Введение

Наиболее частым приемом раздачи интернета клиентам является использование специализированного ПО, т.н. прокси серверов. В мире unix пожалуй самый распространенным прокси сервером является - squid.

Для его использования клиент должен указать в настройках своего интренет проводника адрес и порт прокси сервера. К сожалению, иногда такая возможность отсутствует или ее выполнение затрудненно. Выходом из данной ситуации является использование прокси сервера в т.н. transparent («прозрачном») режиме.

Собственно само название появилось из-за того, что для клиента работа с интеренетом выглядит «прозрачно», т.е. ему нет необходимости прописывать адрес и порт прокси сервера. Единственное что ему нужно это адрес шлюза по умолчанию и адрес днс сервера. Пожалуй, идеальным выходом будет раздача данных параметров с помощью dhcp сервера, но это уже на ваше усмотрение.

К сожалению, у этого способа также существуют и недостатки, к которым можно отнести следующее:

- невозможность использовать аутентификацию;
- невозможность проксировать протокол https. Это связанно с особенностью самого протокола;
- невозможность проксировать протокол ftp. В «обычном» режиме squid позволяет проксировать ftp протокол. Но данную проблему можно решить с помощью программы frox (A transparent FTP proxy);

Но несмотря на эти недостатки вы попрежнему можете:

- ограничивать скорость закачек с помощью delay pools;
- использовать редиректор (squidGuard, rejik) для ограничения закачек музыки, видео, игр и т.п.;

Итак, у нас имеется следующая система:

```
# cat /etc/redhat-release
CentOS release 5 (Final)

# uname -r
2.6.18-53.1.4.el5
```

На данной системе установлено 3 сетевых адаптера:

- eth0 (192.168.0.1) - смотрит в локальную сеть;
- eth1 (192.168.1.1) - смотрит в wi-fi сеть;
- eth2 (212.42.65.50) - смотрит в мир;

Наша задача состоит в следующем: ограничить wi-fi клиентов от клиентов локальной сети. Разрешить им доступ только в инет через прозрачный прокси север. Клиенты из локальной сети будут выходить в интернет через тот же прокси сервер, но который будет работать не в

прозрачном режиме, а обычном.

Установка и настройка dhcp

Устанавливаем и настраиваем dhcp сервер. С помощью его мы будем выдавать адреса, а также другую необходимую информацию (адрес dns сервера, шлюза по умолчанию) wi-fi клиентам.

```
# yum install dhcp
```

Редактируем файл /etc/dhcpd.conf под свои нужды. Ниже приведены минимальные настройки для работы dhcp сервера.

```
#
# /etc/dhcpd.conf
#

ddns-update-style none;
ignore client-updates;

subnet 192.168.1.0 netmask 255.255.255.0 {

        option routers                  192.168.1.1;
        option subnet-mask              255.255.255.0;
        option domain-name-servers      192.168.1.1;

        default-lease-time 3600;
        max-lease-time 25200;

        range 192.168.1.100 192.168.1.200;
}
```

После этого настраиваем запуск dhcp при старте системы и запускаем сам сервер.

```
# chkconfig --level 35 dhcpd on
# service dhcpd start
Starting dhcpd:                                            [  OK  ]
```

