP- tiene abiertos el mismo, así como intentando conocer que servicio y que versión del mismo esta corriendo en determinado puerto. También hace un análisis de algunas de las directivas más básicas (pero no por eso menos importantes) de seguridad del sistema.

Por otra parte el análisis de puertos en los clientes remotos es hecho por los mismos clientes, así que puede decirse que es un ahorro de tiempo considerable, ya que al

momento de activar el escaneo todos los clientes están trabajando localmente al mismo tiempo, para así no alentar o degradar la conexión de red.

Desarrollo en Windows

Se debe mencionar que el desarrollo del cliente para sistemas Windows, en realidad se hizo en un sistema Linux, únicamente utilizando un compilador, el cual genera un ejecutable, que además puede correr en un ambiente Windows usando la librería de emulación POSIX (`cygwin1.dll`). Cabe mencionar que el escaneo de puertos de un sistema Windows, a diferencia de los clientes Linux, se hace remotamente desde el servidor de la herramienta, se tuvo que hacer de esta forma porque el escaneo de puertos TCP en un sistema Windows en forma local es demasiado lento, y el escaneo UDP no puede hacerse en una forma eficiente y confiable usando las librerías básicas del mismo.

El sistema Windows tiene una muy diferente arquitectura de un sistema Linux, por lo que el análisis de directivas básicas de seguridad no pudo hacerse en un ambiente Windows de acuerdo a la forma en que fueron diseñaron los clientes.

Podemos comparar esta herramienta con algunas ya existentes, es este caso será con solo dos: el ya famoso nmap y el destacado nessus.

NMAP (http://www.insecure.org/nmap)

Esta herramienta es un analizador basado en red, el cual analiza los servicios de red que están corriendo en un anfitrión (sea un sistema Windows, Linux, UNIX, Mac, etc.), puede incluso llegar a analizar toda una red, tanto sus puertos y servicios completamente.

NESSUS (http://www.nessus.org)

Esta herramienta es un analizador basado en anfitrión el cual analiza las vulnerabilidades de una máquina específica, además de también hacer un análisis de red del mismo.

Las ventajas de la herramienta desarrollada en este trabajo, con respecto a las dos anteriores, son: el hecho de hacer un análisis de la red (escaneo de puertos), así como un análisis del anfitrión (directivas básicas de seguridad) usando la misma herramienta, además de mostrarle al usuario información y consejos en castellano para asegurar su sistema de ataques a su seguridad. Además, como se menciona anteriormente el análisis de los diferentes clientes remotos se hace en una forma un relativamente más rápida, ya que los clientes hacen su análisis al mismo tiempo (en forma local) que el servidor, con lo que podemos ahorrar tiempo en ello.

Cabe mencionar que la información y consejos mostrados por la herramienta solo son la punta del iceberg, en lo que se refiere a seguridad, esto se hace con la intención de motivar al administrador del sistema, o al usuario común, a investigar por su cuenta un poco más sobre estos temas.

IX – REFERENCIAS

IX.1 - Referencias

[1] Seifried Kart, traducción: Revilla José Antonio.; 1999; Guía de Seguridad del Administrador de Linux (GSAL); 17, 19, 21, 27, 50-100; http://www.seifried.org/lasg.

[2] The Sans Institute; 2000; Securing Linux Step by Step versión 1.0; 3, 18-72; http://www.sans.org.

[3] The Sans Institute; 2005; The twenty most critical Internet security vulnerabilities (updated); http://www.sans.org.

[4] Barret Daniel J., Silverman Richard E. & Byrnes Robert G.; Linux Security Cookbook; O'Reilly; capítulo 9.

[5] Novelo Vázquez Andrés Demetrio; 2001; Tesis de Licenciatura en ciencias de la computación; Mérida, Yucatán; 23-32, 83.

[6] Villalón Huerta Antonio; 2002; Seguridad en UNIX y Redes 2.1; 94, 240-260, 324-355, 575-578.

[7] Mourani Gerhard; 2000; Securing and Optimizing Linux; 29-42, 44-66.

[8] Walton Sean, traducción: Clave Informática I+D S.A.; 2000; Programación de Socket Linux; 10-20, 401.

[9] Cox Phillip, Sheldon Tom; 2002; Windows 2000 manual de seguridad; McGraw-Hill; Mexico D.F.; 134, 283-290, 295, 611, 612.

[10] Siles Peláez Raúl; 2001; Seguridad en TCP/IP edición 1; 23-26, 30, 32, 35-37, 40-48, 53, 54, 56-58, 281, 282, 284-286.

[11] Jamsa Kris, Cope Ken; 1996; Programación en Internet; McGraw-Hill; México D.F; 27-31, 41-44, 61-63, 105, 112, 118, 156.

Apéndice A. - Más directivas

A.1 – Directivas básicas locales (para Linux)

Organización básica de las particiones en un sistema Linux

Se recomienda crear diferentes particiones en el disco duro donde estará instalado el servidor Linux. Por ejemplo

/swap	partición de intercambio de Linux.
/boot	imágenes de inicio del Kernel.
/	documentos del súper usuario del sistema.
/usr	librerías y software del sistema.
/var	archivos de bitácoras del sistema.
/home	documentos de los usuarios del sistema.

