Ataques externos

by Eric Detoisien
<valgasu(at)club−internet.fr>

About the author:

Eric Detoisien es un experto en seguridad informática. Muy ducho en todo lo relativo a la seguridad, es uno de los expertos del grupo rstack – www.rstack.org –

Translated to English by:
Georges Tarbouriech
<gt(at)linuxfocus.org>

Abstract:

Este artículo fue publicado en una revista de Linux en Francia en una entrega especial sobre seguridad. El editor y los autores permitieron a LinuxFocus publicar todos los artículos de esta entrega especial. Consecuentemente, LinuxFocus los irá publicando tan pronto como sean traducidos al inglés. Gracias a toda la gente envuelta en esta labor. Este fragmento será reproducido en cada artículo que tenga el mismo origen.

Este artículo presenta los diferentes tipos de ataques externos que un cracker puede usar contra las máquinas en una red. Tomaremos el principal ataque de red, algunos ataques a través de aplicaciones y ataques de Denegación de Servicio (DoS).

Ataques de red

Los ataques de red recaen sobre vulnerabilidades directamente relacionadas con los protocolos y sus implementaciones. Hay muchos. Sin embargo, la mayor parte de ellos son variantes de cinco ataques de red bien conocidos.

Ataques de fragmentos

Este ataque recae sobre los mecanismos de protección de filtrado IP. Para implementarlo, los crackers usan dos métodos diferentes: Pequeños fragmentos y Superposición de fragmentos. Estos ataques son algo antiguos, en consecuencia, los firewalls (cortafuegos) actuales hace bastante tiempo que los resuelven.

Pequeños fragmentos

De acuerdo con el RFC (*Request For Comment*) 791 (IP), todos los nodos de Internet (routers) deben estar capacitados para transferir paquetes de 68 bytes sin fragmentarlos. El tamaño mínimo de la cabecera de un paquete IP es de 20 bytes sin opciones. Cuando las opciones están presentes, el tamaño máximo de la cabecera es de 60 bytes. El campo IHL(*Internet Header Length*) lleva la longitud de cabecera de palabras de 32 bits. Este campo usa 4 bits, así que el número de posibles valores es de 2^4 – 1 = 15 (no puede tomar el valor 0000). Así, el tamaño máximo de cabecera es realmente 15*4 = 60 bytes. Finalmente, el campo *Fragment Offset* indicando el offset del primer byte del fragmento relativo al datagrama completo está escrito en bloques de 8 bytes. Así que un fragmento de datos lleva al menos 8 bytes. Esto produce realmente 68 bytes.

El ataque consiste en pedir una conexión TCP fragmentada en dos paquetes IP. El primer paquete IP de 68 bytes sólo lleva los 8 primeros bytes de la cabecera TCP (puertos de origen y destino, y número de secuencia). Los datos del segundo paquete IP llevan la petición TCP (flag SYN a 1 y flag ACK a 0).

Sin embargo, los filtros IP aplican la misma regla a todos los fragmentos en un paquete. El filtro de un fragmento (Offset del fragmento = 0) define la regla, en consecuencia se aplica a los otros fragmentos (offset del fragmento = 1) sin ningún otro tipo de control. Así, cuando se defragmenta a nivel IP en la máquina objetivo, el paquete de petición de conexión es reconstruido y pasado a la capa TCP. La conexión se estabiliza a pesar del filtro IP activo que debería haberlo previsto.

Figuras 1 y 2 muestran ambos fragmentos y la figura 3 muestra el paquete defragmentado en la máquina objetivo:

Fig.1: Fragmento 1

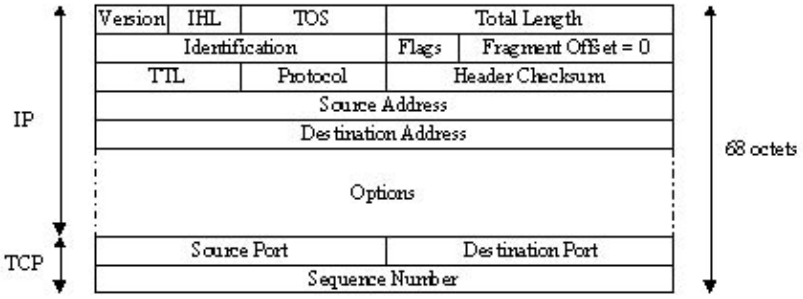


Fig.2: Fragmento 2

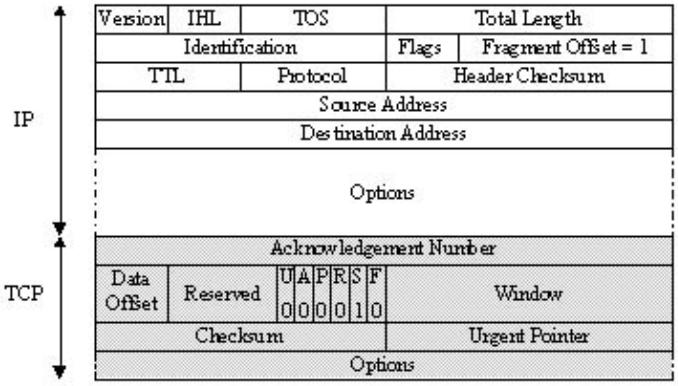
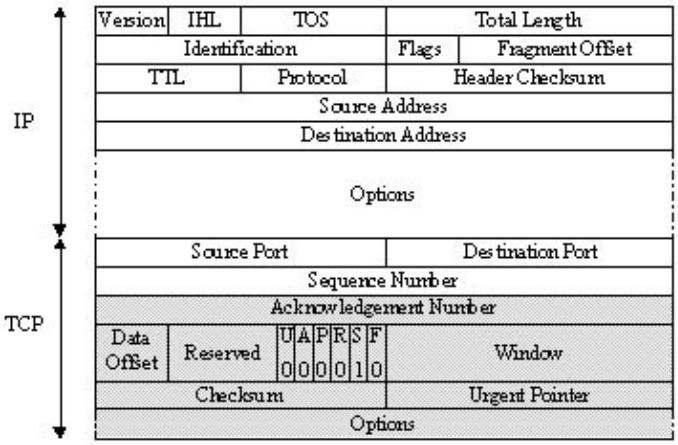