Для чистоты совести смотрим лог файл.

```
# cat /var/log/messages | grep dhcpd
Dec 15 09:05:45 centos5 dhcpd: Internet Systems Consortium DHCP Server
V3.0.5-RedHat
Dec 15 09:05:45 centos5 dhcpd: Copyright 2004-2006 Internet Systems
Consortium.
Dec 15 09:05:45 centos5 dhcpd: All rights reserved.
Dec 15 09:05:45 centos5 dhcpd: For info, please visit
http://www.isc.org/sw/dhcp/
Dec 15 09:05:45 centos5 dhcpd: Wrote 1 leases to leases file.
Dec 15 09:05:45 centos5 dhcpd:
```

```
Dec 15 09:05:45 centos5 dhcpd: No subnet declaration for eth2
(212.42.65.50).
Dec 15 09:05:45 centos5 dhcpd: ** Ignoring requests on eth2.  If this is not
what
Dec 15 09:05:45 centos5 dhcpd:    you want, please write a subnet
declaration
Dec 15 09:05:45 centos5 dhcpd:    in your dhcpd.conf file for the network
segment
Dec 15 09:05:45 centos5 dhcpd:    to which interface eth2 is attached. **
Dec 15 09:05:45 centos5 dhcpd:
Dec 15 09:05:45 centos5 dhcpd: Listening on
LPF/eth1/00:0c:29:06:7d:da/192.168.1/24
Dec 15 09:05:45 centos5 dhcpd: Sending on
LPF/eth1/00:0c:29:06:7d:da/192.168.1/24
Dec 15 09:05:45 centos5 dhcpd:
Dec 15 09:05:45 centos5 dhcpd: No subnet declaration for eth0 (192.168.0.1).
Dec 15 09:05:45 centos5 dhcpd: ** Ignoring requests on eth0.  If this is not
what
Dec 15 09:05:45 centos5 dhcpd:    you want, please write a subnet
declaration
Dec 15 09:05:45 centos5 dhcpd:    in your dhcpd.conf file for the network
segment
Dec 15 09:05:45 centos5 dhcpd:    to which interface eth0 is attached. **
Dec 15 09:05:45 centos5 dhcpd:
Dec 15 09:05:45 centos5 dhcpd: Sending on   Socket/fallback/fallback-net
Dec 15 09:05:45 centos5 dhcpd: dhcpd startup succeeded
```

Как видно из сообщений, dhcp сервер успешно стартовал и привязался к интерфейсу eth1.

Установка и настройка squid

Устанавливаем squid.

```
# yum install squid
```

Редактируем файл /etc/squid/squid.conf под свои нужды. Ниже приведены минимальные настройки для работы squid сервера.

```
#
#   /etc/squid/squid.conf
#

# NETWORK OPTIONS
# ------------------------------------------------------------------------
---
# Указываем  squid  на каком порту и интерфейсе он будет работать. Именно эти
# параметры необходимо будет указывать в настройках интернет проводника.
http_port 192.168.0.1:3128
http_port 192.168.1.1:3128 transparent
```

```
# TAG: auth_param
# Здесь мы указываем squid, как следует производить аутентификацию.
# Единственный параметр, который необходимо указать ncsa_auth -
# это имя файла, в котором хранятся имена пользователей и пароли.
# Данным файлом можно управлять (добавлять новых пользователей,
# удалять текущих, изменять пароли и т.д.) с помощью утилиты
# htpasswd, которая входит в состав пакета apache.
auth_param basic program /usr/lib/squid/ncsa_auth /etc/squid/internet_users
auth_param basic children 5
auth_param basic realm Squid proxy-caching web server
auth_param basic credentialsttl 2 hours
auth_param basic casesensitive off

# ACCESS CONTROLS
# ------------------------------------------------------------------------
---
# ACL - Access Control List. Списки доступа к нашему прокси серверу.
# Здесь мы указываем, кто имеет право, а кто нет, использовать наш прокси.
# Разрешаем использовать наш прокси только прошедшим авторизацию.

acl wifi src 192.168.1.0/24
acl lan proxy_auth REQUIRED

http_access allow wifi
http_access allow lan
http_access deny all
```

