Apéndice A

Guia de referencia

A.1. NocatAuth

En este apéndice nos introduciremos dentro del código y veremos los detalles de implementación del sitema NocatAuth necesarios para la comprensión de este proyecto. En este apéndice no se pretende hacer un recorrido exhaustivo por el código explicando el significado de cada línea. El objetivo de este apéndice es explicar las partes fundamentales del código para poder comprender las modificaciones que se han hecho sobre él. Además hay partes del código que no son necesarias para el funcionamiento básico.

Como ya se ha dicho con anterioridad el código está escrito en su totalidad en Perl y utiliza la programación orientada a objetos de Perl. En Perl un objeto es una referencia a un tipo de datos que sabe a qué clase pertenece. Una clase no es más que un paquete que contiene los métodos correspondientes requeridos para crear y manipular objetos. Un método en Perl es una subrutina definida dentro del paquete, donde el primer argumento del método es una referencia de objeto, o un nombre de paquete, dependiendo de si el método afecta a la clase o al objeto actual.

El sistema NocatAuth se divide en dos partes: una se ejecuta en la pasarela y la otra en el servidor de autenticación.

A.1.1. Pasarela

Lo primero que vamos a ver es el archivo de configuración de la pasarela que tiene la siguiente estructura[1]:

```
###### gateway.conf -- NoCatAuth Gateway Configuration.
```

[1] A continuación una traducción del archivo de configuración.

```
#
# El formato de este archivo es: <Directiva> <Valor>,
#   una por linea. Los tabuladores y espacios en blanco son ignorados.
#   Cualquier linea que comience con un signo de puntuación se considera
#   un comentario.

###### General settings.
#
# Vea el final del archivo para las opciones de registro.
#
# Log verbosity -- 0 Casi ningún registro. 10 se registra
#  todo. 5 intermedio.
#
Verbosity       10

##### Gateway application settings.
#
# GatewayName -- El nombre de la pasarela, será mostrado
#   opcionalmente en la página para mostrar el estado.
#   Vale cualquier cadena de texto corta.
GatewayName the NoCat Network

##
#
# GatewayMode -- Determina el modo de operacion de la pasarela. Los
#   valores posibles son:
#
#   Captive     - Permite la autenticación contra un servidor de
#                   autenticacion.
#   Passive     - Igual que Captive, pero debes usar esto si tu pasarela
#                   está detras de NAT. Funciona tambien si no lo esta.
#                   RECOMENDADO.
#   Open        - Simplemente se muestra al usuario una pagina y se le pide
#                   que acepte las condiciones.
#
# Si esta en modo Captive o Passive, necesitas valores para AuthServiceAddr,
#   AuthServiceURL, y LogoutURL. Seria conveniente dejar un valor bajo en
#   LoginTimeout (probablemente 600).
#
# Si esta en modo Open, necesitas valores para SplashForm, Homepage y
#   posiblemente DocumentRoot (o poner una ruta absoluta para SplashForm).
#   Tambien sera convemiente tener una valor grande para LoginTimeout
#   (probablemente > 3600)
```

```
#
GatewayMode Passive

##
# GatewayLog -- Opcional. Si no se especifica ninguno los mensajes van
#                 a STDERR.
#
GatewayLog /usr/local/nocat/nocat.log

##
# LoginTimeout - Numero de segundos que se esperan desde el ultimo
#   login/renovacion antes de desconectar al usuario. Probablemente
#   no queremos tener esto a valores menores de 60 o se consumira
#   demasiado ancho de banda en renovaciones. Por defecto a 300 seg.
#
# Para modo Captive, poner a un valor relativamente corto (10 minutos)
#   para evitar conexiones de impostores o personas sin autenticar.
#
LoginTimeout 600

# Para portales en modo Open, comenta la linea anterior
#   y pon un valor mayor (por ejemplo 86400, para una notificacion por dia).
#
# LoginTimeout 86400

###### Open Portal settings.
#
##
# HomePage -- Redireccion a ...
#
HomePage http://nocat.net/

# DocumentRoot -- Donde se encuentran las paginas para mostrar.
#   Puede ser difente al DocumentRoot de apache.
#
DocumentRoot /usr/local/nocat/htdocs

# SplashForm -- Formulario mostrado a los usuarios cuando se capturan.
#
SplashForm splash.html

# StatusForm -- Pagina donde se muestra el estado de los usuarios conectados.
#
```

```
StatusForm status.html


###### Active/Passive Portal settings.
#
##
# TrustedGroups - Una lista que contiene los grupos registrados en el
#   servidor de autenticacion que pertenecen a la clase Member.
#   El valor por defecto Any indica que un miembro de cualquier grupo
#   puede acceder en la clase Member.
#
# TrustedGroups NoCat NYCWireless PersonalTelco
#
TrustedGroups Any

##
# Owners - Opcional. Lista con los usuarios que son propietarios de la
#    pasarela, separados por espacios. Acceden con privilegios de la
#    clase Owner.
#
# Owners rob@nocat.net schuyler@nocat.net

##
# AuthServiceAddr - Obligatorio, para modo Captive. Debe ser la direccion
#   del servidor de autenticacion. Debes usar la direccion IP si la
#   la resolucion de nombres no esta disponible al arrancar la pasarela.
#
# AuthServiceAddr 208.201.239.21
#
AuthServiceAddr auth.nocat.net

##
# AuthServiceURL - La direccion del script de login en el servidor de
#   autenticaion.
#
AuthServiceURL  https://$AuthServiceAddr/cgi-bin/login

##
# LogoutURL - Pagina a la que te redirigen cuando haces logout.
#
LogoutURL https://$AuthServiceAddr/logout.html

### Network Topology
```

```
#
# ExternalDevice - Necesario si y solo si NocatAuth no puede averiguarlo
#   mirando las tablas de rutas y cogiendo la interfaz en la ruta por
#   defecto. Debe especificar la interfaz conectada a Internet.
#   Normalmente 'eth0' o 'eth1', o incluso 'ppp0' si tienes una
#   conexion por modem.
#
# ExternalDevice eth0

##
# InternalDevice - Necesario si y solo si tienes dispositivos ethernet en la
#   pasarela entre tu dispositivo wireless y tu conexion de red.
#   Debe especificar la interfaz conectada con la red local, normalmente sera
#   la tarjeta wireless. En Linux, algunos dispositivos wireless se llaman
#   'wvlan0' o 'wlan0' mas que 'ethX'.
#
# InternalDevice eth1

##
# LocalNetwork - Necesario si y solo si NocatAuth no puede averiguar la
#   direccion de tu red local, dada por tu InternalDevice(s). Debe
#   especificar el valor la direccion de la red local y la mascara.
#   Puedes usar el numero de bits en la mascar(p.e. /16, /24, etc.)
#   o la mascara completa x.x.x.x.
#
# LocalNetwork 10.0.1.0/24

##
# DNSAddr - Opcional. Si tu eliges no ejecutar un servidor de nombres DNS
#   en tu red interna, debes proporcionar la direccion de uno o mas en
#   Internet. Debe ser el mismo DNS que use tu servidor DHCP. Si lo dejas
#   en blanco, NoCatAuth tomara los nombres de /etc/resolv.conf.
#
# DNSAddr 111.222.333.444

##
# AllowedWebHosts - Opcional. Lista cualquier direccion que quieras acceso
#   libre.
#
# AllowedWebHosts nocat.net

##
# RouteOnly - Necesario solo si tu NO QUIERES que tu pasarela actue como NAT.
```

```
#   Descomenta esto solo si tu estas ejecutando una red que solo necesita
#   enrutamiento y no necesita que la pasarela haga de NAT.
#
# RouteOnly 1

##
# IgnoreMAC - Activa esto solo y solo si la pasarela no esta conectada
#   directamente (o con un puente de nivel 2) a tu red interna (normalmente
#   wireless). En ese caso, la pasarela no sera capaz de conocer la MAC de
#   los clientes, y sera forzada a usar la direccion IP solo. Teoricamente
#   esto es menos seguro, ya que las direcciones IP son mas faciles de
#   spoof que las direcciones MAC. No uses esto si no sabes lo que haces.
#
# IgnoreMAC 1

##
# MembersOnly - Opcional. Descomenta esto si quieres deshabilitar el
#   acceso publico(i.e. deshabilitar el boton 'skip').
#
# MembersOnly 1

##
# IncludePorts - Opcional. Especifica los puertos TCP a los que se
#   el acceso para la clase public. El acceso al resto de puertos
#   esta denegado.
#
#   Para una lista de servicios y puertos, mira en /etc/services.
#   Dependiendo de tu firewall, tu podrias especificar servicios
#   en vez de puertos.
#
# IncludePorts    22 80 443

##
# ExcludePorts - Opcional. Especifica los puertos a los que la clase
#   public no puede acceder. El resto estaran permitidos.
#
#   Solo puedes usar IncludePorts o ExcludePorts, pero no los dos a la
#   vez. Si ninguno es especificado todos los puertos son accesibles para
#   la clase public.
#
# ExcludePorts 23 25 111
#
ExcludePorts    25
```

```
####### Syslog Options -- Cambia esto solo si tu quieres que Nocat use
#        el registro del sistema!
#
# Log Facility - syslog o internal. Internal envia todos los mensajes al
#    archivo GatewayLog o a STDERR si no se especifica ningun archivo.
#    Syslog manda los mensajes al registro del sistema
#
# LogFacility internal

##
# SyslogSocket - inet or unix.  Inet connects to an inet socket returned
#    by getsrvbyname().  Unix connects to a unix domain socket returned by
#    _PATH_LOG in syslog.ph (typically /dev/log).  Defaults to unix.
#
# SyslogSocket unix

##
# SyslogOptions - Ninguna o mas palabras de pid, ndelay, cons, nowait
#    Por defecto "cons,pid"
#
# SyslogOptions cons,pid

##
# SyslogPriority - La prioridad de mensajes a usar. En importancia
#    decreciente las mas tipicas son: EMERG, ALERT, CRIT, ERR, WARNING,
#    NOTICE, INFO, and DEBUG. Por defecto a INFO.
#
# SyslogPriority INFO

##
# SyslogFacility - La facilidad para el registro.  Por defecto a user.
# SyslogFacility user

##
# SyslogIdent - El identificador del programa que llama a syslog.
#    Aparece en todas las entradas del registro hechas desde Nocat.
#    Por defecto Nocat.
#
# SyslogIdent NoCat

###### Other Common Gateway Options. (Probablemente no tenga que cambiarlas)
#
```

```
# ResetCmd, PermitCmd, DenyCmd -- Shell-scripts para resetear, abrir y
#   cerrar el firewall
#
# ResetCmd initialize.fw
# PermitCmd access.fw permit $MAC $IP $Class
# DenyCmd access.fw deny $MAC $IP $Class

##
# GatewayPort - El puerto TCP donde la pasarela recibe petieciones.
#   De hecho es el puerto estandar para NoCatAuth.
#   Cambie esto solo si es absolutamente necesario.
#
# GatewayPort     5280

##
# PGPKeyPath -- El directorio donde se guardan las claves PGP. Nocat intenta
#   encontrar esto en el directorio pgp/.
#
# PGPKeyPath /usr/local/nocat/pgp

##
# MessageVerify -- Comando para verificar la firma PGP de un mensaje.
#
# GpgvPath /usr/bin/gpgv
#
# MessageVerify $GpgvPath --homedir=$PGPKeyPath 2>/dev/null

##
#
# IdleTimeout -- Frecuencia para comprobar la cache ARP, en segundos,
#   para la expiracion de idle clientes.
#
# MaxMissedARP -- Cuantas veces puede un cliente no estar en la ARP
#   cache antes de que asumamos que se ha ido, y lo desactivemos.
#   Pon el valor 0 para desabilitar le dewconexion basada en la
#   expiracion de la cache ARP.
#
# MaxMissedARP 2
#
# IdleTimeout   300

### Fin!
```

Ya hemos visto todas las directivas. A partir de ahora nos centraremos en el modo Passive, al que llamaremos modo pasivo, que es el recomendado y el que se ejecuta si la pasarela está detrás de un NAT. Asimismo como firewall usaremos las iptables, esto quiere decir que solo se verán los ejecutables referidos a iptables.El contenido de /libexec/ipfilter/, /libexec/ipchains/, /libexec/pf/, y /libexec/loopback/ no se verán.

