“SEGURIDAD EN LA FAMILIA DE PROTOCOLOS TCP/IP Y SUS SERVICIOS ASOCIADOS”
Segunda y última parte.

(**Autor: Lic. JUAN JOSE TALENTI –** correo electrónico: jjtalent@arnet.com.ar // Egresado del CMN como Subt A en 1986 - Docente en el IAC Filial Santa Fe - Beta Tester de Microsoft - Participa del Programa “Desarrollador Cinco Estrellas” que capacita a programadores de la plataforma NET).

Resumen

En esta segunda y última parte, se continúa con el desarrollo de otras **VULNERABILIDADES GENÉRICAS**, entre ellas: Denial of Service, Net Flood, Smurf, TCP Syn Flood, Ddos, Trinoo, Ping of death, Loki, Land, Routing Protocols, Source Routing, Caballos de Troya, etc.
Además de una breve reflexión sobre el **FUTURO** de las redes bajo TCP/IP y, para finalizar, las **CONCLUSIONES** de todo el trabajo.

Desarrollo

(Segunda Parte)

3. VULNERABILIDADES GENÉRICAS.

3.10. DoS: Denial of Service.

En los protocolos TCP/IP, se analizan los ataques basados en la denegación de servicio (DoS) desde el exterior de un sistema, a través de la red, y no una vez se disponga de acceso de administrador en el mismo. En este segundo caso, en los sistemas Unix sería tan sencillo efectuar un ataque de este tipo como eliminar todos los archivos del sistema mediante el comando “rm -rf / &”, dependiendo el restablecimiento del servicio de la política de backup del sistema.

Si se tiene acceso a los dispositivos de red, éstos pueden rearrancarse o apagarse, con la implicación que tendría en las comunicaciones de la red de la organización afectada. Un ataque de denegación de servicio se centra en sobrepasar los límites de recursos establecidos para un servicio determinado, obteniendo como resultado la eliminación temporal del servicio. Por ejemplo, si un servidor es capaz de procesar 10 peticiones por segundo, y se le envían 30, parte del tráfico legítimo no recibirá servicio, o incluso, puede que la saturación del tráfico provoque que el servidor deje de responder a ninguna petición.

Los destinos de estos ataques suelen ser objetivos visibles, como servidores Web, o DNS, o elementos básicos de la red, *routers* o enlaces de red. Este tipo de ataques no supone ningún peligro para la seguridad de las máquinas, ya que no modifica los contenidos de la información, por ejemplo páginas Web, ni permite obtener información sensible. Simplemente persiguen entorpecer el acceso de los usuarios a los servicios de un sistema. Normalmente, una vez que el ataque finaliza, se vuelve a la situación normal.

En algunas ocasiones se han empleado para encubrir otros ataques simultáneos cuyo objetivo sí era comprometer el sistema. Asimismo, la probabilidad de que el administrador, intentando defenderse del DoS cometa un error de configuración es mayor en el momento del ataque, pudiendo dejar al descubierto una vulnerabilidad protegida previamente.

Los tres protocolos en los que se basan las técnicas de saturación de paquetes, o *flooding*, son TCP, UDP e ICMP. Los ataques y herramientas más empleados en estos años para llevar a cabo ataques DoS y DdoS son: *Smurf, Fraggle, Trinoo, TFN, Stacheldraht, TFN2K, mstream, t0rnkit, Trinity DDoS, erkms, li0n, carko, w0rmkit*, así como algunos virus y/o gusanos, *VBS/LoveLetter, Ramen worm, VBS/OnTheFly, cheese worm, sadmin/IIS worm, W32/Sircam, Leaves, CodeRed, CodeRed II, Knight/Kaiten, Nimda.*

Asimismo, las vulnerabilidades que se han explotado correspondían a los servidores de nombres (BIND), IIS, e-mail, telnet, SMB. A grandes rasgos en este periodo se ha observado que se ha pasado de disponer de los agentes de un DoS solo en plataformas Unix, a disponer de ellos también en el entorno Windows. Esto, unido al incremento de conexiones de banda ancha en los hogares (tecnologías xDSL y cable) ha incrementado la potencia de los ataques que pueden realizarse.

Asimismo, debido a la potencia del ancho de banda del que disponen los *routers*, éstos están

siendo empleados como agentes para originar ataques DoS contundentes. En los apartados posteriores se analizarán de forma particular vulnerabilidades que se englobarían dentro de los DoS.

3.11. Net Flood.

El objetivo de este ataque es degradar la capacidad de conexión a la red de un sistema, saturando sus enlaces de comunicaciones. Por ejemplo, si el enlace de una organización dispone de un ancho de banda de 34 Mb. y un atacante dispone de un enlace de 155 Mb., prácticamente la totalidad del tráfico cursado por la organización pertenecerá al atacante, por lo que no podrá enviarse tráfico útil.

Para disponer de altos anchos de banda puede recurrirse a la obtención de múltiples sistemas desde los que efectuar el ataque o apoderarse de sistemas mal administrados y protegidos que posean redes de gran capacidad, como por ejemplo, los existentes en las universidades.

Las dos técnicas aplicadas en este tipo de ataques se basan en los protocolos ICMP y UDP, al tratarse de protocolos no orientados a conexión y que permiten el envío de paquetes sin requisitos previos: *ICMP Flood y UDP Flood*.

3.12. Smurf.

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 de *broadcast*, es decir, aquella que representa a todas las máquinas de una red, se utiliza en algunos protocolos para localizar el sistema que proporciona un servicio concreto de forma sencilla, es decir, preguntando a la red, y no consultando uno por uno a todos los sistemas existentes.

Si esta dirección se encuentra disponible también para usuarios externos a la red, es posible que un atacante pueda enviar un paquete de datos a la misma, provocando que todos los sistemas pertenecientes a dicha red respondan simultáneamente, aumentando la potencia de la respuesta en un factor de N, siendo N el número de máquinas disponibles en la red. Es decir, se realiza un ataque a una red desde otra red intermedia que permite multiplicar los recursos existentes (elementos válidos para desarrollar ataques DDoS).

Este método no implica tener que controlar las redes empleadas como multiplicadoras del efecto de ataque. Si se aúna está técnica junto a la de *IP spoofing*, al enviar un paquete ICMP con la dirección IP origen de la máquina a atacar y dirección IP destino la dirección de *broadcast* de una red con un elevado número de máquinas, digamos cientos, todas las respuestas de la red de *broadcast* se dirigirán realmente a la dirección IP del sistema "*spoofeado*". Este ataque es denominado realmente *Smurf*.

3.13. TCP Syn Flood.

Dentro de los ataques DoS, existe uno asociado directamente al protocolo TCP. Consiste en el envío masivo de paquetes de establecimiento de conexión (SYN) contra un sistema. La recepción de estas solicitudes provoca que el sistema destino, objetivo del ataque, reserve cierta cantidad de memoria (*buffers*) para almacenar las estructuras de datos asociadas a cada una de las nuevas conexiones en curso.

El protocolo TCP requiere del establecimiento de una conexión, que se realiza en tres pasos. Tras la recepción del paquete SYN, responderá con su paquete SYN-ACK, permaneciendo a la espera del paquete final (ACK) que confirma el establecimiento de la conexión TCP (*three-way handshake*).

La conexión permanece en el estado semiabierto, concretamente SYN_RCVD. El atacante no enviará nunca ese ACK esperado, por lo que la memoria del destino es copada en su totalidad por conexiones falsas, no siendo posible el establecimiento de conexiones de clientes reales, y por tanto anulándose el servicio. Asimismo, ciertos sistemas imponen un número máximo de conexiones en este estado, por lo que una vez alcanzado éste, no será posible establecer más conexiones.

