Aprendiendo a usar IPTABLES desde cero.

Introducción

Al conectarnos a internet en nuestras casas, de forma explícita nos estamos conectando, en AMBOS sentidos: directamente a la red, "desnudos" si se me permite la analogía. El objetivo de este artículo es, medinte iptables,
lograr cierta protección y seguridad. Nótese que un firewall no garantiza 100% de seguridad.
Para este artículo, se requieren conocimientos básicos/intermedios acerca de linux y TCP/IP, obviamente implementado bajo dicho sistema operativo.

Cabe destacar que para utilizar iptables, es necesario tener un kernel preparado para éste, y el módulo ip_tables cargado. Del mismo modo, si estas utilizando ipchains, se deben descargar sus módulos antes de cargar los de iptables.
Iptables es una utilidad de linux que se encarga de darle directivas al kernel, acerca del filtrado de paquetes TCP/IP. Un paquete TCP/IP consta de varios campos, con información adicional a los datos que se transmiten en sí. No viene al caso describir cada uno de ellos, sino los que consideraremos mas relevantes, es decir, aquellos campos que mediante
iptables, empezaremos a "vigilar". Ejemplo: direccion origen, direccion destino, puerto origen, puerto de destino, etc.

Como funciona?

El kernel de linux posee (predefinidas "de fábrica") 3 cadenas (chains) de reglas: INPUT, FORWARD y OUTPUT. Cada paquete TCP/IP que ingresa/atraviesa/sale desde/hacia una maquina con iptables funcionando, pasará por la cadena INPUT, FORWARD o OUTPUT respectivamente. Las cadenas,no son mas que un listado de reglas, con las cuales controlamos cada uno de
los paquetes que pasan. Ahora se preguntaran, que es una regla?. Para evitar las confusiones, vamos a simplificar su definición al máximo, y luego mostrarles algunos ejemplos. Una regla consta de 2 partes, y no es mas que una condición y una acción. Si se cumple la condición se ejecuta la acción. Simple, verdad?

Algo para tener en cuenta mas adelante: si un paquete atravesó todas las reglas de una cadena, sin hallar coincidencia, iptables se fijará en la politica por defecto de esa cadena(default policy).
Resumiendo lo anteriormente dicho:

1) Lo mas importante: tener conocimientos previos en linux y TCP/IP !!!
2) El kernel de linux trae 3 cadenas de reglas (INPUT, FORWARD y OUTPUT)
3) Con iptables se pueden agregar/eliminar reglas (entre otras cosas).

Supongamos la siguiente situación...Había una vez, una red local 192.168.1.0 con 50 PC's, de las cuales destacaremos:

1) aquihacker.mired.lan (192.168.1.15)
2) neutral.mired.lan (192.168.1.18)
3) jose.mired.lan (192.168.1.20)

Nosotros somos los operadores de "jose.mired.lan", estamos corriendo un servicio ftp (wu-ftpd*), y estamos enterados de que el muchacho que usa "aquihacker", es alguien muy curioso, que le gusta investigar en máquinas ajenas, utilizando su interfaz de red.

Experimentando con paquetes icmp

Vamos a ver cual es el estado de nuestras 3 cadenas, listando lo que hay en ellas. Para eso ejecutamos el siguiente comando:

```
[root@jose /]# /sbin/iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Si la salida obtenida no se parece a lo que ven aqui arriba, significa que no tienen instalados los módulos correspondientes en el kernel, o que ya hay un firewall funcionando, o para confundirlos más: significa que ya se ejecutó un script probablemente en el inicio del sistema) donde se cargan las reglas del firewall. Consulten la documentación de la distribución que instalaron :P

Suponiendo que la salida que obtuvieron es igual a la escrita aqui arriba,su situación es la siguiente: quiere decir que el firewall esta aceptando todos los paquetes, o lo que es lo mismo, no filtra absolutamente nada. Esto nos indica 2 cosas:

- Las 3 cadenas (INPUT/FORWARD/OUTPUT) estan vacías
- Las 3 cadenas tienen como politica default "ACCEPT".

Es normal que al hacer un ping a localhost (nosotros mismos) recibamos respuesta:

```
[root@jose /]# ping localhost
PING localhost.localdomain (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=0 ttl=255 time=0.3ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=1 ttl=255 time=0.1ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=2 ttl=255 time=0.1ms
(y así sucesivamente...)
```

Vamos a agregar una regla, para entender el funcionamiento:

```
[root@jose /]# /sbin/iptables --append INPUT --protocol icmp --source 127.0.0.1 --jump
DROP
```

Qué es esto? Simple: Agregar (append) la siguiente regla a la cadena INPUT: al recibir un paquete del tipo icmp (--protocol) con origen (--source) "127.0.0.1" y con cualquier destino (ya que no lo especificamos), enviamos ese paquete (--jump) a DROP, que por cierto es lo mismo que dejarlo tirado =).

Que? No pasó nada, verdad? Seguro? Entonces listemos otra vez las reglas que tenemos cargadas (te olvidaste como se hacia??)