Pasaremos a ver el funcionamiento del programa, el método a seguir será secuencial. Comenzaremos viendo que pasa cuando se inicia la pasarela y veremos el proceso que se sigue cada vez que se conecta un usuario. Para empezar diremos que en la pasarela lo que se ejecuta es un proceso que se demoniza y atiende peticiones en el puerto 5280.

Inicialización de la pasarela

Para iniciar la pasarela se ejecuta el archivo bin/gateway. En la figura A.1 se muestra el comportamiento de este archivo[2].

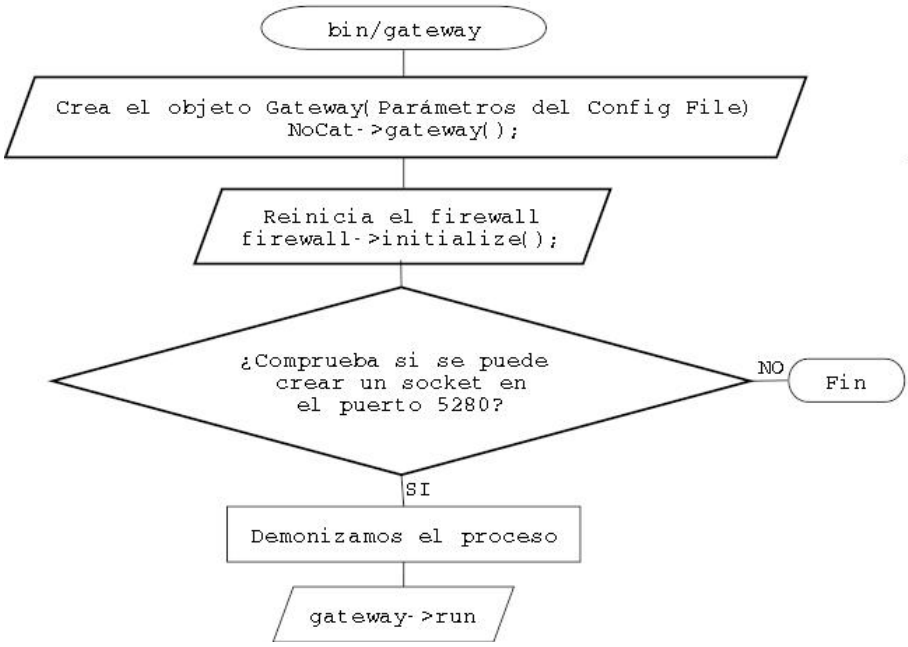

Figura A.1: bin/gateway

Podemos ver que lo que hace es iniciar las reglas del firewall y después demonizar el proceso y ejecutar el método run que es el que se está ejecutando continuamente. Las reglas que se crean en el firewall son las siguientes:

```
# Generated by iptables-save v1.2.8 on Fri Aug 22 15:14:37 2003
```

[2]En la figura 3.1 se define la notación que se usará en los diagramas de flujo

```
*mangle
:PREROUTING ACCEPT [2813633:2352593165]
:INPUT ACCEPT [604797:294873430]
:FORWARD ACCEPT [2207569:2057553106]
:OUTPUT ACCEPT [491984:255531624]
:POSTROUTING ACCEPT [2697828:2312652436]
:NoCat - [0:0]
-A PREROUTING -j NoCat
-A OUTPUT -p tcp -m tcp --dport 22 -j TOS --set-tos 0x10
-A OUTPUT -p tcp -m tcp --dport 80 -j TOS --set-tos 0x08
-A OUTPUT -p tcp -m tcp --dport 443 -j TOS --set-tos 0x08
-A NoCat -i eth1 -j MARK --set-mark 0x4
COMMIT
```

Esta es la tabla mangle y vemos los cinco ganchos[3], en el PREROUTING tenemos -j*(jump)* NoCat y salta a la cadena NoCat que tenmos más abajo. En la cadena NoCat establece que los paquetes de la interfaz eth1 (interfaz local en este caso) sean marcados con la merca 0x4. Además se marcan más paquetes a la salida de la interfaz. Estas marcas (TOS - Type of Service) se utilizan para marcar los paquetes según si es tráfico masivo, o tráfico en tiempo real, etc... En algunos routers de la red se da cierta prioridad a los paquetes marcados.

```
# Completed on Fri Aug 22 15:14:37 2003
# Generated by iptables-save v1.2.8 on Fri Aug 22 15:14:37 2003
*nat
:PREROUTING ACCEPT [40799:4575484]
:POSTROUTING ACCEPT [4085:250171]
:OUTPUT ACCEPT [4083:250059]
:NoCat_Capture - [0:0]
:NoCat_NAT - [0:0]
-A PREROUTING -j NoCat_Capture
-A POSTROUTING -j NoCat_NAT
-A NoCat_Capture -s 192.168.1.0/255.255.255.0 -d 192.168.1.32 -p tcp -m
tcp --dport 80 -j RETURN
-A NoCat_Capture -s 192.168.1.0/255.255.255.0 -d 192.168.1.32 -p tcp -m
tcp --dport 443 -j RETURN
-A NoCat_Capture -p tcp -m mark --mark 0x4 -m tcp --dport 80 -j REDIRECT
--to-ports 5280
-A NoCat_Capture -p tcp -m mark --mark 0x4 -m tcp --dport 443 -j REDIRECT
--to-ports 5280
-A NoCat_NAT -s 192.168.1.0/255.255.255.0 -o eth0 -m mark --mark 0x1 -j
```

[3]Para descripción de las iptables vea el apéndice A.3.

```
MASQUERADE
-A NoCat_NAT -s 192.168.1.0/255.255.255.0 -o eth0 -m mark --mark 0x2 -j
MASQUERADE
-A NoCat_NAT -s 192.168.1.0/255.255.255.0 -o eth0 -m mark --mark 0x3 -j
MASQUERADE
-A NoCat_NAT -s 192.168.1.0/255.255.255.0 -d 192.168.1.32 -p tcp -m tcp
--dport 80 -j MASQUERADE
-A NoCat_NAT -s 192.168.1.0/255.255.255.0 -d 192.168.1.32 -p tcp -m tcp
--dport 443 -j MASQUERADE
-A NoCat_NAT -s 192.168.1.0/255.255.255.0 -d 192.168.0.254 -p tcp -m tcp
--dport 53 -j MASQUERADE
-A NoCat_NAT -s 192.168.1.0/255.255.255.0 -d 192.168.0.254 -p udp -m udp
--dport 53 -j MASQUERADE
COMMIT
```

Esta es la tabla NAT, que servía para el enmascaramiento y la alteración de paquetes en general. Se definen dos cadenas: la cadena NoCat_Capture en el gancho de PREROUTING, y la cadena NoCat_NAT en el gancho de POSTROUTING.

Las dos primeras filas de NoCat_Capture sirven para dejar pasar sin alteración los paquetes que vengan de la subred local y vayan dirigidos al serividor de autenticación (192.168.1.32), al puerto 80 (http) o al puerto 443 (https). Las dos siguientes líneas son para redirigir al puerto 5280 cualquier paquete que tenga la marca 0x4 y vaya dirigido al puerto 80 o 443, es decir, redirige todos los paquetes que vienen de la subred local (los hemos marcado antes en la tabla mangle), que han realizado una petición web, al puerto 5280 de la pasarela.

A la salida de la interfaz nos encontramos con la cadena NoCat_NAT. Hay que tener en cuenta que un paquete que sale de la red local sin enmascaramiento, tendrá en la dirección origen una dirección privada, lo que quiere decir que cuando el destinatario del paquete intente devolver información al origen no podrá, al no disponer de una dirección adecuada. Teniendo en cuenta esto cualquier paquete que sale de la red local debe ser enmarcarado, y de esto se encarga el módulo netfilter, mediante la tabla NAT y el control de estado. El control de estado mantiene la información necesaria para saber cuando llega un paquete de la red externa quién fue el que solicitó dicha información. Una vez visto el funcinamiento veremos que hace la cdena NoCat_NAT:

- Enmascara los paquetes con las marcas 0x1, 0x2 y 0x3. Por ahora hemos visto que en la tabla mangle no se usaban estas marcas.
- Las dos siguientes líneas son para enmascarar las peticiones web que van hacia el servidor de autenticación.
- Las dos últimas líneas enmascarán las peticiones dirigidas al servidor de nombres.

```
# Completed on Fri Aug 22 15:14:37 2003
# Generated by iptables-save v1.2.8 on Fri Aug 22 15:14:37 2003
*filter
:INPUT ACCEPT [27040:29368878]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [24972:119077725]
:NoCat - [0:0]
:NoCat_Inbound - [0:0]
:NoCat_Ports - [0:0]
-A FORWARD -j NoCat
-A NoCat -j NoCat_Ports
-A NoCat -j NoCat_Inbound
-A NoCat -s 192.168.1.0/255.255.255.0 -i eth1 -m mark --mark 0x1 -j ACCEPT
-A NoCat -s 192.168.1.0/255.255.255.0 -i eth1 -m mark --mark 0x2 -j ACCEPT
-A NoCat -s 192.168.1.0/255.255.255.0 -i eth1 -m mark --mark 0x3 -j ACCEPT
-A NoCat -s 192.168.1.0/255.255.255.0 -d 192.168.1.32 -p tcp -m tcp --dport
80 -j ACCEPT
-A NoCat -s 192.168.1.32 -d 192.168.1.0/255.255.255.0 -p tcp -m tcp --sport
80 -j ACCEPT
-A NoCat -s 192.168.1.0/255.255.255.0 -d 192.168.1.32 -p tcp -m tcp --dport
443 -j ACCEPT
-A NoCat -s 192.168.1.32 -d 192.168.1.0/255.255.255.0 -p tcp -m tcp --sport
443 -j ACCEPT
-A NoCat -s 192.168.0.254 -d 192.168.1.0/255.255.255.0 -o eth1 -j ACCEPT
-A NoCat -s 192.168.1.0/255.255.255.0 -d 192.168.0.254 -i eth1 -p tcp -m tcp
--dport 53 -j ACCEPT
-A NoCat -s 192.168.1.0/255.255.255.0 -d 192.168.0.254 -i eth1 -p udp -m udp
--dport 53 -j ACCEPT
-A NoCat -s ! 192.168.1.32 -i eth0 -p tcp -m tcp --dport 5280 -j DROP
-A NoCat -j DROP
-A NoCat_Ports -i eth1 -p tcp -m tcp --dport 5280 -j ACCEPT
-A NoCat_Ports -i eth1 -p udp -m udp --dport 5280 -j ACCEPT
OMMIT
# Completed on Fri Aug 22 15:14:37 2003
```

