Chapter

11 Network Services: Postfix, Apache, NFS, Samba, Squid, LDAP, SIP, XMPP, TURN

Contents

Network services are the programs that users interact with directly in their daily work. They are the tip of the information system iceberg, and this chapter focuses on them; the hidden parts they rely on are the infrastructure we already described.
Many modern network services require encryption technology to operate reliably and securely, especially when used on the public Internet. X.509 Certificates (which may also be referred to as SSL Certificates or TLS Certificates) are frequently used for this purpose. A certificate for a specific domain can often be shared between more than one of the services discussed in this chapter.

11.1. Mail Server

The Falcot Corp administrators selected Postfix for the electronic mail server, due to its reliability and its ease of configuration. Indeed, its design enforces that each task is implemented in a process with the minimum set of required permissions, which is a great mitigation measure against security problems.

ALTERNATIVE
The Exim4 server

Debian uses Exim4 as the default email server (which is why the initial installation includes Exim4). The configuration is provided by a separate package, *exim4-config*, and automatically customized based on the answers to a set of Debconf questions very similar to the questions asked by the *postfix* package.

The configuration can be either in one single file (`/etc/exim4/exim4.conf.template`) or split across a number of configuration snippets stored under `/etc/exim4/conf.d/`. In both cases, the files are used by `update-exim4.conf` as templates to generate `/var/lib/exim4/config.autogenerated`. The latter is the file used by Exim4. Thanks to this mechanism, values obtained through Exim's debconf configuration — which are stored in `/etc/exim4/update-exim4.conf.conf` — can be injected in Exim's configuration file, even when the administrator or another package has altered the default Exim configuration.

The Exim4 configuration file syntax has its peculiarities and its learning curve; however, once these peculiarities are understood, Exim4 is a very complete and powerful email server, as evidenced by the tens of pages of documentation.

➨ `http://www.exim.org/docs.html`

11.1.1. Installing Postfix

The *postfix* package includes the main SMTP daemon. Other packages (such as *postfix-ldap* and *postfix-pgsql*) add extra functionality to Postfix, including access to mapping databases. You should only install them if you know that you need them.

BACK TO BASICS
SMTP

SMTP (*Simple Mail Transfer Protocol*) is the protocol used by mail servers to exchange and route emails.

Several Debconf questions are asked during the installation of the package. The answers allow generating a first version of the `/etc/postfix/main.cf` configuration file.

The first question deals with the type of setup. Only two of the proposed answers are relevant in case of an Internet-connected server, "Internet site" and "Internet with smarthost". The former is appropriate for a server that receives incoming email and sends outgoing email directly to its recipients, and is therefore well-adapted to the Falcot Corp case. The latter is appropriate for a server receiving incoming email normally, but that sends outgoing email through an intermediate SMTP server — the "smarthost" — rather than directly to the recipient's server. This is mostly useful for individuals with a dynamic IP address, since many email servers reject messages coming straight from such an IP address. In this case, the smarthost will usually be

the ISP's SMTP server, which is always configured to accept email coming from the ISP's customers and forward it appropriately. This setup (with a smarthost) is also relevant for servers that are not permanently connected to the internet, since it avoids having to manage a queue of undeliverable messages that need to be retried later.

VOCABULARY
ISP
ISP is the acronym for "Internet Service Provider". It covers an entity, often a commercial company, that provides Internet connections and the associated basic services (email, news and so on).

The second question deals with the full name of the machine, used to generate email addresses from a local user name; the full name of the machine ends up as the part after the at-sign ("@"). In the case of Falcot, the answer should be mail.falcot.com. This is the only question asked by default, but the configuration it leads to is not complete enough for the needs of Falcot, which is why the administrators run `dpkg-reconfigure postfix` so as to be able to customize more parameters.

One of the extra questions asks for all the domain names related to this machine. The default list includes its full name as well as a few synonyms for localhost, but the main falcot.com domain needs to be added by hand. More generally, this question should usually be answered with all the domain names for which this machine should serve as an MX server; in other words, all the domain names for which the DNS says that this machine will accept email. This information ends up in the mydestination variable of the main Postfix configuration file — `/etc/postfix/main.cf`.

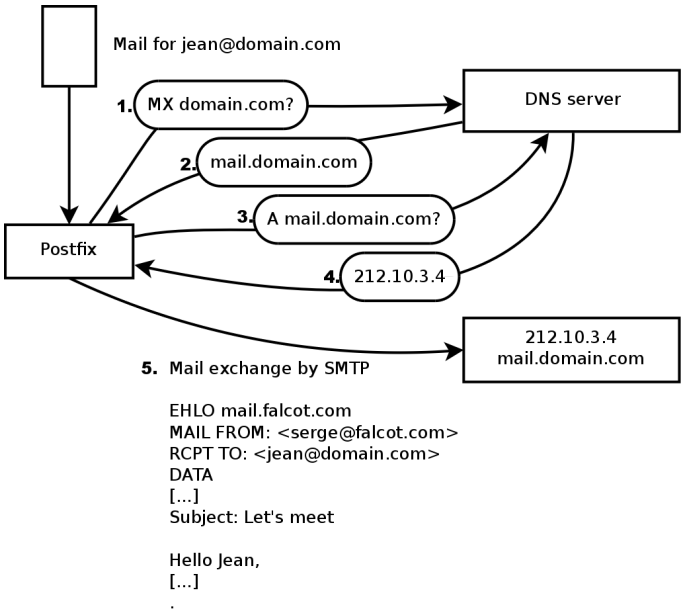

Figure 11.1 *Role of the DNS* MX *record while sending a mail*

EXTRA

Querying the MX records

When the DNS does not have an MX record for a domain, the email server will try sending the messages to the host itself, by using the matching A record (or AAAA in IPv6).

In some cases, the installation can also ask what networks should be allowed to send email via the machine. In its default configuration, Postfix only accepts emails coming from the machine itself; the local network will usually be added. The Falcot Corp administrators added 192.168.0.0/16 to the default answer. If the question is not asked, the relevant variable in the configuration file is mynetworks, as seen in the example below.

Local email can also be delivered through `procmail`. This tool allows users to sort their incoming email according to rules stored in their `~/.procmailrc` file.

After this first step, the administrators got the following configuration file; it will be used as a starting point for adding some extra functionality in the next sections.

Example 11.1 *Initial `/etc/postfix/main.cf` file*

```
# See /usr/share/postfix/main.cf.dist for a commented, more complete version


# Debian specific:  Specifying a file name will cause the first
# line of that file to be used as the name.  The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

smtpd_banner = $myhostname ESMTP $mail_name (Debian/GNU)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_use_tls=yes
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package for
# information on enabling SSL in the smtp client.
```

```
smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated
       defer_unauth_destination
myhostname = mail.falcot.com
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = mail.falcot.com, falcot.com, localhost.localdomain, localhost
relayhost =
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128 192.168.0.0/16
mailbox_command = procmail -a "$EXTENSION"
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
inet_protocols = all
```

SECURITY

***Snake oil* SSL certificates**

The *snake oil* certificates, like the *snake oil* "medicine" sold by unscrupulous quacks in old times, have absolutely no value: you cannot rely on them to authenticate the server since they are automatically generated self-signed certificates. However they are useful to improve the privacy of the exchanges.

In general they should only be used for testing purposes, and normal service must use real certificates; these can be generated with the procedure described in section 10.2.1.1, "Public Key Infrastructure: *easy-rsa*" page 224.

11.1.2. Configuring Virtual Domains

The mail server can receive emails addressed to other domains besides the main domain; these are then known as virtual domains. In most cases where this happens, the emails are not ultimately destined to local users. Postfix provides two interesting features for handling virtual domains.

CAUTION

Virtual domains and canonical domains

None of the virtual domains must be referenced in the `mydestination` variable; this variable only contains the names of the "canonical" domains directly associated to the machine and its local users.

Virtual Alias Domains

A virtual alias domain only contains aliases, i.e. addresses that only forward emails to other addresses.

Such a domain is enabled by adding its name to the virtual_alias_domains variable, and referencing an address mapping file in the virtual_alias_maps variable.

Example 11.2 *Directives to add in the /etc/postfix/main.cf file*

```
virtual_alias_domains = falcotsbrand.com
virtual_alias_maps = hash:/etc/postfix/virtual
```

The /etc/postfix/virtual file describes a mapping with a rather straightforward syntax: each line contains two fields separated by whitespace; the first field is the alias name, the second field is a list of email addresses where it redirects. The special @domain.com syntax covers all remaining aliases in a domain.

Example 11.3 *Example /etc/postfix/virtual file*

```
webmaster@falcotsbrand.com  jean@falcot.com
contact@falcotsbrand.com    laure@falcot.com, sophie@falcot.com
# The alias below is generic and covers all addresses within
# the falcotsbrand.com domain not otherwise covered by this file.
# These addresses forward email to the same user name in the
# falcot.com domain.
@falcotsbrand.com           @falcot.com
```

Virtual Mailbox Domains

CAUTION

Combined virtual domain?

Postfix does not allow using the same domain in both virtual_alias_domains and virtual_mailbox_domains. However, every domain of virtual_mailbox_domains is implicitly included in virtual_alias_domains, which makes it possible to mix aliases and mailboxes within a virtual domain.

Messages addressed to a virtual mailbox domain are stored in mailboxes not assigned to a local system user.

Enabling a virtual mailbox domain requires naming this domain in the virtual_mailbox_domains variable, and referencing a mailbox mapping file in virtual_mailbox_maps. The virtual_mailbox_base parameter contains the directory under which the mailboxes will be stored.

The virtual_uid_maps parameter (respectively virtual_gid_maps) references the file containing the mapping between the email address and the system user (respectively group) that "owns" the corresponding mailbox. To get all mailboxes owned by the same owner/group, the static:5000 syntax assigns a fixed UID/GID (of value 5000 here).

Example 11.4 *Directives to add in the `/etc/postfix/main.cf` file*

```
virtual_mailbox_domains = falcot.org
virtual_mailbox_maps = hash:/etc/postfix/vmailbox
virtual_mailbox_base = /var/mail/vhosts
```

Again, the syntax of the `/etc/postfix/vmailbox` file is quite straightforward: two fields separated with whitespace. The first field is an email address within one of the virtual domains, and the second field is the location of the associated mailbox (relative to the directory specified in *virtual_mailbox_base*). If the mailbox name ends with a slash (/), the emails will be stored in the *maildir* format; otherwise, the traditional *mbox* format will be used. The *maildir* format uses a whole directory to store a mailbox, each individual message being stored in a separate file. In the *mbox* format, on the other hand, the whole mailbox is stored in one file, and each line starting with "From " (From followed by a space) signals the start of a new message.

Example 11.5 *The `/etc/postfix/vmailbox` file*

```
# Jean's email is stored as maildir, with
# one file per email in a dedicated directory
jean@falcot.org falcot.org/jean/
# Sophie's email is stored in a traditional "mbox" file,
# with all mails concatenated into one single file
sophie@falcot.org falcot.org/sophie
```

11.1.3. Restrictions for Receiving and Sending

The growing number of unsolicited bulk emails (*spam*) requires being increasingly strict when deciding which emails a server should accept. This section presents some of the strategies included in Postfix.