Inicio (*booteo*) del sistema

Una vez que se ha decidido que la computadora inicie usando la herramienta LILO (Linux Loader), esta tiene algunas características de flexibilidad, como la importación de algunos argumentos, que pudieran comprometer el sistema, la más dañina seria pasar como argumento la opción "*imagename single*", ya que al pasar esta opción como argumento Linux iniciaría en modo sencillo de usuario, esto quiere decir, que entraríamos al sistema con privilegios de superusuario, sin siquiera preguntarnos un nombre y/o alguna contraseña.

Algunas maneras de solucionar esto seria:

```
delay = x
```

El parámetro *delay* en el archivo lilo.conf, nos indica el tiempo que LILO esperará a que el usuario escoja algunas de las opciones de inicio del sistema, si ponemos el argumento *delay* a cero, LILO iniciara sin ninguna demora la imagen del sistema que tenga por

defecto. Aunque si la máquina tiene instalados dos sistemas operativos, esta no será una opción muy viable.

`restricted`

Se requerirá el uso de alguna contraseña si se desea iniciar con alguna imagen específica (como por ejemplo "*linux single*").

`password`

Requerirá que el usuario introduzca alguna contraseña para iniciar la imagen, esta se puede usar en conjunción con la opción "*restricted*", además debemos asegurarnos que el archivo lilo.conf sea de solo de lectura para el superusuario.

Un ejemplo de un archivo lilo.conf usando algunas de estas opciones sería:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
timeout=100
default=linux
image=/boot/vmlinuz-2.2.5
        label=linux
        root=/dev/hda1
        read-only
        restricted
        password=algun_password
```

Al momento de tener las directivas anteriores en nuestro archivo lilo.conf, el sistema iniciara sin ningún problema si escogemos iniciar con la imagen "*linux*", en cambio si se envía como parámetro "*linux single*", nos pedirá la contraseña para poder iniciar con esa imagen. Hay que tener en cuanta que después de colocar la contraseña en el archivo lilo.conf, este deberá ser de solo lectura únicamente para el superusuario. Así los usuarios normales no podrán ver la contraseña de LILO.

Archivos del Sistema

/etc/login.defs

Este archivo nos permite definir algunos parámetros como la expiración de la contraseña.

fail delay marca el número de segundos (por defecto, 3) que el sistema introduce como retardo desde que se introduce un nombre de usuario o contraseña incorrectos hasta que se vuelve a solicitar el nombre de usuario de entrada al sistema; el número máximo de intentos antes de que se cierre la conexión viene determinado por el valor de *login retries*, por defecto a 5, y el tiempo máximo durante el que se permite la entrada antes de cerrar la conexión se especifica mediante la directiva *login timeout* (60 segundos por defecto).

Cuando un usuario se equivoca en su nombre de entrada o su clave entran en juego un par de directivas más: *faillog enab* y *log unkfail enab*. La primera de ellas, con un valor por defecto a *yes* (el adecuado para nuestra seguridad), se encarga de registrar los intentos fallidos de acceso al sistema en /var/log/faillog, así como de mostrar un mensaje con información acerca del último intento de acceso fallido cuando un usuario accede a la máquina. Por su parte, *log unkfail enab* habilita o deshabilita el registro del nombre de usuario que ha fallado al tratar de entrar al sistema; es importante que su valor sea "no" (es decir, que ese nombre de usuario no se registre) por una sencilla razón: en muchas ocasiones los usuarios teclean su contraseña en lugar de su nombre de usuario cuando tratan de acceder a la máquina, con lo que si el nombre de usuario incorrecto -la clave- se registra en un archivo en texto plano, y ese archivo tiene unos permisos inadecuados, cualquiera con acceso de lectura sobre el archivo de bitácoras podría deducir fácilmente el nombre y la clave del usuario. Adicionalmente, si la directiva *ftmp file* define un archivo (por defecto, /var/log/btmp), en el mismo se registra cada intento de acceso fallido en formato *utmp*, dicha información se puede consultar mediante el comando `lastb`.

Si se produce la situación contraria, la directiva *log ok logins* habilita o deshabilita el registro de este hecho en función de si su valor es *yes* o *no*; además, si *lastlog enab* tiene un valor *yes* se registra la entrada en /var/log/lastlog y se muestra al usuario

información acerca de su ultima conexión. En caso de que el usuario no pueda acceder a su directorio $HOME (bien porque no existe, bien por los permisos del mismo) entra en juego *default home*, y en caso de que su valor sea *no* no se permite el acceso; por el contrario, si su valor es *yes*, el usuario entra al directorio raíz de la máquina.

En /etc/login.defs también se pueden definir líneas para las que no es necesaria ninguna contraseña, mediante la directiva *no password console*: si alguien trata de conectar al sistema a través de ellas, se le solicitara su nombre de usuario pero no su clave; esto es aplicable para todos los usuarios de la máquina excepto para el superusuario, al que siempre se le pide su contraseña. Evidentemente, para incrementar la seguridad de nuestro sistema la directiva correspondiente ha de estar comentada.

A la hora de definir nuevos usuarios en el sistema también entran en juego ciertas directivas del archivo /etc/login.defs. Por ejemplo, el **UID** y **GID** máximo y mínimo para los usuarios regulares viene determinado por *uid max*, *gid max*, *uid min* y *gid min*. También existen entradas para especificar ciertas características relacionadas con las claves de los nuevos usuarios del sistema: se trata de *pass max days*, *pass min days*, *pass min len* y *pass warn age*. Como sus nombres indican, estas directivas marcan los números máximo y mínimo de días que una contraseña puede ser usada, la longitud mínima que toda contraseña ha de tener, y el número de días de antelación con que se avisara a los usuarios antes de que sus claves caduquen, respectivamente. Cada vez que se crea a un usuario nuevo en el sistema, se toman estos valores por defecto, aunque después es habitual particularizar para cada caso concreto.

Cuando un usuario ejecuta el comando `passwd` para cambiar su contraseña en el sistema entra en juego la directiva *obscure checks enab* (cuyo valor ha de ser *yes*) para impedir que se elijan claves débiles (por ejemplo, las formadas únicamente por letras minúsculas); en cualquier caso, se permiten tantos intentos de cambio como indica *pass change tries*, y si se supera este número se devuelve al usuario a su consola y la clave permanece inalterada; si el usuario que trata de cambiar la contraseña es el superusuario (tanto la propia clave como la de cualquier usuario) se le permite elegir cualquier

contraseña, sin importar su robustez, pero se le advierte de que la clave elegida es débil si el valor de directiva *pass always warn* es *'yes'*.

Al ejecutar `passwd` para modificar valores del campo *gecos* o para cambiar la consola de entrada al sistema (o equivalentemente, al ejecutar `chfn` o `chsh`) se solicita al usuario su clave si el valor de la directiva *chfn auth* es *yes*. Además, la entrada *chfn restrict* define los campos concretos que un usuario puede modificar: *Full Name*, *Room Number*, *Work Phone* y *Home Phone* (*frwh* si permitimos que modifique todos ellos); si la directiva no esta definida, no se permite ningún tipo de cambio. Relacionada también con las contraseñas de los usuarios, la directiva *pass max len* marca el número de caracteres significativos en una clave (8 por defecto).

/etc/suauth

En el archivo /etc/suauth se puede realizar un control más minucioso de la ejecución de `su`, no solo para acceder a cuentas con privilegios de administración, sino también para permitir o denegar cambios de identidad entre dos usuarios cualesquiera del sistema. Se trata de un archivo en el que cada línea es de la forma:

```
to-id:from-id:ACCION
```

Donde *ACCION* puede ser *deny* (se deniega el cambio de identidad), *nopass* (se permite el cambio sin necesidad de ninguna clave) u *ownpass* (se solicita la contraseña del usuario que ejecuta la orden en lugar del correspondiente al usuario al que se quiere acceder).

Es importante destacar que el contenido del archivo /etc/suauth solo afecta a usuarios regulares, no al superusuario: no podemos definir reglas que eviten que el cambie su identidad a cualquier usuario del mismo.

/etc/shells

Contiene una lista de las consolas validas, si la consola por defecto de algún usuario no esta definida aquí, no podrá ingresar interactivamente al sistema.

/etc/securetty

Contiene una lista de las consolas desde las cuales el superusuario puede ingresar; En la mayoría de las distribuciones si un usuario necesita privilegios de superusuario deberá ingresar como usuario regular y después cambiarse a superusuario.

Las consolas del sistema van usualmente de *tty1* a *tty6*. Si se desea ingresar como superusuario desde la red (no recomendable) se agregara *ttyp1* y así sucesivamente. Generalmente solo debe permitirse al superusuario ingresar por *tty1* para esto comentamos todos la líneas (agregando el carácter '#' al inicio de la línea) excepto la línea con la cadena *tty1*.

/etc/ftpusers

Contiene la lista de todos los usuarios que NO pueden ingresar al sistema de forma remota vía FTP. Debemos asegurarnos que en esta lista este la cuenta de superusuario y la cuenta de los demonios del sistema.

/etc/securety/access.conf

Contiene parámetros de ingreso para todas las cuentas del sistema. Podemos configurarlo para establecer que usuarios podrán ingresar al sistema por de medio de que servicio. Para deshabilitar el acceso a todos los usuarios excepto al superusuario bastaría con colocar en el archivo:

```
-:ALL EXCEPT root :console
```

/etc/securety/group.conf

Se define cuales usuarios pueden acceder a determinados archivos con permisos de grupo ya especificados.

/etc/securety/limits.conf

Se definen los límites de recursos del sistema como por ejemplo: tiempo de CPU, espacio de disco, etc. Para grupos y usuarios específicos del sistema.

/etc/security/times.conf

Se definen el horario que usuarios específicos pueden acceder al sistema. Los usuarios regulares pueden ser restringidos a un horario específico.

Los bits suid, sgid y sticky

Habitualmente, los permisos de los archivos en Linux se corresponden con un número en octal que varía entre 000 y 777; sin embargo, existen unos permisos especiales que hacen variar ese número entre 0000 y 7777: se trata de los bits de permanencia (1000), SGID (2000) y SUID (4000).

El bit de SUID o SETUID se activa sobre un archivo añadiéndole 4000 a la representación octal de los permisos del archivo y otorgándole además permiso de ejecución al propietario del mismo.

El bit SUID activado sobre un archivo indica que todo aquel que ejecute el archivo, va a tener durante la ejecución los mismos privilegios que quien lo creo; dicho de otra forma, si el superusuario crea un archivo y lo "*setuida*", todo aquel usuario que lo ejecute va a disponer, hasta que el programa finalice, de un nivel de privilegio total en el sistema.

Todo lo que acabamos de comentar con respecto al bit SETUID es aplicable al bit SETGID pero a nivel de grupo del archivo en lugar de propietario: en lugar de trabajar con el UID del propietario, todo usuario que ejecute un programa "*setgidado*" tendrá los privilegios del grupo al que pertenece el archivo. Para activar el bit de SETGID sumaremos 2000 a la representación octal del permiso del archivo y además habremos de darle permiso de ejecución a la terna de grupo; si el archivo es un directorio, el bit SETGID afecta a los archivos y subdirectorios que se creen en el: estos tendrán como

grupo propietario al mismo que el directorio "*setgidado*", siempre que el proceso que los cree pertenezca a dicho grupo.

Los bits de SETUID y SETGID dan a Linux una gran flexibilidad, pero constituyen al mismo tiempo la mayor fuente de ataques internos al sistema (entendiendo por ataques internos aquellos realizados por un usuario - autorizado o no - desde la propia máquina, generalmente con el objetivo de aumentar su nivel de privilegio en la misma).

En un sistema Linux (o UNIX) instalado "*out of the box*" (fuera de la caja) el número de archivos "*setuidados*" suele ser mayor de cincuenta; sin perjudicar al correcto funcionamiento de la máquina, este número se puede reducir a menos de cinco, lo que viene a indicar que una de las tareas de un administrador sobre un sistema recién instalado es minimizar el número de archivos *setuidados* o *setgidados*. No obstante, tampoco es conveniente eliminarlos, sino simplemente reiniciar su bit de SETUID mediante el comando `chmod`.

También debe de estar atentos a nuevos archivos de estas características que se localicen en la máquina; demasiadas aplicaciones de Linux se instalan por defecto con ejecutables "*setuidados*" cuando realmente este bit no es necesario, por lo que a la hora de instalar nuevo software o actualizar el existente debe de reiniciar el bit de los archivos que no lo necesiten. Especialmente grave es la aparición de archivos "*setuidados*" de los que el administrador no tenía constancia (ni son aplicaciones del sistema ni un aplicaciones añadidas), ya que esto casi en el 100% de los casos indica que nuestra máquina ha sido comprometida por un atacante. Para localizar los archivos con alguno de estos bits activos, podemos ejecutar el siguiente comando:

```
find / \( -perm -4000 -o -perm -2000 \) -type f -print
```

Por otra parte, el "*sticky bit*" o bit de permanencia se activa sumándole 1000 a la representación octal de los permisos de un determinado archivo y otorgándole además permiso de ejecución.

Si el bit de permanencia de un archivo esta activado le estamos indicando al sistema operativo que se trata de un archivo muy utilizado, por lo que es conveniente que permanezca en la memoria principal el mayor tiempo posible; esta opción se utilizaba en sistemas antiguos que disponían de muy poca RAM, pero hoy en día prácticamente no se utiliza. Lo que si que sigue vigente es el efecto del "*sticky bit*" activado sobre un directorio: en este caso se indica al sistema operativo que, aunque los permisos 'normales' digan que cualquier usuario pueda crear y eliminar archivos, solo el propietario de cierto archivo y el administrador pueden borrar un archivo guardado en un directorio con estas características. Este bit, que solo tiene efecto cuando es activado por el administrador, se utiliza principalmente en directorios del sistema de archivos en los que interesa que todos puedan escribir pero que no todos puedan borrar los datos escritos, como en /tmp.

Para localizar los archivos que tiene permiso de escritura para todos los usuarios

```
$ find /dir -xdev -perm +o=w ! \( -type d -perm +o=t \) ! -
type l -print
```

Bloqueando el ingreso a la cuenta superusuario

Si deseamos restringir los usuarios que podrán elevar su nivel de privilegios al intentar cambiarse a superusuario (usando el comando su) debemos hacer lo siguiente:
Primero debemos editar el archivo /etc/pam.d/su y agregar las siguientes líneas al archivo:

```
auth sufficient /lib/security/pam_rootok.so debug
auth required /lib/security/pam_wheel.so group=wheel
```

Que indica que solo los miembros del grupo "*wheel*" pueden usar el comando su. El grupo "wheel" es una cuenta especial del sistema que puede ser usada para este propósito. Por lo tanto para implementar este mecanismo el grupo debe ser "*wheel*" necesariamente.

Ahora que hemos definido el grupo "*wheel*" en la configuración del archivo /etc/pam.d/su debemos agregar los usuarios que conformaron este grupo, los cuales podrán usar el comando su para ingresar al sistema como superusuario.

```
usermod -G10 usuario
```

Donde usuario es el nombre del usuario que se agregara al grupo "*wheel*".

Límites de los Recursos del Sistema

Podemos establecer los límites de los recursos usados por cada usuario, para que no sea posible un ataque DoS (p.e. el número de procesos o la cantidad de memoria). Para prohibir por ejemplo la creación de archivos del núcleo (*core*), restringir el máximo de procesos de un usuario a 20 y restringir el uso de memoria a 5 Mb. Editaríamos el archivo /etc/security/limits.conf y agregaríamos lo siguiente:

```
*       hard        core        0
*       hard        rss         5000
*       hard        nproc       20
```

El asterisco (*) significa que cualquier usuario que ingrese al sistema se regirá por estas directivas a excepción del superusuario.

Después de editar el archivo debemos agregar la línea:

```
session required /lib/security/pam_limits.so
```

Al archivo /etc/pam.d/login.

Arreglando los permisos del directorio de scripts /etc/rc.d/init.d

Para arreglar los permisos de este directorio en el cual se encuentran los *scripts* que inician y paran los procesos que necesitan correr al inicio del sistema, como superusuario.