Por último esta era la tabla filter, donde se aplica el filtrado de paquetes, el camino que sigue un paquete es secuencial: si tiene correspondencia con la regla entonces se desecha o se acepta según marque la regla; en el caso de que no coincida pasa a ver la regla siguiente, si no coincide con ninguna regla se usa la política por defecto. Vemos que el filtrado se aplica en el gancho FORWARD, es decir a los paquetes que deben ser enrutados hacia algún destino. Lo primero que hace un paquete es saltar a la cadena NoCat y a partir de ahí el orden que sigue es el siguiente:

1. Salta a Nocat_Ports:

   a) Si viene de la interfaz interna, con protocolo tcp y va al puerto 5280 -> Acepta.

   b) Si viene de la interfaz interna, con protocolo udp y va al puerto 5280 -> Acepta.

2. Salta a Nocat_Inbound. No hay ninguna regla definida aún.

3. Si viene de la red local interna y esta marcado con 0x1 -> Acepta.

4. Si viene de la red local interna y esta marcado con 0x2 -> Acepta.

5. Si viene de la red local interna y esta marcado con 0x3 -> Acepta.

6. Si viene de la red local interna y va hacia el servidor de autenticación(puerto 80) -> Acepta.

7. Si viene del servidor de autenticación(puerto 80) y va hacia la subred local-> Acepta.

8. Si viene de la red local interna y va hacia el servidor de autenticación(puerto 443) -> Acepta.

9. Si viene del servidor de autenticación(puerto 443) y va hacia la subred local-> Acepta.

10. Si viene del servidor de nombres y va hacia la subred local-> Acepta.

11. Si viene de la red local interna y va hacia el servidor de nombres(puerto 53,protocolo tcp) -> Acepta.

12. Si viene de la red local interna y va hacia el servidor de nombres(puerto 53,protocolo udp) -> Acepta.

13. Si el origen no es el servidor de autenticación, viene de la interfaz externa y va al puerto 5280 -> Rechaza

14. Por último las que no hayan coincidido con nada -> Rechaza.

Ya hemos visto que cualquier paquete que entre por la interfaz interna de la pasarela será redirigido al puerto 5280., más adelante veremos como se atienden esas peticiones.

Funcionamiento de la pasarela

En el método run (figura A.2) hay un bucle infinito que va atendiendo peticiones y realizando algunas comprobaciones. Para atender las peticiones crea una pila donde va guardando los sockets que se conectan al puerto 5280. Dentro de poll_socket va creando procesos hijos que son los que atienden realmente las peticiones de los clientes. El método que utiliza para atender a los clientes es accept_client (figura A.3).

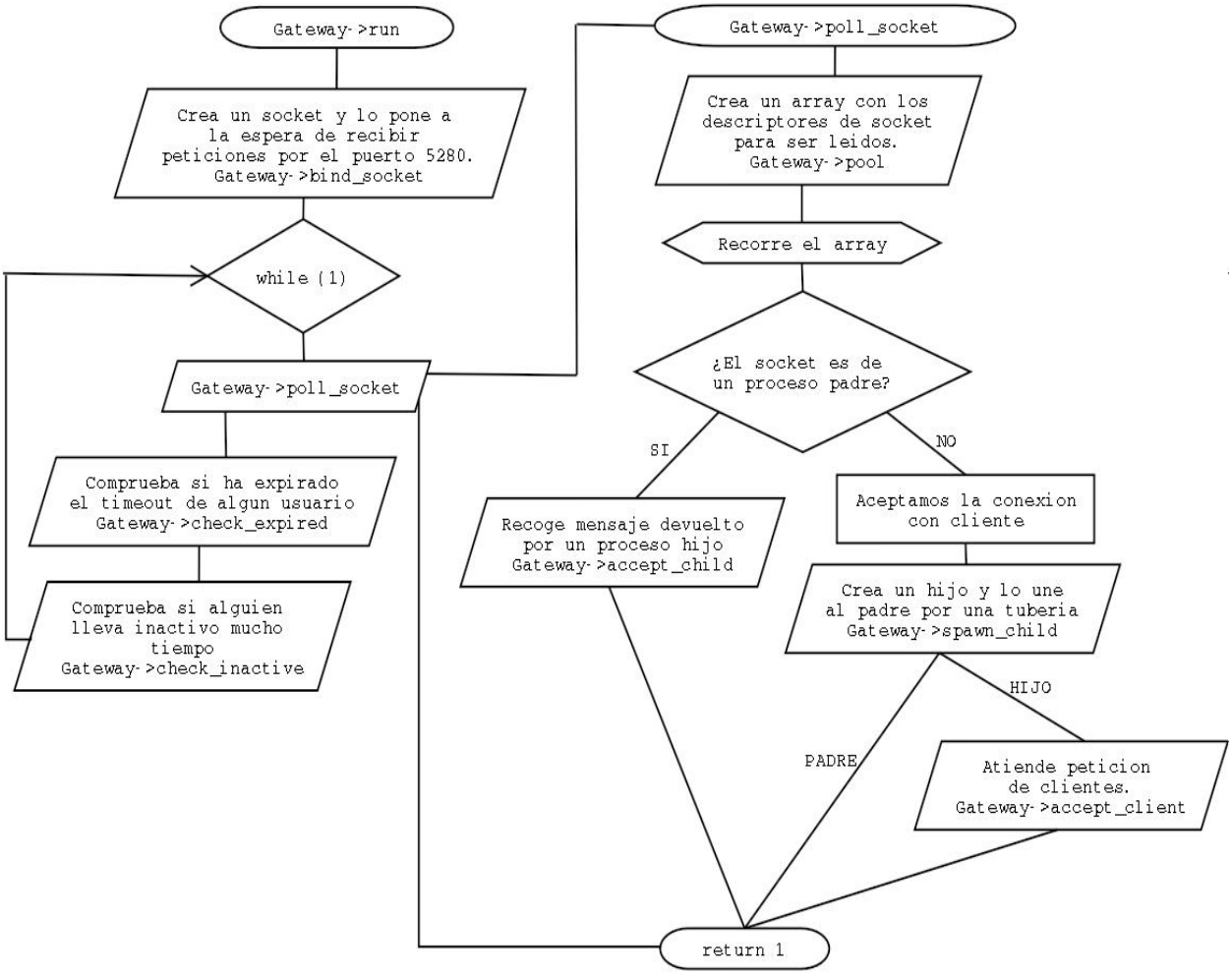

Figura A.2: Gateway-run

En el método accept_client de la clase Gateway se crea un objeto de la clase Peer que guarda los datos del cliente e inmediatamente llama al método handle de la clase Captive. La clase Captive hereda de la clase Gateway.

Es el método handle de la clase Captive el que realmente maneja las conexiones. Primero lee la petición que se recibe en la línea de dirección del explorador, comprueba

si va dirigida a él o es una conexión nueva de un cliente que no está autenticado. Cuando un usuario no está autenticado y hace una petición a una web, ésta es redirigida al puerto 5280. Lo que hace en este caso la pasarela es capturar la petición y enviarla al servidor de autenticación. Por el contrario si la petición va dirigida a la pasarela puede ser por tres razones. Es una petición para mostrar el estado de la pasarela, en ese caso se muestra la página status; puede ser una petición de logout, en tal caso se desconecta al cliente del sistema llamando a Gateway->deny(figura A.6); y la tercera opción es que el usuario esté en el proceso de autenticación. Si es así la pasarela ya ha capturado la petición previamente, ha redirigido al usuario al servidor de autenticación y el usuario ha recibido un mensaje cifrado y firmado del servidor de autenticación que es el que en este momento recibe la pasarela para analizar.

En primer lugar vamos a ver cómo se capturan las peticiones y despues veremos cómo se verifican los mensajes y se actúa en consecuencia. Para capturar las peticiones se guardan los parámetros del usuario: MAC, token, la dirección que ha solicitado, LoginTimeout y la dirección de la pasarela. Todos estos parámetros se ponen junto con la dirección del servidor de autenticación y se redirige al cliente a esta dirección. De forma que el servidor de autenticación recibe una petición del cliente directamente donde está contenida la información de la pasarela. Para ver cómo actúa el servidor de autenticación vea la sección A.1.2. Por ahora nos basta saber que devuelve un mensaje cifrado y firmado al cliente para ser enviado a la pasarela en cinco segundos.

Una vez que ha recibido este mensaje podemos ver cómo actúa el método handle cuando lo recibe. Cuando recibe este mensaje ve que va dirigido hacia él y no es ni una petición del estado ni una petición de logout. Entonces, para analizar este mensaje se llama al método verify de la clase Passive(figura A.4).

En el método verify se descodifica el mensaje y se comprueba que el token que trae es el mismo token que nosotros habíamos generado para ese usuario. Se guardan el usuario, su MAC y el grupo al que pertenece y dependiendo de la acción que diga el servidor se llama al método permit(figura A.5) de la clase Gateway, o a método deny(figura A.6).