CULTURE

The spam problem

"Spam" is a generic term used to designate all the unsolicited commercial emails (also known as UCEs) that flood our electronic mailboxes; the unscrupulous individuals sending them are known as spammers. They care little about the nuisance they cause, since sending an email costs very little, and only a very small percentage of recipients need to be attracted by the offers for the spamming operation to make more money than it costs. The process is mostly automated, and any email address made public (for instance, on a web forum, or on the archives of a mailing list, or on a blog, and so on) will be discovered by the spammers' robots, and subjected to a never-ending stream of unsolicited messages.

All system administrators try to face this nuisance with spam filters, but of course spammers keep adjusting to try to work around these filters. Some even rent networks of machines compromised by a worm from various crime syndicates. Recent statistics estimate that up to 95% of all emails circulating on the Internet are spam

IP-Based Access Restrictions

The smtpd_client_restrictions directive controls which machines are allowed to communicate with the email server.

Example 11.6 *Restrictions Based on Client Address*

```
smtpd_client_restrictions = permit_mynetworks,
    warn_if_reject reject_unknown_client,
    check_client_access hash:/etc/postfix/access_clientip,
    reject_rbl_client sbl-xbl.spamhaus.org,
    reject_rbl_client list.dsbl.org
```

When a variable contains a list of rules, as in the example above, these rules are evaluated in order, from the first to the last. Each rule can accept the message, reject it, or leave the decision to a following rule. As a consequence, order matters, and simply switching two rules can lead to a widely different behavior.

The permit_mynetworks directive, used as the first rule, accepts all emails coming from a machine in the local network (as defined by the *mynetworks* configuration variable).

The second directive would normally reject emails coming from machines without a completely valid DNS configuration. Such a valid configuration means that the IP address can be resolved to a name, and that this name, in turn, resolves to the IP address. This restriction is often too strict, since many email servers do not have a reverse DNS for their IP address. This explains why the Falcot administrators prepended the warn_if_reject modifier to the reject_unknown_client directive: this modifier turns the rejection into a simple warning recorded in the logs. The administrators can then keep an eye on the number of messages that would be rejected if the rule were actually enforced, and make an informed decision later if they wish to enable such enforcement.

TIP

***access* tables**

The restriction criteria include administrator-modifiable tables listing combinations of senders, IP addresses, and allowed or forbidden hostnames. These tables can be created from an uncompressed copy of the `/usr/share/doc/postfix-doc/examples/access.gz` file. This model is self-documented in its comments, which means each table describes its own syntax.

The `/etc/postfix/access_clientip` table lists IP addresses and networks; `/etc/postfix/access_helo` lists domain names; `/etc/postfix/access_sender` contains sender email addresses. All these files need to be turned into hash-tables (a format optimized for fast access) after each change, with the `postmap /etc/postfix/`*`file`* command.

The third directive allows the administrator to set up a blacklist and a whitelist of email servers, stored in the `/etc/postfix/access_clientip` file. Servers in the whitelist are considered as trusted, and the emails coming from there therefore do not go through the following filtering rules.

The last two rules reject any message coming from a server listed in one of the indicated blacklists. RBL is an acronym for *Remote Black List*; there are several such lists, but they all list badly configured servers that spammers use to relay their emails, as well as unexpected mail relays such as machines infected with worms or viruses.

TIP

White list and RBLs

Blacklists sometimes include a legitimate server that has been suffering an incident. In these situations, all emails coming from one of these servers would be rejected unless the server is listed in a whitelist defined by `/etc/postfix/access_clientip`.

Prudence therefore recommends including in the whitelist all the trusted servers from which many emails are usually received.

Checking the Validity of the EHLO or HELO Commands

Each SMTP exchange starts with a HELO (or EHLO) command, followed by the name of the sending email server; checking the validity of this name can be interesting.

Example 11.7 *Restrictions on the name announced in EHLO*

```
smtpd_helo_restrictions = permit_mynetworks,
    reject_invalid_hostname,
    check_helo_access hash:/etc/postfix/access_helo,
    reject_non_fqdn_hostname,
    warn_if_reject reject_unknown_hostname
```

The first permit_mynetworks directive allows all machines on the local network to introduce themselves freely. This is important, because some email programs do not respect this part of the SMTP protocol adequately enough, and they can introduce themselves with nonsensical names.

The reject_invalid_hostname rule rejects emails when the EHLO announce lists a syntactically incorrect hostname. The reject_non_fqdn_hostname rule rejects messages when the announced hostname is not a fully-qualified domain name (including a domain name as well as a host name). The reject_unknown_hostname rule rejects messages if the announced name does not exist in the DNS. Since this last rule unfortunately leads to too many rejections, the administrators turned its effect to a simple warning with the warn_if_reject modifier as a first step; they may decide to remove this modifier at a later stage, after auditing the results of this rule.

Using permit_mynetworks as the first rule has an interesting side effect: the following rules only apply to hosts outside the local network. This allows blacklisting all hosts that announce themselves as part of the falcot.com, for instance by adding a falcot.com REJECT You are not in our network line to the `/etc/postfix/access_helo` file.

Accepting or Refusing Based on the Announced Sender

Every message has a sender, announced by the MAIL FROM command of the SMTP protocol; again, this information can be validated in several different ways.

Example 11.8 *Sender checks*

```
smtpd_sender_restrictions =
    check_sender_access hash:/etc/postfix/access_sender,
    reject_unknown_sender_domain, reject_unlisted_sender,
    reject_non_fqdn_sender
```

The `/etc/postfix/access_sender` table maps some special treatment to some senders. This usually means listing some senders into a white list or a black list.

The reject_unknown_sender_domain rule requires a valid sender domain, since it is needed for a valid address. The reject_unlisted_sender rule rejects local senders if the address does not exist; this prevents emails from being sent from an invalid address in the falcot.com domain, and messages emanating from joe.bloggs@falcot.com are only accepted if such an address really exists.

Finally, the reject_non_fqdn_sender rule rejects emails purporting to come from addresses without a fully-qualified domain name. In practice, this means rejecting emails coming from user@machine: the address must be announced as either user@machine.example.com or user@example.com.

Accepting or Refusing Based on the Recipient

Each email has at least one recipient, announced with the RCPT TO command in the SMTP protocol. These addresses also warrant validation, even if that may be less relevant than the checks made on the sender address.

Example 11.9 *Recipient checks*

```
smtpd_recipient_restrictions = permit_mynetworks,
    reject_unauth_destination, reject_unlisted_recipient,
    reject_non_fqdn_recipient
```

reject_unauth_destination is the basic rule that requires outside messages to be addressed to us; messages sent to an address not served by this server are rejected. Without this rule, a server becomes an open relay that allows spammers to send unsolicited emails; this rule is therefore mandatory, and it will be best included near the beginning of the list, so that no other rules may authorize the message before its destination has been checked.

The reject_unlisted_recipient rule rejects messages sent to non-existing local users, which makes sense. Finally, the reject_non_fqdn_recipient rule rejects non-fully-qualified addresses; this makes it impossible to send an email to jean or jean@machine, and requires using the full address instead, such as jean@machine.falcot.com or jean@falcot.com.

Restrictions Associated with the DATA Command

The DATA command of SMTP is emitted before the contents of the message. It doesn't provide any information per se, apart from announcing what comes next. It can still be subjected to checks.

Example 11.10 *DATA checks*

```
smtpd_data_restrictions = reject_unauth_pipelining
```

The reject_unauth_pipelining directives causes the message to be rejected if the sending party sends a command before the reply to the previous command has been sent. This guards against a common optimization used by spammer robots, since they usually don't care a fig about replies and only focus on sending as many emails as possible in as short a time as possible.

Applying Restrictions

Although the above commands validate information at various stages of the SMTP exchange, Postfix only sends the actual rejection as a reply to the RCPT TO command.

This means that even if the message is rejected due to an invalid EHLO command, Postfix knows the sender and the recipient when announcing the rejection. It can then log a more explicit message than it could if the transaction had been interrupted from the start. In addition, a number of SMTP clients do not expect failures on the early SMTP commands, and these clients will be less disturbed by this late rejection.

A final advantage to this choice is that the rules can accumulate information during the various stages of the SMTP exchange; this allows defining more fine-grained permissions, such as rejecting a non-local connection if it announces itself with a local sender.

Filtering Based on the Message Contents

The validation and restriction system would not be complete without a way to apply checks to the message contents. Postfix differentiates the checks applying to the email headers from those applying to the email body.

Example 11.11 *Enabling content-based filters*

```
header_checks = regexp:/etc/postfix/header_checks
body_checks = regexp:/etc/postfix/body_checks
```

Both files contain a list of regular expressions (commonly known as *regexps* or *regexes*) and associated actions to be triggered when the email headers (or body) match the expression.

QUICK LOOK

Regexp tables

The `/usr/share/doc/postfix-doc/examples/header_checks.gz` file contains many explanatory comments and can be used as a starting point for creating the `/etc/postfix/header_checks` and `/etc/postfix/body_checks` files.

Example 11.12 *Example `/etc/postfix/header_checks` file*

```
/^X-Mailer: GOTO Sarbacane/ REJECT I fight spam (GOTO Sarbacane)
/^Subject: *Your email contains VIRUSES/ DISCARD virus notification
```

BACK TO BASICS

Regular expression

The *regular expression* term (shortened to *regexp* or *regex*) references a generic notation for expressing a description of the contents and/or structure of a string of characters. Certain special characters allow defining alternatives (for instance, `foo|bar` matches either "foo" or "bar"), sets of allowed characters (for instance, `[0-9]` means any digit, and `.` — a dot — means any character), quantifications (`s?` matches either `s` or the empty string, in other words 0 or 1 occurrence of `s`; `s+` matches one or more consecutive `s` characters; and so on). Parentheses allow grouping search results.

The precise syntax of these expressions varies across the tools using them, but the basic features are similar.

➨ http://en.wikipedia.org/wiki/Regular_expression

The first one checks the header mentioning the email software; if GOTO Sarbacane (a bulk email software) is found, the message is rejected. The second expression controls the message subject; if it mentions a virus notification, we can decide not to reject the message but to discard it immediately instead.

Using these filters is a double-edged sword, because it is easy to make the rules too generic and to lose legitimate emails as a consequence. In these cases, not only the messages will be lost, but their senders will get unwanted (and annoying) error messages.

11.1.4. Setting Up *greylisting*

"Greylisting" is a filtering technique according to which a message is initially rejected with a temporary error code, and only accepted on a further try after some delay. This filtering is

particularly efficient against spam sent by the many machines infected by worms and viruses, since this software rarely acts as a full SMTP agent (by checking the error code and retrying failed messages later), especially since many of the harvested addresses are really invalid and retrying would only mean losing time.

Postfix doesn't provide greylisting natively, but there is a feature by which the decision to accept or reject a given message can be delegated to an external program. The *postgrey* package contains just such a program, designed to interface with this access policy delegation service.