Tras un periodo de tiempo controlado por un temporizador (que suele ser de 2 minutos), las conexiones que continúan en este estado expiran, permitiendo la creación de nuevas conexiones. Esto solo será posible si el ataque *TCP SynFlood* ha cesado, ya que mientras se

mantenga, serán sus nuevos inicios de conexión los que ocuparán el espacio de memoria liberado por las sesiones expiradas. Suponiéndose un número máximo de conexiones igual a 30, y el temporizador igual a 2 minutos, se podría desarrollar un ataque de este tipo enviando un paquete SYN cada 4 segundos: tiempo necesario por cada conexión para poder enviar el máximo de 30 conexiones en los 120 segundos de expiración.

Normalmente, para que su detección sea más compleja este ataque se realiza variando la dirección IP del emisor, mediante direcciones falsas (*IP Spoofing*), de forma que se simule de forma más fehaciente una situación real de conexiones realizadas por multitud de clientes. Algunas herramientas dedicadas a este tipo de ataques son: “Neptune” y “Synk4”.

3.14. Connection Flood.

Los servicios TCP orientados a conexión, que son la mayoría (telnet, ftp, http, smtp, nntp...) tienen un límite máximo de conexiones simultáneas soportadas; cuando este límite se alcanza, cualquier conexión nueva es rechazada. De forma similar al *Syn Flood*, si un atacante es capaz de monopolizar el límite definido con conexiones de su propiedad, que simplemente son establecidas pero por las que no se realiza ninguna comunicación posterior, el sistema no proporcionará servicio.

Al igual que antes, las conexiones expiran progresivamente con el paso del tiempo, pero un ataque constante de apertura de conexiones mantendrá continuamente el límite en su valor máximo. La diferencia está en que en este caso la conexión se ha establecido y por tanto se conoce la identidad del atacante (dirección IP), y a su vez, la capacidad del sistema o sistemas atacante/s debe ser lo suficientemente elevada como para mantener abiertas todas las sesiones que colapsan el servidor atacado.

Existe una variante de estos ataques basada en el uso de un cliente que establezca conexiones contra un sistema, pero que no las finalice de forma correcta, de modo que en el servidor los *sockets* correspondientes a estas comunicaciones seguirán estando activos y consumiendo recursos, concretamente en el estado TCP denominado TIME_WAIT.

3.15. SMTP Flood.

Mediante el envío masivo de mensajes de correo electrónico a grandes listas de usuarios de forma continua, se provoca la saturación de los servidores de correo destino o intermedios.

3.16. DdoS.

Una variante más potente a la de los ataques de Denegación de Servicio, son los DDoS, o ataques de denegación de servicio **distribuídos**, 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. Si un ataque desde una fuente es potente, desde 1000 lo será mucho más, es decir, es la aplicación del “divide y vencerás” a la técnica DoS. Su origen se remonta a los comienzos de la seguridad en Internet, cuando el famoso *Gusano* de Robert Morris, Jr. desencadenó una denegación de servicio por un error de programación.

El gusano fue capaz de colapsar por aquel entonces gran parte de los sistemas existentes en Internet. Sin embargo su expansión se ha producido principalmente en el año 2000, haciéndose eco los medios de comunicación, al surgir numerosas herramientas que permiten su ejecución, de forma coordinada y a gran escala.

Un único atacante puede desencadenar una agresión desde centenares de máquinas repartidas por todo el mundo, como ha ocurrido en las Webs de Yahoo, Amazon, CNN, eBay, Buy, ZDNet.

Dado el elevado número de sistemas existentes en Internet, la capacidad de “reclutar” recursos es inmensa. Debido a las vulnerabilidades de los sistemas operativos y de las aplicaciones, como los *buffer-overflows* y los *format-strings*, un atacante es capaz de apoderarse de un conjunto de sistemas (de cientos a miles) e instalar en ellos un servicio que acepte órdenes del atacante para ejecutar un DDoS contra una máquina objetivo.

La sofisticación de las herramientas actuales hace que no se requieran conocimientos técnicos avanzados para llevar a cabo este tipo de ataques: ellas se encargan de analizar y vulnerar los sistemas, para copiarse e instalarse automáticamente (en unos segundos).

El proceso esta compuesto de 4 pasos principales:

- **Fase de escaneo con un conjunto objetivo de sistemas muy elevado, 100.000 o más. Se prueban éstos frente a una vulnerabilidad conocida.**
- **Se obtiene acceso a parte de esos sistemas a través de la vulnerabilidad.**
- **Se instala la herramienta de DDoS en cada sistema comprometido.**
- **Se utilizan estos sistemas para escanear y comprometer nuevos sistemas.**

El modo de operación genérico de las herramientas de DDoS tiene la siguiente topología: el intruso se comunica mediante comandos con un elemento denominado *handler*. Éste se encarga de gestionar el registro, realizado previamente, de un conjunto de agentes, normalmente elevado en número, que son realmente el origen de los paquetes del DDoS.

Por tanto, los agentes y el *handler* conforman una red de ataque, que actúa en el momento en que el *handler* retransmite a todos y cada uno de los agentes las órdenes invocadas por el intruso remotamente.

Las comunicaciones entre estos elementos se realizaban originalmente por puertos fijos y, a la larga, conocidos, por lo que este modo de funcionamiento podía ser detectado por sistemas IDS con facilidad. La difusión en el uso del IRC o *chat*, a dado lugar a la utilización de este medio (y sus puertos TCP asociados, del 6660 al 6669) para constituir los canales de control de los elementos de un DDoS.

3.17. Trinoo.

Esta herramienta de DDoS, aunque antigua, permite el acceso a través de autenticación basada en claves (mediante crypt()), y a su vez, permite determinar si un binario concreto de una máquina actúa como maestro o como esclavo. Inicialmente surgió por la explotación de un *buffer-overfow* en los sistemas que actuaban de víctimas.

Existe una versión asociada a Windows, llamada *WinTrinoo*. Los sistemas maestros disponen de una lista con los *hosts* que pueden ser controlados, a través de los que se puede realizar el ataque distribuido.

La comunicación entre los maestros y los esclavos (o demonios) no está encriptada, y se realiza típicamente (por defecto) a través de los siguientes puertos, por lo que es sencillo de detectar si no se han modificado:

TCP: 1524 27665 (client-master).

UDP: 27444 31335 (master-server).

3.18. Tribe Flood Network y TFN2K.

La comunicación entre clientes y servidores se realiza a través de paquetes de ping: *ICMP echo request* e *ICMP echo reply*, aunque posibilita ataques DoS basados en *ICMP flood*, *SYN flood*, *UDP flood*, y *Smurf*, así como obtener una *shell* de *root* asociada a un puerto TCP seleccionado.

La comunicación entre clientes y servidores no emplea puertos concretos. Éstos pueden determinarse en el momento de la ejecución o pueden elegirse aleatoriamente en el propio programa, pero consisten en una combinación de los protocolos ICMP, TCP y UDP. Asimismo añade capacidades de encriptación, eliminando así la detección por los sistemas IDS.

3.19. Stacheldraht.

Esta herramienta es una combinación de las anteriores, creando una sesión *telnet* encriptada entre clientes y servidores. La comunicación se realiza típicamente (por defecto) a través de los siguientes puertos, por lo que es sencillo de detectar si no se han modificado:

TCP: 16660 65000.

ICMP echo request e ICMP echo reply.

3.20. Ping of death.

Conocido como ping de la muerte, este ataque se basa en enviar un paquete de ping (*ICMP echo request*) de un tamaño muy grande. Teniendo en cuenta que el tamaño máximo de paquete en TCP/IP es de 64 *Kbytes* (65535 *bytes*), 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. El resultado obtenido en muchas ocasiones es que el sistema destino deja de proveer servicio al bloquearse, ya sea rearrancándose (*reboot*) o incluso apagándose (*shutdown*).