Suponiendo que todo el proceso ha sido correcto el usuario debe estar autenticado y en el método permit se llama al método permit de la clase Firewall que es el que actúa sobre las reglas del cortafuegos. Para esto llama al script access.fw que mediante dos líneas como estas:

iptables -t mangle -A NoCat -m –mac-source mac -s ipaddress -j MARK –set-mark marca

iptables -t filter -A NoCat_Inbound -d ipaddress -j ACCEPT

donde mac, ipaddress y marca son la dirección MAC del cliente, la dirección ip del cliente y una marca, que será 0x1, 0x2 o 0x3 dependiendo de la clase a la que pertenezca el usuario. Como vimos al iniciar el firewall hay una reglas para que los paquetes con las

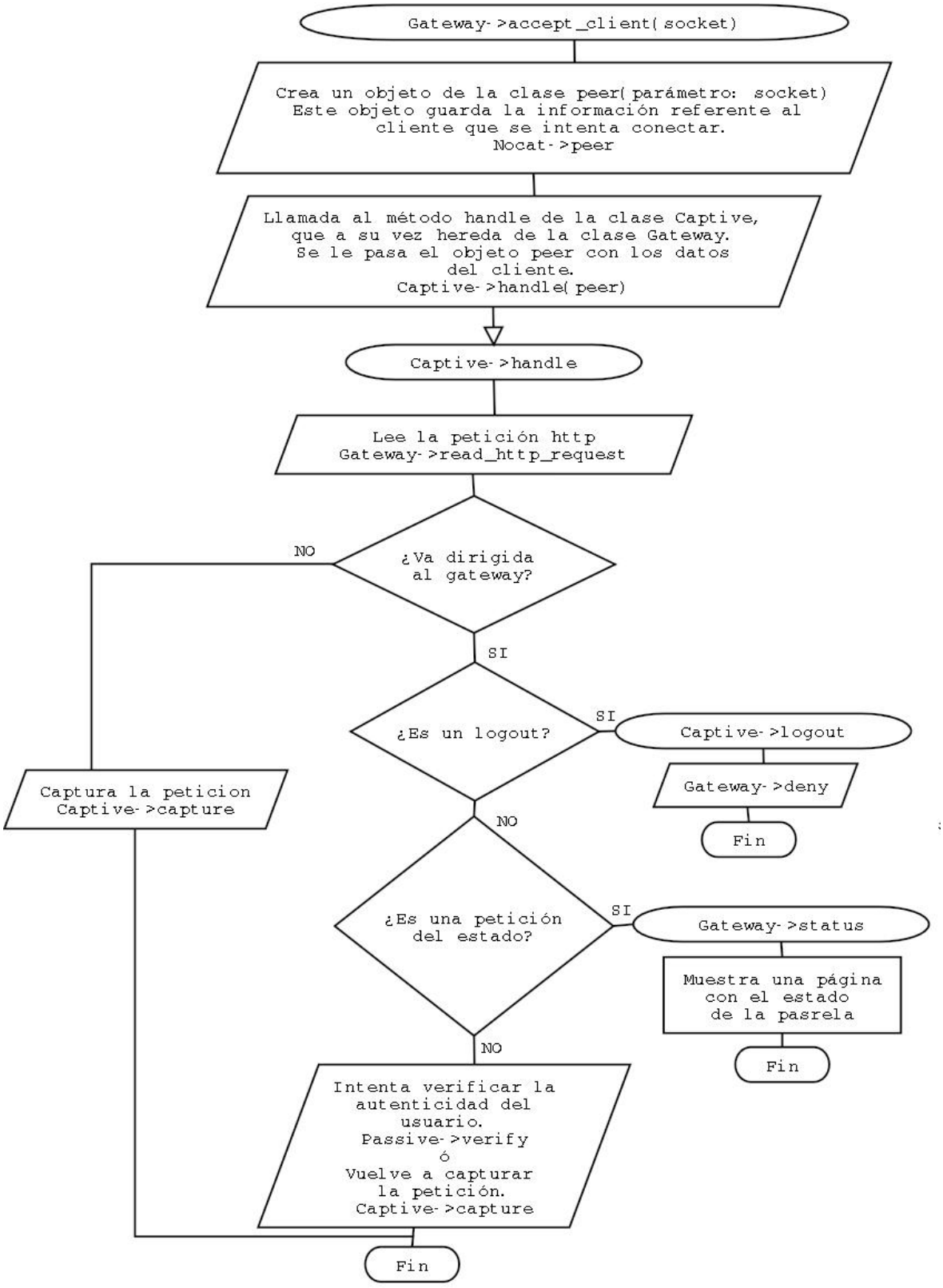

Figura A.3: Gateway-accept_client

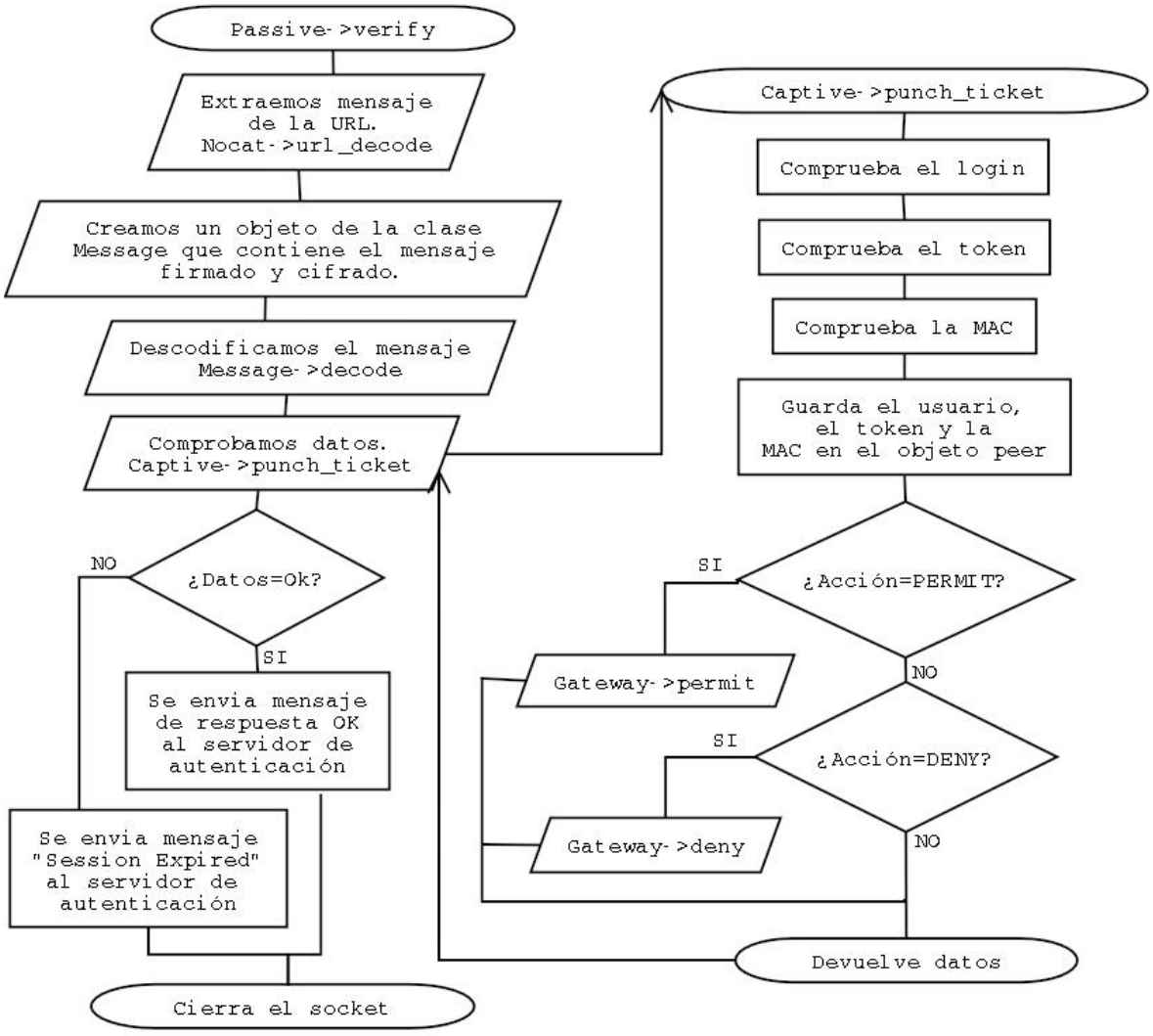

Figura A.4: Passive-verify

marcas 0x1, 0x2 o 0x3 sean aceptados. El resultado de las iptables cuando un usuario con dirección IP igual a 192.168.1.13 y MAC 00:05:1C:0F:B4:07 está autenticado es el siguiente:

```
# Generated by iptables-save v1.2.8 on Fri Aug 22 15:25:06 2003
*mangle
:PREROUTING ACCEPT [2813929:2352625211]
:INPUT ACCEPT [605055:294899204]
:FORWARD ACCEPT [2207607:2057559378]
:OUTPUT ACCEPT [492163:255556825]
:POSTROUTING ACCEPT [2698025:2312682229]
:NoCat - [0:0]
-A PREROUTING -j NoCat
-A OUTPUT -p tcp -m tcp --dport 22 -j TOS --set-tos 0x10
-A OUTPUT -p tcp -m tcp --dport 80 -j TOS --set-tos 0x08
-A OUTPUT -p tcp -m tcp --dport 443 -j TOS --set-tos 0x08
-A OUTPUT -p tcp -m tcp --dport 22 -j TOS --set-tos 0x10
-A OUTPUT -p tcp -m tcp --dport 80 -j TOS --set-tos 0x08
-A OUTPUT -p tcp -m tcp --dport 443 -j TOS --set-tos 0x08
-A NoCat -i eth1 -j MARK --set-mark 0x4
-A NoCat -s 192.168.1.13 -m mac --mac-source 00:05:1C:0F:B4:07 -j
MARK --set-mark 0x2
COMMIT
# Completed on Fri Aug 22 15:25:06 2003
```

La nueva línea que vemos es la última de la cadena NoCat y marca los paquetes con origen la dirección IP y la dirección MAC del cliente que se conecta. A partir de ahora todos los paquetes de ese usuario serán marcados con la marca 0x2, y vimos anteriormente que al iniciar la pasarela había un regla en la tabla NAT para enmascarar los paquetes con la marca 0x2 y una regla en la tabla filter para dejarlos pasar.

```
# Completed on Fri Aug 22 15:25:06 2003
# Generated by iptables-save v1.2.8 on Fri Aug 22 15:25:06 2003
*filter
:INPUT ACCEPT [27298:29394652]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [25151:119102926]
:NoCat - [0:0]
:NoCat_Inbound - [0:0]
:NoCat_Ports - [0:0]
-A FORWARD -j NoCat
-A NoCat -j NoCat_Ports
-A NoCat -j NoCat_Inbound
```

```
-A NoCat -s 192.168.1.0/255.255.255.0 -i eth1 -m mark --mark 0x1 -j ACCEPT
-A NoCat -s 192.168.1.0/255.255.255.0 -i eth1 -m mark --mark 0x2 -j ACCEPT
-A NoCat -s 192.168.1.0/255.255.255.0 -i eth1 -m mark --mark 0x3 -j ACCEPT
-A NoCat -s 192.168.1.0/255.255.255.0 -d 192.168.1.32 -p tcp -m tcp
   --dport 80 -j ACCEPT
-A NoCat -s 192.168.1.32 -d 192.168.1.0/255.255.255.0 -p tcp -m tcp
   --sport 80 -j ACCEPT
-A NoCat -s 192.168.1.0/255.255.255.0 -d 192.168.1.32 -p tcp -m tcp
   --dport 443 -j ACCEPT
-A NoCat -s 192.168.1.32 -d 192.168.1.0/255.255.255.0 -p tcp -m tcp
   --sport 443 -j ACCEPT
-A NoCat -s 192.168.0.254 -d 192.168.1.0/255.255.255.0 -o eth1 -j ACCEPT
-A NoCat -s 192.168.1.0/255.255.255.0 -d 192.168.0.254 -i eth1 -p tcp
   -m tcp --dport 53 -j ACCEPT
-A NoCat -s 192.168.1.0/255.255.255.0 -d 192.168.0.254 -i eth1 -p udp
   -m udp --dport 53 -j ACCEPT
-A NoCat -s ! 192.168.1.32 -i eth0 -p tcp -m tcp --dport 5280 -j DROP
-A NoCat -j DROP
-A NoCat_Inbound -d 192.168.1.13 -j ACCEPT
-A NoCat_Ports -i eth1 -p tcp -m tcp --dport 5280 -j ACCEPT
-A NoCat_Ports -i eth1 -p udp -m udp --dport 5280 -j ACCEPT
COMMIT
# Completed on Fri Aug 22 15:25:06 2003
```

La única línea nueva corresponde a la cadena NoCat_Inbound y establece que se acepten todos los paquetes que vayan dirigidos hacia el cliente.