Fig.3: paquete defrafmentado



Superposición de fragmentos

De acuerdo al RFC 791 (IP), si dos fragmentos IP se superponen, el segundo sobreescribe al primero. El ataque consiste en dividir al paquete IP en dos fragmentos. El filtro IP aceptará el primero, que lleva 68 bytes (ver Pequeños fragmentos) el cual no pide una conexión TCP (flag SYN = 0 y flag ACK = 0). De nuevo esta regla se aplica a los otros fragmentos del paquete. El segundo (con el offset de fragmento = 1) contiene los datos de la conexión real, es entonces aceptado por el filtro IP porque no ve que la conexión se abre aquí, así cuando defragmentamos, los datos del segundo fragmento sobreescriben a los datos del primero, empezando desde el octavo byte (desde el offset del fragmento = 1). El paquete reensamblado es entonces una conexión válida para la máquina destino. La conexión se estabiliza a pesar del filtro IP.

Las figuras 1 y 2 muestran ambos fragmentos y la figura 3 muestra el paquete defragmentado en la máquina objetivo:

Fig.4: Fragmento 1

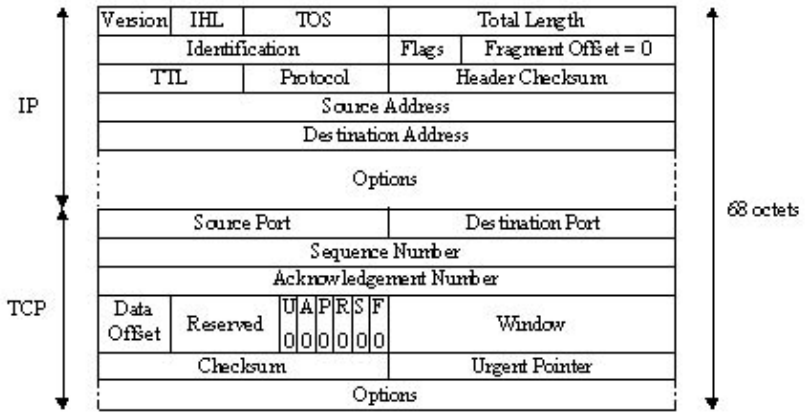


Fig.5: Fragmento 2

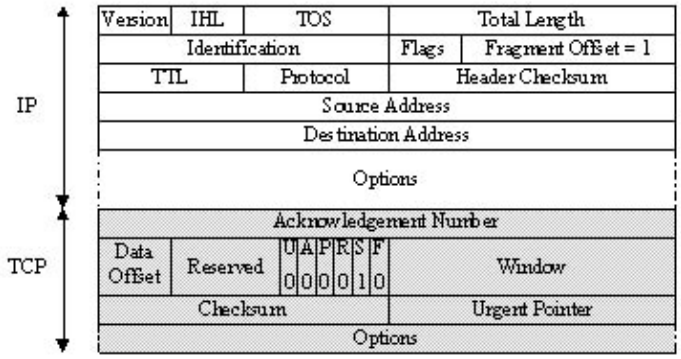
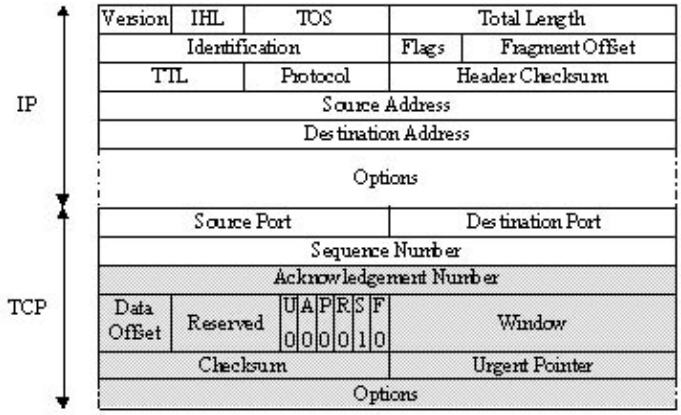


Fig.6: Paquete defragmentado



IP Spoofing

La clave de este ataque es usurpar la dirección IP de una máquina. Esto permite al cracker ocultar el origen de su ataque (usado en ataques de Denegación de Servicio) o para beneficiarse de una relación de confianza entre dos máquinas. Aquí explicaremos el segundo uso del IP Spoofing.

El principio básico de este ataque consiste, para el cracker, en crear sus propios paquetes IP (con programas como `hping2` o `nemesis`) en los cuales cambiaremos, entre otras cosas, la dirección IP de origen. IP Spoofing es llamado frecuentemente Blind Spoofing. Las respuestas a los falsos paquetes no pueden ir a la máquina del cracker, ya que el origen ha sido alterado, van hacia la máquina "burlada". Sin embargo hay dos métodos para hacer que las respuestas regresen:

1. *Source Routing*: el protocolo IP tiene una opción llamada Source Routing (ruteo de origen) que permite definir la ruta que los paquetes IP deben tomar. Esta ruta es una serie de rutas de dirección IP que los paquetes deben seguir. Suficiente para que el cracker provea una ruta para los paquetes hacia un router que el controle. Actualmente la mayor parte de las pilas TCP/IP rechazan los paquetes que usen esta opción.
2. *Re−routing*: tablas de enrutado usando el protocolo RIP, pueden ser cambiadas enviándoles paquetes RIP con nueva información de enrutado. Se hace esto para enrutar los paquetes hacia un router que controle el cracker.

