Configuring SRS with Exim (Debian and Ubuntu)

Introduction

filesender is designed to send mails **on behalf of** an authenticated user but will by definition send those mails from a server outside the administrative domain of those authenticated users. This will introduce problems with receving sites using very strict anti-spam measures (i.e. strict SPF checking). This document describes one possible solution that can be implemented independent of filesender and is based on having the MTA using the Sender Rewrite Scheme as described on http://www.libsrs2.org/

Requirements

- Debian (Lenny) or Ubuntu server
- exim4 installed as MTA (default on Lenny)
- server (MTA) is configured to accept and handle **incoming** mail (either directly or through external spamfiltering solutions)

Installation

All steps need to be done as root.

Install and configure SRS

Install the required srs package:

```
apt-get install srs
```

Create a startup script for the srsd daemon:

```
vi /etc/init.d/srsd
```

with the following content:

```
#! /bin/sh

### BEGIN INIT INFO
# Provides:          srsd
# Required-Start:
# Required-Stop:
# Should-Start:
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: SRS daemon
# Description:       SRS daemon
### END INIT INFO

set -e

# /etc/init.d/srsd: start and stop the srsd daemon

DAEMON=/usr/bin/srsd
USER=Debian-exim
SECRETFILE=/etc/srsd.secret
PIDFILE=/var/run/srsd.pid
SOCKETFILE=/tmp/srsd
SRSD_OPTS="--secretfile ${SECRETFILE}"

test -x $DAEMON || exit 0

. /lib/lsb/init-functions

srsd_start() {
    if start-stop-daemon --start --quiet --background \
        --chuid $USER \
        --pidfile $PIDFILE --make-pidfile \
        --exec $DAEMON \
        -- $SRSD_OPTS
    then
        rc=0
        sleep 1
        if ! kill -0 $(cat $PIDFILE) >/dev/null 2>&1; then
            log_failure_msg "srsd daemon failed to start"
            rc=1
        fi
    else
        rc=1
    fi
    if [ $rc -eq 0 ]; then
        log_end_msg 0
    else
        log_end_msg 1
        rm -f $PIDFILE
    fi
} # srsd_start


case "$1" in
  start)
    log_daemon_msg "Starting srsd daemon" "srsd"
    if [ -s $PIDFILE ] && kill -0 $(cat $PIDFILE) >/dev/null 2>&1; then
        log_progress_msg "apparently already running"
        log_end_msg 0
        exit 0
    fi
        srsd_start
    ;;
  stop)
    log_daemon_msg "Stopping srsd daemon" "srsd"
    start-stop-daemon --stop --quiet --oknodo --pidfile $PIDFILE
    log_end_msg $?
    rm -f $PIDFILE
    rm -f $SOCKETFILE
    ;;

  restart)
    set +e
    log_daemon_msg "Restarting srsd daemon" "srsd"
    if [ -s $PIDFILE ] && kill -0 $(cat $PIDFILE) >/dev/null 2>&1; then
        start-stop-daemon --stop --quiet --oknodo --pidfile $PIDFILE || true
        sleep 1
    else
        log_warning_msg "srsd daemon not running, attempting to start."
            rm -f $PIDFILE
    fi
        srsd_start
    ;;

  status)
    status_of_proc -p $PIDFILE "$DAEMON" srsd
    exit $?    # notreached due to set -e
    ;;
  *)
    echo "Usage: /etc/init.d/srsd {start|stop|restart|status}"
    exit 1
esac

exit 0
```

Configure the startup runlevels (default startup at boot time):

```
chmod 755 /etc/init.d/srsd
update-rc.d srsd defaults
```

Create a 'secrets' file with a random generated secret to use with srsd:

```
touch /etc/srsd.secret
chown Debian-exim /etc/srsd.secret
chmod 600 /etc/srsd.secret
openssl rand -base64 12 > /etc/srsd.secret
```

Start the SRS daemon:

```
invoke-rc.d srsd start
```

Configure exim4

There are various ways exim4 on Debian can be configured. The description below is based on the so called 'split configuration' method but can be easily adapted to other methods. Please have a look at section 2.1 of the README.Debian:

```
zmore /usr/share/doc/exim4/README.Debian.gz
```

The actual method of configuration and some of the settings below are usually set when installing exim4. They can be set/changed with:

```
dpkg-reconfigure exim4-config
```

Step 0: make sure exim is configured to accept incoming mail. Within the debconf scheme this should be one of the following modes:

```
2.1.1.1.1.  internet site; mail is sent and received directly using SMTP
2.1.1.1.2.  mail sent by smarthost; received via SMTP or fetchmail
```