```
chmod -R 700 /etc/rc.d/init.d/*
```

Lo que significa que solamente el superusuario puede leer, escribir y ejecutar los *scripts* de este directorio. Cada vez que se instale un nuevo programa arreglar los permisos del mismo.

A.2 – Directivas de los Servicios de Red

Telnet

Este es un servicio que tiene algunos inconvenientes con respecto a la seguridad:

- Autentificación en texto claro de nombre y contraseña.
- Texto claro en todos los comandos.
- Ataques de adivinación de contraseña.

Para asegurar las cuentas de los usuarios con respecto al servicio Telnet (aunque lo mas recomendable es deshabilitarlo y usar un servicio que utilice la comunicación en forma cifrada como SSH), hay algunas cosas que se pueden hacer, primero no permitir que el superusuario entre al sistema en forma remota vía Telnet, esto se puede controlar con el archivo /etc/securetty, aunque por defecto en la mayoría de las distribuciones esta restringido. Si no se desea que algún usuario ingrese al sistema es estableciendo su shell a /bin/false (o algo no listado en /etc/shells).

Telnet además despliega una "cadena identificativa" (*banner*), cuando alguien se ha conectado al sistema. Esta cadena contiene información del sistema, como el nombre del sistema, que sistema operativo esta corriendo y tal vez algunos otros detalles del mismo. Telnet despliega el contenido del archivo /etc/issue.net (y/o /etc/issue), así que simplemente edita el contenido de ambos archivos para mayor seguridad; pero recordemos que los archivos mencionados (issue e sigue.net) son creados cada vez que

se inicia el sistema, por lo tanto debemos crear un guión (*script*) que cambie los archivos al iniciar o podemos quitar (o comentar) las líneas que crean estos archivos del fichero /etc/rc.d/rc.local y así cuando los modifiquemos ya no serán cambiados cada vez que se inicie el sistema.

SSH (Secure Shell)

El archivo de configuración de este servicio es /etc/sshd/sshd_config.

```
PermitRootLogin no
IgnoreRhosts yes
AllowUsers  ------
DenyUsers   ------
AllowHosts  ------
DenyHosts   ------
```

Host

Esta opción restringe las políticas activadas en el archivo de configuración a solo los anfitriones declarados bajo esta directiva, en la cual pueden usarse comodines. Con esta opción se pueden declarar diferentes políticas de seguridad para diferentes anfitriones usando el mismo archivo de configuración (sshd_config).

ForwardAgent

Esta opción especifica cual agente de autentificación de conexión (si hay alguno) debe ser redireccionado a la máquina remota.

ForwardX11

Esta opción es usada por las personas que usan el servicio Xwindow GUI y desean ser automáticamente redirecionadas a una sesión X11 en la máquina remota.

RhostsAuthentication

Esta opción especifica si deseamos usar autentificación basada en rhosts.

PasswordAuthentication

Especifica si vamos a usar autentificación basada en contraseñas.

`FallBackToRsh`

Especifica si es que la conexión con el demonio de SSH falla la conexión se hará automáticamente por medio de RSH.

`UseRsh`

Especifica si los servicios Rlogin/RSH serán usados en este anfitrión.

`BatchMode`

Especifica si la consulta de conexión basada en nombre y contraseña será deshabilitada.

`CheckHostIP`

Especifica si deseamos o no que SSH adicionalmente verifique la dirección IP del anfitrión que se conecta al servidor para detectar un ataque de IP Spoofing.

`StrictHostKeyChecking`

Especifica si deseamos o no que SSH agregue una nueva llave de anfitrión a $HOME/.ssh/known_hosts automáticamente o no (o sea manualmente se hará).

`IdentityFile`

Especifica un archivo de identidad alternativa para la autentificación RSA.

`Port`

Especifica el puerto por el cual SSH escuchara las peticiones.

`Cipher`

Especifica cual cifrado será usado para las sesiones cifradas.

`EscapeChar`

Especifica el carácter de escape para suspender las sesiones.

RSH, REXEC, RCP

Los servicios r-* de UNIX BSD (aparecieron inicialmente en la versión 4.2 de esta variante de UNIX) son herramientas con una parte cliente y una servidora que permiten la conexión remota entre máquinas, principalmente para servicios de terminal remota y transferencia de archivos. Las herramientas clientes son RSH, Rlogin y RPC, mientras que las servidoras son demonios como rexecd, rshd o rlogind.

Rlogin (puerto 513, TCP) se utiliza como terminal virtual de un sistema UNIX, de una forma muy parecida a Telnet. Rsh (puerto 514, TCP) es utilizado para ejecutar comandos en una máquina remota sin necesidad de acceder a ella, y RPC (vía rsh) para copiar archivos entre diferentes máquinas.

Estos servicios pretenden evitar el transito de contraseñas por la red, ya que este movimiento de claves implica molestias a los usuarios y también problemas de seguridad; para conseguirlo, entran en juego lo que los diseñadores del sistema de red de UNIX BSD denominaron `máquinas confiables' y `usuarios confiables': cualquier usuario, puede hacer uso de recursos de una máquina remota sin necesidad de una clave si su conexión proviene de una máquina fiable o su nombre de usuario es fiable. Una máquina se puede considerar fiable de dos formas: o bien su nombre se encuentra en el archivo /etc/hosts.equiv, o bien se encuentra en un archivo denominado .rhosts y situado en el $HOME de algún usuario. Si estamos en el primer caso, cualquier usuario (excepto el superusuario) del sistema remoto (y fiable) puede hacer acceder a nuestro equipo bajo el mismo nombre de cuenta que tiene en el primero, sin necesidad de contraseñas. En el segundo caso, utilizando los archivos .rhosts, cualquier usuario del sistema remoto podrá conectarse al nuestro pero solo bajo el nombre de usuario en cuyo $HOME se encuentra el archivo.

Como podemos ver, las relaciones de confianza entre equipos UNIX pueden ser muy útiles y cómodas, pero al mismo tiempo muy peligrosas: estamos confiando plenamente en sistemas remotos, por lo que si su seguridad se ve comprometida también se ve la

nuestra. Las máquinas fiables se han de reducir a equipos de la misma organización, y administrados por la misma persona; además, es necesario tener siempre presente que si tenemos habilitados los servicios r-* cualquier usuario puede establecer relaciones de confianza, lo que puede suponer una violación de nuestra política de seguridad. Es conveniente revisar los directorios $HOME en busca de archivos .rhosts.

Las relaciones de confianza son transitivas: si una máquina confía en otra, lo hace también en todas en las que confía ella. De esta forma se crean anillos de confianza entre máquinas, y como las relaciones suelen estar basadas en el nombre del equipo se trata de objetivos ideales para un atacante mediante *IP Spoofing*: si un intruso consigue hacer pasar su equipo por uno de los confiables, automáticamente ha conseguido acceso (casi ilimitado) al resto de las máquinas.

Para desactivarlos debemos editar el archivo del servicio RPC que usualmente se encontrara dentro de la carpeta /etc/xinetd.d y estableciendo el valor *yes* en la directiva *disable.*