En el caso de que se llame al método deny. Esto puede ocurrir por varias razones: el usuario pulse logout, expire el tiempo marcado en la directiva Logintimeout sin recibir una autenticación, o que el servidor mande en el mensaje que no está autenticado. En cualquiera de estos casos se llama al método deny de la clase Gateway, que realiza la operación inversa al método permit. Es decir llama al método deny de la clase firewall que vuelve a llamar al script access.fw, pero esta vez diciendo que debe borrar las reglas insertadas antes. De tal manera que los paquetes de dicho usuario dejan de estar marcados con una marca de privilegio, y se vuelven a marcar con 0x4. Entonces la próxima petición que haga ese usuario volverá a ser redirigdo al puerto 5280 de la pasarela y vuelve a empezar el proceso.

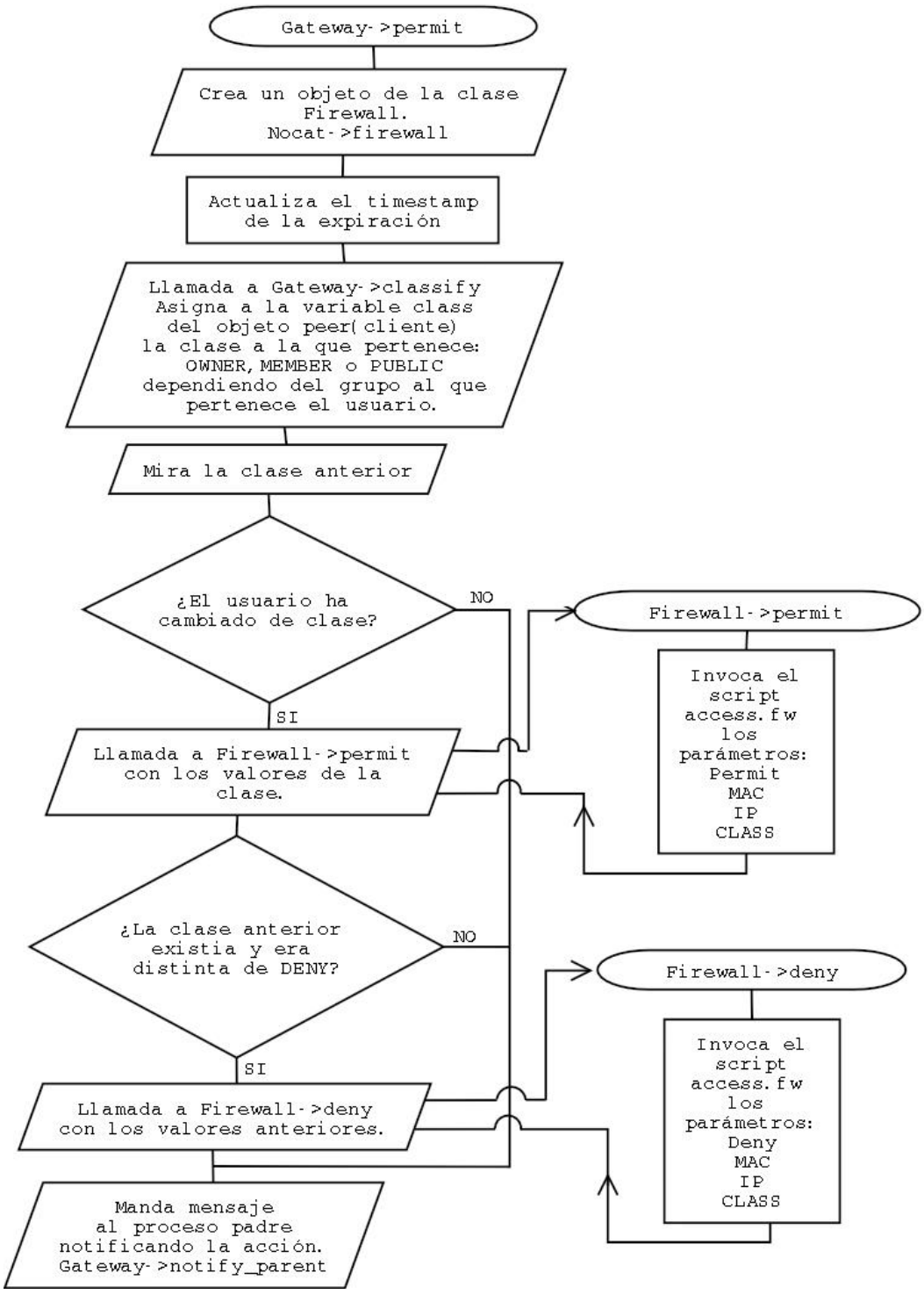

Figura A.5: Gateway-permit

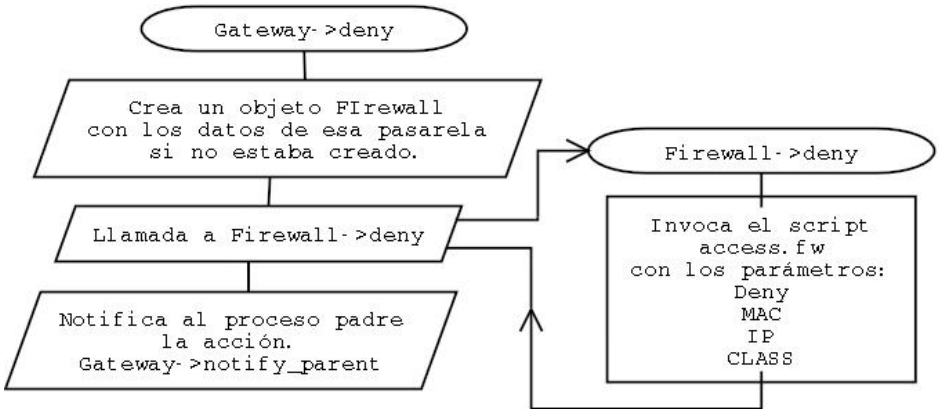

Figura A.6: Gateway-deny

A.1.2. Servidor de autenticación

Para empezar veremos el archivo de configuración del servidor. A continuación se presenta una traducción del archivo:

```
###### authserv.conf -- NoCatAuth Authentication Service Configuration.
#
# Formato de este archivo: <Directiva> <Valor>, uno por
#   linea. Tabuladores y espacios iniciales son ignorados. Cualquier
#   línea que comience por un signo de puntuación se considera un comentario.

###### Parámetros generales.
#
# Log verbosity -- 0 casi no se registra. 10 se registra
#   todo. 5 es un término medio.
#
Verbosity       10

##
# PGPKeyPath -- El directorio donde se guardan las claves PGP.
#   Nocat la intenta buscar en el directorio pgp/ en nocat/.
#   Cambia esto solo si has puesto las claves en otro lugar.
#
PGPKeyPath /usr/local/nocat/pgp


###### Authservice - Parámetros específicos.
#
# HomePage -- Redirección por defecto
#
HomePage http://nocat.net/

# DocumentRoot -- Donde se guardan todas las plantillas (incluyendo
#   SplashPage). Puede ser distinto a la directiva DocumentRoot
#   de apache.
#
DocumentRoot /usr/local/nocat/htdocs

##### Authservice - Fuente de autenticación.
#
# DataSource -- Específica el modo de autenticación
#   Posibles valores son DBI, Passwd, LDAP, RADIUS, PAM, Samba, IMAP, NIS.
#
```

```
DataSource DBI

##
# Auth service base de datos.
#
# Si has seleccionado DataSource DBI, entonces debes rellenar Database,
#   DB_User, y DB_Password
#
# Database tiene formato para especificar driver en DBI.
#
# Para postgre:
# Database dbi:Pg:dbname=nocat
#
# Para mysql:
Database dbi:mysql:database=nocat
DB_User nocat
DB_Passwd 123456


## LDAP . Requiere Net::LDAP & IO::Socket::SSL instalados del modulo CPAN.
#

.......................
```

Aquí están definidas las directivas para todos los tipos de autenticación que soporta el servidor. Nuestro trabajo está basado en MySQL y por tanto no se ha considerado de interés el incluir esas directivas.

```
###### Auth service Tablas de usuario.
#
# UserTable Nombre de la tabla que contiene el ID de usuario.
#
# UserIDField Nombre de la columna que contiene el ID del cliente
#    que lo identifica univocamente.
#
# UserPasswdField Guarda la clave encriptada con MD5.
#
# UserAuthField esta obsoleto.
#
UserTable member
UserIDField login
UserPasswdField pass
```

```
UserAuthField   status
UserStampField created

GroupTable      network
GroupIDField    network
GroupAdminField admin

####### Auth service Configuración Web.
#
# MinPasswdLength -- Longitud minima de la clave.
#   No se comprueba mucho mas del par usuario/clave.
#
MinPasswdLength 6

# MessageSign -- Comando del shell para firmar los mensajes del servidor.
#   El mensaje se pasa por la entrada estandar, mientras que la salida
#   se lee por la salida estandar
#
# GpgPath /usr/bin/gpg
#
# MessageSign $GpgPath --clearsign --homedir=$PGPKeyPath -o-

# LocalGateway -- Si ejecutas el servidor de autenticación en la misma subred
#   que tienes la pasarela necesitas especificar el nombre de la pasarela.
#   En otro caso omitelo.(Se requiere el modulo Net::Netmask)
#
#LocalGateway    192.168.1.10

# Auth service Nombre de las plantillas. Mira las plantillas individualmente
#   para ver con detalle lo que hace cada una.
#
LoginForm login.html
LoginOKForm login_ok.html
FatalForm fatal.html
ExpiredForm expired.html
RenewForm renew.html
PassiveRenewForm renew_pasv.html

RegisterForm register.html
RegisterOKForm register_ok.html
RegisterFields name url description

UpdateForm update.html
```

```
UpdateFields url description

###### Auth service Mensajes de usuario.
#
LoginGreeting   Necesita permitir el uso de ventanas emergentes y javascript
                  en este sitio para continuar.
LoginMissing    Tienes que rellenar todos los campos.
LoginBadUser    El usuario y/o la contraseña no son correctos.
LoginBadPass El usuario y/o la contraseña no son correctos.
LoginBadStatus  Usted no esta registrado.

RegisterGreeting    Welcome! Please enter the following information
                    to register.
RegisterMissing     Name, E-mail, and password fields must be filled in.
RegisterUserExists  Sorry, that e-mail address is already taken. Are you
                    already registered?
RegisterBadUser     The e-mail address provided appears to be invalid. Did
                    you spell it correctly?
RegisterInvalidPass All passwords must be at least six characters long.
RegisterPassNoMatch The passwords you provided do not match. Please try again.
RegisterSuccess     Congratulations, you have successfully registered.

UpdateGreeting     Enter your E-mail and password to update your info.
UpdateBadUser     That e-mail address is unknown. Please try again.
UpdateBadPass     That e-mail and password do not match. Please try again.
UpdateInvalidPass   New passwords must be at least six characters long.
UpdatePassNoMatch   The new passwords you provided do not match. Please
                    try again.
UpdateSuccess     Congratulations, you have successfully updated your account.

LocalNetwork 192.168.1.0/24

###### Fin.
```

Funcionamiento del servidor

El servidor de autenticación recibe peticiones web que atiende un script cgi que se ejecuta en el servidor y está escrito en Perl. También utiliza la programación orientada a objetos de Perl pero tiene un funcionamiento mucho más sencillo que la pasarela, desde el script de login se atienden todas las peticiones y no se necesita saltar de un método a otro. En la figuras A.7 A.8 y A.9 tenemos el diagrama de flujo de este script.