Step 1: add additional routers, these should be **before** the dnslookup router:

```
vi /etc/exim4/conf.d/router/175_exim4-config_srs
```

with the following content:

```
srs_bounce:
  debug_print = "R: srs_bounce for $local_part@$domain"
  driver = redirect
  allow_fail
  allow_defer
  domains = $primary_hostname
  local_part_prefix = srs0+ : srs0- : srs0= : srs1+ : srs1- : srs1=
  caseful_local_part
  address_data = ${readsocket{/tmp/srsd}{REVERSE $local_part_prefix$local_part@$domain}{5s}{\n}{:defer: SRS daemon failure}}
  data = ${if match{$address_data}{^ERROR}{:fail: Invalid SRS address}{$address_data}}


srs_forward:
  debug_print = "R: srs_forward for $local_part@$domain"
  no_verify
  senders = ! : ! *@+local_domains
  address_data = ${readsocket{/tmp/srsd}\
              {FORWARD $sender_address_local_part@$sender_address_domain $primary_hostname\n}\
                                    {5s}{\n}{:defer: SRS daemon failure}}
  errors_to = ${quote_local_part:${local_part:$address_data}}@${domain:$address_data}
  headers_add = "X-SRS: Sender address rewritten from <$sender_address> to <${quote_local_part:${local_part:$address_data}}@${domain:$address_data}> by $primary_hostname."
  driver = redirect
  repeat_use = false
  allow_defer
  data = ${quote_local_part:$local_part}@$domain
```

Step 2: add some small modifications to the macro-definitions used:

```
vi /etc/exim4/conf.d/main/000_localmacros
```

with the following content:

```
CHECK_RCPT_LOCAL_LOCALPARTS = ^[.] : ^.*[@%!|`#&?]
MAIN_LOG_SELECTOR = +tls_peerdn +address_rewrite +return_path_on_delivery +sender_on_delivery +smtp_confirmation +smtp_connection +smtp_incomplete_transaction +smtp_no_mail
```

The above macros ensure that the '/' character (might be used with generated SRS addresses) is allowed and will take care of the logging of all rewrites (and some more).

Step 3: restart exim. If exim is configured to use the split configuration this will automatically generate a new configuration. If you are using one of the other configuration methods please have a look at

```
man update-exim4.conf.template
```

on how to generate a single exim4 configuration template from the above 'split configuration files'.

```
invoke-rc.d exim4 restart
```

You're done...

How does it work?

Afer installing and configuring the above all incoming and outgoing mail will be checked whether the **envelope** sender and recipient addresses should be rewritten or not. For outgoing mails the sender address is rewritten only when the original sender is outside the local domain(s) of the server. For incoming mails the recipient address will be checked whether it is a valid SRS address (i.e. it was generated on the server based on the secret hash) and is not expired yet (default expiry is 49 days).

exim hands over addresses to be rewritten or validated to the srsd-daemon through a socket in

/tmp/srsd. The srsd daemon then does the rewriting or validation.

The generated rewritten address is based on the original addres, the day the address is generated and the first secret in the /etc/srsd.secret file. Validation is done using all secrets in the secrets file (one per line) and is checked for expiry.

The expiry time (in days) is 'hard coded' in the Mail::SRS Perl module (/usr/share/perl5/Mail/SRS.pm). If you need to change the default you have to edit the SRS.pm file, for example (making the expiry 14 days):

```
        use Mail::SRS;
        my $srs = new Mail::SRS(
                Secret     => [ .... ],    # scalar or array
                MaxAge     => 14,          # days
                HashLength => 4,           # base64 characters: 4 x 6bits
                HashMin    => 4,           # base64 characters
                        );
        my $srsaddress = $srs->forward($sender, $alias);
        my $sender = $srs->reverse($srsaddress);
```

SRS uses a stateless mechanism to generate and validate SRS-addresses so there is no need for storing/saving the addresses.

Known problems

- NDR's (bounces) from remote MTA's are sent to the rewritten address (also in the RFC2822 To: field). This will work and the NDR will be sent to the original sender but this might confuse both users and draconian spamfilters. Needs investigation.

Security considerations

- The srsd daemon runs with the priviliges of the Debian-exim user and should only get properly sanitised addresses from Exim to rewrite/validate.
- the requirement to accept and handle incoming mail opens up an extra port exposed to the outside world. You can minimise the additional risks by putting the MTA on your FileSender box behind a dedicated mail handling server, i.e. both incoming and outgoing mail are sent through another (dedicated) mail server. Access to the SMTP port on your FileSender server can then be restricted to accept only connections from the dedicated mail server.
- As always, keep your server up to date with the latest security patches, as with all software there is a chance of exploitable bugs but the Debian and Exim community have a good track record with implementing security fixes.

Acknowledgements

- Paul Dekkers @ SURFnet for creating a working exim4 config with stock Debian packages based on the not always working information floating around on the Internet.
- Peter Thomassen for improvements and bug fixes.