Parental control on the Linux desktop

Introduction

It is fascinating to observe how children take things for granted where their parents need many years to get used to them ... take computers and the Internet for instance. Kids are so much more *savvy* with the computer that it becomes increasingly more difficult for parents to understand and value their kids' activities on the Internet. The results are not always that positive - children should be protected from some of the darker sides of the Internet. It already helps if parents sit and watch by the side of their kids from time to time, to see what they are up to, to talk about what they encounter and to give friendly advice.

Then again, it is impossible to monitor your children's computer usage 24/7. So, it would be neat to have a way of filtering their access to the Internet.
One way of achieving this, is by installing a content filter. The content filter is a program that checks URLs and disallows access in case the URL is blacklisted, and examines the content of the returned Internet pages so that it can block access to undesired content (violence, explicit language or images, etcetera).
This filtering should be automatic - in the sense of executing beyond the childrens' control. A youngster will find out soon enough where in the browser you configured a web proxy, and then acts by simply removing the proxy definition. We Linux users are in the happy circumstance that all the tools we need for such an automatic filter already exist - it is merely a matter of downloading, installing, configuring and avtivating the various bits and pieces.

In another Wiki article I talk about setting up such a content filter in a *network*. This solution is called *transparent proxying*. The content filter in that article is chained to a proxy program and is installed on the Internet gateway server. This presumes that you *have* a local network with multiple computers, and a server in your network that can act as the Internet Gateway.
This article however, will concentrate on controlling the Internet access on a solitary PC, like the family PC you might have in your living room. It *should* run Linux of course 🙂 and preferably Slackware. What I will show you is this:

- Install and configure a content filter - dansguardian
- Install and configure a proxy - tinyproxy
- [*Optionally*] install and configure a virusscanner - clamav
- Setup iptables firewall rules that relay all internet browser traffic through the content filter
- Yet allow unlimited Internet access for the parents' login account, and for *root*.

The sections on compiling and installing tinyproxy and dansguardian are taken from that other article, to make *this* article self-contained. I still encourage you to read the transparent proxy article if you want to know a little more on the background of (transparent) proxying and content filtering.

Proxy installation

The homepage for tinyproxy has download links for “stable” and “devel” versions. I have used the “devel” version in the past, but I decided to switch to the “stable” release since that was actually

released many months after the latest development release. Although this software has not had an update for a long time, the tinyproxy web site shows new activities, so I expect new releases to appear anytime soon. The “devel” program however is actually quite stable.
Tinyproxy must explicitly be build for transparent proxy support. If you want to compile and install the software manually, this is what you would do:

```
tar -zxvf tinyproxy-1.6.3.tar.gz
cd tinyproxy-1.6.3
./configure --prefix=/usr \
            --localstatedir=/var \
            --sysconfdir=/etc \
            --enable-xtinyproxy \
            --enable-filter \
            --enable-upstream \
            --enable-reverse \
            --enable-transparent-proxy \
            --program-prefix="" \
            --program-suffix=""
make
make install
```

If you rather install a Slackware package or want to use a SlackBuild script to create a Slackware package, you'll find all you need in my repository

Contentfilter installation

The dansguardian software is actively maintained. You will need the basic software package you can download from the dansguardian homepage. It's default configuration will already be sufficient for a lot of people. If you want more extensive URL blacklists or badword lists you can look at the website. Some extensions you'll find have to be paid for however.
Although the most current release is in the *BETA* download section, it's actually quite stable. I used that for my install. For the manually compiling people:

```
tar -zxvf dansguardian-2.9.9.4.tar.gz
cd dansguardian-2.9.9.4
./configure --prefix=/usr \
            --localstatedir=/var \
            --sysconfdir=/etc \
            --enable-pcre \
            --enable-clamd \
            --enable-email \
            --enable-commandline \
            --with-proxyuser=nobody \
            --with-proxygroup=nobody
make
make install
```

I have a SlackBuild and a Slackware package for dansguardian in my repository which you can use as well. The advantage being that I added a start script and a logrotate script to the package. If you want

those without building from my SlackBuild script, I added them in the last section.
I configured dansguardian to run as user *nobody* - because that is an existing account without privileges, and Apache uses it too. If you want dansguardian to run using another account, you will have to do so in the *configure* stage, see above. Change the `–with-proxyuser=` and `–with-proxygroup=` parameters to contain the desired user account. Do not forget to create this account (user and group) on your Slackware machine in case the account does not yet exist! Suppose you want to run dansguardian as the user *filter* and group *parental*, you can run the following commands to create those:

```
groupadd parental
useradd -g parental -s /bin/false filter
```

NOTE: We will configure tinyproxy to run as user *nobody* as well, but in tinyproxy's case, we don't have to define that at compile-time. Tinyproxy defines the effective user as which the program executes in a parameter in it's configuration file (see below).