Lo que sucede realmente es que el paquete emitido es fragmentado, a nivel de IP, en las redes intermedias, los fragmentos van siendo encolados en el sistema destino hasta que se reciben los último pedazos (un paquete de 65536 *bytes* es suficiente), que son los que desbordan el buffer, provocando un comportamiento anómalo. En el momento de la aparición de este ataque (1996) muchas de las implementaciones TCP/IP, tanto de Unix, de Windows, como de los dispositivos de red fueron vulnerables. Simplemente mediante el envío de un paquete cuyo campo de datos sea mayor de (65507 *bytes* = 65535 – 20 –8) se puede llevar a cabo. Debe tenerse en cuenta que los SS.OO. actuales no permiten al cliente de *ping* enviarlo, obteniéndose respuestas como "*ping: illegal packet size*" (HP-UX) o "*Bad value for option -l, valid range is from 0 to 65500*" (Windows 2000):

Unix: # ping victima.com 65510.

Windows: c:\>ping -l 65510 victima.com

Para realizar este ataque es necesario disponer de una herramienta que lo implemente o modificar el límite impuesto en el código fuente del cliente de *ping*, por ejemplo en Linux.

Asimismo, existen otros ataques basados en la fragmentación de paquetes ICMP. Todos los tipos de paquetes ICMP son de un tamaño reducido, por lo que la detección de un paquete de gran tamaño debería dar lugar a sospechas.

3.21. Loki.

Este ataque fue inicialmente el nombre de un proyecto, pasando posteriormente a convertirse en una herramienta cuyo objetivo es demostrar la posibilidad de encubrir tráfico en túneles ICMP y UDP, bajo lo que se ha dado en denominar canales encubiertos.

En el caso de que este tráfico este permitido a través de los *firewalls*, el ataque es posible. Debe tenerse en cuenta que en la mayoría de las ocasiones es necesario habilitar al menos ciertos tipos de paquetes ICMP, como los de la familia *unreachable*, para que funcionalidades de la pila TCP/IP se desarrollen, por ejemplo, el algoritmo PMTUD, *Path MTU Discovery*.

El objetivo del ataque es introducir tráfico encubierto, típicamente IP, en paquetes ICMP (o UDP) que son permitidos. La herramienta consta de un cliente, loki, y un servidor, lokid, que se encargan de encapsular y desencapsular el tráfico en ambos extremos.

3.22. Land.

Este ataque permite bloquear un sistema, mediante el envío 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 también son iguales.

Para ello es necesario enviar paquetes IP mediante la técnica de *spoofing*. Debe tenerse en cuenta que algunos sistemas IDS detectan la primera situación y otros la segunda. Por tanto, podría darse algún caso en el que se establezca una conexión a la propia máquina, se envíe por tanto un paquete [127.0.0.1:puerto_cliente ==> 127.0.0.1:puerto_servidor], y el sistema IDS lo detecte como un ataque cuando en realidad no lo es.

Este ejemplo, aplicable a un gran número de las vulnerabilidades mencionadas, refleja la estrecha línea existente entre un ataque real y una situación convencional, denotando que su detección y automatización no es trivial.

Esta técnica afecta a implementaciones tanto de sistemas como de dispositivos de red, como los equipos Cisco. Las versiones afectadas de Cisco pueden obtenerse de una página en la que publican los denominados "*Field Notices*", es decir, noticias de implementación relacionadas con sus productos. El *exploit*, al igual que muchos de los comentados a lo largo de los diferentes apartados posee la extensión ".c", por estar programado en lenguaje C, por tanto se denomina "*land.c*". Existe un ataque similar que en lugar de enviar un único paquete TCP, envía grupos de éstos a la vez que realiza un escaneo de puertos en un rango concreto de direcciones. Éste se denomina "*latierra.c"*.

3.23. Routing protocols.

Los protocolos de enrutamiento pueden ser vulnerados principalmente mediante la introducción de paquetes de actualización de rutas, de forma que es posible adecuar y condicionar los caminos que seguirá el tráfico según un criterio específico.

Uno de los protocolos que puede ser falseado (*spoofing*) es RIP, en su versión 1, RFC 1058, y 2, RFC 1723. Se trata de un protocolo UDP (puerto 520), por tanto acepta paquetes de cualquier sistema sin necesitar ninguna conexión previa. La versión 1 no dispone de sistema de autenticación, mientras que la versión 2 presenta un método basado en el envío de claves en claro de 16 *bytes*.

Para vulnerar RIP, como se especifica a continuación, es necesario inicialmente identificar un *router* que hable este protocolo a través de la identificación del puerto UDP 520. En el caso de pertenecer al mismo segmento de red, deben escucharse las actualizaciones RIP que circulan por la red o solicitárselas directamente a alguno de los *routers*. De esta forma se obtendrá la tabla de rutas que se anuncia en ese momento.

Si no se está en el mismo segmento, se dispone de herramientas como rprobe para realizar una petición RIP remota: el resultado se obtendrá mediante un *sniffer* en el sistema desde el que se ataca. Una vez definida la información que se pretende inyectar en la tabla de rutas anunciada, por ejemplo, redireccionar todo el tráfico a un sistema desde el que se pueda analizar el mismo, mediante utilidades como srip, se inyectará la ruta deseada.

A partir de ese momento todo el flujo de tráfico pasará por el nuevo camino definido. Para que el funcionamiento habitual no se vea modificado, es necesario que el nuevo sistema al que van destinados los paquetes los redireccione consecuentemente: *ip forwarding*.

A lo largo del texto se muestra, en los diferentes análisis de las vulnerabilidades y sus defensas, como modificar los parámetros TCP/IP en el *kernel* de los distintos sistemas operativos mencionados.

El primer ejemplo de configuración de este tipo se aplica a la capacidad de los *kernels* para hacer *routing* entre varios interfaces. Ésta se configura como sigue:

- **En HP-UX basta con introducir en el archivo */etc/rc.config.d/nddconf*:**

  TRANSPORT_NAME[1]=ip

  NDD_NAME[1]=ip_forwarding

  NDD_VALUE[1]=1

- **En Solaris basta con ejecutar mediante ndd en un script de arranque:**

  ndd –set /dev/ip ip_forwarding 0

- **En Linux, se debe añadir también en un script de arranque:**

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

  En el caso de Linux es posible especificar los parámetros en el archivo /etc/sysctl.conf (p.ej., Red Hat) que es cargado por la utilidad /sbin/sysctl. La documentación al respecto se encuentra en el directorio /Documentation del *kernel*, en el archivo proc.txt.

- **En Windows las modificaciones se realizan a través del editor del registro (regedt32.exe); para ello debe localizarse la clave:**

  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters

  Bajo la misma es necesario añadir el valor:

  Value Name: IPForwarding

  Data Type: REG_DWORD

  Value: 1

3.24. Session Hijaking.

Considerando la importancia de la información trasmitida a través de las redes de datos, y las medidas de seguridad que deben desarrollarse, esta técnica pretende mostrar la posibilidad de apoderarse de una sesión ya establecida.

Este avance podría suponer el obviar todo el proceso de autenticación previo. El *TCP hijaking* puede realizarse en entornos de red de difusión, 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”, ya que el atacante debe situarse entre el equipo que estableció la conexión original y la víctima. Para poder tomar el control de una conexión previamente es necesario obtener la información asociada a como transcurre ésta a lo largo del tiempo, concretamente en TCP, deben conocerse los números de secuencia actuales, ya sea directamente o a través de los ISNs y del número de *bytes* trasmitidos.