Estas técnicas son muy usadas: el ataque es llevado a cabo sin saber quÃ© paquetes son los que vienen del servidor objetivo.

Blind Spoofing se usa contra servicios como rlogin o rsh. Su mecanismo de autentificación solo recae en la dirección IP de origen de la máquina cliente. Este relativamente bien conocido ataque (Kevin Mitnick lo usó contra la máquina de Tsutomu Shimomura's en 1994) requiere varios pasos:

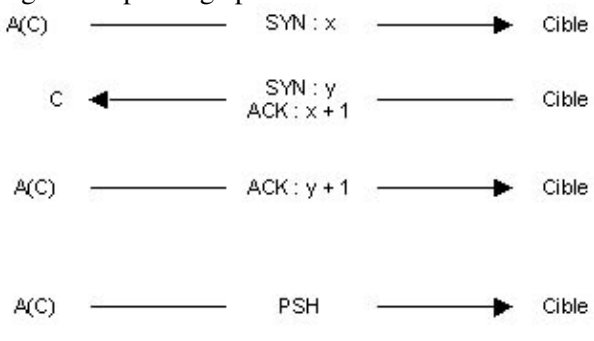
- Encontrar la dirección IP que está utilizando la "máquina de confianza", por ejemplo `showmount -e` que indica a dÃ³nde se exporta el sistema de archivos, o `rpcinfo` que da más información.
- Dejar al host de confianza fuera de servicio usando, por ejemplo, un SYN Flooding (se hablará mas sobre este ataque más adelante en Denegación de servicios). Esto es primordial para evitar que la máquina responda a los paquetes enviados por el servidor objetivo/víctima. De otro modo, enviaría paquetes TCP RST que pararían/cortarían la conexión.
- Predicción de los números de secuencia TCP: todo paquete TCP está asociado a un número de secuencia inicial. La pila TCP/IP del Sistema Operativo lo genera de forma lineal, dependiendo del tiempo, aleatorio o seudo−aleatorio, según el sistema. El cracker sÃ³lo puede atacar a sistemas generando números de secuencia predecibles (generados linealmente o dependientes del tiempo).
- El ataque consiste en abrir una conexión TCP en el puerto deseado (rsh, por ejemplo). Para un mejor entendimiento vamos a recordar como trabaja el mecanismo de apertura de conexion TCP. Se hace en tres pasos (TCP Three Way Handshake):

1. El emisor manda un paquete que lleva el flag TCP SYN y un número de secuencia X a la máquina víctima.
2. La víctima responde con un paquete que lleva los flags TCP SYN y ACK (con un número de reconocimiento X+1) activos, su número de secuencia es Y.
3. El emisor envía un paquete que contiene el flag TCP ACK (con un número de reconocimiento Y+1) hacia la víctima.

Durante el ataque, el cracker no recibe el SYN−ACK enviado por la víctima. Para establecer la conexión, predecirá el número de secuencia y para poder enviar un paquete con el número ACK correcto (Y+1). La conexión es entonces estabilizada a través de la autentificación por dirección IP. El cracker puede ahora enviar un comando al servicio rsh, como un `echo ++ >> /.rhosts` para mayores permisos de acceso. Para hacer esto ha de crear un paquete con el flag TCP PSH (Push): los datos recibidos son inmediatamente enviados a la capa superior (aquí el servicio rsh). Puede conectar a la máquina a través de un servicio como rlogin o rsh sin IP Spoofing.

La figura 7 muestra los diferentes pasos de IP Spoofing:

Fig.7: IP Spoofing aplicado al servicio rsh



`rpcinfo`

El cracker usa la máquina A, mientras que C representa la máquina de confianza. A(C) significa que el paquete se envia desde A con la dirección IP de C burlada. Nota: existe un programa llamado `mendax` que implementa estos mecanismos de IP Spoofing.

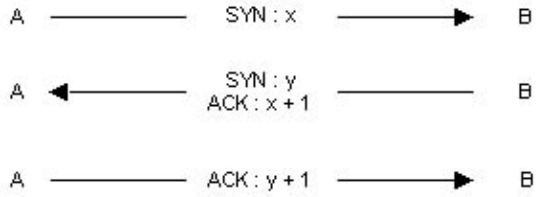
Secuestro de sesiones TCP

El secuestro de sesiones TCP permite al cracker redirigir un flujo TCP. Entonces, un cracker puede sobrepasar la protección por contraseÃ±a (como en telnet o ftp). La necesidad de escuchar (sniffing) restringe este ataque a la red física del atacado. Antes de detallar este ataque, vamos a explicar algunos principios básicos del protocolo TCP.

No vamos a revelar el misterio del protocolo TCP, pero sí concretaremos los principales puntos requeridos para entender el ataque. La cabecera TCP contiene varios campos:

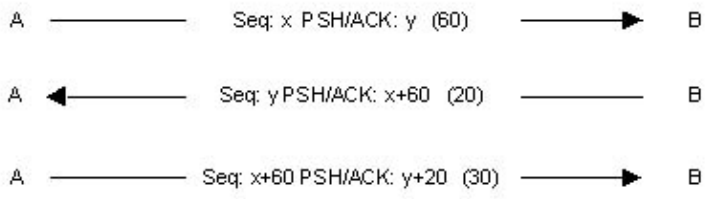
- El puerto origen y el puerto destino, identificando la conexión entre dos máquinas;
- El número de secuencia que identifica cada byte enviado;
- El número de reconocimiento que corresponde al último byte recibido;
- Los flags interesantes son:
    - ♦ SYN que sincroniza los números de secuencia cuando una conexión esta establecida;
    - ♦ ACK, el flag de reconocimiento de un segmento TCP;
    - ♦ PSH que dice al receptor que envÃ-e los datos a la aplicación.