Virusscanner installation

ClamAV is an Open Source virus scanner with a decent catch rate and a good virus pattern update policy. It is able to catch viruses in downloaded files as well as malicious HTML code (phishing and pharming for instance). Dansguardian supports clamav as a plugin contentscanner, for all downloaded content. If you want to compile clamav yourself, the basic commands are like this:

```
groupadd clamav
useradd -g clamav -s /bin/false clamav
./configure --prefix=/usr \
            --localstatedir=/var \
            --sysconfdir=/etc \
            --with-user=clamav --with-group=clamav \
            --with-dbdir=/usr/share/clamav \
            --with-libcurl \
            --with-tcpwrappers \
            --enable-milter \
            --enable-id-check
make
make install
```

A pre-built package is available from my repository where you can also find the SlackBuild script to automatically build the Slackware package yourself if you prefer that. The package comes with a convenient rc script that can start/stop the deamons for you. If you decide to build from source, you can find a copy of this rc script in the last section.

You'll have noticed that the above `configure` command makes clamav run as a user *clamav*. The virus database files will be owned by that user and are inaccessible to other users on the system (so their integrity can not be compromised). To make dansguardian talk to the clamav program, their respective user accounts will need to have “something” in common. What that “something” is, will be detailed in the section on the clamav configuration.

Configuration

Next step is to configure our contentfilter.

Proxy

The content of the tinyproxy configuration file `/etc/tinyproxy/tinyproxy.conf` (stripped of comments and empty lines) for our example setup can be found in the last section.
I entered the domain name for my internal lan *my.net* in this configuration file. If yours is different, please change accordingly.
To show you where this differs from the tinyproxy defaults, here is a diff from the original file:

```
diff /etc/tinyproxy/tinyproxy.conf-dist /etc/tinyproxy/tinyproxy.conf
20c20
< Port 8888
---
> Port 3128
27c27
< #Listen 192.168.0.1
---
> Listen 127.0.0.1
112c112
< #XTinyproxy mydomain.com
---
> XTinyproxy my.net
192c192
< Allow 192.168.1.0/25
---
> #Allow 192.168.2.0/24
```

The important lines here are as follows:

```
Port 3128
Listen 127.0.0.1
Allow 127.0.0.1
```

These achieve the following:

- make tinyproxy listen on `127.0.0.1:3128` where dansguardian will contact it
- allow the localhost (IP address 127.0.0.1, for dansguardian) access - implicitly denying access attempts from any other IP address but 127.0.0.1.

Contentfilter

You will find the content of the dansguardian configuration file `/etc/dansguardian/dansguardian.conf` (stripped of comments and empty lines) for our example setup in the last section. The difference with the originally distributed file is quite small:

```
diff /etc/dansguardian/dansguardian.conf.new
/etc/dansguardian/dansguardian.conf
48a49
> anonymizelogs = off
74c75
< filterip =
---
> filterip = 127.0.0.1
97c98
< accessdeniedaddress =
'http://YOURSERVER.YOURDOMAIN/cgi-bin/dansguardian.pl'
---
> accessdeniedaddress = 'http://icculus.qemu.lan/cgi-bin/dansguardian.pl'
399c400
< #contentscanner = '/etc/dansguardian/contentscanners/clamdscan.conf'
---
> contentscanner = '/etc/dansguardian/contentscanners/clamdscan.conf'
```

The important lines in this file are:

```
filterip = 127.0.0.1
filterport = 8080
proxyip = 127.0.0.1
proxyport = 3128
```

They show that dansguardian

- will listen at the *filterip/filterport* address, i.e. `127.0.0.1:8080`. This is the IP address and TCP port to where we will transparently redirect all web browser requests from your children. Dansguardian is the primary entry point for the browsers' http requests.
- will look for a compatible proxy at IP address:port `127.0.0.1:3128`. This matches exactly with how we configured the IP address and TCP port for tinyproxy.