Once *postgrey* is installed, it runs as a daemon and listens on port 10023. Postfix can then be configured to use it, by adding the check_policy_service parameter as an extra restriction:

```
smtpd_recipient_restrictions = permit_mynetworks,
    [...]
    check_policy_service inet:127.0.0.1:10023
```

Each time Postfix reaches this rule in the ruleset, it will connect to the `postgrey` daemon and send it information concerning the relevant message. On its side, Postgrey considers the IP address/sender/recipient triplet and checks in its database whether that same triplet has been seen recently. If so, Postgrey replies that the message should be accepted; if not, the reply indicates that the message should be temporarily rejected, and the triplet gets recorded in the database.

The main disadvantage of greylisting is that legitimate messages get delayed, which is not always acceptable. It also increases the burden on servers that send many legitimate emails.

IN PRACTICE

Shortcomings of greylisting

Theoretically, greylisting should only delay the first mail from a given sender to a given recipient, and the typical delay is in the order of minutes. Reality, however, can differ slightly. Some large ISPs use clusters of SMTP servers, and when a message is initially rejected, the server that retries the transmission may not be the same as the initial one. When that happens, the second server gets a temporary error message due to greylisting too, and so on; it may take several hours until transmission is attempted by a server that has already been involved, since SMTP servers usually increase the delay between retries at each failure.

As a consequence, the incoming IP address may vary in time even for a single sender. But it goes further: even the sender address can change. For instance, many mailing-list servers encode extra information in the sender address so as to be able to handle error messages (known as *bounces*). Each new message sent to a mailing-list may then need to go through greylisting, which means it has to be stored (temporarily) on the sender's server. For very large mailing-lists (with tens of thousands of subscribers), this can soon become a problem.

To mitigate these drawbacks, Postgrey manages a whitelist of such sites, and messages emanating from them are immediately accepted without going through greylisting. This list can easily be adapted to local needs, since it is stored in the `/etc/postgrey/whitelist_clients` file.

GOING FURTHER

Selective greylisting with *milter-greylist*

The drawbacks of greylisting can be mitigated by only using greylisting on the subset of clients that are already considered as probable sources of spam (because they are listed in a DNS blacklist). This is not possible with *postgrey* but *milter-greylist* can be used in such a way.

In that scenario, since DNS blacklists never triggers a definitive rejection, it becomes reasonable to use aggressive blacklists, including those listing all dynamic IP addresses from ISP clients (such as `pbl.spamhaus.org` or `dul.dnsbl.sorbs.net`).

Since milter-greylist uses Sendmail's milter interface, the postfix side of its configuration is limited to "`smtpd_milters =unix:/var/run/milter-greylist/milter-greylist.sock`". The `greylist.conf(5)` manual page documents `/etc/milter-greylist/greylist.conf` and the numerous ways to configure milter-greylist. You will also have to edit `/etc/default/milter-greylist` to actually enable the service.

11.1.5. Customizing Filters Based On the Recipient

section 11.1.3, "Restrictions for Receiving and Sending" page 257 and section 11.1.4, "Setting Up *greylisting*" page 262 reviewed many of the possible restrictions. They all have their use in limiting the amount of received spam, but they also all have their drawbacks. It is therefore more and more common to customize the set of filters depending on the recipient. At Falcot Corp, greylisting is interesting for most users, but it hinders the work of some users who need low latency in their emails (such as the technical support service). Similarly, the commercial service sometimes has problems receiving emails from some Asian providers who may be listed in blacklists; this service asked for a non-filtered address so as to be able to correspond.

Postfix provides such a customization of filters with a "restriction class" concept. The classes are declared in the smtpd_restriction_classes parameter, and defined the same way as smtpd_recipient_restrictions. The check_recipient_access directive then defines a table mapping a given recipient to the appropriate set of restrictions.

Example 11.13 *Defining restriction classes in `main.cf`*

```
smtpd_restriction_classes = greylisting, aggressive, permissive

greylisting = check_policy_service inet:127.0.0.1:10023
aggressive = reject_rbl_client sbl-xbl.spamhaus.org,
        check_policy_service inet:127.0.0.1:10023
permissive = permit

smtpd_recipient_restrictions = permit_mynetworks,
        reject_unauth_destination,
        check_recipient_access hash:/etc/postfix/recipient_access
```

Example 11.14 *The /etc/postfix/recipient_access file*

```
# Unfiltered addresses
postmaster@falcot.com  permissive
support@falcot.com     permissive
sales-asia@falcot.com  permissive

# Aggressive filtering for some privileged users
joe@falcot.com         aggressive

# Special rule for the mailing-list manager
sympa@falcot.com       reject_unverified_sender

# Greylisting by default
falcot.com             greylisting
```

11.1.6. Integrating an Antivirus

The many viruses circulating as attachments to emails make it important to set up an antivirus at the entry point of the company network, since despite an awareness campaign, some users will still open attachments from obviously shady messages.

The Falcot administrators selected `clamav` for their free antivirus. The main package is *clamav*, but they also installed a few extra packages such as *arj*, *unzoo*, *unrar* and *lha*, since they are required for the antivirus to analyze attachments archived in one of these formats.

The task of interfacing between antivirus and the email server goes to `clamav-milter`. A *milter* (short for *mail filter*) is a filtering program specially designed to interface with email servers. A milter uses a standard application programming interface (API) that provides much better performance than filters external to the email servers. Milters were initially introduced by *Sendmail*, but *Postfix* soon followed suit.

QUICK LOOK

A milter for Spamassassin

The *spamass-milter* package provides a milter based on *SpamAssassin*, the famous unsolicited email detector. It can be used to flag messages as probable spams (by adding an extra header) and/or to reject the messages altogether if their "spamminess" score goes beyond a given threshold.

Once the *clamav-milter* package is installed, the milter should be reconfigured to run on a TCP port rather than on the default named socket. This can be achieved with `dpkg-reconfigure clamav-milter`. When prompted for the "Communication interface with Sendmail", answer "inet:10002@127.0.0.1".

NOTE

Real TCP port vs named socket

The reason why we use a real TCP port rather than the named socket is that the postfix daemons often run chrooted and do not have access to the directory hosting the named socket. You could also decide to keep using a named socket and pick a location within the chroot (`/var/spool/postfix/`).

The standard ClamAV configuration fits most situations, but some important parameters can still be customized with `dpkg-reconfigure clamav-base`.

The last step involves telling Postfix to use the recently-configured filter. This is a simple matter of adding the following directive to `/etc/postfix/main.cf`:

```
# Virus check with clamav-milter
smtpd_milters = inet:[127.0.0.1]:10002
```

If the antivirus causes problems, this line can be commented out, and `service postfix reload` should be run so that this change is taken into account.

IN PRACTICE

Testing the antivirus

Once the antivirus is set up, its correct behavior should be tested. The simplest way to do that is to send a test email with an attachment containing the `eicar.com` (or `eicar.com.zip`) file, which can be downloaded online:

➨ `http://www.eicar.org/86-0-Intended-use.html`

This file is not a true virus, but a test file that all antivirus software on the market diagnose as a virus to allow checking installations.

All messages handled by Postfix now go through the antivirus filter.

11.1.7. Authenticated SMTP

Being able to send emails requires an SMTP server to be reachable; it also requires said SMTP server to send emails through it. For roaming users, this may need regularly changing the configuration of the SMTP client, since Falcot's SMTP server rejects messages coming from IP addresses apparently not belonging to the company. Two solutions exist: either the roaming user installs an SMTP server on their computer, or they still use the company server with some means of authenticating as an employee. The former solution is not recommended since the computer won't be permanently connected, and it won't be able to retry sending messages in case of problems; we will focus on the latter solution.

SMTP authentication in Postfix relies on SASL (*Simple Authentication and Security Layer*). It requires installing the *libsasl2-modules* and *sasl2-bin* packages, then registering a password in the SASL database for each user that needs authenticating on the SMTP server. This is done with the `saslpasswd2` command, which takes several parameters. The `-u` option defines the authentication domain, which must match the `smtpd_sasl_local_domain` parameter in the Postfix configuration. The `-c` option allows creating a user, and `-f` allows specifying the file to use if the SASL database needs to be stored at a different location than the default (`/etc/sasldb2`).

```
# saslpasswd2 -u 'postconf -h myhostname' -f /var/spool/postfix/etc/sasldb2 -c jean
[... type jean's password twice ...]
```

Note that the SASL database was created in Postfix's directory. In order to ensure consistency, we also turn `/etc/sasldb2` into a symbolic link pointing at the database used by Postfix, with the `ln -sf /var/spool/postfix/etc/sasldb2 /etc/sasldb2` command.

Now we need to configure Postfix to use SASL. First the postfix user needs to be added to the sasl group, so that it can access the SASL account database. A few new parameters are also needed to enable SASL, and the smtpd_recipient_restrictions parameter needs to be configured to allow SASL-authenticated clients to send emails freely.

Example 11.15 *Enabling SASL in `/etc/postfix/main.cf`*

```
# Enable SASL authentication
smtpd_sasl_auth_enable = yes
# Define the SASL authentication domain to use
smtpd_sasl_local_domain = $myhostname
[...]
# Adding permit_sasl_authenticated before reject_unauth_destination
# allows relaying mail sent by SASL-authenticated users
smtpd_recipient_restrictions = permit_mynetworks,
    permit_sasl_authenticated,
    reject_unauth_destination,
[...]
```

EXTRA

Authenticated SMTP client

Most email clients are able to authenticate to an SMTP server before sending outgoing messages, and using that feature is a simple matter of configuring the appropriate parameters. If the client in use does not provide that feature, the workaround is to use a local Postfix server and configure it to relay email via the remote SMTP server. In this case, the local Postfix itself will be the client that authenticates with SASL. Here are the required parameters:

```
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
relay_host = [mail.falcot.com]
```

The `/etc/postfix/sasl_passwd` file needs to contain the username and password to use for authenticating on the `mail.falcot.com` server. Here is an example:

```
[mail.falcot.com]   joe:LyinIsji
```

As for all Postfix maps, this file must be turned into `/etc/postfix/sasl_passwd.db` with the `postmap` command.

11.2. Web Server (HTTP)

The Falcot Corp administrators decided to use the Apache HTTP server, included in Debian *Jessie* at version 2.4.10.

ALTERNATIVE

Other web servers

Apache is merely the most widely-known (and widely-used) web server, but there are others; they can offer better performance under certain workloads, but this has its counterpart in the smaller number of available features and modules. However, when the prospective web server is built to serve static files or to act as a proxy, the alternatives, such as *nginx* and *lighttpd*, are worth investigating.

11.2.1. Installing Apache

Installing the *apache2* package is all that is needed. It contains all the modules, including the *Multi-Processing Modules* (MPMs) that affect how Apache handles parallel processing of many requests (those used to be provided in separate *apache2-mpm-** packages). It will also pull *apache2-utils* containing the command line utilities that we will discover later.

The MPM in use affects significantly the way Apache will handle concurrent requests. With the *worker* MPM, it uses *threads* (lightweight processes), whereas with the *prefork* MPM it uses a pool of processes created in advance. With the *event* MPM it also uses threads, but the inactive connections (notably those kept open by the HTTP *keep-alive* feature) are handed back to a dedicated management thread.

The Falcot administrators also install *libapache2-mod-php5* so as to include the PHP support in Apache. This causes the default *event* MPM to be disabled, and *prefork* to be used instead, since PHP only works under that particular MPM.