Una vez conseguido el control, el objetivo será ejecutar algún comando, típicamente se pretende apoderarse de sesiones de terminal, que permita acceder al sistema remoto de forma directa. Habitualmente el control de la sesión se realiza empleando técnicas como *source-routing*, de forma que los paquetes de vuelta lleguen al atacante y no al destino real.

En caso de no disponer de esta facilidad la técnica se conoce como *blind-hijacking* y se basa en adivinar o intuir las respuestas de los sistemas que intervienen en la comunicación.

Para obtener la información de la conexión existente debe emplearse un *sniffer*, situándose entre los sistemas que se están comunicando, ataque conocido como "*man-in-the-middle attack*".

Existen dos herramientas principales para llevarlo a cabo, aunque numerosos *sniffers* también incluyen esta funcionalidad, por ejemplo *ethereal*: se denominan *Juggernaut y Hunt*. Asimismo, existen métodos para apoderarse de las conexiones encriptadas, por ejemplo de SSH v.1 o SSL. Existen herramientas como “dsniff” o “ettercap” que facilitan aplicar la técnica de *hijacking* en estos entornos.

3.25. Source Routing.

Esta funcionalidad propia del protocolo IP permite enviar dentro del mismo paquete de datos la información necesaria para su enrutamiento, es decir, la dirección IP de cada uno de los dispositivos de red intermedios que deben cruzarse hasta llegar al destino final.

Esta característica puede emplearse para tareas de verificación y configuración de los enlaces, pero desde el punto de vista de la seguridad supone que un atacante es capaz de manejar por dónde deben viajar sus paquetes IP, saltándose todas las reglas de enrutamiento definidas en los dispositivos de red.

Asimismo, puede permitir la realización de pruebas para conocer las redes internas y también permitir a un atacante el alcanzar redes con IPs internas. Unida a la técnica de *IP spoofing* permite que un atacante se haga pasar por otro sistema IP, siendo capaz de enviar y recibir todas las respuestas asociadas a una comunicación falseada.

Una de las principales dificultades asociadas al *spoofing* es que realmente no se puede simular de forma totalmente real la dirección falseada, ya que la red enviará las respuestas generadas por el sistema atacado al equipo que verdaderamente posee la dirección IP falsa.

Por lo tanto, para tener éxito en el ataque es necesario eliminar momentáneamente al equipo al que se le ha robado la dirección IP, por ejemplo mediante un ataque DoS. En el caso de poder emplear el *source routing*, ésto no sería necesario.

3.26. ICMP Redirects.

Existe un tipo de paquete ICMP que es empleado por los dispositivos de enrutamiento para informar de las alternativas de rutas por las que debe dirigir el tráfico un sistema ejecutando el protocolo IP.

En algunos casos estos mensajes tienen la utilidad de aconsejar a un sistema del camino a seguir, es decir, el siguiente equipo al que debe enviarle los paquetes IP. En el caso de cambios en la red la redirección funciona de manera dinámica.

La vulnerabilidad asociada a esta funcionalidad se basa en generar paquetes de redirección hacia un sistema objetivo, de forma que se oriente su flujo de tráfico hacia sistemas controlados por el mismo atacante, para por ejemplo, analizarlo mediante un *sniffer* o incluso realizar cualquier tipo de modificación en los datos.

Realmente, si un sistema acepta la recepción de este tipo de paquetes, lo que hace es actualizar su tabla de rutas con una entrada de tipo dinámica, que le indica la nueva ruta a seguir.

3.27. Directed Broadcast.

Este tipo de tráfico puede dar lugar a la existencia de redes amplificadoras de tráfico empleadas en ataques de tipo DoS como *Smurf.*

Es necesario comentar que existen otros tipos de paquetes de *broadcast* que permiten a un atacante extraer información valiosa de la red y su composición.

Mediante el uso de paquetes *broadcast* de máscara de red, un atacante puede obtener los bloques de redes empleados para posteriormente emplear rangos de IPs precisos en sus escaneos. Asimismo, mediante la utilización de *broadcast* de tipo *timestamp*, el atacante puede extraer información de identificación de los sistemas existentes.

3.28. SNMP.

El protocolo SNMP, *Simple Network Management Protocol*, también conocido como "*Security Not My Problem*", al menos hasta la versión 3, permite la gestión y administración de dispositivos de red: RFC 1157 (versión 1) y RFC 1446 (versión 2).

La seguridad de la primera versión se basa en el uso de claves conocidas como *community names*, mientras que la versión 2 gestiona la integridad mediante el uso del algoritmo MD5, y permite encriptación, mediante DES, pero ésta no limita el uso de claves simples.

La versión 3, RFC 2570, profundiza más en la seguridad de los dispositivos. A grandes rasgos existen dos tipos de comunidades: lectura (RO) y lectura-escritura (RW). Cada clave asociada permite realizar la operación referida, por lo que la clave RW permitiría modificar la información del dispositivo de red, mediante el comando snmpset.

Una vulnerabilidad habitual son los ataques de fuerza bruta sobre el "*public / private*" *community string*. Los *routers* suelen disponer de agentes SNMP con grandes cantidades de información acerca de la red debido a su función y situación, por lo que constituirán uno de los objetivos principales de un escaneo SNMP.

Para explotar las vulnerabilidades asociadas a este protocolo, basta con disponer de una utilidad como snmpwalk, disponible para numerosos sistemas Unix, que permite obtener la información almacenada en la MIB de un sistema conectado a la red de forma remota. En el caso de que las comunidades de lectura y escritura del sistema no se hayan modificado, dispondrá de las establecidas en el estándar, que son respectivamente "public" y "private".

Podrá obtenerse información de cualquier objeto de la MIB como sigue (en Unix):

```
# snmpwalk <<dirección_IP | nombre_host>> COMUNIDAD [id_objeto]
```

Por ejemplo, para obtener la información de los interfaces de red de un equipo:

```
# snmpwalk host99 public interfaces
```

3.29. TCP Initial Sequence Numbers.

El protocolo TCP genera un ISN, o número de secuencia inicial, para poder realizar el control de flujo de la conexión. Este es uno de los ataques más antiguos, data de 1985, y se basa en la utilización de *pseudo-random number generators* (PRNGs) para la generación de los ISNs.

Si los números de secuencia pueden ser predichos, puede llegar a ser posible el modificar la información de la conexión, apoderándose de ella mediante *hijaking* o realizar *blind spoofing* sobre futuras conexiones.

La modificación de los datos en la conexión puede realizarse inyectando paquetes válidos, al conocerse el ISN inicial y el número de *bytes* intercambiado, y por tanto, el número de secuencia actual.

Si no se conoce exactamente este valor, pero sí de forma aproximada, puede enviarse también un grupo de paquetes en un rango de secuencia concreto (que vendrá limitado por el tamaño de ventana TCP), con el objetivo de que alguno coincida con el número de secuencia actual. Inicialmente se modificaron las implementaciones para hacer lo más aleatoria posible la

generación de estos números.

La generación de números aleatorios, tanto en los números de secuencia iniciales de TCP, *ISNs*, como en los algoritmos de encriptación y generación de claves, tiene un peso muy relevante respecto a la seguridad.

En el sistema operativo Linux se han introducido mecanismos para la obtención de números aleatorios más reales, mediante dos archivos de dispositivo, /dev/random y /dev/urandom, que gestionan la entropía existente en el sistema según su actividad. Actualmente se ha encontrado una nueva vulnerabilidad que se presenta cuando se usan incrementos aleatorios, al aumentar constantemente el valor de los ISNs generados.

Debido a las implicaciones del teorema del límite central, el sumatorio de una serie de números no proporciona la suficiente varianza en el rango de valores de ISN deseados, por lo que un atacante puede apoderarse de las conexiones.