После этого настраиваем запуск squid при старте системы и запускаем сам прокси сервер.

```
# chkconfig --level 35 squid on
# service squid start
Starting squid:                                            [  OK  ]
```

```
# cat cache.log
2007/12/15 09:12:36| Starting Squid Cache version 2.6.STABLE6 for i686-
redhat-linux-gnu...
2007/12/15 09:12:36| Process ID 4752
2007/12/15 09:12:36| With 1024 file descriptors available
2007/12/15 09:12:36| Using epoll for the IO loop
2007/12/15 09:12:36| DNS Socket created at 0.0.0.0, port 1028, FD 8
2007/12/15 09:12:36| Adding nameserver 127.0.0.1 from /etc/resolv.conf
2007/12/15 09:12:36| User-Agent logging is disabled.
2007/12/15 09:12:36| Referer logging is disabled.
2007/12/15 09:12:36| Unlinkd pipe opened on FD 13
2007/12/15 09:12:36| Swap maxSize 102400 KB, estimated 7876 objects
2007/12/15 09:12:36| Target number of buckets: 393
2007/12/15 09:12:36| Using 8192 Store buckets
2007/12/15 09:12:36| Max Mem  size: 8192 KB
2007/12/15 09:12:36| Max Swap size: 102400 KB
2007/12/15 09:12:36| Local cache digest enabled; rebuild/rewrite every
3600/3600 sec
```

```
2007/12/15 09:12:36| Rebuilding storage in /var/spool/squid (CLEAN)
2007/12/15 09:12:36| Using Least Load store dir selection
2007/12/15 09:12:36| Set Current Directory to /var/spool/squid
2007/12/15 09:12:36| Loaded Icons.
2007/12/15 09:12:37| Accepting transparently proxied HTTP connections at
192.168.1.1, port 3128, FD 15.
2007/12/15 09:12:37| Accepting proxy HTTP connections at 192.168.0.1, port
3128, FD 16.
2007/12/15 09:12:37| Accepting ICP messages at 0.0.0.0, port 3130, FD 17.
2007/12/15 09:12:37| WCCP Disabled.
2007/12/15 09:12:37| Ready to serve requests.
2007/12/15 09:12:37| Done reading /var/spool/squid swaplog (39 entries)
2007/12/15 09:12:37| Finished rebuilding storage from disk.
2007/12/15 09:12:37|        39 Entries scanned
2007/12/15 09:12:37|         0 Invalid entries.
2007/12/15 09:12:37|         0 With invalid flags.
2007/12/15 09:12:37|        39 Objects loaded.
2007/12/15 09:12:37|         0 Objects expired.
2007/12/15 09:12:37|         0 Objects cancelled.
2007/12/15 09:12:37|         0 Duplicate URLs purged.
2007/12/15 09:12:37|         0 Swapfile clashes avoided.
2007/12/15 09:12:37|   Took 0.5 seconds (  73.4 objects/sec).
2007/12/15 09:12:37| Beginning Validation Procedure
2007/12/15 09:12:37|   Completed Validation Procedure
2007/12/15 09:12:37|   Validated 39 Entries
2007/12/15 09:12:37|   store_swap_size = 248k
2007/12/15 09:12:37| storeLateRelease: released 0 objects
```