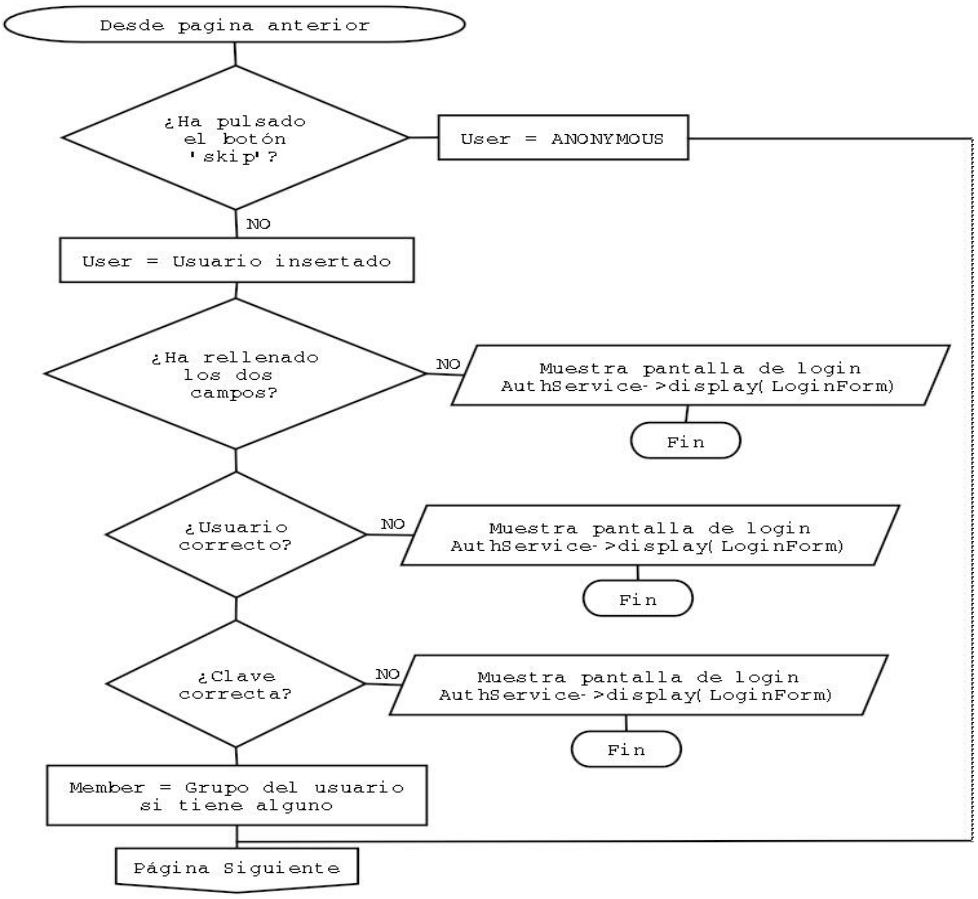

Figura A.7: /cgi-bin/login

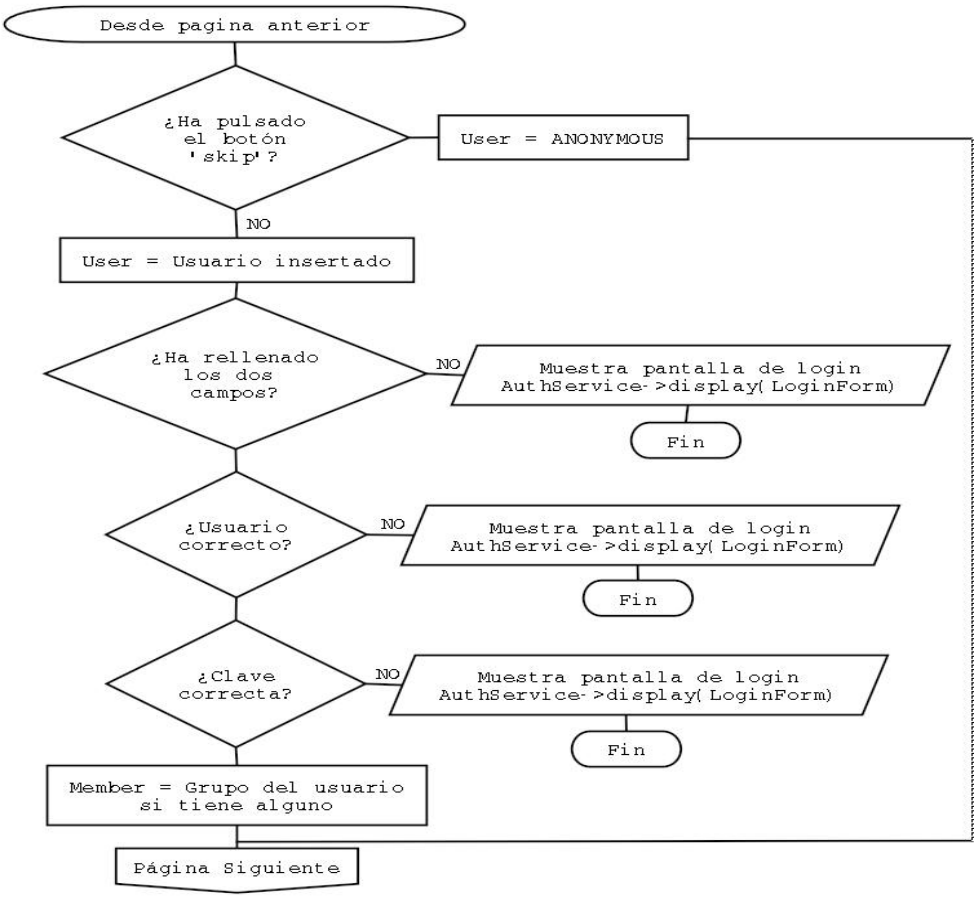

Figura A.8: /cgi-bin/login

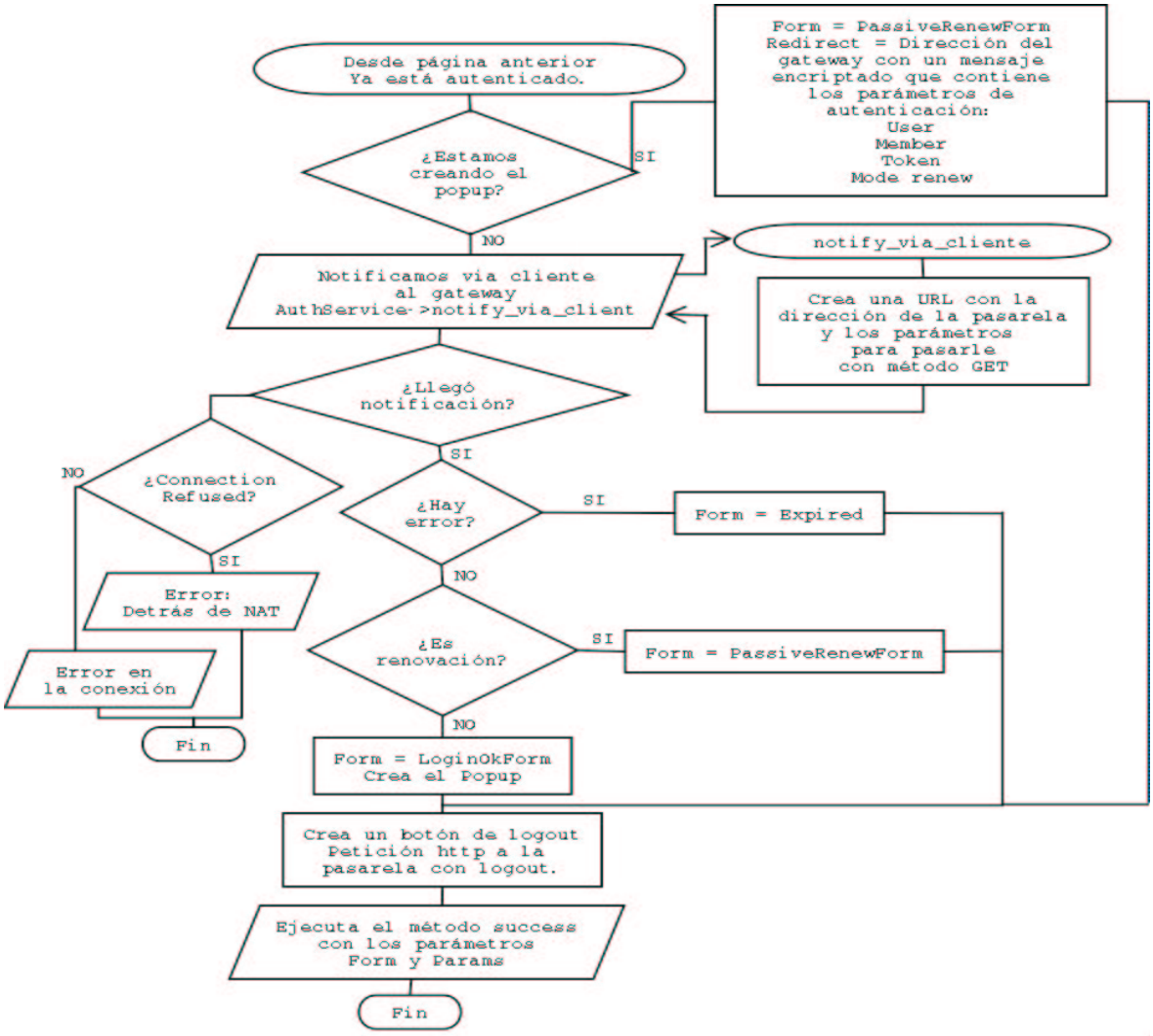

Figura A.9: /cgi-bin/login

El método success redirige al cliente a una página con el siguiente contenido:

- Por un lado, a los cinco segundos, envia una petición http dirigida a la pasarela que contiene el token, el permiso, la MAC, la dirección que el usuario había solicitado, etc...Cuando la pasarela verifica este mensaje es cuando cambia las reglas del firewall para permitir el acceso al usuario, y lo redirige a la página que había solicitado.
- Por otro lado abre un popup, y lo redirige al servidor. Concretamente vuelve a ejecutar el script de login pero esta vez el parámetro mode es popup. Como se puede ver en el diagrama de flujo de la figura A.9 el popup se convierte en un PassiveRenewForm con la siguiente información:
  - En primer lugar manda una petición http a la pasarela con un mensaje cifrado y firmado que contiene toda la información necesaria para la pasarela. Este mensaje lo manda a los 5 segundos.
  - En segundo lugar programa al popup para que a los 45 segundos vuelva a mandar un formulario al sevidor de autenticación con todos sus datos. Cuando esto es enviado a los 45 segundos se vuelve a refrescar el popup para enviarle los datos a la pasarela en 5 segundos y al servidor a los 45 segundos.