The line

```
accessdeniedaddress = 'http://127.0.0.1/cgi-bin/dansguardian.pl'
```

does not really matter, because in dansguardian's (and our) default configuration, the *access denied* web page is generated from a language-dependent template page. That is why you define `language = 'ukenglish'` - if you want dansguardian to show it's messages in another language, look in directory `/usr/share/dansguardian/languages/` for the available languages.

The line

```
contentscanner = '/etc/dansguardian/contentscanners/clamdscan.conf'
```

should only be enabled when you install the optional ClamAV virus scanner. In this case, you should also edit that file `/etc/dansguardian/contentscanners/clamdscan.conf` and uncomment the line

```
#clamdudsfile = '/var/run/clamav/clamd.sock'
```

so that it looks like

```
clamdudsfile = '/var/run/clamav/clamd'
```

The file `/var/run/clamav/clamd` is the socket file where the virusscanner listen for scan requests. It should match the file name configured in `/etc/clamd.conf`, by default this is:

```
LocalSocket /var/run/clamav/clamd
```

Of course, there is a lot of fine-tuning possibilities in this configuration file, as well as many others in the `/etc/dansguardian` directory tree. But in it's default setup dansguardian already does some impressive (perhaps *aggressive* is the better word) filtering.

Virusscanner

The ClamAV virusscanner consists of several components. The *clamd* program which does all the scanning and the *freshclam* program which periodically retrieves updates to the virus definition files are what we need for the content filter. These programs are conveniently started/stopped by the rc-script that is installed with my clamav Slackware package, and of which you find a copy in the last section of this article. Both programs will only start and work correctly after you edited their respective configuration files `/etc/clamd.conf` and `/etc/freshclam.conf`. The difference to the distributed files are as follows:

- **/etc/clamd.conf**

```
diff /etc/clamd.conf.new /etc/clamd.conf
8c8
< Example
---
> #Example
43c43
< #LogSyslog
---
> LogSyslog
48c48
< #LogFacility LOG_MAIL
---
> LogFacility LOG_MAIL
```

The important lines in `/etc/clamd.conf` are these:

```
LocalSocket /var/run/clamav/clamd
User clamav
AllowSupplementaryGroups
```

  - The *LocalSocket* is the file through which dansguardian will communicate with the virusscanner.

    - The *User* keyword shows as which user the scanning daemon is running.
    - The *AllowSupplementaryGroups* keyword is what we need in to let the dansguardian user account (*nobody* by default) communicate with the clamav user account (*clamav*). If enabled, clamav will accept connections from another program if the user ID of that program has a *group* in common with the clamav user ID. Remember that clamav was configured so that only the *clamav* account had access to the ClamAV files? Well, this is the "hook" to make it co-operate with other programs.

- **/etc/freshclam.conf**

```
diff /etc/freshclam.conf.new /etc/freshclam.conf
9c9
< Example
---
> #Example
26c26
< #LogSyslog
---
> LogSyslog
31c31
< #LogFacility LOG_MAIL
---
> LogFacility LOG_MAIL
55c55
< #DatabaseMirror db.XY.clamav.net
---
> DatabaseMirror db.nl.clamav.net
```

  The `DatabaseMirror db.nl.clamav.net` line should be adapted for the ISO contry code of your own country (mine is **nl**).

- Finally, that common group I previously mentioned, to which both the user IDs *clamav* and *nobody* belong, is what is left to configure. To see what groups a user ID belongs to, you use the `id` command, like this:

```
id nobody
id clamav
```

  This shows easily enough that the two accounts have no common group. The following three commands add user *clamav* to the *nobody* and *nogroup* groups, and user *nobody* to the *clamav* group. Perhaps it is possible to leave out one of these additions, but at least it works this way.

```
gpasswd -a clamav nobody
gpasswd -a clamav nogroup
gpasswd -a nobody clamav
```

  Verify that this worked, by again running

```
id nobody
id clamav
```

The iptables rules

Now that we setup the content filter for the inspection of requested URLs and the retrieved web content, and the proxy to take care of the retrieval process, we still need to tell our computer that the outgoing web traffic should silently (transparently) be re-routed through this filter. This is where the Linux netfilter - aka the iptables firewall - comes into play.
What our iptables rules need to do is:

- intercept the web traffic (requests destined at Internet web servers, targeted at their http port which is port 80) just before it exits the computer, and instead redirect this traffic to our dansguardian filter which is listening at a local TCP port (127.0.0.1:8080)
- but only do this interception for users that we want to have restricted access. We define these as all local accounts except for root and any other privileged account you can think of (like, your own login account).