Por tanto, los sistemas basados en la generación de números mediante incrementos aleatorios son vulnerables a ataques estadísticos. Esta debilidad en las implementaciones del protocolo TCP permite emplear una técnica conocida como *Blind Spoofing*, en la que se realiza un ataque de *IP Spoofing* pero sin la posibilidad de interceptar el tráfico de red intercambiado entre los sistemas.

Para poder mantener la comunicación, es necesario “adivinar” los números de secuencia empleados por el sistema a atacar.

3.30. Tiny Fragment Attack.

Para comprender este ataque debe considerarse como tiene lugar la fragmentación de paquetes TCP sobre IP.

Cuando un paquete IP supera el tamaño máximo de transmisión, MTU, debe dividirse en paquetes menores. El primero de ellos incluirá la cabecera TCP asociada al paquete original, mientras que el resto de fragmentos simplemente contendrán la cabecera IP y los datos, pero no información de TCP (cabecera TCP).

A través del campo de *fragment offset* de la cabecera IP se determina si existen más fragmentos y la relación entre éstos. Cuando se gestiona un sistema de filtrado de paquetes, lo habitual es permitir que los fragmentos de un paquete IP pasen el filtro, ya que no se dispone de información TCP para tomar una decisión de filtrado en función, por ejemplo, de los puertos origen y destino. La técnica presentada pretende enviar un paquete TCP inicial con la siguiente información: SYN=0, ACK=1, FO=”*more packets follow*”. De esta forma, el paquete puede atravesar un filtro concreto (*stateless*), al no disponer del *flag* SYN activo.

Este paquete no sería peligroso de no ser porque el tamaño de *offset* (20 *bytes*) es lo suficientemente pequeño como para sobrescribir ciertos campos de la cabecera TCP mediante el paquete que representa el supuesto fragmento esperado a continuación.

Este segundo paquete en el proceso de fragmentación cambiará los valores de TCP a SYN=1, ACK=0, por tanto, se tendrá un paquete de establecimiento de conexión reconstruído en la máquina destino, aunque los filtros explícitamente no permiten el establecimiento de conexión desde ese sistema IP origen.

Por ejemplo, al enviar un paquete de 8 *bytes*, suficiente para contener los puertos fuente y destino (además del número de secuencia), se obligará a recibir los *flags* TCP en el siguiente paquete.

Este segundo paquete o fragmento no posee cabecera TCP, por lo que el filtro no se podrá aplicar sobre él, ni tampoco en el resto de fragmentos. Realmente, el campo de datos del segundo fragmento, contiene el resto de la cabecera TCP tras los 8 *bytes*, es decir, los *flags* TCP.

3.31. Winnuke.

Este ataque afecta a los sistemas que utilizan el protocolo NetBIOS sobre TCP/IP, típicamente en el sistema operativo Windows.

Este protocolo emplea los puertos UDP 137, 138 y 139. 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, 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 (ésta referencia contiene menciones a otros ataques de los protocolos TCP/IP).

Asimismo, existen *exploits* similares, como *supernuke*. El termino *Nuking* (*nuke.c*) no debe ser confundido con *Winnuke*. Es una técnica antigua, por lo que no funciona en los sistemas modernos, y para ser ejecutado debe tenerse privilegio de *root*.

El ataque se basa en enviar fragmentos o paquetes ICMP no válidos, con la finalidad de ralentizar al objetivo o incluso bloquearlo. Posteriormente surgió una variante de éste denominada *Smurfing*.

3.32. Teardrop.

El ataque *teardrop* se basa en el envío de fragmentos de paquetes en lugar de paquetes completos.

Se comprobó que algunas implementaciones de la pila TCP/IP no eran capaces de reconstruír paquetes con fragmentos cuyos *bytes* se superponen.

El resultado es de nuevo que el sistema destino puede llegar a bloquearse; apareció en Linux inicialmente. Para llevarlo a cabo bastaría con 2 paquetes, A y B, donde el *offset* del paquete B indica que comienza dentro del paquete A.

Existen dos versiones de este ataque: *teardrop* y *teardrop2*. La variación de la segunda respecto a la primera se basa en la inclusión del *flag* de urgencia (URG) en la cabecera TCP de los fragmentos.

Por ejemplo, Windows NT 4 SP3 se parcheó frente a la primera versión, pero era vulnerable a la segunda. Existen variantes de *teardrop* en las que el paquete enviado tiene el *flag* SYN activo, como “*syndrop.c*”, así como otras orientadas a sistemas Windows, ”*bonk.c*”.

3.33. DNS.

DIG es una utilidad para la obtención de información del servicio de nombres DNS. Como ya se mencionó en el apartado de *footprinting*, el DNS es una fuente de información de red muy valiosa.

La utilidad mencionada permite copiar una base de datos entera de nombres (dominio) desde un servidor DNS, para su posterior análisis.

Asimismo sus características avanzadas facilitan extraer toda la información asociada al protocolo DNS, no permitiendo únicamente la realización de peticiones, como *nslookup*.

3.34. NTP.

El protocolo NTP, *Network Time Protocol*, permite sincronizar la hora de forma simultánea en todos los equipos de una red.

Este protocolo presenta diferentes vulnerabilidades, ya que por ejemplo, sistemas de alta disponibilidad configurados en cluster se basan en el momento horario de cada uno de sus nodos para gestionar el *cluster* que ofrece el servicio.

En el caso de poder modificar la hora en un nodo, podrían obtenerse resultados inesperados en el conjunto de ellos, como por ejemplo que se desconfiguraran ciertos nodos, no formando parte del *cluster*, pudiendo llegar a anularse la alta disponibilidad.

Mediante el comando ntpdate se puede modificar la hora de un sistema:

```
# ntpdate –d <<dirección_IP>>
```

Mediante el comando ntpq se pueden hacer consultas al servicio NTP de un sistema. Por ejemplo para ver las asociaciones de un equipo, es decir, de quién obtiene la hora:

```
# ntpq –p <<dirección_IP>>
```

3.35. Caballos de Troya o Trojanos.

Pese a que esta vulnerabilidad está más asociada a los sistemas y no a TCP/IP, se presenta una visión general, ya que en numerosas ocasiones, es empleada para introducir servicios TCP/IP no deseados en sistemas destino y poder así ejecutar ataques remotos posteriormente o

incluso tomar su control por completo.

También conocidos como puertas traseras (*back doors*), son fragmentos de programas no autorizados que se introducen en otros para que el programa original ejecute ciertas acciones no deseadas.

En el caso de los *trojanos* que afectan a los servicios TCP/IP más directamente, se trata de programas completos, que normalmente se justifican como herramientas de administración remota o RAT (típicamente de Windows) como por ejemplo:

- **Back Orifice** (Creado por el grupo de Hackers "El Culto de la Vaca Muerta" y uno de los más conocidos).
- **Back Orifice 2000** (Idem anterior pero perfeccionado).
- **NetBus.**
- **SubSeven.**

Asimismo, las herramientas típicas de administración y acceso remoto podría incluirse en este grupo, ya que facilitan el acceso y el completo control del sistema destino.

Entre estas se encuentran *PCAnywhere, VNC, Windows Terminal Services*. Habitualmente este tipo de software se descarga en los sistemas objetivo al visitar alguna página Web o servicio electrónico público sin que el usuario se percate de ello.

Las consecuencias y acciones de cada herramienta pueden variar en función de la idea con la que se diseñaron: desde conectarse a canales IRC, a distribuirse para actuar como fuente futura de ataques DDoS o incluso manipular y extraer información del sistema que les hospeda.

3.36. IPSec.

La seguridad del estándar IPSec ha sido analizada en numerosos estudios, poniéndose en entre dicho como característica negativa la complejidad de sus especificaciones y del propio protocolo.

