iptables (Español)

Iptables es un poderoso **firewall** integrado en el kernel de Linux y que forma parte del proyecto **netfilter**. Iptables puede ser configurado directamente, como también por medio de un **frontend**[broken link: invalid section] o una **GUI**[broken link: invalid section]. iptables es usado por **IPv4**, en tanto que ip6tables es usado para **IPv6**.

nftables (http://netfilter.org/projects/nftables/) está programada para **ser liberada con el kernel de Linux 3.13 (http://www.phoronix.com/scan.php?page=news_item&px=MTQ5MDU)**, y vendrá a sustituir definitivamente iptables como la principal utilidad de cortafuegos de Linux. Por ahora, un howto está disponible **aquí (https://home.regit.org/netfilter-en/nftables-quick-howto/)**.

Artículos relacionados

- **Firewalls (Español)**
- **Simple stateful firewall**
- **Sysctl#TCP/IP stack hardening**
- **Sshguard**
- **Fail2ban**

Contents

Instalación

Todos los kernels de serie de Arch Linux son compatibles con iptables. Solo necesita **instalar** las herramientas en el **espacio de usuario**, que son proporcionadas por el paquete **`iptables (https://www.archlinux.org/packages/?name=iptables)`** presente en los **repositorios oficiales**.

Conceptos básicos

Tablas

iptables cuenta con cinco *tablas*, que son zonas en las que una cadena de reglas se puede aplicar:

1. `raw` filtra los paquetes antes que cualquier otra tabla. Se utiliza principalmente para configurar exenciones de seguimiento de conexiones en combinación con el target `NOTRACK` .
2. `filter` es la tabla por defecto (si no se pasa la opción `-t`).

3. `nat` se utiliza para la **traducción de dirección de red** (por ejemplo, el redirección de puertos). Debido a las limitaciones en iptables, el filtrado no se debe hacer aquí.
4. `mangle` se utiliza para la alteración de los paquetes de red especializados (véase **Mangles packet**).
5. `security` se utiliza para reglas de conexión de red **Mandatory Access Control**.

Cadenas

Las tablas contienen *cadenas*, que son listas de reglas que ordenan los **paquete de red**. Por defecto, la tabla `filter` contiene tres cadenas integradas: `INPUT`, `OUTPUT` y `FORWARD`.

1. Todo el tráfico entrante, dirigido a la máquina, se hace pasar a través de la cadena `INPUT`.
2. Todo el tráfico saliente, generado localmente, pasa a través de la cadena `OUTPUT`
3. Todo el tráfico enrutado, que no se ha suministrado localmente, pasa a través de la cadena `FORWARD`.

Véase `iptables(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/iptables.8)` para obtener una descripción de las cadenas integradas en otras tablas.

El usuario puede definir las reglas de las cadenas para hacerlas más eficientes.

Las cadenas compiladas tienen un target predefinido, que se utiliza si no hay reglas definidas. Ni las cadenas compiladas ni las definidas por el usuario pueden ser un target predefinido.

Reglas

El filtrado de los paquetes de red se basa en *rules -reglas-*, que se especifican por diversos *matches -«coincidencias»-* (condiciones que el paquete debe satisfacer para que la regla se puede aplicar), y un *target -«objetivo»* (acción a tomar cuando el paquete coincide con la condición plenamente). Si bien las condiciones individuales suelen ser muy simples, la especificación de la regla completa puede ser muy compleja.

Los targets se especifican mediante la opción `-j` o `--jump`. Los targets pueden ser tanto las cadenas definidas por el usuario, como uno de los targets integrados especiales, o una extensión de target. Los targets integrados son `ACCEPT`, `DROP`, `QUEUE` y `RETURN`; las extensiones de target son, por ejemplo, `REJECT` y `LOG`. Si el target es un target integrado, el destino del paquete es decidido inmediatamente y el procesamiento del paquete red en la tabla actual se detiene. Si el target es una cadena definida por el usuario y el paquete supera con éxito esta segunda cadena, se moverá a la siguiente regla de la cadena inicial. Las extensiones de target pueden ser tanto *terminating* (como los targets integrados) como *non-terminating* (como las cadenas especificadas por el usuario), Véase `iptables-extensions(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/iptables-extensions.8)` para obtener más detalles

Módulos

Hay muchos módulos que pueden ser utilizados para reforzar iptables, como connlimit, conntrack, limit y recent. Estos módulos añaden funcionalidad extra al permitir reglas de filtrado avanzadas.

Configuración

Desde la linea de órdenes

Mostrar las reglas vigentes

Puede comprobar el *ruleset*[1] en uso y el número de paquetes de red que ha satisfecho cada regla usando la orden:

```
# iptables -nvL
```

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0K packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Si la salida se parece a lo anterior, significa que no hay reglas establecidas. Nada está bloqueado.

Para mostrar los números de línea cuando se listan reglas, añada `--line-numbers` a esa entrada. Esto es útil cuando se eliminan y añaden reglas individuales.