SECURITY

Execution under the `www-data` user

By default, Apache handles incoming requests under the identity of the `www-data` user. This means that a security vulnerability in a CGI script executed by Apache (for a dynamic page) won't compromise the whole system, but only the files owned by this particular user.

Using the *suexec* modules allows bypassing this rule so that some CGI scripts are executed under the identity of another user. This is configured with a `SuexecUserGroup` *`usergroup`* directive in the Apache configuration.

Another possibility is to use a dedicated MPM, such as the one provided by *libapache2-mpm-itk*. This particular one has a slightly different behavior: it allows "isolating" virtual hosts (actually, sets of pages) so that they each run as a different user. A vulnerability in one website therefore cannot compromise files belonging to the owner of another website.

QUICK LOOK

List of modules

The full list of Apache standard modules can be found online.

➨ `http://httpd.apache.org/docs/2.4/mod/index.html`

Apache is a modular server, and many features are implemented by external modules that the main program loads during its initialization. The default configuration only enables the most common modules, but enabling new modules is a simple matter of running `a2enmod` *`module`*; to disable a module, the command is `a2dismod` *`module`*. These programs actually only create (or delete) symbolic links in `/etc/apache2/mods-enabled/`, pointing at the actual files (stored in `/etc/apache2/mods-available/`).

With its default configuration, the web server listens on port 80 (as configured in `/etc/apache2/ports.conf`), and serves pages from the `/var/www/` directory (as configured in `/etc/apache2/sites-enabled/000-default.conf`).

GOING FURTHER

Adding support for SSL

Apache 2.4 includes the SSL module required for secure HTTP (HTTPS) out of the box. It just needs to be enabled with `a2enmod ssl`, then the required directives have to be added to the configuration files. A configuration example is provided in `/etc/apache2/sites-available/default-ssl.conf`.

➨ http://httpd.apache.org/docs/2.4/mod/mod_ssl.html

Some extra care must be taken if you want to favor SSL connections with *Perfect Forward Secrecy* (those connections use ephemeral session keys ensuring that a compromission of the server's secret key does not result in the compromission of old encrypted traffic that could have been stored while sniffing on the network). Have a look at Mozilla's recommandations in particular:

➨ https://wiki.mozilla.org/Security/Server_Side_TLS#Apache

11.2.2. Configuring Virtual Hosts

A virtual host is an extra identity for the web server.

Apache considers two different kinds of virtual hosts: those that are based on the IP address (or the port), and those that rely on the domain name of the web server. The first method requires allocating a different IP address (or port) for each site, whereas the second one can work on a single IP address (and port), and the sites are differentiated by the hostname sent by the HTTP client (which only works in version 1.1 of the HTTP protocol — fortunately that version is old enough that all clients use it already).

The (increasing) scarcity of IPv4 addresses usually favors the second method; however, it is made more complex if the virtual hosts need to provide HTTPS too, since the SSL protocol hasn't always provided for name-based virtual hosting; the SNI extension (*Server Name Indication*) that allows such a combination is not handled by all browsers. When several HTTPS sites need to run on the same server, they will usually be differentiated either by running on a different port or on a different IP address (IPv6 can help there).

The default configuration for Apache 2 enables name-based virtual hosts. In addition, a default virtual host is defined in the /etc/apache2/sites-enabled/000-default.conf file; this virtual host will be used if no host matching the request sent by the client is found.

CAUTION

First virtual host

Requests concerning unknown virtual hosts will always be served by the first defined virtual host, which is why we defined `www.falcot.com` first here.

QUICK LOOK

Apache supports SNI

The Apache server supports an SSL protocol extension called *Server Name Indication* (SNI). This extension allows the browser to send the hostname of the web server during the establishment of the SSL connection, much earlier than the HTTP request itself, which was previously used to identify the requested virtual host among those hosted on the same server (with the same IP address and port). This allows Apache to select the most appropriate SSL certificate for the transaction to proceed.

Before SNI, Apache would always use the certificate defined in the default virtual host. Clients trying to access another virtual host would then display warnings, since the certificate they received didn't match the website they were trying to access. Fortunately, most browsers now work with SNI; this includes Microsoft Internet Explorer starting with version 7.0 (starting on Vista), Mozilla Firefox starting with version 2.0, Apple Safari since version 3.2.1, and all versions of Google Chrome.

The Apache package provided in Debian is built with support for SNI; no particular configuration is therefore needed.

Care should also be taken to ensure that the configuration for the first virtual host (the one used by default) does enable TLSv1, since Apache uses the parameters of this first virtual host to establish secure connections, and they had better allow them

Each extra virtual host is then described by a file stored in `/etc/apache2/sites-available/`. Setting up a website for the falcot.org domain is therefore a simple matter of creating the following file, then enabling the virtual host with `a2ensite www.falcot.org`.

Example 11.16 *The `/etc/apache2/sites-available/www.falcot.org.conf` file*

```
<VirtualHost *:80>
ServerName www.falcot.org
ServerAlias falcot.org
DocumentRoot /srv/www/www.falcot.org
</VirtualHost>
```

The Apache server, as configured so far, uses the same log files for all virtual hosts (although this could be changed by adding CustomLog directives in the definitions of the virtual hosts). It therefore makes good sense to customize the format of this log file to have it include the name of the virtual host. This can be done by creating a `/etc/apache2/conf-available/customlog.conf` file that defines a new format for all log files (with the LogFormat directive) and by enabling it with `a2enconf customlog`. The CustomLog line must also be removed (or commented out) from the `/etc/apache2/sites-available/000-default.conf` file.

Example 11.17 *The /etc/apache2/conf.d/customlog.conf file*

```
# New log format including (virtual) host name
LogFormat "%v %h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" vhost

# Now let's use this "vhost" format by default
CustomLog /var/log/apache2/access.log vhost
```

11.2.3. Common Directives

This section briefly reviews some of the commonly-used Apache configuration directives.

The main configuration file usually includes several Directory blocks; they allow specifying different behaviors for the server depending on the location of the file being served. Such a block commonly includes Options and AllowOverride directives.

Example 11.18 *Directory block*

```
<Directory /var/www>
Options Includes FollowSymlinks
AllowOverride All
DirectoryIndex index.php index.html index.htm
</Directory>
```

The DirectoryIndex directive contains a list of files to try when the client request matches a directory. The first existing file in the list is used and sent as a response.

The Options directive is followed by a list of options to enable. The None value disables all options; correspondingly, All enables them all except MultiViews. Available options include:

- ExecCGI indicates that CGI scripts can be executed.
- FollowSymlinks tells the server that symbolic links can be followed, and that the response should contain the contents of the target of such links.
- SymlinksIfOwnerMatch also tells the server to follow symbolic links, but only when the link and the its target have the same owner.
- Includes enables *Server Side Includes* (*SSI* for short). These are directives embedded in HTML pages and executed on the fly for each request.
- Indexes tells the server to list the contents of a directory if the HTTP request sent by the client points at a directory without an index file (ie, when no files mentioned by the DirectoryIndex directive exists in this directory).
- MultiViews enables content negotiation; this can be used by the server to return a web page matching the preferred language as configured in the browser.

BACK TO BASICS

.htaccess file

The `.htaccess` file contains Apache configuration directives enforced each time a request concerns an element of the directory where it is stored. The scope of these directives also recurses to all the subdirectories within.

Most of the directives that can occur in a `Directory` block are also legal in a `.htaccess` file.

The AllowOverride directive lists all the options that can be enabled or disabled by way of a `.htaccess` file. A common use of this option is to restrict ExecCGI, so that the administrator chooses which users are allowed to run programs under the web server's identity (the www-data user).

Requiring Authentication

In some circumstances, access to part of a website needs to be restricted, so only legitimate users who provide a username and a password are granted access to the contents.

Example 11.19 *.htaccess file requiring authentication*

```
Require valid-user
AuthName "Private directory"
AuthType Basic
AuthUserFile /etc/apache2/authfiles/htpasswd-private
```

SECURITY

No security

The authentication system used in the above example (`Basic`) has minimal security as the password is sent in clear text (it is only encoded as *base64*, which is a simple encoding rather than an encryption method). It should also be noted that the documents "protected" by this mechanism also go over the network in the clear. If security is important, the whole HTTP connection should be encrypted with SSL.

The `/etc/apache2/authfiles/htpasswd-private` file contains a list of users and passwords; it is commonly manipulated with the `htpasswd` command. For example, the following command is used to add a user or change their password:

```
# htpasswd /etc/apache2/authfiles/htpasswd-private user
New password:
Re-type new password:
Adding password for user user
```

Restricting Access

The Require directive controls access restrictions for a directory (and its subdirectories, recursively).

It can be used to restrict access based on many criteria; we will stop at describing access restriction based on the IP address of the client, but it can be made much more powerful than that, especially when several Require directives are combined within a RequireAll block.

Example 11.20 *Only allow from the local network*

```
Require ip 192.168.0.0/16
```

ALTERNATIVE

Old syntax

The `Require` syntax is only available in Apache 2.4 (the version in *Jessie*). For users of *Wheezy*, the Apache 2.2 syntax is different, and we describe it here mainly for reference, although it can also be made available in Apache 2.4 using the `mod_access_compat` module.

The `Allow from` and `Deny from` directives control access restrictions for a directory (and its subdirectories, recursively).

The `Order` directive tells the server of the order in which the `Allow from` and `Deny from` directives are applied; the last one that matches takes precedence. In concrete terms, `Order deny,allow` allows access if no `Deny from` applies, or if an `Allow from` directive does. Conversely, `Order allow,deny` rejects access if no `Allow from` directive matches (or if a `Deny from` directive applies).

The `Allow from` and `Deny from` directives can be followed by an IP address, a network (such as `192.168.0.0/255.255.255.0`, `192.168.0.0/24` or even `192.168.0`), a hostname or a domain name, or the `all` keyword, designating everyone.

For instance, to reject connections by default but allow them from the local network, you could use this:

```
Order deny,allow
Allow from 192.168.0.0/16
Deny from all
```

11.2.4. Log Analyzers

A log analyzer is frequently installed on a web server; since the former provides the administrators with a precise idea of the usage patterns of the latter.

The Falcot Corp administrators selected *AWStats* (*Advanced Web Statistics*) to analyze their Apache log files.

The first configuration step is the customization of the `/etc/awstats/awstats.conf` file. The Falcot administrators keep it unchanged apart from the following parameters:

```
LogFile="/var/log/apache2/access.log"
LogFormat = "%virtualname %host %other %logname %time1 %methodurl %code %bytesd %
       refererquot %uaquot"
SiteDomain="www.falcot.com"
HostAliases="falcot.com REGEX[^.*\.falcot\.com$]"
DNSLookup=1
LoadPlugin="tooltips"
```

All these parameters are documented by comments in the template file. In particular, the `LogFile` and `LogFormat` parameters describe the location and format of the log file and the information it contains; `SiteDomain` and `HostAliases` list the various names under which the main web site is known.

For high traffic sites, `DNSLookup` should usually not be set to 1; for smaller sites, such as the Falcot one described above, this setting allows getting more readable reports that include full machine names instead of raw IP addresses.