Aunque el análisis también está enfocado desde un punto de vista político centrado en el control de la encriptación, denota que es la implementación de los diferentes fabricantes la que determinará su seguridad al 100%,en función de sí los estándares son respetados y la interoperatibilidad posible.

Los dispositivos que "hablan" IPSec pueden ser identificados por tener el puerto 500 escuchando, ya que es el asociado al estándar de intercambio de claves o IKE, *Inter Key Exchange protocol*. Por otro lado, la debilidad desde el punto de vista de la seguridad, no es tanto la vulnerabilidad del propio IPSec, como la de los algoritmos de encriptación asociados al mismo, como el RSA.

Debe tenerse en cuenta que en el intercambio de información mediante IP, gran parte de la información asociada a los protocolos es conocida, por lo que puede emplearse como texto en claro para romper la encriptación mediante, por ejemplo, ataques estadísticos.

Existe un estudio detallado sobre las vulnerabilidades de protocolos de túneles como PPTP. En éste se muestra la posibilidad de aplicar la técnica de *spoofing* para adquirir credenciales de autenticación en un entorno PPTP entre un cliente, ya conectado a Internet (no realizando una conexión conmutada o de *dial-up*) y un servidor; no así entre servidores.

3.37. Finger Bomb.

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.

3.38. RPC.

Existe una tecnología de red, inventado originalmente por Sun Microsystems, que permite invocar procedimientos y acciones de forma remota desde otro equipo. Ésta se denomina RPC, *Remote Procedure Call*.

Es posible obtener la lista de servicios activos en un equipo mediante el siguiente comando en Unix:

rpcinfo –p <<dirección_IP>>

3.39. Relaciones de confianza entre sistemas.

Los comandos *r-Unix* (*rsh, rcp, rlogin...*) requieren de relaciones de confianza entre sistemas para accederse entre sí directamente, esquivando los controles de seguridad y autenticación habituales.

Esto significa que si se es capaz de vulnerar uno de los sistemas incluídos en el círculo de confianza, se dispondrá de acceso al resto. Si se desea filtrar del exterior la utilización de estos programas puede hacerse mediante los puertos 512, 513 y 514. Los archivos implicados en la obtención de permisos son:

/etc/hosts.equiv, $HOME/.rhosts, /etc/hosts.allow o /etc/hosts.deny, y .shosts.

Por ejemplo, si un atacante es capaz de obtener a través de un *exploit* uno de estos archivos, es capaz de ver las puertas de entrada desde las que el acceso está permitido, por lo que su siguiente paso será adquirir el control de alguno de los sistemas contenidos en el archivo.

Existen asimismo ataques que se basan en modificar el archivo de confianza, por ejemplo el .rhost en /usr/bin, para poder acceder libremente desde el sistema actual.

3.40. Buffer-Overflows.

Los desbordamientos de un *buffer*, normalmente la pila de ejecución, se mencionan, junto a los *format strings*, 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.

Se ha incluído ya que está asociada a los servicios proporcionados sobre TCP/IP y porque es ejecutada mediante el envío de paquetes de información desde la red, explotando una debilidad. La primera vulnerabilidad encontrada en Internet de este tipo fue el famoso *Gusano* de Robert Morris, Jr.

El 2 de noviembre de 1988, este estudiante generó un *exploit* que aprovechaba dos vulnerabilidades: la primera asociada al modo de depuración del demonio *sendmail* (que permite el envío de *e-mails*), y la segunda relativa al demonio *fingerd* (que implementa la identificación mediante peticiones *finger*) de los sistemas Unix.

El gusano fue capaz de colapsar por aquel entonces gran parte de los sistemas existentes en Internet, provocando gran conmoción respecto a la seguridad en La Red.

Los servidores que proporcionan un servicio TCP/IP, a través de protocolos de nivel superior, como HTTP, SMTP, FTP, DNS, NNTP, pueden presentar errores de diseño o implementación que dan lugar a vulnerabilidades de seguridad.

Como objetivo final de éstas, se busca la posibilidad de ejecutar código arbitrario, que consiste en proporcionar un *shellcode*, o código ensamblador que permite obtener una *shell* de *root* en el sistema, es decir, con todos los permisos de administrador. Los SS.OO. abiertos suelen ser estudiados en mayor profundidad, por lo que las vulnerabilidades existentes son más conocidas, y por tanto, están más controladas (se habrá distribuído un parche software para corregir el error).

El mejor ejemplo de un sistema así es Linux. En el caso de SS.OO. propietarios, como por ejemplo la familia Windows, la información al respecto depende del fabricante.

El ataque que permite explotar este tipo de vulnerabilidad se presenta normalmente en forma de *exploit*, que no es ni más ni menos que un programa escrito en C y en ensamblador que fuerza las condiciones necesarias para aprovecharse del error de seguridad subyacente.

Existen infinidad de *exploits* para servicios como imap, pop3, sendmail, inn, automuountd, ftp, bind, httpd, samba...Sin duda alguna, uno de los mejores artículos existentes en la red es "*Smashing the Stack for Fun and Profit*" aparecido en Phrack. Técnicamente, este método se basa en la posibilidad de escribir información sobrepasando los límites de un *array*, almacenado en la pila asociada a la rutina de un programa dónde el *array* está definido, consiguiendo así

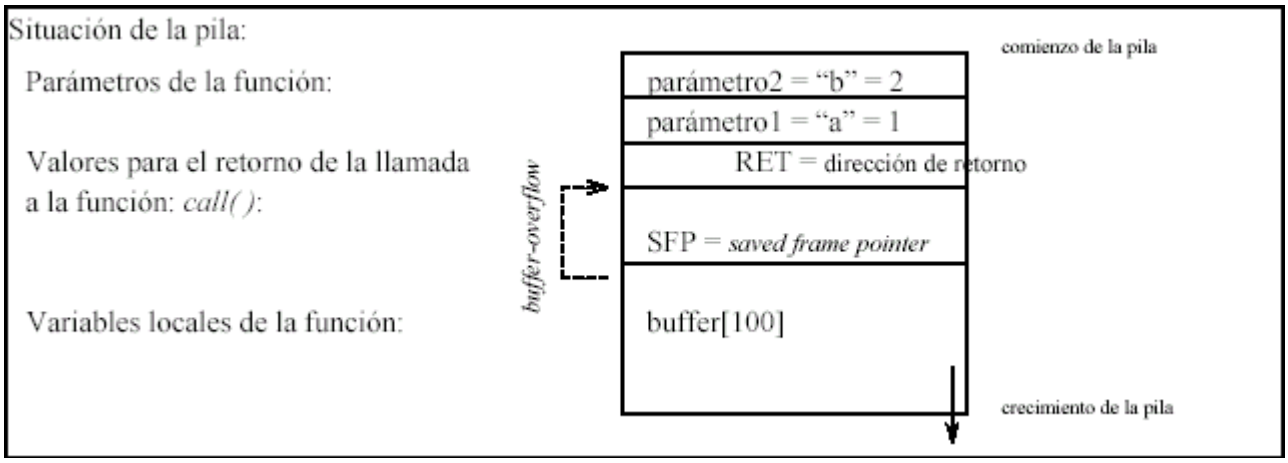
corromper la pila de ejecución, sobrescribiendo el valor retorno de la función, y causando que el flujo de ejecución continúe en una dirección arbitraria (introducida en los datos del *buffer-overflow*).

Esta técnica es posible al emplearse en los programas funciones de manipulación de *buffers* que no comprueban los límites de las estructuras de datos, como por ejemplo “strcpy( )”, en lugar de las que sí lo hacen, por ejemplo “strncpy( )”.