```
[root@jose /]# /sbin/iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  127.0.0.1           0.0.0.0/0

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Notamos que en la cadena INPUT, se nos agregó una regla, cuya función es impedir que se haga ping desde nuestra propia máquina. Podrán notar que esta regla no es demasiado útil a los fines prácticos, pero si lo es para ejemplificar el uso. Y que es lo que faltaría? Comprobar su funcionamiento!! por supuesto!!

```
[root@jose /]# ping localhost
PING localhost.localdomain (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
```

(y se quedará esperando una respuesta que nunca llegará...)

Nuestro firewall está invisible? Claro que no. De hecho, veremos que desde la pc "aquihacker", hacemos ping sin problemas, y nuestra maquina le responde.

```
[root@aquihacker /]# ping jose.mired.lan
PING jose.mired.lan (192.168.1.20) from 192.168.1.15 : 56(84) bytes of data.
64 bytes from 192.168.1.20: icmp_seq=0 ttl=255 time=2.524 msec
64 bytes from 192.168.1.20: icmp_seq=1 ttl=255 time=1.319 msec
```

Con lo cual, con un simple ping, nuestro amigo operador de "aquihacker" puede notar la presencia de nuestra PC. A no desesperar que esto recien comienza.

Restringiendo paquetes provenientes del exterior

Puesto que nuestro firewall está filtrando los paquetes ICMP locales, y deja pasar los remotos. Vamos a cambiar un poco las reglas "del juego".
Borramos la regla que anteriormente agregamos:

```
[root@jose /]# /sbin/iptables --delete INPUT --protocol icmp --source 127.0.0.1 --jump DROP
```

Esto se logra con la opción "-D", y el resto, es la transcripción EXACTA dela regla que queremos borrar.
NOTA: conociendo que era la única regla, hubieramos logrado el mismo resultado escribiendo:

```
[root@jose /]# /sbin/iptables --delete INPUT 1
```

Ahora lo que importa: Restringir el acceso a paquetes icmp desde la red!!!

```
[root@jose /]# /sbin/iptables -A INPUT -p icmp -s 192.168.1.15 -j DROP
```

Notese que es indistinto escribir -A o --append, -p o --protocol, etc. Ahora, listamos nuestra cadena INPUT:

```
[root@jose /]# /sbin/iptables -nL INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  192.168.1.15        0.0.0.0/0
```

Listo, ahora cualquier paquete icmp con origen en 192.168.1.15 será "DROPeado". Por ello, cuando nuestro amigo curioso hace ping desde su máquina a la nuestra, jamás tendrá respuesta..

```
[root@aquihacker /]# ping jose.mired.lan
PING jose.mired.lan (192.168.1.20) from 192.168.1.15 : 56(84) bytes of data.
```

Listo. Para nuestro amigo, nuestra máquina está invisible, por consiguiente, logramos nuestro objetivo de asegurarla.
Cierto? NOOO!! Sacrilegio!! NO no no y nooooO!

Nuestro amigo, extrañado por que el comando ping no respondió, en un principio pensó que tendríamos la PC apagada, pero luego ejecutó...

```
[root@aquihacker /]# ftp jose.mired.lan
Connected to jose.mired.lan.
220 jose FTP server (Version wu-2.6.1-16) ready.
Name (jose:root):
```

Oh Oh! Nos hemos olvidado que teníamos un FTP server corriendo... Y corriendo apurados agregamos la siguiente regla:

```
[root@jose /]# /sbin/iptables -A INPUT -p tcp -s 192.168.1.15 --dport 21 -j DROP
```

Listamos la cadena INPUT y obtenemos:

```
[root@jose /]# /sbin/iptables -nL INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  192.168.1.15         0.0.0.0/0
DROP       tcp  --  192.168.1.15         0.0.0.0/0           tcp dpt:21
```

Por su parte, nuestro "amigo" se aburrirá de esperar hasta que nuestro FTP le responda (cosa que no sucederá).
Interesante, verdad?