The basic NAT firewall rules that accomplish this are like this:

```
# Full access to the userid of the dansguardian and tinyproxy (==nobody),
and of freshclam (==clamav):
# Note that dansguardian needs to connect to tinyproxy at port 3128,
# tinyproxy needs to be able to connect to external servers at port 80 on
behalf of the web browsers,
# and freshclam needs to be able to fetch virus definition updates.
/usr/sbin/iptables -A OUTPUT -t nat -p tcp --dport 3128 -m owner --uid-owner
nobody -j ACCEPT
/usr/sbin/iptables -A OUTPUT -t nat -p tcp --dport 80   -m owner --uid-owner
nobody -j ACCEPT
/usr/sbin/iptables -A OUTPUT -t nat -p tcp --dport 80   -m owner --uid-owner
clamav -j ACCEPT
# Privileged user(s) will bypass the content filter:
PRIVUSERS="root alien"
for user in $PRIVUSERS; do
  /usr/sbin/iptables -A OUTPUT -t nat -p tcp --dport 80 -m owner --uid-owner
$user -j ACCEPT
done
# What comes next is the catch-all. Any user account not listed above
# (nobody, clamav and $PRIVUSERS) is forced through the content filter.
# Redirect requests for web pages (http traffic) to the dansguardian listen
port:
/usr/sbin/iptables -A OUTPUT -t nat -p tcp --dport 80 -j REDIRECT --to-ports
8080
# Also catch the sneaky bastards that try to bypass dansguardian:
/usr/sbin/iptables -A OUTPUT -t nat -p tcp --dport 3128 -j REDIRECT --to-
ports 8080
```

Note that these iptables rules populate the NAT table (NAT is Network Address Translation). NAT rules are what you would ordinarily think of when configuring a firewall/router which hides an internal network behind a single external network interface (the old name for NAT was masquerading although the two terms do not fully

compare). These rules would be added to the **PREROUTING** chain.
In our case of catching the local traffic, the NAT rules are not used to change the source or destination IP address (masquerading), but we change the destination TCP port of the packets that match the rules (redirection). The iptables rules in this case are instead being added to the **OUTPUT** chain.

A nice script that implements these rules, and which accepts a start and a stop parameter, is listed in the last section. You can save it as the file `/etc/rc.d/rc.firewall` and make it executable:

```
chmod +x /etc/rc.d/rc.firewall
```

Of course, if you already have a firewall script in that location (see my transparent proxy for a link on setting up a firewall) you can try and add the listing to that script. Or else you can save the listing to another file with a different name, say `/etc/rc.d/rc.redirect`, make that file executable. You can then add the line

```
/etc/rc.d/rc.redirect start
```

to the file `/etc/rc.d/rc.local` in front of the other lines you will be adding there - see the section on starting the programs

No limits for admins and parents

The message was hidden in the section on iptables, but I will repeat it in it's own section:

This setup will make *any* user account on your Linux computer subject to http content filtering, *except* for those user accounts that are listed in the variable PRIVUSERS. We defined PRIVUSERS in the firewall script (see the last section for it's listing). You need to add the user account names to that variable that you want to grant unfiltered Internet browsing. The definition of this variable in my example looked like this:

```
PRIVUSERS="root alien"
```

Starting the programs

We have assembled a series of scripts and programs that we should start somehow, so that they will provide the parental control we were setting up. They should be started when the computer boots, so that the protection is already active by the time our first computer user logs in.
This is done by adding these commands and scripts to the file `/etc/rc.d/rc.local`.

If you (built and) installed my Slackware package for dansguardian, it's rc script is installed non-executable by default. In order to run dansguardian on boot (as shown below) you will have to make the script executable by running

```
chmod +x /etc/rc.d/rc.dansguardian
```

If you optionally (built and) installed my Slackware package for clamav, it's rc script is installed non-executable by default. In order to run clamav on boot (as shown below) you will have to make the script executable by running

```
chmod +x /etc/rc.d/rc.clamav
```

If you configured your firewall rules in the file `/etc/rc.d/rc.firewall`, then this script will be detected by Slackware and automatically started with the `start` parameter on boot. This happens in the the Slackware init script `/etc/rc.d/rc.inet2` to be precise, like this:

```
if [ -x /etc/rc.d/rc.firewall ]; then
  /etc/rc.d/rc.firewall start
fi
```

so we don't have to worry about that. The important bit is the order in which tinyproxy and dansguardian are started: if dansguardian does not find a proxy service listening at the configured address, it will refuse to start.