SECURITY

Access to statistics

AWStats makes its statistics available on the website with no restrictions by default, but restrictions can be set up so that only a few (probably internal) IP addresses can access them; the list of allowed IP addresses needs to be defined in the `AllowAccessFromWebToFollowingIPAddresses` parameter

AWStats will also be enabled for other virtual hosts; each virtual host needs its own configuration file, such as `/etc/awstats/awstats.www.falcot.org.conf`.

Example 11.21 *AWStats configuration file for a virtual host*

```
Include "/etc/awstats/awstats.conf"
SiteDomain="www.falcot.org"
HostAliases="falcot.org"
```

AWStats uses many icons stored in the `/usr/share/awstats/icon/` directory. In order for these icons to be available on the web site, the Apache configuration needs to be adapted to include the following directive:

```
Alias /awstats-icon/ /usr/share/awstats/icon/
```

After a few minutes (and once the script has been run a few times), the results are available online:

➠ http://www.falcot.com/cgi-bin/awstats.pl

➠ http://www.falcot.org/cgi-bin/awstats.pl

CAUTION

Log file rotation

In order for the statistics to take all the logs into account, *AWStats* needs to be run right before the Apache log files are rotated. Looking at the `prerotate` directive of `/etc/logrotate.d/apache2` file, this can be solved by putting a symlink to `/usr/share/awstats/tools/update.sh` in `/etc/logrotate.d/httpd-prerotate`:

```
$ cat /etc/logrotate.d/apache2
/var/log/apache2/*.log {
  daily
  missingok
  rotate 14
  compress
  delaycompress
  notifempty
  create 644 root adm
  sharedscripts
  postrotate
    if /etc/init.d/apache2 status > /dev/null ; then \
      /etc/init.d/apache2 reload > /dev/null; \
    fi;
  endscript
  prerotate
    if [ -d /etc/logrotate.d/httpd-prerotate ]; then \
      run-parts /etc/logrotate.d/httpd-prerotate; \
    fi; \
  endscript
}
$ sudo mkdir -p /etc/logrotate.d/httpd-prerotate
$ sudo ln -sf /usr/share/awstats/tools/update.sh \
  /etc/logrotate.d/httpd-prerotate/awstats
```

Note also that the log files created by `logrotate` need to be readable by everyone, especially AWStats. In the above example, this is ensured by the `create 644 root adm` line (instead of the default `640` permissions).

11.3. FTP File Server

FTP (*File Transfer Protocol*) is one of the first protocols of the Internet (RFC 959 was issued in 1985!). It was used to distribute files before the Web was even born (the HTTP protocol was created in 1990, and formally defined in its 1.0 version by RFC 1945, issued in 1996).

This protocol allows both file uploads and file downloads; for this reason, it is still widely used to deploy updates to a website hosted by one's Internet service provider (or any other entity hosting websites). In these cases, secure access is enforced with a user identifier and password; on successful authentication, the FTP server grants read-write access to that user's home directory.

Other FTP servers are mainly used to distribute files for public downloading; Debian packages are a good example. The contents of these servers is fetched from other, geographically remote, servers; it is then made available to less distant users. This means that client authentication is not required; as a consequence, this operating mode is known as "anonymous FTP". To be perfectly correct, the clients do authenticate with the anonymous username; the password is often, by convention, the user's email address, but the server ignores it.

Many FTP servers are available in Debian (*ftpd*, *proftpd-basic*, *pyftpd* and so on). The Falcot Corp administrators picked *vsftpd* because they only use the FTP server to distribute a few files (including a Debian package repository); since they don't need advanced features, they chose to focus on the security aspects.

Installing the package creates an ftp system user. This account is always used for anonymous FTP connections, and its home directory (`/srv/ftp/`) is the root of the tree made available to users connecting to this service. The default configuration (in `/etc/vsftpd.conf`) requires some changes to cater to the simple need of making big files available for public downloads: anonymous access needs to be enabled (anonymous_enable=YES) and read-only access of local users needs to be disabled (local_enable=NO). The latter is particularly important since the FTP protocol doesn't use any form of encryption and the user password could be intercepted over the wire.

11.4. NFS File Server

NFS (*Network File System*) is a protocol allowing remote access to a filesystem through the network. All Unix systems can work with this protocol; when Windows systems are involved, Samba must be used instead.

NFS is a very useful tool but, historically, it has suffered from many limitations, most of which have been addressed with version 4 of the protocol. The downside is that the latest version of NFS is harder to configure when you want to make use of basic security features such as authentication and encryption since it relies on Kerberos for those parts. And without those, the NFS protocol must be restricted to a trusted local network since data goes over the network unencrypted (a *sniffer* can intercept it) and access rights are granted based on the client's IP address (which can be spoofed).

DOCUMENTATION

NFS HOWTO

Good documentation to deploy NFSv4 is rather scarce. Here are some pointers with content of varying quality but that should at least give some hints on what should be done.

➨ https://help.ubuntu.com/community/NFSv4Howto

➨ http://wiki.linux-nfs.org/wiki/index.php/Nfsv4_configuration

11.4.1. Securing NFS

If you don't use the Kerberos-based security features, it is vital to ensure that only the machines allowed to use NFS can connect to the various required RPC servers, because the basic protocol trusts the data received from the network. The firewall must also block *IP spoofing* so as to prevent an outside machine from acting as an inside one, and access to the appropriate ports must be restricted to the machines meant to access the NFS shares.

BACK TO BASICS

RPC

RPC (*Remote Procedure Call*) is a Unix standard for remote services. NFS is one such service.

RPC services register to a directory known as the *portmapper*. A client wishing to perform an NFS query first addresses the *portmapper* (on port 111, either TCP or UDP), and asks for the NFS server; the reply usually mentions port 2049 (the default for NFS). Not all RPC services necessarily use a fixed port.

Older versions of the protocol required other RPC services which used dynamically assigned ports. Fortunately, with NFS version 4, only port 2049 (for NFS) and 111 (for the portmapper) are needed and they are thus easy to firewall.

11.4.2. NFS Server

The NFS server is part of the Linux kernel; in kernels provided by Debian it is built as a kernel module. If the NFS server is to be run automatically on boot, the *nfs-kernel-server* package should be installed; it contains the relevant start-up scripts.

The NFS server configuration file, `/etc/exports`, lists the directories that are made available over the network (*exported*). For each NFS share, only the given list of machines is granted access. More fine-grained access control can be obtained with a few options. The syntax for this file is quite simple:

```
/directory/to/share machine1(option1,option2,...) machine2(...) ...
```

Note that with NFSv4, all exported directories must be part of a single hierarchy and that the root directory of that hierarchy must be exported and identified with the option fsid=0 or fsid=root.

Each machine can be identified either by its DNS name or its IP address. Whole sets of machines can also be specified using either a syntax such as *.falcot.com or an IP address range such as 192.168.0.0/255.255.255.0 or 192.168.0.0/24.

Directories are made available as read-only by default (or with the ro option). The rw option allows read-write access. NFS clients typically connect from a port restricted to root (in other words, below 1024); this restriction can be lifted by the insecure option (the secure option is implicit, but it can be made explicit if needed for clarity).

By default, the server only answers an NFS query when the current disk operation is complete (sync option); this can be disabled with the async option. Asynchronous writes increase per-

formance a bit, but they decrease reliability since there is a data loss risk in case of the server crashing between the acknowledgment of the write and the actual write on disk. Since the default value changed recently (as compared to the historical value of NFS), an explicit setting is recommended.

In order to not give root access to the filesystem to any NFS client, all queries appearing to come from a root user are considered by the server as coming from the nobody user. This behavior corresponds to the root_squash option, and is enabled by default. The no_root_squash option, which disables this behavior, is risky and should only be used in controlled environments. The anonuid=*uid* and anongid=*gid* options allow specifying another fake user to be used instead of UID/GID 65534 (which corresponds to user nobody and group nogroup).

With NFSv4, you can add a sec option to indicate the security level that you want: sec=sys is the default with no special security features, sec=krb5 enables authentication only, sec=krb5i adds integrity protection, and sec=krb5p is the most complete level which includes privacy protection (with data encryption). For this to work you need a working Kerberos setup (that service is not covered by this book).

Other options are available; they are documented in the `exports(5)` manual page.

CAUTION

First installation

The `/etc/init.d/nfs-kernel-server` boot script only starts the server if the `/etc/exports` lists one or more valid NFS shares. On initial configuration, once this file has been edited to contain valid entries, the NFS server must therefore be started with the following command:

```
# service nfs-kernel-server start
```

11.4.3. NFS Client

As with other filesystems, integrating an NFS share into the system hierarchy requires mounting. Since this filesystem has its peculiarities, a few adjustments were required in the syntaxes of the `mount` command and the `/etc/fstab` file.

Example 11.22 *Manually mounting with the `mount` command*

```
# mount -t nfs4 -o rw,nosuid arrakis.internal.falcot.com:/shared /srv/
      shared
```

Example 11.23 *NFS entry in the `/etc/fstab` file*

```
arrakis.internal.falcot.com:/shared /srv/shared nfs4 rw,nosuid 0 0
```

The entry described above mounts, at system startup, the `/shared/` NFS directory from the arrakis server into the local `/srv/shared/` directory. Read-write access is requested (hence the

rw parameter). The nosuid option is a protection measure that wipes any setuid or setgid bit from programs stored on the share. If the NFS share is only meant to store documents, another recommended option is noexec, which prevents executing programs stored on the share. Note that on the server, the `shared` directory is below the NFSv4 root export (for example `/export/shared`), it is not a top-level directory.

The `nfs(5)` manual page describes all the options in some detail.

11.5. Setting Up Windows Shares with Samba

Samba is a suite of tools handling the SMB protocol (also known as "CIFS") on Linux. This protocol is used by Windows for network shares and shared printers.

Samba can also act as an Windows domain controller. This is an outstanding tool for ensuring seamless integration of Linux servers and the office desktop machines still running Windows.

11.5.1. Samba Server

The *samba* package contains the main two servers of Samba 4, `smbd` and `nmbd`.

DOCUMENTATION

Going further

The Samba server is extremely configurable and versatile, and can address a great many different use cases matching very different requirements and network architectures. This book only focuses on the use case where Samba is used as a standalone server, but it can also be a NT4 Domain Controller or a full Active Directory Domain Controller, or a simple member of an existing domain (which could be a managed by a Windows server).

The *samba-doc* package contains a wealth of commented example files in `/usr/share/doc/samba-doc/examples/`.

TOOL

Authenticating with a Windows Server

Winbind gives system administrators the option of using a Windows server as an authentication server. Winbind also integrates cleanly with PAM and NSS. This allows setting up Linux machines where all users of a Windows domain automatically get an account.

More information can be found in the `/usr/share/doc/samba-doc/examples/pam_winbind/` directory.

Configuring with `debconf`

The package sets up a minimal configuration during the initial installation but you should really run `dpkg-reconfigure samba-common` to adapt it:

The first piece of required information is the name of the workgroup where the Samba server will belong (the answer is FALCOTNET in our case).

The package also proposes identifying the WINS server from the information provided by the DHCP daemon. The Falcot Corp administrators rejected this option, since they intend to use the Samba server itself as the WINS server.