Ahora: pensemos lo siguiente, que pasa si en nuestra red, de las 50 PC's que la conforman, la mayoría son "amigos curiosos" ??. Agregaremos una regla para cada uno de ellos? Sería un trabajo bastante tedioso, y poco eficiente.
Además, un detalle realmente importante: si bien todas estas reglas, se aplicarían sin problemas, y son correctas sintácticamente, es MUY ACONSEJABLE seguir la filosofía de "Todo lo que no está EXPLICITAMENTE permitido, entonces está prohibido".
Como logramos esto? Como primer paso, setearemos la politica por defecto en DROP (descartar). Cualquier paquete que circula por la cadena INPUT es DROPeado si no coincide con ninguna regla.
En caso de que estén editando el iptables desde una consola local, pueden ignorar el parrafo siguiente:

---
* NOTA IMPORTANTE: para aquellos que estan toqueteando el iptables remotamente (ya sea por telnet o ssh), si setean como política DROP para la cadena INPUT se les caerá la conexión. De hecho me pasó a mi, es por eso que previamente deberan agregar la siguiente regla:

```
[root@jose /]# /sbin/iptables -A INPUT -p tcp -s #ORIGEN# --dport
  #PUERTO# -j DROP
```

Donde #ORIGEN# es el IP desde donde se están conectando al linux con iptables y #PUERTO# debe ser el 22 (si es conexion ssh) o 23 (si usan telnet).

Ahora, como decíamos, continuaremos seteando la política de INPUT en DROP (con la opción -P)

```
[root@jose /]# /sbin/iptables -P INPUT -j DROP
```

Vaciamos todas las cadenas con la opción -F (flush)

```
[root@jose /]# /sbin/iptables -F
```

Listamos lo que tenemos:

```
[root@jose /root]# /sbin/iptables -nL INPUT
```

```
Chain INPUT (policy DROP)
target     prot opt source               destination
```

Aunque no tengamos ninguna regla, nadie va a poder acceder a nuestra PC por ningun motivo. Justamente, cualquier paquete que entra, es expuesto a cada regla de la cadena (en nuestro caso nuestra cadena esta vacía), al no encontrar coincidencias, el destino del paquete, lo decide la política de la cadena, osea DROP. El problema aquí es que aunque nosotros podamos realizar
conexiones salientes, las respuestas de los servidores serán descartadas. Para solucionar esto, agregamos la siguiente regla:

```
[root@jose /root]# /sbin/iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Por la cual dejamos pasar cualquier paquete cuya conexión ya se ha establecido (ESTABLISHED), o cuya conexión es nueva, pero está relacionada a una conexión ya establecida (RELATED), según las man pages del iptables (les dije que las lean, no?)
Por enésima vez, listamos lo que tenemos en nuestra cadena INPUT!!

```
[root@jose /root]# /sbin/iptables -nL INPUT
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state
RELATED,ESTABLISHED
```

Con lo cual, nadie podrá iniciar una conexion a nuestra maquina, salvo que especifiquemos...
Como verán, por defecto tenemos un DROP. Ahora podemos empezar a permitir conexiones.

Acceso local:

```
[root@jose /root]# /sbin/iptables -A INPUT -i lo -s 127.0.0.1 -j ACCEPT
```

-i lo : Con esto especifico como interfaz de red a localhost (lo).

```
[root@jose /root]# /sbin/iptables -nL INPUT
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state
RELATED,ESTABLISHED
ACCEPT     all  --  127.0.0.1            0.0.0.0/0
```

Le permito a "neutral", el acceso ftp (puerto 21)

```
[root@jose /root]# /sbin/iptables -A INPUT -p tcp -s 192.168.1.18 --dport 21 -j ACCEPT
[root@jose /root]# /sbin/iptables -nL INPUT
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state
RELATED,ESTABLISHED
ACCEPT     all  --  127.0.0.1            0.0.0.0/0
ACCEPT     tcp  --  192.168.1.18         0.0.0.0/0           tcp dpt:21
```

Consejos útiles y comentarios finales

Por último, aún no apaguen su máquina. Los cambios realizados 'on-the-fly' a iptables, no quedan guardados en ningún archivo, sino que se almacenan en memoria, y como sabrán, al reiniciar, los cambios se borrarán. Tengan en
cuenta los siguientes consejos:

1) Guarden todas sus reglas en un script prolijo, y con comentarios explicando cada regla o grupo de reglas aplicadas

2) En caso de ejecutar el script automáticamente al iniciar la máquina, consideren hacerlo antes de levantar ningún servicio, o mejor aún, antes de levantar ninguna interfaz de red. No se preocupen por las reglas que hacen referencia a interfaces no levantadas, igualmente son aceptadas por iptables.