1 Los «rulesets» son una agrupación de conjuntos de reglas que funcionan a modo de «subrutina» de cualquier lenguaje de programación.

Modificar las reglas

Las reglas pueden ser añadidas o bien como un apéndice a las reglas o a las cadenas, o insertarlas en una específica posición en la cadena. Exploraremos ambos métodos.

Primero de nada, nuestro equipo no es un router (salvo que, por supuesto, lo **sea**). Cambiaremos la política, por defecto, en la cadena `FORWARD`, de `ACCEPT` a `DROP`.

```
# iptables -P FORWARD DROP
```

Advertencia: El resto de esta sección está destinada a enseñar la sintaxis y los conceptos que se esconden detrás de las reglas de iptables. No tiene la pretensión de enseñar cómo proteger los servidores. Para mejorar la seguridad de su sistema, consulte **Simple stateful firewall** para una configuración de iptables mínimamente segura y **Security** para proteger Arch Linux en general.

El servicio LAN de sincronización de **Dropbox** tiene la característica de **transmitir paquetes de red cada 30 segundos (https://isc.sans.edu/port.html?port=17500)** a todos los equipos que estén en su área inalámbrica. Si estamos en una zona LAN con clientes de Dropbox y no usamos esta característica, entonces podríamos quererla para rechazar los paquetes.

```
# iptables -A INPUT -p tcp --dport 17500 -j REJECT --reject-with icmp-port-unreachable
```

```
# iptables -nvL --line-numbers

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        0     0 REJECT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:17500 reject-with i
cmp-port-unreachable

Chain FORWARD (policy DROP 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
```

Nota: Utilizamos `REJECT` en lugar de `DROP` aquí, porque **RFC 1122 3.3.8 (https://tools.ietf.org/html/rfc1122#page-69)** requiere que los hosts devuelvan los errores ICMP siempre que sea posible, en vez de dejar caer los paquetes de red. En realidad, lo mejor es `REJECT` para los paquetes de red del hosts que sepan acerca de la existencia de su servidor, y `DROP` para los paquetes de red del hosts que no sepan siquiera que existe el servidor.

Ahora, supongamos que decidimos utilizar Dropbox e instalarlo en nuestro ordenador. También queremos la sincronización LAN, pero solo con una IP en particular en nuestra red. Así que debemos utilizar `-R` para sustituir nuestra antigua regla. Donde `10.0.0.85` es nuestra otra IP:

```
# iptables -R INPUT 1 -p tcp --dport 17500 ! -s 10.0.0.85 -j REJECT --reject-with icmp-port-unreachable
```

```
# iptables -nvL --line-numbers

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        0     0 REJECT     tcp  --  *      *      !10.0.0.85            0.0.0.0/0            tcp dpt:17500 reject-with i
cmp-port-unreachable

Chain FORWARD (policy DROP 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
```

Hemos reemplazado nuestra regla original con otra que nos permite usar `10.0.0.85` para acceder al puerto `17500` de nuestro equipo. Pero ahora nos damos cuenta de que este no es escalable. Si un usuario amigable de Dropbox está intentando acceder al puerto `17500` de nuestro dispositivo, se debe permitir acceder de inmediato, sin comprobación frente a cualquier regla de firewall que esté detrás.

Así que escribimos una nueva regla para permitir que nuestro usuario de confianza acceda inmediatamente. Utilizaremos `-I` para insertar la nueva regla antes de la antigua:

```
# iptables -I INPUT 1 -p tcp --dport 17500 -s 10.0.0.85 -j ACCEPT -m comment --comment "Friendly Dropbox"
```

```
# iptables -nvL --line-numbers

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        0     0 ACCEPT     tcp  --  *      *       10.0.0.85            0.0.0.0/0            tcp dpt:17500 /* Friendly D
ropbox */
2        0     0 REJECT     tcp  --  *      *      !10.0.0.85            0.0.0.0/0            tcp dpt:17500 reject-with i
cmp-port-unreachable

Chain FORWARD (policy DROP 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
```

Y reemplazar nuestra segunda regla con otra que rechace todo lo demás que entre al puerto `17500` :

```
# iptables -R INPUT 2 -p tcp --dport 17500 -j REJECT --reject-with icmp-port-unreachable
```

La lista de nuestra regla final ahora se vería así:

```
# iptables -nvL --line-numbers

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        0     0 ACCEPT     tcp  --  *      *       10.0.0.85            0.0.0.0/0            tcp dpt:17500 /* Friendly D
ropbox */
2        0     0 REJECT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:17500 reject-with i
cmp-port-unreachable

Chain FORWARD (policy DROP 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
```

```
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
```

Restablecer las reglas

Puede vaciar y restablecer la configuración, por defecto, de iptables utilizando las siguientes órdenes:

```
# iptables -F
# iptables -X
# iptables -t nat -F
# iptables -t nat -X
# iptables -t mangle -F
# iptables -t mangle -X
# iptables -t raw -F
# iptables -t raw -X
# iptables -t security -F
# iptables -t security -X
# iptables -P INPUT ACCEPT
# iptables -P FORWARD ACCEPT
# iptables -P OUTPUT ACCEPT
```

La orden `-F` sin argumentos vuelca todas las cadenas en su tabla actual. Del mismo modo, `-X` elimina todos las cadenas vacías no predeterminadas en una tabla. Las cadenas individuales pueden ser eliminados o borrados con `-F` y `-X` seguidas de un argumento `[chain]`.