Por ejemplo, en el siguiente programa, la situación genérica de la pila quedaría como sigue:

fuente.c

```
-----------------------------------
void funcion(int a, int b) {
char buffer[100];
}
void main() {
funcion(1,2);
}
-----------------------------------
```



Para poder llevar a cabo está técnica es necesario conocer información muy precisa de la arquitectura del sistema destino, tanto de la arquitectura de la CPU subyacente, como del sistema operativo sobre el que se provee el servicio a atacar.

Por ejemplo, el sentido de crecimiento de la pila de ejecución, es decir, sí es hacia direcciones menores o mayores de memoria, la definición del puntero de pila (SP), es decir, sí éste referencia a la última posición ocupada en la pila o a la primera posición libre.

Asimismo, se requiere conocer en detalle el orden en el que se depositan los diferentes elementos en la pila: el puntero de *frame* anterior (SFP), la dirección de retorno (RET), las variables locales, los parámetros que se le pasan a la función, el valor de retorno.

Con toda este información, se podrá introducir un valor en la posición de la dirección de retorno que envíe el flujo de ejecución justo al punto que se desee, es decir, una posición de memoria dónde se haya depositado previamente un *shellcode*; el objetivo suele ser disponer de un *shell* en el sistema con los permisos asociados al usuario con el que el servicio atacado (sobre el que se realiza el *buffer-overflow*) está ejecutando, que en numerosas ocasiones implica disponer de todos los permisos, al ejecutar como *root* (en Unix). Una vez se disponga de toda la información detallada, se utilizará un puntero contra la dirección de memoria que contiene el valor de la dirección de retorno para modificarlo.

El valor introducido apuntará a la dirección de la pila en la que se haya depositado el *shellcode*. Debido a que el código a ejecutar se deposita directamente en la pila, éste debe ser el propio código ensamblador extraído de su equivalente fuente. Para ello puede compilarse el fuente con

la opción de generación del propio código ensamblador asociado o extraer éste mediante una utilidad como gdb.

Por ejemplo, en lenguaje C un *shell* se obtendría mediante el siguiente código:

shellcode.c

```
-----------------------------------
#include <stdio.h>
void main() {
char *name[2];
name[0] = "/bin/sh";
name[1] = NULL;
execve(name[0], name, NULL);
exit(0);
}
-----------------------------------
```

La ejecución real de *exploits* basados en este método requiere la utilización de técnicas más avanzadas, como la utilización de referencias relativas al *shellcode*, ya que se desconoce la posición absoluta en memoria, o lo que es lo mismo, el estado de la pila, en el momento de su ejecución.

Asimismo es necesario realizar un tratamiento al contenido del buffer a volcar en memoria, para evitar la existencia de elementos NULL que podrían concluír la lectura de un *string* de caracteres y por tanto no ejecutar el programa en su totalidad.

A su vez es necesario simular una condición de finalización correcta, tanto si realmente es así, como si ocurre algún problema en las llamadas al sistema, de forma que en ningún caso aborte la ejecución del servidor al intentar ejecutar el *exploit*.

Otro aspecto a tener en cuenta es la posible variación de la posición en la pila del código a introducir, por lo que es posible apuntar a direcciones de memoria no permitidas. Para evitarlo sería necesario averiguar el tamaño del *buffer* sobre el que aplicar el *overflow*, así como el desplazamiento necesario en la pila: para facilitar su situación en memoria suele emplearse el código NOP (*No Operation*), que le indica al procesador que no haga nada, de forma que puede utilizarse parte de memoria con este tipo de instrucción como margen en el desplazamiento a aplicar.

Resumiendo, los *buffers-overflows* se producen al escribir en memoria por encima de los límites de un *array* o *buffer* existente, y son consecuencia de que ciertas funciones, de lenguajes como C, implícitamente no comprueban los límites de los *arrays*.

La librería estándar de C posee numerosas funciones para ejecutar operaciones de copia y concatenación de *strings*, sin comprobación de límites: strcat(), strcpy(), sprintf(), and vsprintf().

Todas operan considerando el final del *string* al observar el primer carácter NULL. Otras funciones afectadas podrían ser: gets(), scanf(), así como el uso de las funciones getc(), fgetc(), or getchar() en bucles de lectura de caracteres.

Por tanto, el encontrar software vulnerable es tan sencillo como buscar en los fuentes por funciones como las comentadas y analizar su utilización. Asimismo, existen generadores de *shellcodes* automáticos.

3.41. Format Strings.

Los ataques de *format string* se producen al imprimir o copiar a otro *buffer* un *string*. El programador pretende imprimir su contenido con una sentencia como:

```
printf("%s", str);
```

Pero en su lugar, por "La Ley del Menor Esfuerzo", escribe:

```
printf(str);
```

El resultado funcional es el mismo, pero el resultado técnico es muy diferente, ya que esta sentencia genera un agujero de seguridad en el código que permite controlar su flujo de ejecución.

Aunque el programador le indica a la función el *string* a imprimir, ésta lo interpreta como un *string* de formato, es decir, pretende encontrar en su contenido caracteres de formatos especiales, como por ejemplo "%d", "%s", "%x".

Por cada uno de los caracteres de formato, un número variable de argumentos será extraído de la pila. La vulnerabilidad obvia es que podrán leerse los valores de la pila que se encuentren por encima del *string* de formato, y la no tan obvia es que ofrece el control suficiente como para escribir en la memoria del programa.

Analizando la función *printf* sobre la que se desarrollará el ataque: aparte de las utilidades propias de la función para imprimir enteros, *strings* y delimitar la longitud de los campos a imprimir, ésta permite:

- Obtener en cualquier momento el número de caracteres en la salida: al encontrarse un "%n", el número de caracteres en la salida antes de encontrar éste campo se almacenará en la dirección pasada en el siguiente argumento:

```
int valor, x = 100, y = 20;
printf("%d %n%d", x, &valor, y);
```

- El carácter de formato "%n" devuelve el número de caracteres que deberían haberse emitido a la salida, y no el número de los que realmente se emitieron. Por ejemplo, al formatear un string en un buffer de tamaño fijo, el string podía ser truncado. A pesar de esto, el valor devuelto por "%n" será el desplazamiento si el string no se hubiera truncado:

```
char buf[20];
int valor, x = 0;
sprintf(buf, sizeof buf; "%.100d%n", x, &valor);
printf("Valor: %d", valor);
```

Este ejemplo imprimirá "Valor: 100" y no "Valor: 20".

Por tanto, mediante la manipulación correcta de funciones como sprintf() y printf() se pueden escribir caracteres en la pila, concretamente el número de *bytes* indicados por "%n", en la dirección que se le indique a la función a través del *string* de formato (ya que ese valor se deposita en la pila, para que actúe como el siguiente argumento en el primer listado).

El valor a escribir puede ser manejado como se desee por el segundo listado, ya que con "%.numerod", se añade el valor "número" a los caracteres existentes realmente antes de "%n" en el *string*. Al igual que en los *buffers-overflows*, es necesario conocer la arquitectura del sistema, por ejemplo, sí la representación numérica es *litle-endian* o *big-endian*, así como ciertas características del sistema operativo y del entorno.

La conclusión de esta técnica es que es posible escribir el valor deseado en casi cualquier dirección de memoria, luego puede usarse para sobrescribir el comando a ser ejecutado, o el UID asociado a un programa, o la dirección de retorno de forma que apunte a una posición de memoria dónde se haya ubicado previamente un *shellcode* (tal y como ocurre en los *buffer-overflows)*).

3.41. Comunicaciones Inalámbricas: *Wireless.*

Las redes inalámbricas sobre las que seguirá hablándose TCP/IP serán el objetivo de los *hackers* en un futuro no muy lejano.