  Este proceso tiene una excepción, que es la primera vez que se ejecuta el popup. La primera vez que se ejecuta no envia nada a la pasarela, puesto que ya ha sido enviado por la página de redirección anterior. En lugar de eso manda al servidor de autenticación otro mensaje duplicado a los 45 segundos.

A.2. Cron

El cron es un demonio del sistema operativo que sirve para programar tareas. Cron puede ser iniciado al arrancar el sistema desde /etc/rc o /etc/rc.local.

Cron busca archivos crontab que han sido nombrados bajo una cuenta en /etc/passwd; los crontab encontrados son cargados en memoria. Cron también busca el archivo /etc/crontab y archivos en /etc/cron.d, estos están en un formato distinto. Cron se despierta cada minuto, examina todos los crontab guardados, y comprueba cada comando para ver si hay que ejecutarlo en ese minuto. Cuando ejecuta algún comando, cualquier salida es remitida al propietario del crontab.

Adicionalmente, cron comprueba cada minuto si el modtime de su directorio spool ha cambiado, si es así, cron examina el modtime de todos los crontab y recarga en memoria los que hayan cambiado. De esta manera cron no necesita ser reiniciado cada vez que se cambia un archivo crontab.

A.2.1. Archivos crontab

El archivo crontab contiene las instrucciones para que el demonio cron actúe y tienen el siguiente formato: "Ejecuta este comando a esta hora en esta fecha". Cada usuario tiene su crontab y los comandos que tiene un usuario en su crontab serán ejecutados como ese usuario.

Las lineas en blanco y las que comienzan con tabulador o espacios en blanco son ignoradas. Las líneas cuyo primer caracter que no sea un espacio en blanco sea una almohadilla (#) son comentarios y son ignoradas. Los comentarios no se permiten e la misma línea que los comandos cron. Los comentarios tampoco están permitidos en la misma línea donde se definen variables de entorno.

Un línea activa puede ser un comando cron una definición de variable de entorno. Las variables de entorno se definen con el formato,

name = value

donde los espacios entre el signo igual (=) son opcionales y cualquier espacio en blanco detrás del nombre será asignado a la variable. El valor puede estar entrecomillado para evitar esto.

Varias variables de entorno son iniciadas automaticamente por el demonio cron. SHELL está establecido como /bin/sh, y LOGNAME y HOME se cogen /etc/passwd del propietario del crontab. HOME y SHELL se pueden redefinir pero LOGNAME no.

El formato de los comandos cron es el siguiente: cada línea tiene cinco campos de fecha y hora, seguidos por el nombre de usuario si si es un crontab del sistema,

y finalmente un comando. El comando es ejecutado por el cron cuando el minuto, la hora, y el campo de mes coinciden con la hora actual, y al menos uno de los campos de dia (dia del mes o dia de la semana) coincide con el dia actual.

Los campos de fecha y hora son:

Campo	Valores permitidos
minuto	0-59
hora	0-23
dia del mes	1-31
mes	1-12
dia de la semana	0-7

Un campo puede ser un asterisco (*), y toma todos los valores permitidos.

Los rangos de números están permitidos. Los rangos son dos números separados por un guión, los extremos del rango inclusive. Por ejemplo, 8-11 para las horas se ejcutaría a las 8, 9, 10 y 11.

Las listas están permitidas. Las listas son un conjunto de números (o rangos) separados por comas.Ejemplos: "1,2,5,9", "0-4,8-12".

Se pueden usar saltos conjuntamente con los rangos. Se añade al rango "/<number>" especificando los saltos que hay que ir dando a través del rango. Por ejemplo, "0-23" en el campo hora sirve para ejecutar el comando cada dos horas.

Nombres pueden ser usados para los campos meses y dias de la semana. Se pueden usar las tres primeras letras en mayúsculas o minúsculas, por supuesto en inglés.

El resto de la línea es para el comando a ejecutar. El comando será ejecutado por /bin/sh o por el interprete especificado en la variable SHELL del cronfile. El signo de tanto por ciento (%) en el comando,a no ser que esté precedido por el caracter de escape \ será cambiado por un retorno de carro y lo que prosiga al % será enviado al comando por la entrada estandar.

A.3. Netfilter/Iptables

El proyecto Netfilter/iptables de Linux trata de crear un subsistema de cortafuegos que tenga capacidad para el filtrado de paquetes, cualquier clase de NAT[4] y marcado de paquetes. La mayor parte de netfilter/iptables está incluida en el kernel de Linux y

[4]Network Address Translation.

la mayoria de distribuciones incluyen la herramienta iptable, necesaria para el manejo y configuración del sistema.

A.3.1. Arquitectura de Netfilter

Netfilter es básicamente una serie de ganchos[5] en varios puntos del recorrido de un paquete que atraviesa una interfaz de red de nuestra máquina. El diagrama de recorrido (idealizado) de IPv4 se puede observar en la figura A.10.

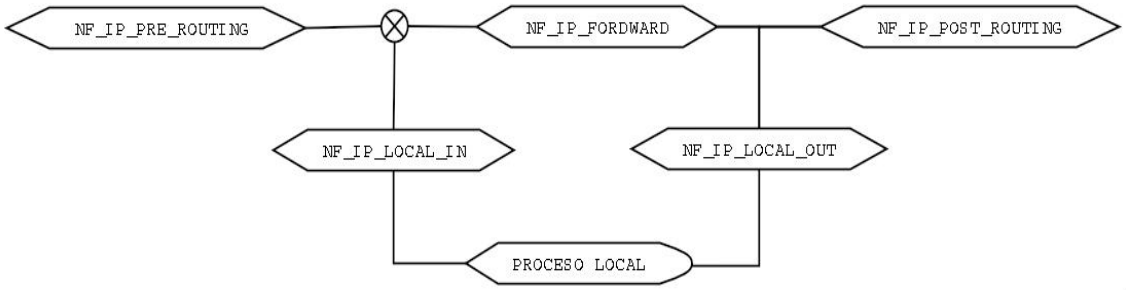

Figura A.10: Estructura de las Iptables

Observando la figura A.10 tenemos que los paquetes entran desde la izquierda: tras haber pasado las sencillas comprobaciones rutinarias (no está truncado, la suma de control IP es correcta y no es una recepción promiscua), son pasados al gancho NF_IP_PRE_ROUTING del sistema netfilter.

Luego entran en el código de enrutamiento, que decide si el paquete está destinado a otra interfaz o a un proceso local. El código de enrutamiento puede rechazar paquetes que no se pueden enrutar.

Si está destinado a la propia máquina, se llama de nuevo al sistema netfilter para el gancho NF_IP_LOCAL_IN, antes de ser enviado al proceso (si hay alguno).

Si, en cambio, está destinado hacia otra interfaz, se llama al sistema netfilter para el gancho NF_IP_FORWARD.

Luego el paquete pasa por un gancho final, el gancho NF_IP_POST_ROUTING, antes de ser enviado al siguiente nodo de enrutamiento.

Para los paquetes creados localmente, se llama al gancho NF_IP_LOCAL_OUT. Aquí puede ver que el enrutamiento ocurre después de haber llamado a este gancho: de hecho, se llama primero al código de enrutamiento (para averiguar la dirección IP y algunas opciones IP), y luego se le llama otra vez si el paquete ha sido alterado.

[5]En documentación inglesa se le llama *hooks*.

A.3.2. La base de Netfilter

Ahora que ya tenenos definidos los cinco ganchos, uno o varios módulos del kernel pueden registrarse para escuchar en alguno de estos ganchos. Luego, cuando se llama al gancho de netfilter desde el código de red, los módulos registrados en ese punto tienen libertad para manipular el paquete. Un módulo puede decirle a netfilter que haga una de estas cinco cosas:

NF_ACCEPT: continúa el recorrido normalmente. NF_DROP: rechaza el paquete; no continúes el recorrido. NF_STOLEN,: me hago cargo del paquete; no continúes el recorrido. NF_QUEUE: pon el paquete en una cola (normalmente para tratar con el espacio de usuario). NF_REPEAT: llama de nuevo a este gancho.

Las tres últimas no se van a explicar en este apéndice ya que no se usan en el trabajo y queda fuera de este proyecto.

A.3.3. Selección de paquetes: IP Tables

Hay un sistema de selección de paquetes llamado IP Tables sobre el sistema netfilter. Los módulos del kernel pueden registrar una tabla nueva, e indicarle a un paquete que atraviese una tabla dada. Este método de selección de paquetes se utiliza para el filtrado de paquetes (la tabla 'filter'), para la Traducción de Direcciones de Red (la tabla 'nat') y para la manipulación general de paquetes antes del enrutamiento (la tabla 'mangle').

Las funciones que se implementan en cada gancho y el orden en que se llaman es el mostrado en la figura A.11.

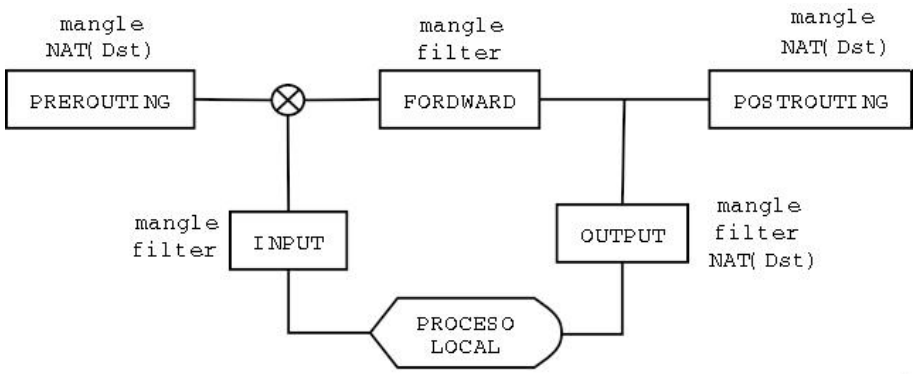

Figura A.11: Tablas de Iptables.

Filtrado de Paquetes

Esta tabla, 'filter', nunca altera los paquetes: sólo los filtra.