La figura 8 muestra cÃ³mo establecer una conexión TCP (Three Way Handshake – Saludo en tres pasos):
Fig.8: Three Way Handshake – Saludo en tres pasos



Aquí la máquina A inició una conexión TCP en la máquina B.

Figura 9. Muestra la transferencia de datos TCP:



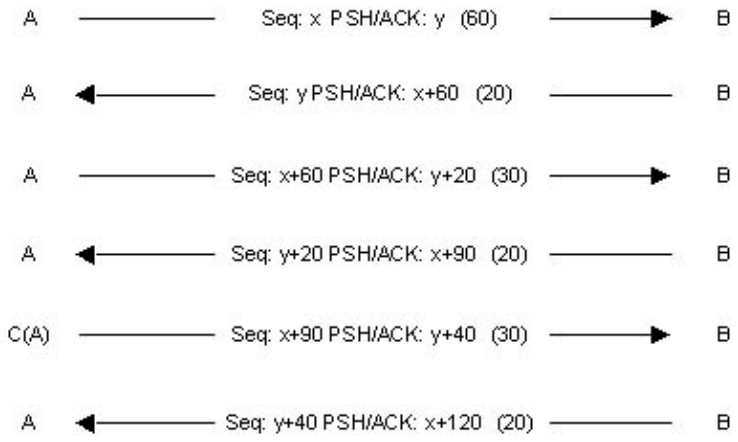
Los números de secuencia cambiarán de acuerdo al número de bytes de datos enviados. El número de secuencia se representa por *Seq*, el número de reconocimiento se encuentra después de los flags PSH y ACK y el número de bytes de datos enviados se encuentra entre paréntesis.

Este ataque crea un estado de desincronización en ambos lados de la conexión TCP, permitiendo el secuestro de la sesión. Una conexión es desincronizada cuando el número de secuencia del próximo byte enviado por la máquina A es diferente del número de secuencia del siguiente byte en ser recibido por B, y viceversa.

En el ejemplo de la figura 9, al final del primer paso, cuando B recibe su paquete, A espera por un paquete con un número de reconocimiento de X+60. Si el siguiente paquete enviado por B no tiene este número de reconocimiento, A y B se consideran desincronizados.

Así, si un cracker con una máquina C quiere secuestrar una sesión estabilizada de telnet, sesión entre las máquinas A y B. Primero, la máquina C escucha el tráfico telnet (puerto TCP 23) entre A y B. Una vez que el cracker cree que A ha tenido tiempo de autentificarse con el servicio telnet de la máquina B, desicroniza la máquina A con B. Para hacer esto, crea un paquete tomando la IP de origen de la máquina A y el número TCP de reconocimiento esperado por B. De acuerdo, la máquina B acepta el paquete, Además de desincronizar la conexión TCP, este paquete permite al cracker introducir un comando a través de la sesión telnet previamente establecida por A. A este paquete le está permitido portar datos (flag PSH = 1).

La figura 10 muestra este ataque:
Fig.10: Secuestro de sesión TCP.



La máquina B acepta el comando enviado por C, reconoce este paquete enviando un paquete a A con el flag ACK.

Lo que significa que si A envió un paquete a B, será rechazado porque el número de secuencia no es el esperado por B.

Surge entonces un problema: La tormenta ACK. Se generan muchos ACK. Esto sucede cuando A envía un paquete TCP con un número de secuencia inválido (A no está sincronizado), B lo rechaza y envía a A un ACK con el número de secuencia que espera. A recibe este ACK, y aunque el número de secuencia no coincide con el esperado, también envia un ACK a B y B lo hace de nuevo...

Este problema de tormenta ACK puede ser resuelto si el cracker utiliza ARP Spoofing. En este caso, la máquina C envenenará el caché ARP de la máquina B diciéndole que la IP de A está ahora asociada a la dirección MAC de C. Estas técnicas las implementa el programa `hunt`.

ARP Spoofing

Este ataque, también llamado redirección ARP, redirecciona el tráfico de una o varias máquinas de la red hacia la máquina del cracker. Esto se hace sobre la red física de las víctimas. Recordemos que es el protocolo ARP y cÃ³mo trabaja:

El protocolo ARP (*Address Resolution Protocol −Protocolo de Resolución de Dirección−)* implementa el mecanismo de resolución de una dirección IP a una dirección MAC Ethernet. El hardware de red se comunica intercambiando tramas Ethernet (obviamente en una red Ethernet), en la capa de enlace de datos. Para poder compartir esta información, es necesario que las tarjetas de red tengan una única dirección Ethernet: esta es la dirección MAC (*MAC=Media Access Control* −Control de Acceso al Medio−).

Cuando se manda un paquete IP, la máquina emisora necesita conocer la dirección MAC de la receptora. Para obtenerla, se envía una petición "broadcast" ARP a todo la red. Esta petición preguntará: "¿Cuál es la dirección MAC asociada a 'Ã©sta' dirección IP?". La máquina que tenga la dirección IP en cuestión responderá a través de un paquete ARP, proporcionando a la máquina emisora la dirección MAC solicitada. A partir de aquí la máquina origen conocerá cual es la dirección MAC que corresponde a la dirección IP a la que se quiere enviar paquetes. Esta correspondencia permanecerá durante algún tiempo en el caché (para evitar hacer una nueva petición cada vez que se envía un paquete).