```
disable = yes
```

Si irremediablemente deben estar activados los servicios mencionados anteriormente, debemos asegurarnos que ningún usuario tenga en sus documentos un archivo de ingreso al sistema (.rhosts); ni que exista un archivo /etc/hosts.equiv. Estos archivos son los que usan los servicios ya mencionados para establecer anfitriones y usuarios "confiables".

El siguiente guión busca en cada directorio de inicio de cada usuario archivos .rhosts.

```
#!/bin/sh
        for i in `cut -d: -f6 /etc/passwd`; do
           if [ -e $i/.rhosts ]; then
              echo "SECURITY CHECK found .rhosts in $i"
           fi
```

done

/etc/hosts.equiv

En este archivo se indican, una en cada línea, las máquinas confiables, confiables significa que confiamos en su seguridad tanto como en la nuestra, por lo que para facilitar la comparición de recursos, no se van a pedir contraseñas a los usuarios que quieran conectar desde estas máquinas con el mismo nombre de usuario, utilizando las ordenes r* (Rlogin, RSH, RPC. . .). Por ejemplo, si en el archivo /etc/hosts.equiv del servidor maquina1 hay una entrada para el nombre de anfitrión maquina2, cualquier usuario1 de este sistema puede ejecutar un comando para conectarse a maquina1 sin necesidad de ninguna clave.

Obviamente, esto supone un gran problema de seguridad, por lo que lo más recomendable es que el archivo /etc/hosts.equiv este vació o no exista.

FTP (File Transfer Protocol)

El servicio es inseguro en si mismo, contraseñas, datos, etc. Son transferidos en texto claro y fácilmente puede ser husmeado, de cualquier manera la forma más usada del servicio es con el usuario *anonymous*. Uno de los problemas principales de este servicio es la configuración de los permisos de los directorios principales de almacenamiento que permiten a los usuarios redistribuir sus propios datos, que en muchos casos pueden ser datos con derechos reservados.

Los problemas generales con FTP son:

- Autentificación en texto claro, usuario y contraseña.
- Comandos en texto claro.
- Ataques de adivinación de contraseña.
- Instalación impropia del servidor que deriva en obtención de máximos privilegios.
- Ataques DoS.

Para eliminar el servicio anónimo del FTP debemos editar el archivo /etc/ftpaccess y cambiar la línea

```
class   all   real,guest,anonymous   *       por
class   all   real  *
```

Y finalmente eliminar el directorio de inicio del usuario anónimo de FTP

```
rpm -e anonftp.
```

Algunas versiones del demonio de FTP permiten un control a través del archivo /etc/ftpaccess, como por ejemplo no permitirles modificar archivos:

```
chmod      no guest,anonymous
delete     no guest,anonymous
overwrite  no guest,anonymous
rename     no guest,anonymous
```

Existe una forma de automatizar FTP para que no solicite nombre ni contraseña, y es mediante el uso de un archivo situado en el directorio $HOME de cada usuario (en la máquina desde donde se invoca a FTP, no en el servidor) llamado .netrc. En este archivo se pueden especificar, en texto claro, nombres de máquina, nombres de usuario y contraseñas de sistemas remotos, de forma que al conectarse a ellos la transferencia de estos datos se realiza automáticamente, sin ninguna interacción con el usuario.

La existencia de archivos .netrc en los directorios $HOME de los usuarios se puede convertir en un grave problema de seguridad: si un atacante consigue leer nuestro archivo .netrc, automáticamente obtiene nuestro nombre de usuario y nuestra clave en un sistema remoto. Por tanto, no es de extrañar que para que el mecanismo funcione correctamente, este archivo solo puede ser leído por su propietario; si no es así, no se permitirá el acceso al sistema remoto (aunque los datos de .netrc sean correctos). Este mecanismo solo se ha de utilizar para conexiones a sistemas remotos como usuario anónimo (*anonymous* o *ftp*).

Quizás sea conveniente rastrear periódicamente los directorios de conexión de nuestros usuarios en busca de archivos .netrc.