To sum it all up, we add these lines to the file `/etc/rc.d/rc.local`:

```
# Start clamav
if [ -x /etc/rc.d/rc.clamav ]; then
  /etc/rc.d/rc.clamav start
fi

# Start tinyproxy
if [ -x /usr/sbin/tinyproxy ]; then
  /usr/sbin/tinyproxy > /dev/null 2>&1
fi

# Start dansguardian
if [ -x /etc/rc.d/rc.dansguardian ]; then
  /etc/rc.d/rc.dansguardian start
fi
```

These programs log their actions; to respectively `/var/log/maillog`, `/var/log/tinyproxy.log` and `/var/log/dansguardian/access.log`. If their logging is a bit too verbose to your taste (the default is to log quite a *lot*) you can turn the log levels down in the configuration files.

We're done!
Test your setup by alternately logging on as yourself and under one of the kids' accounts. Open a browser and try to access some pages you suspect might trigger a block or *access denied* from the content filter. If you plugged in ClamAV as virusscanner, try downloading this link: http://www.eicar.org/download/eicar.com.txt. The “EICAR test virus” is not a virus, but a specific string of characters that **all** anti-virus companies have agreed upon to signal as a virus. Since it is harmless, it is the perfect test case for the correct installation of a virus scanner. Clicking on the above link should trigger a block by dansguardian after it was told by clamav that you, the user, tried to download a virus.

Example configuration files

/etc/rc.d/rc.dansguardian:

```
#!/bin/sh
#
# Startup script for dansguardian
#
# processname: dansguardian
# pidfile: /var/run/dansguardian.pid
# config: /etc/dansguardian/dansguardian.conf

# File includes changes by Thomas Jarosch
function wait_for_pid()
{
    local PID=$1
    local RET=0
    if [ $PID -eq 0 ] ; then
        return $RET
    fi
    # give 60 secs then KILL
    local COUNTDOWN=60

    while [ -d /proc/${PID} ] && [ $COUNTDOWN -gt 0 ] ; do
        sleep 1
        COUNTDOWN=$[$COUNTDOWN-1]
    done

    if [ -d /proc/${PID} ]; then
        COMMAND=`ps h -o command ${PID}`
        logger "dansguardian: timeout waiting for PID ${PID}: ${COMMAND};
sending SIGKILL"
        kill -KILL $PID >/dev/null 2>&1
        RET=1
    fi
    return $RET
}

# See how we were called.

case "$1" in
start)
        if [ -f /usr/sbin/dansguardian ] &&
           [ -f /etc/dansguardian/dansguardian.conf ]; then
                echo -n "Starting dansguardian: "
                if /usr/sbin/dansguardian 2> /dev/null; then
                        echo " OK"
                else
                        echo " FAILED"
                fi
```

```
        fi
        ;;
stop)
        echo -n "Shutting down dansguardian: "
        WAITPID=0
        if [ -f /var/run/dansguardian.pid ] ; then
            WAITPID=`cat /var/run/dansguardian.pid`
        fi
        if /usr/sbin/dansguardian -q 2> /dev/null; then
                if wait_for_pid $WAITPID ; then
                    echo " OK"
                else
                    echo " FAILED"
                fi
                /bin/rm -f /var/run/dansguardian.pid
                /bin/rm -f /tmp/.dguardianipc
        else
                echo " FAILED"
        fi
        ;;
restart)
        $0 stop
        $0 start
        ;;
status)
        if [ -x /usr/sbin/dansguardian ]; then
                /usr/sbin/dansguardian -s
    else
        echo "Dansguardian is not installed!"
        fi
        ;;
*)
        echo "Usage: $0 {start|stop|restart|status}" >&2
        ;;
esac

exit 0
```

/etc/logrotate.d/dansguardian:

```
/var/log/dansguardian/access.log {
  rotate 4
  weekly
  sharedscripts
  prerotate
    /usr/sbin/dansguardian -q > /dev/null
    sleep 5
  endscript

  postrotate
    /usr/sbin/dansguardian > /dev/null
```

```
  endscript
}
```