Configuring Manually

Changes to `smb.conf` The requirements at Falcot require other options to be modified in the `/etc/samba/smb.conf` configuration file. The following excerpts summarize the changes that were effected in the [global] section.

```
[global]

## Browsing/Identification ###

# Change this to the workgroup/NT-domain name your Samba server will part of
   workgroup = FALCOTNET

# Windows Internet Name Serving Support Section:
# WINS Support - Tells the NMBD component of Samba to enable its WINS Server
   wins support = yes  1

[...]

####### Authentication #######

# Server role. Defines in which mode Samba will operate. Possible
# values are "standalone server", "member server", "classic primary
# domain controller", "classic backup domain controller", "active
# directory domain controller".
#
# Most people will want "standalone sever" or "member server".
# Running as "active directory domain controller" will require first
# running "samba-tool domain provision" to wipe databases and create a
# new domain.
   server role = standalone server

# "security = user" is always a good idea. This will require a Unix account
# in this server for every user accessing the server.
   security = user  2

[...]
```

1 Indicates that Samba should act as a Netbios name server (WINS) for the local network.

2 This is the default value for this parameter; however, since it is central to the Samba configuration, filling it explicitly is recommended. Each user must authenticate before accessing any share.

Adding Users Each Samba user needs an account on the server; the Unix accounts must be created first, then the user needs to be registered in Samba's database. The Unix step is done quite normally (using `adduser` for instance).

Adding an existing user to the Samba database is a matter of running the `smbpasswd -a user` command; this command asks for the password interactively.

A user can be deleted with the `smbpasswd -x user` command. A Samba account can also be temporarily disabled (with `smbpasswd -d user`) and re-enabled later (with `smbpasswd -e user`).

11.5.2. Samba Client

The client features in Samba allow a Linux machine to access Windows shares and shared printers. The required programs are available in the *cifs-utils* and *smbclient* packages.

The `smbclient` Program

The `smbclient` program queries SMB servers. It accepts a `-U user` option, for connecting to the server under a specific identity. `smbclient //server/share` accesses the share in an interactive way similar to the command-line FTP client. `smbclient -L server` lists all available (and visible) shares on a server.

Mounting Windows Shares

The `mount` command allows mounting a Windows share into the Linux filesystem hierarchy (with the help of `mount.cifs` provided by *cifs-utils*).

Example 11.24 *Mounting a Windows share*

```
mount -t cifs //arrakis/shared /shared \
      -o credentials=/etc/smb-credentials
```

The `/etc/smb-credentials` file (which must not be readable by users) has the following format:

```
username = user
password = password
```

Other options can be specified on the command-line; their full list is available in the `mount.cifs(1)` manual page. Two options in particular can be interesting: `uid` and `gid` allow forcing the owner and group of files available on the mount, so as not to restrict access to root.

A mount of a Windows share can also be configured in `/etc/fstab`:

```
//server/shared /shared cifs credentials=/etc/smb-credentials
```

Unmounting a SMB/CIFS share is done with the standard `umount` command.

Printing on a Shared Printer

CUPS is an elegant solution for printing from a Linux workstation to a printer shared by a Windows machine. When the *smbclient* is installed, CUPS allows installing Windows shared printers automatically.

Here are the required steps:

- Enter the CUPS configuration interface: http://localhost:631/admin
- Click on "Add Printer".
- Choose the printer device, pick "Windows Printer via SAMBA".
- Enter the connection URI for the network printer. It should look like the following:

  smb://*user*:*password*@*server*/*printer*.
- Enter the name that will uniquely identify this printer. Then enter the description and location of the printer. Those are the strings that will be shown to end users to help them identify the printers.
- Indicate the manufacturer/model of the printer, or directly provide a working printer description file (PPD).

Voilà, the printer is operational!

11.6. HTTP/FTP Proxy

An HTTP/FTP proxy acts as an intermediary for HTTP and/or FTP connections. Its role is twofold:

- Caching: recently downloaded documents are copied locally, which avoids multiple downloads.
- Filtering server: if use of the proxy is mandated (and outgoing connections are blocked unless they go through the proxy), then the proxy can determine whether or not the request is to be granted.

Falcot Corp selected Squid as their proxy server.

11.6.1. Installing

The *squid3* Debian package only contains the modular (caching) proxy. Turning it into a filtering server requires installing the additional *squidguard* package. In addition, *squid-cgi* provides a querying and administration interface for a Squid proxy.

Prior to installing, care should be taken to check that the system can identify its own complete name: the `hostname -f` must return a fully-qualified name (including a domain). If it does not, then the `/etc/hosts` file should be edited to contain the full name of the system (for instance, arrakis.falcot.com). The official computer name should be validated with the network administrator in order to avoid potential name conflicts.

11.6.2. Configuring a Cache

Enabling the caching server feature is a simple matter of editing the `/etc/squid3/squid.conf` configuration file and allowing machines from the local network to run queries through the proxy. The following example shows the modifications made by the Falcot Corp administrators:

Example 11.25 *The `/etc/squid3/squid.conf` file (excerpts)*

```
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS

# Example rule allowing access from your local networks. Adapt
# to list your (internal) IP networks from where browsing should
# be allowed
acl our_networks src 192.168.1.0/24 192.168.2.0/24
http_access allow our_networks
http_access allow localhost
# And finally deny all other access to this proxy
http_access deny all
```

11.6.3. Configuring a Filter

`squid` itself does not perform the filtering; this action is delegated to `squidGuard`. The former must then be configured to interact with the latter. This involves adding the following directive to the `/etc/squid3/squid.conf` file:

```
url_rewrite_program /usr/bin/squidGuard -c /etc/squid3/squidGuard.conf
```

The `/usr/lib/cgi-bin/squidGuard.cgi` CGI program also needs to be installed, using `/usr/share/doc/squidguard/examples/squidGuard.cgi.gz` as a starting point. Required modifications to this script are the `$proxy` and `$proxymaster` variables (the name of the proxy and the administrator's contact e-mail, respectively). The `$image` and `$redirect` variables should point to existing images representing the rejection of a query.

The filter is enabled with the `service squid3 reload` command. However, since the *squidguard* package does no filtering by default, it is the administrator's task to define the policy. This can be done by creating the `/etc/squid3/squidGuard.conf` file (using `/etc/squidguard/squidGuard.conf.default` as template if required).

The working database must be regenerated with `update-squidguard` after each change of the `squidGuard` configuration file (or one of the lists of domains or URLs it mentions). The configuration file syntax is documented on the following website:

➨ `http://www.squidguard.org/Doc/configure.html`

ALTERNATIVE

DansGuardian

The *dansguardian* package is an alternative to *squidguard*. This software does not simply handle a blacklist of forbidden URLs, but it can take advantage of the PICS system (*Platform for Internet Content Selection*) to decide whether a page is acceptable by dynamic analysis of its contents.

11.7. LDAP Directory

OpenLDAP is an implementation of the LDAP protocol; in other words, it is a special-purpose database designed for storing directories. In the most common use case, using an LDAP server allows centralizing management of user accounts and the related permissions. Moreover, an LDAP database is easily replicated, which allows setting up multiple synchronized LDAP servers. When the network and the user base grows quickly, the load can then be balanced across several servers.

LDAP data is structured and hierarchical. The structure is defined by "schemas" which describe the kind of objects that the database can store, with a list of all their possible attributes. The syntax used to refer to a particular object in the database is based on this structure, which explains its complexity.

11.7.1. Installing

The *slapd* package contains the OpenLDAP server. The *ldap-utils* package includes command-line tools for interacting with LDAP servers.

Installing *slapd* usually asks very few questions and the resulting database is unlikely to suit your needs. Fortunately a simple `dpkg-reconfigure slapd` will let you reconfigure the LDAP database with more details:

- Omit OpenLDAP server configuration? No, of course, we want to configure this service.
- DNS domain name: "falcot.com".
- Organization name: "Falcot Corp".
- An administrative passwords needs to be typed in.
- Database backend to use: "MDB".
- Do you want the database to be removed when *slapd* is purged? No. No point in risking losing the database in case of a mistake.
- Move old database? This question is only asked when the configuration is attempted while a database already exists. Only answer "yes" if you actually want to start again from a

clean database, for instance if you run `dpkg-reconfigure slapd` right after the initial installation.

- Allow LDAPv2 protocol? No, there is no point in that. All the tools we are going to use understand the LDAPv3 protocol.

BACK TO BASICS

LDIF format

An LDIF file (*LDAP Data Interchange Format*) is a portable text file describing the contents of an LDAP database (or a portion thereof); this can then be used to inject the data into any other LDAP server.

A minimal database is now configured, as demonstrated by the following query:

```
$ ldapsearch -x -b dc=falcot,dc=com
# extended LDIF
#
# LDAPv3
# base <dc=falcot,dc=com> with scope sub
# filter: (objectclass=*)
# requesting: ALL
#

# falcot.com
dn: dc=falcot,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: Falcot Corp
dc: falcot

# admin, falcot.com
dn: cn=admin,dc=falcot,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

The query returned two objects: the organization itself, and the administrative user.

11.7.2. Filling in the Directory

Since an empty database is not particularly useful, we are going to inject into it all the existing directories; this includes the users, groups, services and hosts databases.

The *migrationtools* package provides a set of scripts dedicated to extract data from the standard Unix directories (`/etc/passwd`, `/etc/group`, `/etc/services`, `/etc/hosts` and so on), convert this data, and inject it into the LDAP database.

Once the package is installed, the `/etc/migrationtools/migrate_common.ph` must be edited; the `IGNORE_UID_BELOW` and `IGNORE_GID_BELOW` options need to be enabled (uncommenting them is enough), and `DEFAULT_MAIL_DOMAIN`/`DEFAULT_BASE` need to be updated.

The actual migration operation is handled by the `migrate_all_online.sh` command, as follows:

```
# cd /usr/share/migrationtools
# LDAPADD="/usr/bin/ldapadd -c" ETC_ALIASES=/dev/null ./migrate_all_online.sh
```

The `migrate_all_online.sh` asks a few questions about the LDAP database into which the data is to be migrated. Table 11.1 summarizes the answers given in the Falcot use-case.

Question	Answer
X.500 naming context	dc=falcot,dc=com
LDAP server hostname	localhost
Manager DN	cn=admin,dc=falcot,dc=com
Bind credentials	the administrative password
Create DUAConfigProfile	no

Table 11.1 *Answers to questions asked by the `migrate_all_online.sh` script*

We deliberately ignore migration of the `/etc/aliases` file, since the standard schema as provided by Debian does not include the structures that this script uses to describe email aliases. Should we want to integrate this data into the directory, the `/etc/ldap/schema/misc.schema` file should be added to the standard schema.

TOOL

Browsing an LDAP directory

The `jxplorer` command (in the package of the same name) is a graphical tool allowing to browse and edit an LDAP database. It is an interesting tool that provides an administrator with a good overview of the hierarchical structure of the LDAP data.

Also note the use of the -c option to the `ldapadd` command; this option requests that processing doesn't stop in case of error. Using this option is required because converting the `/etc/services` often generates a few errors that can safely be ignored.

11.7.3. Managing Accounts with LDAP

Now the LDAP database contains some useful information, the time has come to make use of this data. This section focuses on how to configure a Linux system so that the various system directories use the LDAP database.

Configuring NSS

The NSS system (Name Service Switch, see sidebar "NSS and system databases" page 158) is a modular system designed to define or fetch information for system directories. Using LDAP as a source of data for NSS requires installing the *libnss-ldap* package. Its installation asks a few questions; the answers are summarized in Table 11.2 .

Question	**Answer**
LDAP server Uniform Resource Identifier	ldap://ldap.falcot.com
Distinguished name of the search base	dc=falcot,dc=com
LDAP version to use	3
Does the LDAP database require login?	no
Special LDAP privileges for root	yes
Make the configuration file readable/writeable by its owner only	no
LDAP account for root	cn=admin,dc=falcot,dc=com
LDAP root account password	the administrative password

Table 11.2 *Configuring the* libnss-ldap *package*

The `/etc/nsswitch.conf` file then needs to be modified, so as to configure NSS to use the freshly-installed `ldap` module.

Example 11.26 *The `/etc/nsswitch.conf` file*