Настройка брандмауера

Создаем минимальный набор правил в нашем фаерволе. Этот пример создан лишь для того, чтобы показать принцип работы iptables. Для более углубленного понимания работы iptables рекомендую прочитать статью Оскара Андерсона в переводе Андрея Кисилева - Iptables Tutorial 1.1.19, которая считается классикой.

```
#
# /usr/local/firewall.sh
#

#!/bin/sh
IPTABLES="/sbin/iptables"

# Отчищаем все правила в таблицах filter, nat и mangle

$IPTABLES -t filter -F
$IPTABLES -t nat -F
$IPTABLES -t mangle -F

# Удаляем все пользовательские цепочки в таблицах filter, nat и mangle
```

```
$IPTABLES -t filter -X
$IPTABLES -t nat -X
$IPTABLES -t mangle -X

# Задаем политики по умолчанию

$IPTABLES -t filter -P INPUT DROP
$IPTABLES -t filter -P FORWARD DROP
$IPTABLES -t filter -P OUTPUT DROP

# Создаем пользовательские цепочки. Как строить firewall каждый решает сам.
# Но лично мне удобно настраивать фаервол, когда все разбито по цепочкам
# и как бы разложено по своим полочкам. Тогда я точно знаю, что и где надо
# искать или исправлять. Думаю, названия цепочек говорят сами за себя.

$IPTABLES -N eth0-eth2
$IPTABLES -N eth2-eth0
$IPTABLES -N eth1-eth2
$IPTABLES -N eth2-eth1
$IPTABLES -N eth2-out
$IPTABLES -N eth1-out
$IPTABLES -N eth0-out
$IPTABLES -N eth2-in
$IPTABLES -N eth1-in
$IPTABLES -N eth0-in
$IPTABLES -N icmp_packets

# Направляем все входящие пакеты в соответствующие цепочки.

$IPTABLES -A INPUT -d 192.168.0.1 -j eth0-in
$IPTABLES -A INPUT -d 192.168.1.1 -j eth1-in
$IPTABLES -A INPUT -d 212.42.65.50 -j eth2-in

# Направляем все исходящие пакеты в соответствующие цепочки.

$IPTABLES -A OUTPUT -s 192.168.0.1 -j eth0-out
$IPTABLES -A OUTPUT -s 192.168.1.1 -j eth1-out
$IPTABLES -A OUTPUT -s 212.42.65.50 -j eth2-out

# Разрешаем весь трафик с петлевого интерфейса.

$IPTABLES -A INPUT -i lo -j ACCEPT
$IPTABLES -A OUTPUT -o lo -j ACCEPT

# Разрешаем следующие типы icmp пакетов.

$IPTABLES -A icmp_packets -p icmp --icmp-type  -j ACCEPT
$IPTABLES -A icmp_packets -p icmp --icmp-type 3 -j ACCEPT
$IPTABLES -A icmp_packets -p icmp --icmp-type 8 -j ACCEPT
$IPTABLES -A icmp_packets -p icmp --icmp-type 11 -j ACCEPT
```

```
# Для удобства фильтрации, направляем все транзитные пакеты,
# в соответствующие цепочки.
# eth0-eth1  локальная сеть -> мир
# eth1-eth0  мир -> локальная сеть
# eth1-eth2  wi-fi  сеть -> мир
# eth2-eth1  мир  -> wi-fi  сеть

$IPTABLES -A FORWARD -i eth1 -o eth2 -j eth1-eth2
$IPTABLES -A FORWARD -i eth2 -o eth1 -j eth2-eth1
$IPTABLES -A FORWARD -i eth0 -o eth2 -j eth0-eth2
$IPTABLES -A FORWARD -i eth2 -o eth0 -j eth2-eth0

# eth0-eth2.  В данную цепочку попадают все транзитные пакеты,
# направленные из локальной сети в мир.

$IPTABLES -A eth0-eth2 -p icmp -j icmp_packets
$IPTABLES -A eth0-eth2 -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A eth0-eth2 -j DROP

# eth2-eth0.  В данную цепочку попадают все транзитные пакеты,
# направленные из мира в локальную сеть.

$IPTABLES -A eth2-eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A eth2-eth0 -j DROP

# eth1-eth2.  В данную цепочку попадают все транзитные пакеты,
# направленные из  wi-fi  сети в мир.

$IPTABLES -A eth1-eth2 -p icmp -j icmp_packets
$IPTABLES -A eth1-eth2 -p tcp --dport 443 -j ACCEPT
$IPTABLES -A eth1-eth2 -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A eth1-eth2 -j DROP

# eth2-eth1.  В данную цепочку попадают все транзитные пакеты,
# направленные из мира в  wi-fi  сеть.

$IPTABLES -A eth2-eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A eth2-eth1 -j DROP

# eth0-in.  В данной цепочке открываем порты тех служб, которые
# должны быть доступны на сервере из локальной сети.

$IPTABLES -A eth0-in -p icmp -j icmp_packets
$IPTABLES -A eth0-in -p tcp --dport 22 -j ACCEPT
$IPTABLES -A eth0-in -p udp --dport 53 -j ACCEPT
$IPTABLES -A eth0-in -p tcp --dport 3128 -j ACCEPT
$IPTABLES -A eth0-in -j DROP

# eth1-in.  В данной цепочке открываем порты тех служб, которые
# должны быть доступны на сервере из  wi-fi  сети.
```

```
$IPTABLES -A eth1-in -p icmp -j icmp_packets
$IPTABLES -A eth1-in -p udp --dport 53 -j ACCEPT
$IPTABLES -A eth1-in -p udp --dport 67 --sport 68 -j ACCEPT
$IPTABLES -A eth1-in -p tcp --dport 3128 -j ACCEPT
$IPTABLES -A eth1-in -j DROP

# eth2-in.  В данной цепочке открываем порты тех служб, которые
# должны быть доступны на сервере из мира. Например  www,  smtp,  ftp

$IPTABLES -A eth2-in -p icmp -j icmp_packets
$IPTABLES -A eth2-in -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A eth2-in -j DROP

# eth0-out.

$IPTABLES -A eth0-out -p icmp -j icmp_packets
$IPTABLES -A eth0-out -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A eth0-out -j DROP

# eth1-out.

$IPTABLES -A eth1-out -p icmp -j icmp_packets
$IPTABLES -A eth1-out -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A eth1-out -j DROP

# eth2-out.

$IPTABLES -A eth2-out -p icmp -j icmp_packets
$IPTABLES -A eth2-out -p udp --dport 53 -j ACCEPT
$IPTABLES -A eth2-out -p tcp -m multiport --dport 80,443,8080 -j ACCEPT
$IPTABLES -A eth2-out -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A eth2-out -j DROP

# Производим сетевую трансляцию адресов  (NAT)

$IPTABLES -t nat -A POSTROUTING -s 192.168.0.0/24 -o eth2 -j SNAT --to-
source 212.42.65.50
$IPTABLES -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth2 -j SNAT --to-
source 212.42.65.50

# "Заварачиваем" весь  http  трафик на  squid.

$IPTABLES -t nat -A PREROUTING -s 192.168.1.0/24 -i eth1 -p tcp -m multiport
--dport 80,8080 -j REDIRECT --to-ports 3128
```