Este ataque envenena el cachÃ© de la máquina objetivo. El cracker envía respuestas a la máquina objetivo diciéndole que la nueva MAC asociada a la IP de un gateway (puerta de enlace), por ejemplo, es la dirección IP del cracker. La máquina del cracker recibirá entonces todo el tráfico destinado al gateway. Así pues, suficiente para escuchar (y/o modificarlo). Después de lo cual, enrutará los paquetes hacia su destino real para que nadie se pueda percatar de los cambios.

El ARP Spoofing es útil cuando una red local utiliza switches. Estos redireccionan las tramas Ethernet a puertos distintos (cables "físicos") de acuerdo con la dirección MAC a la que vayan dirigidos. Luego, un sniffer puede capturar tramas si está bajo el mismo cable físico. Así ARP Spoofing permite escuchar el tráfico situado entre puertos distintos del switch.

Para implementar un ataque ARP Spoofing, el cracker utilizará un generador de paquetes ARP como son `ARPSpoof` o `nemesis`. Ejemplo: la máquina "víctima" 10.0.0.171, su gateway por defecto 10.0.0.1, y la máquina del cracker 10.0.0.227. Antes del ataque, el resultado de hacer un traceroute es:

```
[root@cible -> ~]$ traceroute 10.0.0.1
traceroute to 10.0.0.1 (10.0.0.1), 30 hops max, 40 byte packets
 1  10.0.0.1  (10.0.0.1)  1.218 ms  1.061 ms  0.849 ms
```

Y el cachÃ© ARP de la máquina objetivo es:

```
[root@cible -> ~]$ arp
Address      HWtype  HWAddress        Flags Mask  Iface
10.0.0.1     ether   00:b0:c2:88:de:65   C        eth0
10.0.0.227   ether   00:00:86:35:c9:3f   C        eth0
```

El cracker ejecuta entonces `ARPSpoof`:

```
[root@pirate -> ~]$ arpspoof -t 10.0.0.171 10.0.0.1
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
0:0:86:35:c9:3f 0:60:8:de:64:f0 0806 42: arp reply 10.0.0.1 is-at 0:0:86:35:c9:3f
```

Los paquetes enviados son paquetes ARP, envenenan el caché ARP de la máquina 10.0.0.171, con una respuesta ARP diciendo que la dirección MAC asociada a 10.0.0.1 ahora es `00:00:86:35:c9:3f`.

El caché ARP de la máquina 10.0.0.171 se convierte:

```
[root@cible -> ~]$ arp
Address     HWtype  HWAddress       Flags Mask  Iface
10.0.0.1    ether   00:00:86:35:c9:3f   C       eth0
10.0.0.227  ether   00:00:86:35:c9:3f   C       eth0
```

Para analizar qué tráfico va ahora sobre la máquina 10.0.0.227, es suficiente con ejecutar un nuevo traceroute sobre 10.0.0.1:

```
[root@cible -> ~]$ traceroute 10.0.0.1
traceroute to 10.0.0.1 (10.0.0.1), 30 hops max, 40 byte packets
 1  10.0.0.227  (10.0.0.227)  1.712 ms  1.465 ms  1.501 ms
 2  10.0.0.1  (10.0.0.1)  2.238 ms  2.121 ms  2.169 ms
```

Ahora el cracker puede escuchar el tráfico entre las máquinas 10.0.0.171 y 10.0.0.1. No debe olvidar activar el enrutado IP en su máquina: 10.0.0.227.

DNS Spoofing

El protocolo DNS (*Domain Name System* – Sistema de Nombres del Dominio) convierte un nombre de dominio (por ejemplo www.test.com) en su dirección IP (por ejemplo: 192.168.0.1) y viceversa. Este ataque utiliza respuestas falsas a las peticiones de DNS enviadas por una "víctima". Esta ataque recae sobre dos métodos principales:

DNS ID Spoofing

La cabecera del protocolo DNS contiene un campo de identificación para relacionar preguntas y respuestas. El objeto del DNS ID Spoofing es el devolver una respuesta falsa a una petición DNS antes de que el servidor DNS real pueda responder. Para hacer esto, el campo ID del solicitante ha de ser predecido. Localmente es sencillo predecirlo escuchando el tráfico de la red. Sin embargo, se convierte en algo más complicado al salir de una red local. Existen varios métodos:

- Probando todas las posibilidades en el campo ID. No muy realista ya que hay 65535 posibilidades (es un campo de 16 bits);
- Enviar unos cientos de peticiones DNS en el orden adecuado. Obviamente este método no es bastante seguro;
- Encontrar un servidor que genere IDs predecibles (por ejemplo, ID incrementando en 1). Este tipo de vulnerabilidades pueden ser encontradas en algunas versiones de Windows 9x.

En cualquier caso, es necesario responder antes que el servidor de DNS real. Esto puede hacerse, por ejemplo bloqueándolo con un ataque del tipo de denegación de servicio.

Para que ocurra el atacante debe controlar un servidor DNS (ns.attaquant.com) teniendo autoridad sobre el dominio attaquant.com. El servidor DNS objetivo (ns.cible.com) se supone que tiene súmeros de secuencia predecibles (por incremento a 1 en cada petición).