```
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the 'glibc-doc' and 'info' packages installed, try:
# 'info libc "Name Service Switch"' for information about this file.

passwd: ldap compat
group: ldap compat
shadow: ldap compat

hosts: files dns ldap
networks: ldap files
```

```
protocols: ldap db files
services: ldap db files
ethers: ldap db files
rpc: ldap db files

netgroup: ldap files
```

The `ldap` module is usually inserted before others, and it will therefore be queried first. The notable exception is the hosts service since contacting the LDAP server requires consulting DNS first (to resolve ldap.falcot.com). Without this exception, a hostname query would try to ask the LDAP server; this would trigger a name resolution for the LDAP server, and so on in an infinite loop.

If the LDAP server should be considered authoritative (and the local files used by the `files` module disregarded), services can be configured with the following syntax:

service:ldap [NOTFOUND=return] files.

If the requested entry does not exist in the LDAP database, the query will return a "not existing" reply even if the resource does exist in one of the local files; these local files will only be used when the LDAP service is down.

Configuring PAM

This section describes a PAM configuration (see sidebar "`/etc/environment` and `/etc/default/locale`" page 147) that will allow applications to perform the required authentications against the LDAP database.

CAUTION

Broken authentication

Changing the standard PAM configuration used by various programs is a sensitive operation. A mistake can lead to broken authentication, which could prevent logging in. Keeping a root shell open is therefore a good precaution. If configuration errors occur, they can be then fixed and the services restarted with minimal effort.

The LDAP module for PAM is provided by the *libpam-ldap* package. Installing this package asks a few questions very similar to those in *libnss-ldap*; some configuration parameters (such as the URI for the LDAP server) are even actually shared with the *libnss-ldap* package. Answers are summarized in Table 11.3 .

Installing *libpam-ldap* automatically adapts the default PAM configuration defined in the `/etc/pam.d/common-auth`, `/etc/pam.d/common-password` and `/etc/pam.d/common-account` files. This mechanism uses the dedicated `pam-auth-update` tool (provided by the *libpam-runtime* package). This tool can also be run by the administrator should they wish to enable or disable PAM modules.

Question	Answer
Allow LDAP admin account to behave like local root?	Yes. This allows using the usual `passwd` command for changing passwords stored in the LDAP database.
Does the LDAP database require logging in?	no
LDAP account for root	cn=admin,dc=falcot,dc=com
LDAP root account password	the LDAP database administrative password
Local encryption algorithm to use for passwords	crypt

Table 11.3 *Configuration of* libpam-ldap

Securing LDAP Data Exchanges

By default, the LDAP protocol transits on the network as cleartext; this includes the (encrypted) passwords. Since the encrypted passwords can be extracted from the network, they can be vulnerable to dictionary-type attacks. This can be avoided by using an extra encryption layer; enabling this layer is the topic of this section.

Configuring the Server The first step is to create a key pair (comprising a public key and a private key) for the LDAP server. The Falcot administrators reuse *easy-rsa* to generate it (see section 10.2.1.1, "Public Key Infrastructure: *easy-rsa*" page 224). Running `./build-server-key ldap.falcot.com` asks a few mundane questions (location, organization name and so on). The answer to the "common name" question *must* be the fully-qualified hostname for the LDAP server; in our case, ldap.falcot.com.

This command creates a certificate in the `keys/ldap.falcot.com.crt` file; the corresponding private key is stored in `keys/ldap.falcot.com.key`.

Now these keys have to be installed in their standard location, and we must make sure that the private file is readable by the LDAP server which runs under the openldap user identity:

```
# adduser openldap ssl-cert
Adding user ‘openldap’ to group ‘ssl-cert’ ...
Adding user openldap to group ssl-cert
Done.
# mv keys/ldap.falcot.com.key /etc/ssl/private/ldap.falcot.com.key
# chown root:ssl-cert /etc/ssl/private/ldap.falcot.com.key
# chmod 0640 /etc/ssl/private/ldap.falcot.com.key
# mv newcert.pem /etc/ssl/certs/ldap.falcot.com.pem
```

The `slapd` daemon also needs to be told to use these keys for encryption. The LDAP server configuration is managed dynamically: the configuration can be updated with normal LDAP operations on the cn=config object hierarchy, and the server updates `/etc/ldap/slapd.d` in

real time to make the configuration persistent. `ldapmodify` is thus the right tool to update the configuration:

Example 11.27 *Configuring `slapd` for encryption*

```
# cat >ssl.ldif <<END
dn: cn=config
changetype: modify
add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/ssl/certs/ldap.falcot.com.pem
-
add: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/ssl/private/ldap.falcot.com.key
-
END
# ldapmodify -Y EXTERNAL -H ldapi:/// -f ssl.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "cn=config"
```

TOOL

`ldapvi` to edit an LDAP directory

With `ldapvi`, you can display an LDIF output of any part of the LDAP directory, make some changes in the text editor, and let the tool do the corresponding LDAP operations for you.

It is thus a convenient way to update the configuration of the LDAP server, simply by editing the `cn=config` hierarchy.

```
# ldapvi -Y EXTERNAL -h ldapi:/// -b cn=config
```

The last step for enabling encryption involves changing the `SLAPD_SERVICES` variable in the `/etc/default/slapd` file. We'll play it safe and disable unsecured LDAP altogether.

Example 11.28 *The `/etc/default/slapd` file*

```
# Default location of the slapd.conf file or slapd.d cn=config directory. If
# empty, use the compiled-in default (/etc/ldap/slapd.d with a fallback to
# /etc/ldap/slapd.conf).
SLAPD_CONF=

# System account to run the slapd server under. If empty the server
# will run as root.
SLAPD_USER="openldap"

# System group to run the slapd server under. If empty the server will
# run in the primary group of its user.
```

```
SLAPD_GROUP="openldap"

# Path to the pid file of the slapd server. If not set the init.d script
# will try to figure it out from $SLAPD_CONF (/etc/ldap/slapd.conf by
# default)
SLAPD_PIDFILE=

# slapd normally serves ldap only on all TCP-ports 389. slapd can also
# service requests on TCP-port 636 (ldaps) and requests via unix
# sockets.
# Example usage:
# SLAPD_SERVICES="ldap://127.0.0.1:389/ ldaps:/// ldapi:///"
SLAPD_SERVICES="ldaps:/// ldapi:///"

# If SLAPD_NO_START is set, the init script will not start or restart
# slapd (but stop will still work).  Uncomment this if you are
# starting slapd via some other means or if you don't want slapd normally
# started at boot.
#SLAPD_NO_START=1

# If SLAPD_SENTINEL_FILE is set to path to a file and that file exists,
# the init script will not start or restart slapd (but stop will still
# work).  Use this for temporarily disabling startup of slapd (when doing
# maintenance, for example, or through a configuration management system)
# when you don't want to edit a configuration file.
SLAPD_SENTINEL_FILE=/etc/ldap/noslapd

# For Kerberos authentication (via SASL), slapd by default uses the system
# keytab file (/etc/krb5.keytab).  To use a different keytab file,
# uncomment this line and change the path.
#export KRB5_KTNAME=/etc/krb5.keytab

# Additional options to pass to slapd
SLAPD_OPTIONS=""
```

Configuring the Client On the client side, the configuration for the *libpam-ldap* and *libnss-ldap* modules needs to be modified to use an ldaps:// URI.

LDAP clients also need to be able to authenticate the server. In a X.509 public key infrastructure, public certificates are signed by the key of a certificate authority (CA). With *easy-rsa*, the Falcot administrators have created their own CA and they now need to configure the system to trust the signatures of Falcot's CA. This can be done by putting the CA certificate in `/usr/local/share/ca-certificates` and running `update-ca-certificates`.

```
# cp keys/ca.crt /usr/local/share/ca-certificates/falcot.crt
# update-ca-certificates
Updating certificates in /etc/ssl/certs... 1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d....
Adding debian:falcot.pem
done.
done.
```

Last but not least, the default LDAP URI and default base DN used by the various command line tools can be modified in `/etc/ldap/ldap.conf`. This will save quite some typing.

Example 11.29 *The `/etc/ldap/ldap.conf` file*

```
#
# LDAP Defaults
#

# See ldap.conf(5) for details
# This file should be world readable but not world writable.

BASE   dc=falcot,dc=com
URI    ldaps://ldap.falcot.com

#SIZELIMIT      12
#TIMELIMIT      15
#DEREF          never

# TLS certificates (needed for GnuTLS)
TLS_CACERT      /etc/ssl/certs/ca-certificates.crt
```

This chapter sampled only a fraction of the available server software; however, most of the common network services were described. Now it is time for an even more technical chapter: we'll go into deeper detail for some concepts, describe massive deployments and virtualization.

11.8. Real-Time Communication Services

Real-Time Communication (RTC) services include voice, video/webcam, instant messaging (IM) and desktop sharing. This chapter gives a brief introduction to three of the services required to operate RTC, including a TURN server, SIP server and XMPP server. Comprehensive details of how to plan, install and manage these services are available in the Real-Time Communications Quick Start Guide which includes examples specific to Debian.

➡ http://rtcquickstart.org

Both SIP and XMPP can provide the same functionality. SIP is slightly more well known for voice and video while XMPP is traditionally regarded as an IM protocol. In fact, they can both be used

for any of these purposes. To maximize connectivity options, it is recommended to run both in parallel.

These services rely on X.509 certificates both for authentication and confidentiality purposes. See section 10.2.1.1, "Public Key Infrastructure: *easy-rsa*" page 224 for details on how to create them. Alternatively the *Real-Time Communications Quick Start Guide* also has some useful explanations:

➡ http://rtcquickstart.org/guide/multi/tls.html

11.8.1. DNS settings for RTC services

RTC services require DNS SRV and NAPTR records. A sample configuration that can be placed in the zone file for falcot.com:

```
; the server where everything will run
server1            IN     A      198.51.100.19
server1            IN     AAAA   2001:DB8:1000:2000::19

; IPv4 only for TURN for now, some clients are buggy with IPv6
turn-server        IN     A      198.51.100.19

; IPv4 and IPv6 addresses for SIP
sip-proxy          IN     A      198.51.100.19
sip-proxy          IN     AAAA   2001:DB8:1000:2000::19

; IPv4 and IPv6 addresses for XMPP
xmpp-gw            IN     A      198.51.100.19
xmpp-gw            IN     AAAA   2001:DB8:1000:2000::19

; DNS SRV and NAPTR for STUN / TURN
_stun._udp  IN SRV    0 1 3467 turn-server.falcot.com.
_turn._udp  IN SRV    0 1 3467 turn-server.falcot.com.
@           IN NAPTR  10 0 "s" "RELAY:turn.udp" "" _turn._udp.falcot.com.