La facilidad de acceso a estas redes, en el caso de no encontrarse protegidas de forma adecuada, es mucho mayor que en el caso de las redes de datos comunes, ya que no es necesario el obtener un punto físico mediante el que conectarse a la red.

Basta con disponer de una tarjeta de red que hable el protocolo 802.11 a,b o g y de una notebook o palmtop con acceso inhalámbrico (Ejemplo: Wi-Fi) para desplazarnos al área de

transmisión de una red concreta y comenzar a indagar. Existen herramientas que permite escanear el “ambiente” en busca de redes de este tipo, como por ejemplo, Net Sumbler. También desde el punto de vista del hardware, se consiguen por unos U$S 30, herramientas que permiten medir la intensidad de una señal de red inalámbrica desde el exterior de cualquier edificio.

También cabe acotar que ya existe placas madres (motherboards) de escritorio y computadoras portátiles que traen incluídas una tarjeta para comunicaciones inalámbricas.

Para tener una idea de lo hasta aquí expresado, hace unos meses se experimentó en el microcentro porteño con una notebook y de veinte redes encontradas, se pudo acceder sin dificultad alguna a dieciocho de ellas.

Este acceso permite obtener información de la red a la cual se accede o bien utilizarla para ingresar a Internet y poder transmitir y recibir información de forma gratuita y subrepticia.

4. EL FUTURO.

El futuro de la seguridad pasa por la generación de herramientas de control inteligentes, como los IDS basados en la IA, Inteligencia Artificial, así como por el desarrollo de sistemas de comunicaciones diseñados con la seguridad como característica fundamental, como es el caso de IPv6.

La incorporación y el desarrollo de las denominadas "redes inteligentes" podría dificultar considerablemente las actividades de los *hackers*.

El Instituto Tecnológico de Georgia, EEUU, trabaja en un proyecto de desarrollo de redes neuronales, que probablemente aumentará la seguridad del tráfico digital.

El término red neuronal refleja una de las técnicas empleadas en el área de la inteligencia artificial dentro de la computación.

Ésta pretende reproducir el funcionamiento de aprendizaje y reconocimiento de las neuronas del cerebro humano, que aprenden de la experiencia, creando conexiones entre las distintas áreas de conocimiento.

Con todo, cabe precisar que no se trata de redes que estén en condiciones de pensar, sino de sistemas capaces de identificar patrones en el flujo digital y aprender de los intentos de intrusión.

Hoy en día, los administradores de sistemas deben actualizar manualmente los sistemas de protección de las redes contra los ataques informáticos, ya que aparecen nuevas vulnerabilidades y técnicas de ataque con demasiada frecuencia.

Con la incorporación de redes inteligentes se hará más previsible y fácil la contención de los intrusos. Tales redes estarán incluso en condiciones de detectar máquinas que monitorizan ilegalmente el tráfico de la red para captar y apoderarse de información, es decir, *sniffers*.

La novedad es que las redes neuronales detectarán ese tipo de máquinas sin que sus operadores se percaten. Las implicaciones de la nueva versión de IP, IPv6 o *IP Next Generation*, en el área de las VPNs es realmente importante y facilitará su expansión.

Debe partirse de la base de que el protocolo IPSec fue diseñado inicialmente para esta versión de IP, portándose posteriormente a la versión 4, ofreciendo así la posibilidad de trabajar con tráfico IP encriptado.

Actualmente, los fabricantes de productos para VPNs aceptan que los sistemas no disponen de una implementación de la pila TCP/IP basada en la versión 6, por lo que se emplean modificaciones sobre las implementaciones actuales de la versión 4.

El IP Forum encargado de generar las especificaciones del protocolo y de su evolución futura tiene entre sus filas a los grandes fabricantes en el mundo de las redes: Cisco, 3Com, Microsoft, MCI Worldcom, Nokia, British Telecom, Siemens y otros.

5. CONCLUSIONES.

La seguridad de las redes de comunicaciones, y concretamente de Internet, evoluciona a pasos agigantados cada minuto que transcurre.

Nuevas vulnerabilidades y utilidades, tanto para explotarlas como para combatirlas, son distribuidas

públicamente en la red.

La información disponible al respecto es inmanejable, por lo que el diseño de un sistema de seguridad debe basarse en la fortaleza de las tecnologías empleadas, y no en la ocultación de las mismas: "*security through obscurity*".

El protocolo TCP/IP sufre algunos problemas de seguridad por las características intrínsecas de su diseño, los cuales han sido ampliamente analizados a lo largo de los últimos años.

La nueva versión de IP, versión 6, se diseño con la seguridad en mente, de ahí la aparición del estándar IPSec, que junto a otras tecnologías, como las infraestructuras de clave pública, PKIs, permiten controlan y disolver muchas de las vulnerabilidades presentadas a lo largo del presente trabajo.

Asimismo, se puede concluir que la seguridad de una red no se basa en una única técnica exclusivamente, como promulga la idea errónea de securizar una red mediante un *firewall*, sino que viene reforzada por la utilización de multitud de tecnologías que permiten monitorizar y gestionar cada uno de los aspectos críticos de la red: encriptación, IDSs, *firewalls*, software específico de seguridad, protocolos seguros (SSL, SSH, IPSec), PKIs.

Lo que es más, mediante el uso exclusivo de las tecnologías mencionadas no es posible asegurar la seguridad de la red.

Para ello es necesario a su vez disponer de procedimientos y políticas adecuadas que permitan concienciar a los usuarios y administradores de los sistemas informáticos, así como facilitar la aplicación de análisis y controles exhaustivos en la propia red y los elementos que la componen.

Es por tanto necesario dedicar tiempo y esfuerzo a evolucionar la red hacia un entorno seguro, siendo necesario mantenerse actualizado (preferiblemente de forma automática, por ejemplo, mediante listas de distribución) de los avances que se realizan en este campo, así como de los nuevos avisos, vulnerabilidades y tecnologías que salen a la luz.

El objetivo final es asegurar ciertas características en las comunicaciones, como son, la autenticidad, la integridad de la información, la privacidad o confidencialidad, el no repudio, el control de acceso a la información.

Finalmente, cabe concluir con una reflexión al respecto de la seguridad: **el esfuerzo dedicado a la protección de un entorno debe ser directamente proporcional al valor de su contenido.**

Debido a que toda vulnerabilidad posee, más tarde o más temprano, su correspondiente protección, la vertiginosa carrera en la que la seguridad de las redes se debate actualmente, se centra en el mantenimiento constante y actualizado de las protecciones necesarias para evitar las vulnerabilidades existentes y ya conocidas.

6. BIBLIOGRAFÍA:

Practical Unix & Internet Security, 3rd Edition, Simson Garfinkel, Gene Spafford, Alan Schwartz.
Editorial: O'Reilly; 3ra edición (Febrero 21, 2003).
ISBN: 0596003234.

The Protocols (TCP/IP Illustrated, Volume 1), W. Richard Stevens.
Editorial: Addison-Wesley Professional, 1ra edición (Enero 1, 1994).
ISBN: 0201633469.

The Implementation (TCP/IP Illustrated, Volume 2), Gary R. Wright, Wright Gary R., W. Richard Stevens.
Editorial: Addison-Wesley Professional, 1ra edición (Enero 31, 1995).
ISBN: 020163354X.

Firewalls and Internet Security, William R. Cheswick, Steven M. Bellovin, Aviel D. Rubin.
Editorial: Addison-Wesley Pub Co, 2da edición (Febero 24, 2003).
ISBN: 020163466X

Maximum Security, Fourth Edition, Anonymous.
Editorial: Sams, 4th Bk&Cdr edition (Diciembre 16, 2002).

ASIN: 0672324598.