El ataque requiere cuatro pasos:

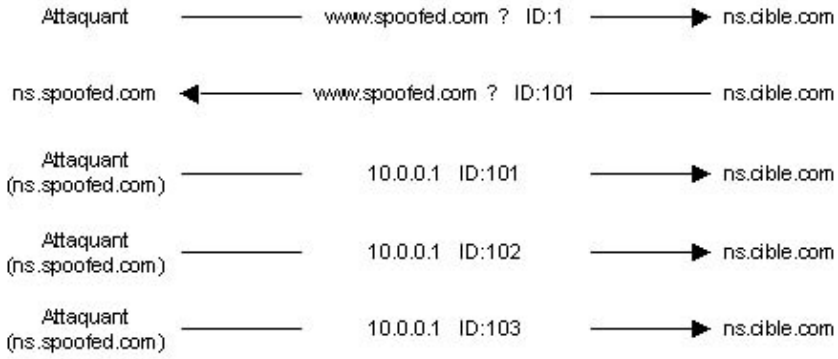
1. El atacante envía una petición DNS para el nombre www.attaquant.com al servidor DNS del dominio cible.com, como se ve en la figura 11;

   Fig.11: Petición DNS enviada a `ns.cible.com`

2. El servidor DNS objetivo transpasa la petición DNS al dominio `attaquant.com`;
3. El atacante puede ahora escuchar para obtener su ID (en el ejemplo el ID es 100);
4. El ataque altera la dirección IP asociada al nombre de la máquina, aquí la máquina "víctima" es www.spoofed.com que debe tener 192.168.0.1 como su dirección IP. El cracker envía una petición DNS para resolver el nombre www.spoofed.com a ns.cible.com. Inmediatamente a espaldas de esto, envía un puñado de respuestas DNS alteradas (dando como dirección IP la del atacante 10.0.0.1) a esta misma petición teniendo alterada la dirección IP original con uno de los servidores DNS del dominio spoofed.com. El ID de cada respuesta se incrementará en 1, comenzando por el recibido durante el segundo paso (ID=100) para mejorar el paso de obtener el número ID correcto. Este es justo el caso de ns.cible.com , que debe haber respondido a otras peticiones y así incrementar su DNS ID. La figura 12 muestra estos pasos.

   Pic.12: DNS ID Spoofing

El caché del servidor DNS objetivo está entonces envenenada y la próxima máquina que pregunte por la resolición del nombre www.spoofed.com obtendrá la dirección IP de la máquina del atacante y será redirigido a su site. Este último puede ser una "copia" del site real para engañar a los usuarios de Internet y robarles información confidencial.

Envenenación del caché DNS

Los servidores DNS pueden usar un cachÃ© para, localmente, guardar las respuestas de las peticiones de datos anteriores durante un periÃ³do de tiempo. Esto es para evitar la pérdida de tiempo de preguntar siempre al servidor de nombres que tiene autoridad sobre el dominio. Este segundo tipo de DNS Spoofing consistirá en envenenar esta caché con información falsa. Veamos un ejemplo:

Mantendremos los datos del ejemplo anterior. Los pasos del ataque son los siguientes:

- Envia una petición DNS para resolver el nombre www.attaquant.com al servidor DNS del dominio cible.com;
- El servidor DNS objetivo envia una petición para resolver el nombre www.attaquant.com al servidor DNS del atacante;
- El servidor DNS del atacante envia una respuesta con los registros alterados permitiendo asignar el nombre de máquina a una dirección IP perteneciente al atacante. Por ejemplo: el site www.cible.com puede tener un registro DNS alterado enviando de vuelta a la dirección IP de www.attaquant.com en lugar de la correcta.

Ataques de aplicaciones

Los ataques de aplicaciones recaen en vulnerabilidades específicas encontradas en algunas aplicaciones. Sin embargo, algunas de ellas pueden ser clasificadas por tipos.

El problema de la configuración

Uno de los principales problemas de seguridad encontrados en las aplicaciones viene de malas configuraciones. Hay dos tipos de errores: configuraciones por defecto y configuraciones erróneas.

El software, como los servidores web, con la instalación por defecto instalan ficheros que con frecuencia proporcionan sites de ejemplo que pueden ser usadas por los crackers para acceder a información confidencial. Por ejemplo, podrian proporcionar scripts que proporcionen los datos de origen via páginas dinámicas. Es más, una instalación puede proveer de un site remoto de administración con un login y password por defecto (localizable en la documentación de la aplicación). El cracker puede entonces cambiar lo que quiera en el site.

Las principales vulnerabilidades generadas por una mala configuración son listas de acceso con parámetros erróneos. Así los cracker pueden acceder a páginas privadas o bases de datos privadas.

Como ejemplo clásico de mala configuración, los fallos en el servidor web Lotus Domino. Cuando se instala este servidor, las bases de datos de configuración Lotus no tienen ninguna lista de control de acceso. Claramente, si el names.nsf puede ser accedido con un navegador web sin autentificar, es posible obtener mucha información, como cada nombre de usuario de Lotus.

Bugs

La mala programación del software siempre lleva a los bugs. Estas son las vulnerabilidades más importantes. Cuando se descubren, permiten ejecutar comandos no autorizados, obtener código fuente de páginas dinámicas, hacer que un servicio no esté operativo, tomar el control de la máquina, etc. El más conocido e interesante de cara a los exploits es el buffer overflow (desbordamiento de búfer).