```
#!/bin/sh
        for i in `cut -d: -f6 /etc/passwd`; do
           if [ -e $i/.netrc ]; then
              echo "SECURITY CHECK found .rhosts in $i"
           fi
done
```

Sendmail

En la mayoría de los casos el servidor de Sendmail aceptara conexiones al puerto 25 mediante Telnet, una vez conectado un intruso puede obtener información de algún usuario casi momentáneamente con el siguiente comando:

```
expn nombre_de_usuario
```

Este comando solicita al programa de correo que despliegue de nombre_de_usuario su dirección de correo electrónico y su nombre real. El resultado será típicamente así:

```
nombre_de_usuario          nombre_usuario@servidor_correo.com
nombre_real
```

El comando expn puede ser deshabilitado por el administrador del sistema. A pesar de esto el intruso puede verificar la existencia de la cuenta de correo usando el comando vrfy. Lo más recomendable seria deshabilitar ambos comandos. Lo cual puede hacerse editando el archivo de configuración de Sendmail (/etc/sendmail.cf), aunque la mayoría de los demonios más recientes traen deshabilitadas ambas opciones por defecto.

Cambiamos la línea:

```
O PrivacyOptions=authwarnings
```

Por

```
O PrivacyOptions=authwarnings,noexpn,novrfy
```

Para modificar la "cadena identificativa" (*banner*) que despliega Sendmail al conectarnos al servidor podemos editar el archivo /etc/sendmail.cf de la siguiente manera:

Cambiamos la línea:

```
O SmtpGreetingMessage=$j Sendmail $v/$Z;$b
```

Por

```
O SmtpGreetingMessage=$j Sendmail $v/$Z;$b NO UCE C=xx L=xx
```

Donde "xx" corresponden al código de ubicación de tu país y estado. Para más información revisar el manual de configuración de Sendmail.

Apache

El archivo httpd.conf es el archivo principal de configuración del servidor Apache. Algunas de las directivas más importantes del mismo son:

`ServerType`

Esta opción especifica si el servidor funcionara como *standalone*, que significa que el demonio de Apache correrá como un demonio independiente o como *super-server* de *xinetd*, que estará embebido en el demonio de *xinetd*. Se recomienda configurarlo como demonio independiente.

`ServerRoot`

Especifica la dirección de los archivos de configuración de Apache.

`Timeout`

Especifica la cantidad de tiempo que Apache esperara por una petición *GET*, *POST*, *PUT* y trasmisiones *ACK*.

`KeepAlive`

Si se especifica *On* en esta directiva, se habilitarán las conexiones persistentes en el servidor y se permitirán más de una petición de conexión.

`MaxKeepAliveRequests`

Especifica el número máximo de peticiones permitidas por conexión si la directiva anterior esta permitida. Si se especifica *0*, un número de peticiones ilimitadas son permitidas en el servidor.

`KeepAliveTimeout`

Especifica el tiempo (en segundos), que Apache esperara una petición subsiguiente después de ser cerrada la conexión.

`StartServers`

Especifica el número de procesos hijos del servidor que serán creados cuando se inicie el demonio Apache. Se recomienda el número *16*.

`MaxClients`

Especifica el número máximo de peticiones simultáneas que soportara Apache. Para una carga alta de operación se recomienda el número *512*.

`MaxRequestsPerChild`

Especifica el número de peticiones que un proceso hijo puede manejar simultáneamente.

`User`

Especifica la Identificación de Usuario (*UID*, por sus siglas en inglés), con el que correrá el servidor Apache; se recomienda crear un nuevo usuario con el nivel mínimo de privilegios y adecuarlo para el propósito de solo correr el servidor Apache.

Group

Especifica la Identificación de Grupo (*GID*, por sus siglas en inglés), con el que correrá el servidor Apache; se recomienda crear un nuevo grupo con el nivel mínimo de privilegios y adecuarlo para el propósito de solo correr el servidor Apache.

POP

El servicio POP (*Post Office Protocol*, puertos 109 y 110 en TCP) se utiliza para que los usuarios puedan acceder a su correo sin necesidad de montar sistemas de archivos compartidos mediante NFS: los clientes utilizan SMTP para enviar correo y POP para recogerlo del servidor, de forma que el procesamiento se realice en la máquina del usuario. Se trata de un servicio que podemos considerar peligroso, por lo que debemos deshabilitarlo a no ser que sea estrictamente necesario ofrecerlo.

- Autentificación en texto claro de nombre y contraseña.
- Texto claro en todos los comandos.
- Ataques de adivinación de contraseña.
- Corre como superusuario, se derivan varios *exploit* para superusuario

IMAP

Inicia corriendo como superusuario, pero decrecen sus privilegios a los mismos del usuario que lo acceda, aunque no es tan sencillo ya que tiene que acceder a bandejas.

NFS

Es una buena manera de distribuir sistemas de archivos, si se asume que la red esta asegurada y encerrada. NFS se usa solo en sistemas de banda ancha y donde los riesgos de seguridad no son altos y la información que será compartida no es sensible. Si se ocupa un alto nivel de seguridad es necesario cifrar los datos, y NFS no es una buena opción. Las únicas opciones de "seguridad" con que cuenta el servicio NFS son:

/etc/exports

La configuración de este archivo debe estar lo más restringida posible. Esto significa no usar comodines, no permitir acceso de escritura al superusuario y montando el sistema de archivos solo en modo lectura lo mas posible que se pueda.

Directivas de exports:

```
ro - solo lectura
noaccess - prohibir el acceso
root_squash - los privilegios del cliente NFS decrecen a
un usuario con privilegios menores (como nobody)
```

Directivas de montaje del sistema de archivos a exportar:

```
noexec - no permite la ejecución de binarios
nosuid - no permite binarios uid ni gid
nodev - no interpreta archivos de bloque o de caracteres
        especiales.
```

SNMP

Fue diseñado para permitir a sistemas y equipos heterogéneos poder comunicarse entre si, reportar datos y permitir modificaciones de sus características a través de una red TCP/IP. Por ejemplo un dispositivo activado SNMP (como un ruteador Cisco), puede ser monitoreado/configurado desde un cliente SNMP. Desafortunadamente SNMP no tiene seguridad implementada, al igual que no la tiene la versión 2 de SNMP (SNMPv2). Las únicas opciones para asegurar un poco SNMP, es cambiar la cadena comunitaria, (que por defecto en la mayoría de las implementaciones es "*public*"), por alguna contraseña fuerte. Aunque esto podría no servir para mucho, ya que la comunicación podría ser husmeada (ya que está en texto claro). Solo permitir acceso de solo lectura a las capturas del agente de SNMP cuando sea posible. La opción mas viable seria deshabilitar (o bloquear) este servicio si el segmento de la red no tiene ningún dispositivo SNMP habilitado. La versión 3 del servicio (SNMPv3) sí tiene directivas de seguridad.

Finger

Este servicio debe estar habilitado solo en casos realmente necesario. Para deshabilitarlo debemos seguir los mismos pasos para los servicios RPC, únicamente buscando el archivo del servicio finger, en vez del servicio RPC, ya que hay una variedad de ataques DoS a este servicio.

LPD

Es el demonio de impresión de UNIX, ya que el servicio acceda a varias partes del sistema (impresoras, demonios de ingreso, etc.) es un agujero de seguridad en potencia, y ha sido fuente de muchos ataques de superusuario. El control de acceso al demonio se encuentra en los archivos /etc/hosts.equiv y /etc/hosts.lpd, donde debemos configurar el acceso al servicio.

Samba

Este servicio nos permite compartir archivos e impresoras a través de Windows y cualquier UNIX (incluyendo Linux), simplemente dando acceso al sistema de archivos vía SMB (por sus siglas en inglés, *Server Message Block*), el protocolo de Windows para compartir carpetas e impresoras. Este verifica el nombre y contraseña (si se requiere) y da acceso a las carpetas de acuerdo a los permisos establecidos.

Para controlar el acceso al servicio Samba el archivo de configuración de Samba (/etc/smb.conf), tiene algunas directivas de seguridad.

```
Security = xxxx
```

Donde xxxx representa el servidor o dominio que será compartido, la autentificación la hace Samba vía el archivo /etc/password o smbpasswd, si es un dominio, samba autentifica vía la controladora de dominio de NT de la red NT.

```
guest account = xxxx
```

Donde xxxx es el nombre de usuario de la cuenta, que será usada si la forma de compartir los archivos es definida como *public*.

`Hosts allow = xxxx`

Donde xxxx representa separados por espacio los anfitriones que serán aceptados para conectarse al servicio, esta directiva nos puede servir para restringir el acceso.

`Hosts deny = xxxx`

Donde xxxx representa separados por espacio los anfitriones que serán rechazados para conectarse al servicio.

`Interfaces = xxxx`

Donde xxxx representa las interfaces separadas por espacio desde la cual escuchara las peticiones Samba.

Recordar activar la opción para cifrar contraseñas.

Para una administración más sencilla, Samba nos provee de una herramienta llamada SWAT (por sus siglas en inglés, *Samba Web Administration Tool*), el inconveniente es que para acceder a la herramienta debemos ingresar como superusuario y la información está en texto claro.

Se deben revisar los archivos /etc/hosts.allow y /etc/hosts.deny si se tiene configurado TCP WRAPPERS.

A.3 - Directivas de administración (solo para sistemas Windows)

Algunas políticas sobre la cuenta Administrador

- Renombrar la cuenta.
- Contraseña fuerte.
- Todas las tareas de administración se deben hacer desde la consola de administración en un ambiente controlado.
- Si van a existir diferentes cuentas de administrador, crearlas y configurarlos por separado.

- Monitorear y auditar las cuantas con privilegio de administrador regularmente.

Las horas permitidas de inicio de sesión se puedan usar para prevenir que usuarios inicien el sistema fuera de las horas regulares de trabajo.

Considerar quitar los archivos ejecutables del sistema. Colocarlos los archivos siguientes de línea de comandos en un directorio que no sea el árbol %Systemroot%, y solo permitir el acceso al administrador: cmd.exe, net.exe, arp.exe, atp.exe, atsbc.exe, cacls.exe, ftp.exe, ipconfig.exe, nbtsat.exe, netstat.exe, nslookup.exe, ping.exe, Regedit.exe, regedt32.exe, rexec.exe, route.exe, rsh.exe, secfixup.exe, debug.exe, edit.com, edlin.exe, finger.exe, posix.exe, qbasic.exe, rcp.exe, rdisk.exe, syskey.exe, telnet.exe, tracerout.exe, xcopy.exe.