/etc/rc.d/rc.clamav:

```
#!/bin/sh
# Start/stop/restart clamav.

# Set to '1' if you want milter support:
MILTER=0

# Start clamav:
clamav_start() {
  if [ -x /usr/sbin/clamd ]; then
    echo -n "Starting clamd daemon:  /usr/sbin/clamd "
    /usr/sbin/clamd
    echo "."
    # Give clamd a chance to create the socket
    sleep 1
    echo -n "Starting freshclam daemon:  /usr/bin/freshclam -d -l
/var/log/freshclam.log "
    /usr/bin/freshclam -d -l /var/log/freshclam.log
    echo "."
    if [ "$MILTER" == "1" ]; then
      echo -n "Starting clamav-milter daemon:  /usr/sbin/clamav-milter -dblo
--max-children=2 local:/var/run/clamav/clmilter.sock "
      /usr/sbin/clamav-milter -dblo --max-children=2
local:/var/run/clamav/clmilter.sock
      echo "."
    fi
  fi
}

# Stop clamav:
clamav_stop() {
  kill `cat /var/run/clamav/clamd.pid`
  #killall freshclam
  kill `cat /var/run/clamav/freshclam.pid`
  [ "$MILTER" == "1" ] && killall clamav-milter
}

# Restart clamav:
clamav_restart() {
  clamav_stop
  sleep 1
  clamav_start
}

case "$1" in
'start')
  clamav_start
```

```
  ;;
'stop')
  clamav_stop
  ;;
'restart')
  clamav_restart
  ;;
*)
  echo "usage $0 start|stop|restart"
esac
```

/etc/rc.d/rc.firewall:

```
#!/bin/sh
# Start/stop/restart our iptables rules for http traffic redirection.

# Privileged user(s) will be able to bypass the content filter:
PRIVUSERS="root alien"

# Start firewall:
start() {
  echo -n "Activating web traffic redirection "
  # Full access to the userid of the dansguardian and tinyproxy (==nobody),
and of freshclam (==clamav):
  # Note that dansguardian needs to connect to tinyproxy at port 3128,
  # tinyproxy needs to be able to connect to external servers at port 80 on
behalf of the web browsers,
  # and freshclam needs to be able to fetch virus definition updates.
  /usr/sbin/iptables -A OUTPUT -t nat -p tcp --dport 3128 -m owner --uid-
owner nobody -j ACCEPT
  /usr/sbin/iptables -A OUTPUT -t nat -p tcp --dport 80   -m owner --uid-
owner nobody -j ACCEPT
  /usr/sbin/iptables -A OUTPUT -t nat -p tcp --dport 80   -m owner --uid-
owner clamav -j ACCEPT
  for user in $PRIVUSERS; do
    /usr/sbin/iptables -A OUTPUT -t nat -p tcp --dport 80 -m owner --uid-
owner $user -j ACCEPT
  done
  # What comes next is the catch-all. Any user account not listed above
  # (nobody, clamav and $PRIVUSERS) is forced through the content filter.
  # Redirect requests for web pages (http traffic) to the dansguardian
listen port:
  /usr/sbin/iptables -A OUTPUT -t nat -p tcp --dport 80 -j REDIRECT --to-
ports 8080
  # Also catch the sneaky bastards that try to bypass dansguardian:
  /usr/sbin/iptables -A OUTPUT -t nat -p tcp --dport 3128 -j REDIRECT --to-
ports 8080
  echo "."
}

# Stop firewall:
```

```
stop() {
  echo -n "Stopping web traffic redirection "
  # Basically, a "-D" instead of the "-A" we had in the start function:
  /usr/sbin/iptables -D OUTPUT -t nat -p tcp --dport 3128 -m owner --uid-
owner nobody -j ACCEPT
  /usr/sbin/iptables -D OUTPUT -t nat -p tcp --dport 80   -m owner --uid-
owner nobody -j ACCEPT
  /usr/sbin/iptables -D OUTPUT -t nat -p tcp --dport 80   -m owner --uid-
owner clamav -j ACCEPT
  for user in $PRIVUSERS; do
    /usr/sbin/iptables -D OUTPUT -t nat -p tcp --dport 80 -m owner --uid-
owner $user -j ACCEPT
  done
  /usr/sbin/iptables -D OUTPUT -t nat -p tcp --dport 80 -j REDIRECT --to-
ports 8080
  /usr/sbin/iptables -D OUTPUT -t nat -p tcp --dport 3128 -j REDIRECT --to-
ports 8080
  echo "."
}

# Restart firewall:
restart() {
  stop
  start
}

case "$1" in
'start')
  start
  ;;
'stop')
  stop
  ;;
'restart')
  restart
  ;;
*)
  echo "usage $0 start|stop|restart"
esac
```