Se encuentra en los puntos NF_IP_LOCAL_IN, NF_IP_FORWARD y NF_IP_LOCAL_OUT. Esto significa que para cualquier paquete dado, existe un (y sólo un) posible lugar donde pueda ser filtrado. Esto hace las cosas mucho más sencillas. Además, el hecho de que el sistema netfilter proporcione las dos interfaces de entrada (input) y salida (output) para el gancho NF_IP_FORWARD significa que hay bastantes tipos de filtrado que se simplifican mucho.

NAT

Esto es el reino de la tabla 'nat', que se alimenta de paquetes mediante tres ganchos de netfilter: para los paquetes no locales, los ganchos NF_IP_PRE_ROUTING y NF_IP_POST_ROUTING son perfectos para las alteraciones en el destino y el origen, respectivamente. Para alterar el destino de los paquetes locales, se utiliza el gancho NF_IP_LOCAL_OUT.

Esta tabla es ligeramente distinta a la tabla 'filter' en el sentido de que sólo el primer paquete de una conexión nueva atravesará la tabla: el resultado de este recorrido se aplica luego a todos los paquetes futuros de la misma conexión.

Enmascaramiento, redireccionamiento de puertos y proxys transparentes

Se puede dividir NAT en NAT de Origen (en el que se altera el origen del primer paquete), y NAT de Destino (en el que se altera el destino del primer paquete).

El enmascaramiento es una forma especial de NAT de Origen; el redireccionamiento de puertos y los proxys transparentes son formas especiales de NAT de Destino.

Manipulación de paquetes

La tabla mangle se encarga de manipular la información de los paquetes y entre otras cosas sirve para el marcado de paquetes. Esto se usa para marcarlos por ejemplo cono pertenecientes a un tipo de servicio[6] que puede ser tráfico masivo, tráfico en tiempo real, etc...

[6]TOS-Type of Service.

A.3.4. Seguimiento de conexiones

El seguimiento de conexiones es fundamental para NAT, pero está implementado en un módulo aparte; esto permite una extensión del filtrado de paquetes para utilizar de manera limpia y sencilla el seguimiento de conexiones (el módulo 'state').

A.4. HTB

Para optimizar el tráfico en Linux existe una herramienta (TC)[7] que está incluida en las versiones superiores a la 2.4 del kernel, con la que es posible crear clases de tráfico y darles prioridad a éstas. Asímismo es posible realizar reservas del ancho de banda.

Vamos a dar algunas definiciones antes de seguir[8]:

- Qdisc. Un algoritmo que controla la cola de un dispositivo, sea de entrada (ingress) o de salida (egress).
- Qdisc raíz (root qdisc). La qdisc raíz es la que está adjunta al dispositivo.
- Qdisc sin clases. Una qdisc sin subdivisiones internas configurables.
- Qdisc con clases. Una qdisc con clases contiene mltiples clases. Algunas de ellas contienen otra qdisc, que a su vez puede ser con clases, pero no tiene por qué.
- Clases. Una qdisc con clases puede tener muchas clases, cada una de las cuales es interna a ella. A una clase, a su vez, se le pueden añadir varias clases. De manera que una clase puede tener como padre una qdisc u otra clase. Una clase terminal (o clase "hoja": leaf class) es una clase que no tiene clases hijas. Esta clase tiene 1 qdisc adjunta. Esta qdisc es responsable de enviar datos a la clase. Cuando creas una clase, se le adjunta una qdisc fifo. Cuando añades una clase hija, se elimina esta qdisc. Para clases terminales, esta qdisc fifo puede ser reemplazada con otra más adecuada. Incluso se puede reemplazar esta qdisc fifo por otra con clases de manera que se puedan añadir más clases.
- Clasificador. Cada qdisc con clases necesita determinar a qué clase necesita enviar un paquete. Esto se hace usando el clasificador.
- Filtro. La clasificacin se puede realizar usando filtros. Un filtro contiene varias condiciones que pueden ser cumplidas.

HTB[9] es una disciplina de colas con clases que permite de forma intuitiva y rápida crear, a partir de un enlace físico, varios enlaces simulados más lentos. A cada uno de estos enlaces se le pueden asociar los filtros que se desee para determinar qué paquetes corresponden a ese enlace y que ancho de banda tienen reservado.

[7] Traffic contol
[8] Definiciones extraidas del LARTC(Linux Advanced Routing & Traffic Control).
[9] Hierarchical Token Bucket.

A.4.1. Base teórica de HTB

Vamos a definir brevemente la teoría de HTB. En primer lugar veamos algunas definiciones:

- Clase. Cada clase tiene asociada una tasa garantizada (assured rate AR), una tasa máxima (ceil rate CR), una prioridad P, un nivel y un cuanto Q. Puede tener una clase padre. La tasa instantanea se llamará rate o R.
- Hoja. Es un clase que no tiene clases hijas.
- El nivel de una clase indica la posición en la jerarquía. Las hojas tienen nivel 0.

Una vez vistas las definiciones veamos como se realiza el reparto de ancho de banda en un enlace. Si tenemos una clase c, el valor de la tasa de bajada de c es:

$R_c = min(CR_c, AR_c + B_c)$

,es decir, la tasa instantánea de una clase es su tasa garantizada más B_c[10], simpre y cuando esto no supere su tasa máxima.

Ahora se va a definir B_c:

$$B_c = \begin{cases} \dfrac{Q_c \cdot R_p}{\sum\limits_{i:P_i=P_c} [Q_i]} & \text{si } \min_{\forall i \in D(p)}[P_i] \geq P_c \\ 0 & \text{e.o.c} \end{cases}$$

Podemos ver que B_c será distinto de cero, es decir, el incremento sobre la tasa garantizada en la clase c será distinto de cero, cuando existan clases con menor o igual prioridad que ella que desciendan del mismo padre. Obviamente si la clase no tiene clase padre, ninguna clase le prestará nada, y B_c será simpre 0.

En el caso de que haya clases en su mismo nivel con más prioridad[11] que él, entonces serán servidas antes. La fracción que se reparte del ancho de banda Rp, entre las hojas de la misma prioridad es proporcional al valor de los cuantos de esas clases.

Para resumir podemos decir que el ancho de banda que no se utiliza en una clase padre es repartido entre las hojas de mayor prioridad teniendo en cuenta el valor de los cuantos asignados.

Por otro lado se intenta mantener la independencia de las clases. Los cambios en la tasa instantánea de una clase no debe afectar al retardo de otras clases, a no ser que

[10] B_c es la tasa que toma prestada la clase c de las clases que están en su mismo nivel y tienen la misma clase padre.

[11] El valor más prioritario es el 0, nosotros nos referimos con tener más prioridad a tener un valor menor.

ambas estén recibiendo simultaneámente ancho de banda prestado del mismo padre. Además se cumple que las clases con más prioridad tienen un retardo menor que las clases con menos prioridad dentro de un mismo nivel.

A.4.2. Guia de usuario

En este apartado vamos a ver como crear unas reglas sencillas para controlar el tráfico en una interfaz de red. En este caso la interfaz elegida será la eth0. Hay que tener en cuenta que el tráfico que se controla es el tráfico de salida de la interfaz. Es posible controlar el envio de paquetes, ya que se puede retrasar el envío de unos para beneficiar el envio de otros. La recepción de paquetes no se puede controlar, y por tanto el tráfico entrante en la interfaz tampoco. En el ejemplo vamos a crear una clase raiz de la que descienden dos clases.

Lo primero que hay que hacer es asociar una clase padre a una interfaz. Para hacer esto se usa el comando:

tc qdisc add dev eth0 root handle 1: htb default 2

Este comando asocia a la interfaz eth0 una disciplina de colas HTB y le asigna el manejador 1. El manejador es sólo un nombre o identificador para referirse a la clase en los siguientes comandos. "default 2" significa que el tráfico que no coincida con ningún filtro irá la clase 1:2[12].

Con los siguientes comandos se crean tres clases:

tc class add dev eth0 parent 1: classid 1:1 htb rate 100kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:10 htb rate 30kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:11 htb rate 10kbps ceil 100kbps

La primera línea crea la clase raiz 1:1 que pertenece a la disciplina de colas 1:. La clase raíz permite que sus hijos puedan compartir el ancho de banda entre ellos, pero la clase raíz no lo puede compartir con otra. Se podrían crear todas las clases que quiesiéramos bajo la disciplina de colas 1: pero el exceso de ancho de banda de una clase no estaría disponible para el resto, entonces lo que se hace es crear una clase raíz bajo la disciplina de colas htb, y el resto son hijas de esta clase raíz. Así que con las líneas anteriores hemos creado una clase raíz y dos clases que dependen de ésta. Las dos clases hijas pueden compartir el ancho de banda que no usen entre ellas.

Los parámetros rate y ceil tienen el siguiente significado:

- Rate. Indica el ancho de banda que tiene reservada una clase. Como mínimo siempre va a tener disponible ese ancho de banda.

[12]Los manejadores se denotan por *x:y* donde x identifica la disciplina de colas e y identifica una clase perteneciente a esa disciplina de colas.

- Ceil. Indica el máximo ancho de banda que puede llegar a tener una clase. Cuando el resto de clases deja libre un ancho de banda, éste es compartido con el resto de clases, pues el máximo que puede tener una clase cuando recibe de otras es igual al valor de ceil. Como máximo el valor de ceil tiene que ser el ancho de banda del que dispone su clase padre. Por defecto si no se indica nada el valor de ceil es igual al valor de rate.

Además de estos parámetros hay otros dos que merece la pena destacar:

- Burst. El parámetro burst mide el tamaño de ráfaga. Es decir, el parámetro burst controla la cantidad de paquetes que se pueden enviar seguidos de una misma clase a la máxima tasa que se dispone por hardware.
- Prio. El parámetro prio sirve para determinar la prioridad de una clase. Como hemos visto en la sección anterior, las clases con más prioridad son servidas antes y el retardo es menor. La clase más prioritaria tiene el valor 0.

Para asignar una dispciplina de colas a cada una de estas clases hoja o finales se usa el comando:

tc qdisc del dev eth0 parent 1:10 handle 10 sfq perturb 10 tc qdisc del dev eth0 parent 1:12 handle 12 sfq perturb 10

Hemos asignado una disciplina sfq, que es la que recomienda el autor de HTB. Esta disciplina de colas *Stochastic Fairness Queueing* es una una implementación sencilla de la familia de algoritmos de colas justas. Hace un reparto equitativo entre todos el tráfico que pertenece a dicha clase.

Hay que asignar qué paquetes corresponden a cada clase, para esto se utiliza el comando:

tc filter dev eth0 protocol ip parent 1:0 prio 1 u32 $filtro

donde filtro puede ser muchas cosas, y en particular se puede indicar:

- Origen de un paquete. Los paquetes que provengan de ese origen iran a esa clase.
- Destino de un paquete. Los paquetes que tengan ese destino perteneceran a dicha clase.
- Marcados de paquetes. Los paquetes que tengan una marca en concreto.
- Puertos origen.
- Puertos destino.