Archivo de configuración

Las reglas iptables son, por defecto, almacenadas en `/etc/iptables/iptables.rules`. Este archivo es leído por `iptables.service`:

```
# systemctl enable iptables.service
# systemctl start iptables.service
```

Las reglas iptables para ipv6 son, por defecto, almacenadas en el archivo `/etc/iptables/ip6tables.rules`, el cual es leido por `ip6tables.service`. Puede iniciarlo de la misma manera que el anterior.

Después de añadir las reglas a través de línea de órdenes, el archivo de configuración no se cambia automáticamente, tiene que guardarlo de forma manual:

```
# iptables-save > /etc/iptables/iptables.rules
```

Si modifica el archivo de configuración de forma manual, tiene que volver a cargarlo:

```
# systemctl reload iptables
```

Guías

- **Simple stateful firewall**
- **Router**

Registro

El target `LOG` se puede utilizar para registrar los paquetes de red que satisfacen una regla. A diferencia de otros targets, como `ACCEPT` o `DROP`, el paquete de red continuará moviéndose a través de la cadena después de alcanzar un target `LOG`. Esto significa que, para activar el registro de todos los paquetes de red perdidos, se

tendría que agregar una regla `LOG` duplicada antes de cada regla DROP. Como esto reduce la eficiencia y hace que las cosas sean menos simples, se puede crear, en su lugar, una cadena `logdrop` .

Cree la cadena con:

```
# iptables -N logdrop
```

Después, defínala:

```
## /etc/iptables/iptables.rules

*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]

... other user defined chains ..

## logdrop chain
:logdrop - [0:0]

-A logdrop -m limit --limit 5/m --limit-burst 10 -j LOG
-A logdrop -j DROP

... reglas ...

## log AND drop packets that hit this rule:
-A INPUT -m state --state INVALID -j logdrop

... más reglas ...
```

Limitar el tamaño del registro

El módulo *limit* se debe usar para prevenir que el registro de iptables se haga demasiado grande o haga que el disco duro se escriba innecesariamente. Sin limitación, un atacante podría llenar el disco (o al menos la partición `/var`) causando la saturación del registro de iptables.

`-m limit` se utiliza para llamar al módulo limit. Puede usar `--limit` para utilizar una tasa promedio y `--limit-burst` para establecer una tasa de ráfaga (*«limit burst»*) inicial. Por ejemplo:

```
-A LOGDROP -m limit --limit 5/m --limit-burst 10 -j LOG
```

Esto agrega una regla a la cadena `logdrop` que registra todos los paquetes de red que pasan a través de ella. Los primeros 10 paquetes serán registrados, y de ahí en adelante quedarán registrados únicamente 5 paquetes por minuto. El «limit burst» es restaurado a uno cada vez que la «tasa límite» no se supera.

syslog-ng

Asumiendo que usa **syslog-ng**, puede controlar donde será guardada la salida del registro de iptables de este modo:

```
filter f_everything { level(debug..emerg) and not facility(auth, authpriv); };
```

a

```
filter f_everything { level(debug..emerg) and not facility(auth, authpriv) and not filter(f_iptables); };
```

Esto evitará la salida del registro de iptables en el archivo `/var/log/everything.log` .

Si quiere que el registro de iptables se vuelque en un archivo distinto de `/var/log/iptables.log` , basta con cambiar el valor de destino de `d_iptables` (siempre en el archivo `syslog-ng.conf`)

```
destination d_iptables { file("/var/log/iptables.log"); };
```

ulogd

ulogd (http://www.netfilter.org/projects/ulogd/index.html) es un demonio especializado en el registro de los paquetes de red ejecutado en el espacio de usuario para netfilter que puede sustituir el target `LOG` predeterminado. El paquete **`ulogd (https://www.archlinux.org/packages/?name=ulogd)`** está disponible en el repositorio `[community]` .

Véase también

wikipedia:es:iptables

- **Port knocking**
- **Sitio web oficial de iptables (http://www.netfilter.org/projects/iptables/index.html)**
- **Tutorial de iptables 1.2.2 (http://www.frozentux.net/iptables-tutorial/iptables-tutorial.html)** por Oskar Andreasson
- **iptables Debian (http://wiki.debian.org/iptables)** Wiki de Debian

Retrieved from "https://wiki.archlinux.org/index.php?title=Iptables_(Español)&oldid=489338"