Buffer overflow

El desbordamiento de búfer es una vulnerabilidad causada por una mala programación. Aparece cuando una variable pasada como argumento a una función se copia en un búfer sin comprobar el tamaño. Si el tamaño es superior al espacio que el bÃºfer tiene asignado es suficiente para que se produzca el desbordamiento. Será explotado pasando a la variable un fragmento de programa. Si un cracker ejecuta este tipo de ataque obtendrá la capacidad de ejecutar remotamente comandos en la máquina objetivo con los permisos del programa atacado. Sobre esto puedes encontrar más información en la serie de Programación segura:

- Evitando los agujeros de seguridad durante el desarrollo de aplicaciones – Parte 1
- Evitando los agujeros de seguridad durante el desarrollo de aplicaciones – Parte 2: memoria, pila y funciones, cÃ³digo shell
- Evitando los agujeros de seguridad durante el desarrollo de aplicaciones – Parte 3: buffer overflows (desbordamientos del bÃºfer)
- Evitando los agujeros de seguridad durante el desarrollo de aplicaciones – Parte 4: secuencias de formato
- Evitando los agujeros de seguridad durante el desarrollo de aplicaciones – Parte 5: condiciones durante la carrera
- Evitando los agujeros de seguridad durante el desarrollo de aplicaciones – Parte 6: scripts CGI

Scripts

La mala programación de scripts frecuentemente afecta a la seguridad del sistema. Hay que significar que explotando las vulnerabilidades de scripts hechos en PERL que permitan leer ficheros fuera del árbol principal de la web o ejecutar comandos no autorizados. Estos problemas de programación se presentan en los anteriormente mencionados artículos de seguridad en CGIs (la Parte 6).

Man in the Middle

El objeto de este ataque es desviar el tráfico ente dos máquinas. Esto es para interceptar, modificar o destruir los datos circulantes durante la comunicación. Este más que un ataque real es un concepto. Existen varios ataques que implementan la teoría del Man in the Middle, como son el DNS Man in the Middle que usa DNS Spoofing para desviar el tráfico entre un servidor web y un cliente web. Recientemente ha sido creaeda una aplicación para desviar el tráfico SSH.

Denegación de servicio

Este tipo de ataque fiel a su denominación provoca la incapacidad/indisponibilidad de un servicio (aplicación específica) o de una máquina. Distinguiremos dos tipos de denegación de servicio: por un lado, los que explotan un fallo en la aplicación y por otro lado los relativos a la mala implementación o en la debilidad de un protocolo.

Denegación de servicio de aplicación

Si las vulnerabilidades de una aplicación pueden llevar a la posibilidad de tomar el control de una máquina (ejemplo del buffer overflow), pueden también llevar a la denegación de servicio. La aplicación se volverá indisponible o por bloqueo de los recursos de la máquina o mismo por bloqueo completo de la misma.

Denegación de servicio de red

Existen diferentes tipos de denegación de servicio utilizando características de la pila TCP/IP.

SYN Flooding (inundación de SYNs)

Ya hemos visto que una conexión TCP se establece en tres etapas (TCP Three Way Handshake). SYN Flooding explota este mecanismo. Las tres etapas son enviar un SYN, recibir un SYN−ACK y enviar un ACK. La idea es dejar en la máquina objetivo un número elevado de conexiones TCP en espera. Para hacer esto, el cracker envia un montón de peticiones de conexiones (SYN flag=1), la máquina objetivo envia el SYN−ACK de vuelta para responder al SYN recibido. El cracker no le responderá con el ACK, así para cada SYN recibido, la máquina objetivo tendrá una conexión abierta. Además estas conexiones medio−abiertas usan recursos de memoria, despues de un rato la máquina se saturará y no podrá aceptar más conexiones. Este tipo de denegación de servicio solo afecta a la máquina objetivo.

El cracker utiliza un inundador de SYNs como el synk4, indicando el puerto TCP destino utilizando una IP de origen aleatoria para evitar que la máquina atacante sea identificada.

UDP Flooding (inundación UDP)

Esta denegación de servicio explota la forma de trabajo que tiene el protocolo UDP (sin conexión). Crea una tormenta de paquetes UDP sobre una, dos o varias máquinas . Un ataque entre dos máquinas llevaría hacia la congestión de la red sobre la que trabajen estas máquinas, así como la saturación de los recursos de ambas máquinas. La congestión de la red es más importante ya que el tráfico UDP tiene prioridad sobre el tráfico TCP. El protocolo TCP tiene un mecanismo para controlar la congestión en el caso de que se conozca que un paquete llega despues de "un largo tiempo", este mecanismo adapta la frecuencia con la que se envían los paquetes TCP de froma que disminuye el radio. El protocolo UDP no posee este mecanismo: despuÃ©s de un rato el tráfico UDP utilizara todo el ancho de banda dejando una pequeñísima parte al tráfico TCP.

El caso más conocido de inundación UDP es el "*Chargen Denial of Service Attack*" la implementación de este ataque es sencilla: Es suficiente con establecer una comunicación entre el servicio `chargen` de una máquina y el servicio `echo` de otra. El servicio chargen genera caracteres mientras que el servicio echo reenvía los datos que recibe. El cracker envía paquetes UDP al puerto 19 (chargen) de una de las víctimas dando la dirección IP y el puerto de origen de la otra. En este caso, puerto de origen UDP 7 (echo). La inundación UDP lleva a la saturación del ancho de banda entre ambas máquinas. Una red completa puede ser la víctima de una inundación UDP.

Fragmentación de paquetes

La denegación de servicio de la fragmentación de paquetes se aprovecha de la debilidad de algunas pilas TCP/IP en lo concerniente a la defragmentación IP (reensamblado de fragmentos IP).

Un ataque conocido que usa esto es Teardrop. La fragmentación offset del segundo fragmento es menor que el tamaño del primero y así para el offset añadido al tamaño del segundo. Esto significa que el primer fragmento contiene al segundo (superposición). En el tiempo de desfragmentación, algunos sistemas no gestionan esta excepción y esto les lleva a una denegación de servicio. Hay variantes de este ataque: bonk, boink y newtear por ejemplo. La denegación de servicio "ping de la muerte" explota la mala administración de la defragmentación de ICMP, enviando más datos que el tamaño máximo de un paquete IP. Estos tipos diferentes de denegación de servicio llevan al "cuelgue" de la máquina objetivo.