; DNS SRV and NAPTR records for SIP
_sips._tcp  IN SRV    0 1 5061 sip-proxy.falcot.com.
@           IN NAPTR  10 0 "s" "SIPS+D2T" "" _sips._tcp.falcot.com.

; DNS SRV records for XMPP Server and Client modes:
_xmpp-client._tcp  IN     SRV    5 0 5222 xmpp-gw.falcot.com.
_xmpp-server._tcp  IN     SRV    5 0 5269 xmpp-gw.falcot.com.
```

11.8.2. TURN Server

TURN is a service that helps clients behind NAT routers and firewalls to discover the most efficient way to communicate with other clients and to relay the media streams if no direct media

path can be found. It is highly recommended that the TURN server is installed before any of the other RTC services are offered to end users.

TURN and the related ICE protocol are open standards. To benefit from these protocols, maximizing connectivity and minimizing user frustration, it is important to ensure that all client software supports ICE and TURN.

For the ICE algorithm to work effectively, the server must have two public IPv4 addresses.

Install the TURN server

Install the *resiprocate-turn-server* package.

Edit the `/etc/reTurn/reTurnServer.config` configuration file. The most important thing to do is insert the IP addresses of the server.

```
# your IP addresses go here:
TurnAddress = 198.51.100.19
TurnV6Address = 2001:DB8:1000:2000::19
AltStunAddress = 198.51.100.20
# your domain goes here, it must match the value used
# to hash your passwords if they are already hashed
# using the HA1 algorithm:
AuthenticationRealm = myrealm

UserDatabaseFile = /etc/reTurn/users.txt
UserDatabaseHashedPasswords = true
```

Restart the service.

Managing the TURN users

Use the htdigest utility to manage the TURN server user list.

```
# htdigest /etc/reTurn/users.txt myrealm joe
```

Use the HUP signal to make the server reload the `/etc/reTurn/users.txt` file after changing it or enable the automatic reload feature in `/etc/reTurn/reTurnServer.config`.

11.8.3. SIP Proxy Server

A SIP proxy server manages the incoming and outgoing SIP connections between other organizations, SIP trunking providers, SIP PBXes such as Asterisk, SIP phones, SIP-based softphones and WebRTC applications.

It is strongly recommended to install and configure the SIP proxy before attempting a SIP PBX setup. The SIP proxy normalizes a lot of the traffic reaching the PBX and provides greater connectivity and resilience.

Install the SIP proxy

Install the *repro* package. Using the package from *jessie-backports* is highly recommended, as it has the latest improvements for maximizing connectivity and resilience.

Edit the `/etc/repro/repro.config` configuration file. The most important thing to do is insert the IP addresses of the server. The example below demonstrates how to setup both regular SIP and WebSockets/WebRTC, using TLS, IPv4 and IPv6:

```
# Transport1 will be for SIP over TLS connections
# We use port 5061 here but if you have clients connecting from
# locations with firewalls you could change this to listen on port 443
Transport1Interface = 198.51.100.19:5061
Transport1Type = TLS
Transport1TlsDomain = falcot.com
Transport1TlsClientVerification = Optional
Transport1RecordRouteUri = sip:falcot.com;transport=TLS
Transport1TlsPrivateKey = /etc/ssl/private/falcot.com-key.pem
Transport1TlsCertificate = /etc/ssl/public/falcot.com.pem

# Transport2 is the IPv6 version of Transport1
Transport2Interface = 2001:DB8:1000:2000::19:5061
Transport2Type = TLS
Transport2TlsDomain = falcot.com
Transport2TlsClientVerification = Optional
Transport2RecordRouteUri = sip:falcot.com;transport=TLS
Transport2TlsPrivateKey = /etc/ssl/private/falcot.com-key.pem
Transport2TlsCertificate = /etc/ssl/public/falcot.com.pem

# Transport3 will be for SIP over WebSocket (WebRTC) connections
# We use port 8443 here but you could use 443 instead
Transport3Interface = 198.51.100.19:8443
Transport3Type = WSS
Transport3TlsDomain = falcot.com
# This would require the browser to send a certificate, but browsers
# don't currently appear to be able to, so leave it as None:
Transport3TlsClientVerification = None
Transport3RecordRouteUri = sip:falcot.com;transport=WSS
Transport3TlsPrivateKey = /etc/ssl/private/falcot.com-key.pem
Transport3TlsCertificate = /etc/ssl/public/falcot.com.pem

# Transport4 is the IPv6 version of Transport3
Transport4Interface = 2001:DB8:1000:2000::19:8443
Transport4Type = WSS
Transport4TlsDomain = falcot.com
Transport4TlsClientVerification = None
Transport4RecordRouteUri = sip:falcot.com;transport=WSS
Transport4TlsPrivateKey = /etc/ssl/private/falcot.com-key.pem
Transport4TlsCertificate = /etc/ssl/public/falcot.com.pem
```

```
# Transport5: this could be for TCP connections to an Asterisk server
# in your internal network.  Don't allow port 5060 through the external
# firewall.
Transport5Interface = 198.51.100.19:5060
Transport5Type = TCP
Transport5RecordRouteUri = sip:198.51.100.19:5060;transport=TCP

HttpBindAddress = 198.51.100.19, 2001:DB8:1000:2000::19
HttpAdminUserFile = /etc/repro/users.txt

RecordRouteUri = sip:falcot.com;transport=tls
ForceRecordRouting = true
EnumSuffixes = e164.arpa, sip5060.net, e164.org
DisableOutbound = false
EnableFlowTokens = true
EnableCertificateAuthenticator = True
```

Use the `htdigest` utility to manage the admin password for the web interface. The username must be *admin* and the realm name must match the value specified in `repro.config`.

```
# htdigest /etc/repro/users.txt repro admin
```

Restart the service to use the new configuration.

Managing the SIP proxy

Go to the web interface at http://sip-proxy.falcot.com:5080 to complete the configuration by adding domains, local users and static routes.

The first step is to add the local domain. The process must be restarted after adding or removing domains from the list.

The proxy knows how to route calls between local users and full SIP address, the routing configuration is only necessary for overriding default behavior, for example, to recognize phone numbers, add a prefix and route them to a SIP provider.

11.8.4. XMPP Server

An XMPP server manages connectivity between local XMPP users and XMPP users in other domains on the public Internet.

Terminology **XMPP or Jabber?**	XMPP is sometimes referred to as Jabber. In fact, Jabber is a trademark and XMPP is the official name of the standard.

Prosody is a popular XMPP server that operates reliably on Debian servers.

Install the XMPP server

Install the *prosody* package. Using the package from *jessie-backports* is highly recommended, as it has the latest improvements for maximizing connectivity and resilience.

Review the `/etc/prosody/prosody.cfg.lua` configuration file. The most important thing to do is insert JIDs of the users who are permitted to manage the server.

```
admins = { "joe@falcot.com" }
```

An individual configuration file is also needed for each domain. Copy the sample from `/etc/prosody/conf.avail/example.com.cfg.lua` and use it as a starting point. Here is falcot.com.cfg.lua:

```
VirtualHost "falcot.com"
        enabled = true
        ssl = {
                key = "/etc/ssl/private/falcot.com-key.pem";
                certificate = "/etc/ssl/public/falcot.com.pem";
                }
```

To enable the domain, there must be a symlink from `/etc/prosody/conf.d/`. Create it that way:

```
# ln -s /etc/prosody/conf.avail/falcot.com.cfg.lua /etc/prosody/conf.d/
```

Restart the service to use the new configuration.

Managing the XMPP server

Some management operations can be performed using the prosodyctl command line utility. For example, to add the administrator account specified in `/etc/prosody/prosody.cfg.lua`:

```
# prosodyctl adduser joe@falcot.com
```

See the Prosody online documentation[1] for more details about how to customize the configuration.

11.8.5. Running services on port 443

Some administrators prefer to run all of their RTC services on port 443. This helps users to connect from remote locations such as hotels and airports where other ports may be blocked or Internet traffic is routed through HTTP proxy servers.

To use this strategy, each service (SIP, XMPP and TURN) needs a different IP address. All the services can still be on the same host as Linux supports multiple IP addresses on a single host.

[1] http://prosody.im/doc/configure

The port number, 443, must be specified in the configuration files for each process and also in the DNS SRV records.

11.8.6. Adding WebRTC

Falcot wants to let customers make phone calls directly from the web site. The Falcot administrators also want to use WebRTC as part of their disaster recovery plan, so staff can use web browsers at home to log in to the company phone system and work normally in an emergency.

IN PRACTICE
Try WebRTC

If you have not tried WebRTC before, there are various sites that give an online demonstration and test facilities.

➨ `http://www.sip5060.net/test-calls`

WebRTC is a rapidly evolving technology and it is essential to use packages from the *jessie-backports* or *Testing* distributions.

JSCommunicator is a generic, unbranded WebRTC phone that does not require any server-side scripting such as PHP. It is built exclusively with HTML, CSS and JavaScript. It is the basis for many other WebRTC services and modules for more advanced web publishing frameworks.

➨ `http://jscommunicator.org`

The package *jscommunicator-web-phone* is the quickest way to install a WebRTC phone into a web site. It requires a SIP proxy with a WebSocket transport. The instructions in section 11.8.3.1, "Install the SIP proxy" page 295 include the necessary details to enable the WebSocket transport in the *repro* SIP proxy.

After installing *jscommunicator-web-phone*, there are various ways to use it. A simple strategy is to include or copy the configuration from `/etc/jscommunicator-web-phone/apache.conf` into an Apache virtual host configuration.

Once the web-phone files are available in the web server, customize the `/etc/jscommunicator-web-phone/config.js` to point at the TURN server and SIP proxy. For example:

```
JSCommSettings = {

  // Web server environment
  webserver: {
    url_prefix: null            // If set, prefix used to construct sound/ URLs
  },

  // STUN/TURN media relays
  stun_servers: [],
  turn_servers: [
    { server:"turn:turn-server.falcot.com?transport=udp", username:"joe", password:"
          j0Ep455d" }
  ],
```

```
// WebSocket connection
websocket: {
    // Notice we use the falcot.com domain certificate and port 8443
    // This matches the Transport3 and Transport4 example in
    // the falcot.com repro.config file
  servers: 'wss://falcot.com:8443',
  connection_recovery_min_interval: 2,
  connection_recovery_max_interval: 30
},

...
```

More advanced click-to-call web sites typically use server-side scripting to generate the config.js file dynamically. The DruCall[2] source code demonstrates how to do this with PHP.

[2] http://drucall.org

Keywords

RAID
LVM
FAI
Preseeding
Monitoring
Virtualization
Xen
LXC