eth0 - смотрит в локальную сеть, eth1 - смотрит в мир. Данным скриптом (firewall.sh) удобно пользоваться на стадии отладки правил.

```
# chmod +x /usr/local/firewall.sh
```

```
# /usr/local/firewall.sh
```

Данный фаерволл построен по принципу - «запрещено все, кроме разрешенного». Написание фаерволов по такому принципу дает более глубокое понимание того, что происходит в процессе работы того или иного сервиса, какие порты и протоколы он использует и т.д.

После того, как у вас все настроено, лучше воспользоваться штатными средствами управления iptables. Для этого воспользуемся скриптом для сохранения текущих правил в файл. В разных дистрибутивах набор правил может сохраняться по отличному от /etc/sysconfig/iptables пути. Данный путь является стандартным для Red Hat дистрибутивов и его клонов, например CentOS.

```
# service iptables save
Saving firewall rules to /etc/sysconfig/iptables:          [  OK  ]

# chkconfig --level 35 iptables on
# service iptables restart
Flushing firewall rules:                                   [  OK  ]
Setting chains to policy ACCEPT: mangle filter nat         [  OK  ]
Unloading iptables modules:                                [  OK  ]
Applying iptables firewall rules:                          [  OK  ]
```

Тестирование

На этом собственно настройку данной системы можно считать завершенной. Для проверки корректности работы необходимо с помощью любого ноутбука подключиться по wi-fi к нашей точке доступа и в любом интернет проводнике открыть любимый сайт, например www.google.com.