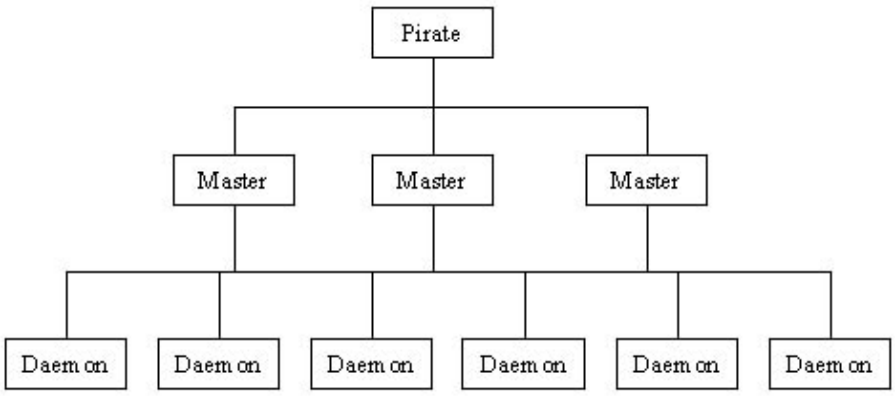
Smurfing

Este ataque usa el protocolo ICMP. Cuando se envía un ping (mensaje ICMP ECHO) a una dirección de broadcast (por ejemplo 10.255.255.255), es reducido a enviar a cada máquina de la red. El principio del ataque es inundar de paquetes ICMP ECHO REQUEST enviados usando como IP de origen la del objetivo. El cracker envía un flujo continuo de pings al broadcast de la red y todas las máquinas responderán con un mensaje ICMP ECHO REPLY. El flujo se multiplica por el número de hosts el la red. En este caso, toda la red en la que se encuentra el objetivo se verá afectada por la denegación de servicio, ya que el gran tráfico que se genera con este ataque producirá una congestión en la red.

Denegación de servicio distribuida

La denegación de servicio distribuida satura a la red atacada. La idea es usar varios origenes (demonios) para el ataque y masters para controlarlos. Las herramientas más conocido para DDoS (*Distributed Denial of Service*) son Tribal Flood Network (TFN), TFN2K, Trinoo y Stacheldraht. La figura 13 muestra un DDoS de red típico:

Fig.13: DDoS de red



El cracker usa masters para controlar facilmente los orígenes. Obviamente, necesita conexiones (TCP) a los masters para configurar y preparar el ataque. Los masters sÃ³lo envian comandos a los origenes via UDP. Sin los masters, el cracker debería conectarse a cada origen. El origen del ataque sería detectado de forma más sencilla y su implementación sería más larga.

Cada demonio y master se comunican entre si para intercambiar mensajes específicos dependiendo de la herramienta usada. Estas comunicaciones pueden también estar encriptadas y/o autentificadas. Para instalar los demonios y los masters, el cracker usará vulnerabilidades conocidas (buffer overflows en servicios como RPC, FTP, etc.). El ataque en sÃ-, es un SYN Flooding o un Smurf. El resultado de cada una de las denegaciones de servico es crear una red no disponible.

Conclusión

Hoy en día, la seguridad contra los ataques remotos se está incrementando pero desafortunadamente esto no es cierto para la seguridad interna. Esta "pobre relación" de protección contra crackers sigue dirigiendo hacia grandes perspectivas de ataques locales como los secuestros de sesiones TCP, ARP Spoofing y DNS Spoofing. Es más la predicción de los números de secuencia (corazón del IP Spoofing) y las variantes de Ataques de Fragmentación aparecen por los fallos encontrados en el soporte de red de los Sistemas Operativos. En lo concerniente a los ataques de aplicación, seguirán gozando de buenos tiempos debido al crecimiento en la complejidad de las aplicaciones web y del poco tiempo que se les da a los programadores y administradores. El ataque de denegación de servicio permanecera en su forma distribuida tanto tiempo como cada usuario siga sin darse cuenta de la necesidad de proteger su máquina.

Enlaces

- RFC 1858 – Security Considerations for IP Fragment Filtering: sunsite.dk/RFC/rfc/rfc1858.html
- IP Spoofing Demystified – Phrack 48: www.phrack.org/
- Simple Active Attack Against TCP – Laurent Joncheray: www.insecure.org/stf/iphijack.txt
- DNS ID Hacking – ADM Crew: packetstorm.securify.com/groups/ADM/ADM–DNS–SPOOF/ADMID.txt
- The DoS Project's "trinoo" – David Dittrich: staff.washington.edu/dittrich/misc/trinoo.analysis
- The Strange Tale of the DENIAL OF SERVICE Attacks against GRC.COM: grc.com
- hping2: www.kyuzz.org/antirez/hping.html
- nemesis: www.packetfactory.net/Projects/nemesis/
- mendax: _packetstorm.securify.com/Exploit_Code_Archive/mendax_linux.tgz
- hunt: lin.fsid.cvut.cz/~kra/index.html
- dsniff: www.monkey.org/~dugsong/dsniff/
- fragrouter: packetstorm.securify.com/UNIX/IDS/fragrouter–1.6.tar.gz

Webpages maintained by the LinuxFocus Editor team © Eric Detoisien "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org	Translation information: fr –––> –– : Eric Detoisien <valgasu(at)club–internet.fr> fr –––> en: Georges Tarbouriech <gt(at)linuxfocus.org> en –––> es: Carlos González Pérez <carlosdw(en)terra.es>

2005–01–10, generated by lfparser_pdf version 2.51