Section 51 - Security considerations

51. Security considerations

This chapter discusses a number of issues concerned with security, some of which are also covered in other parts of this manual.

For reasons that this author does not understand, some people have promoted Exim as a “particularly secure” mailer. Perhaps it is because of the existence of this chapter in the documentation. However, the intent of the chapter is simply to describe the way Exim works in relation to certain security concerns, not to make any specific claims about the effectiveness of its security as compared with other MTAs.

What follows is a description of the way Exim is supposed to be. Best efforts have been made to try to ensure that the code agrees with the theory, but an absence of bugs can never be guaranteed. Any that are reported will get fixed as soon as possible.

51.1 Building a more “hardened” Exim

There are a number of build-time options that can be set in Local/Makefile to create Exim binaries that are “harder” to attack, in particular by a rogue Exim administrator who does not have the root password, or by someone who has penetrated the Exim (but not the root) account. These options are as follows:

-

ALT_CONFIG_PREFIX can be set to a string that is required to match the start of any file names used with the -C option. When it is set, these file names are also not allowed to contain the sequence “/../”. (However, if the value of the -C option is identical to the value of CONFIGURE_FILE in Local/Makefile, Exim ignores -C and proceeds as usual.) There is no default setting for ALT_CONFIG_PREFIX.

If the permitted configuration files are confined to a directory to which only root has access, this guards against someone who has broken into the Exim account from running a privileged Exim with an arbitrary configuration file, and using it to break into other accounts.

-

If ALT_CONFIG_ROOT_ONLY is defined, root privilege is retained for -C and -D only if the caller of Exim is root. Without it, the Exim user may also use -C and -D and retain privilege. Setting this option locks out the possibility of testing a configuration using -C right through message reception and delivery, even if the caller is root. The reception works, but by that time, Exim is running as the Exim user, so when it re-execs to regain privilege for the delivery, the use of -C causes privilege to be lost. However, root can test reception and delivery using two separate commands. ALT_CONFIG_ROOT_ONLY is not set by default.

-

If DISABLE_D_OPTION is defined, the use of the -D command line option is disabled.

-

FIXED_NEVER_USERS can be set to a colon-separated list of users that are never to be used for any deliveries. This is like the never_users runtime option, but it cannot be overridden; the runtime option adds additional users to the list. The default setting is “root”; this prevents a non-root user who is permitted to modify the runtime file from using Exim as a way to get root.

51.2 Root privilege

The Exim binary is normally setuid to root, which means that it gains root privilege (runs as root) when it starts execution. In some special cases (for example, when the daemon is not in use and there are no local deliveries), it may be possible to run Exim setuid to some user other than root. This is discussed in the next section. However, in most installations, root privilege is required for two things:

-

To set up a socket connected to the standard SMTP port (25) when initialising the listening daemon. If Exim is run from inetd, this privileged action is not required.

-

To be able to change uid and gid in order to read users’ .forward files and perform local deliveries as the receiving user or as specified in the configuration.

It is not necessary to be root to do any of the other things Exim does, such as receiving messages and delivering them externally over SMTP, and it is obviously more secure if Exim does not run as root except when necessary. For this reason, a user and group for Exim to use must be defined in Local/Makefile. These are known as “the Exim user” and “the Exim group”. Their values can be changed by the run time configuration, though this

is not recommended. Often a user called exim is used, but some sites use mail or another user name altogether.

Exim uses setuid() whenever it gives up root privilege. This is a permanent abdication; the process cannot regain root afterwards. Prior to release 4.00, seteuid() was used in some circumstances, but this is no longer the case.

After a new Exim process has interpreted its command line options, it changes uid and gid in the following cases:
-

If the -C option is used to specify an alternate configuration file, or if the -D option is used to define macro values for the configuration, and the calling process is not running as root or the Exim user, the uid and gid are changed to those of the calling process. However, if ALT_CONFIG_ROOT_ONLY is defined in Local/Makefile, only root callers may use -C and -D without losing privilege, and if DISABLE_D_OPTION is set, the -D option may not be used at all.
-

If the expansion test option (-be) or one of the filter testing options (-bf or -bF) are used, the uid and gid are changed to those of the calling process.
-

If the process is not a daemon process or a queue runner process or a delivery process or a process for testing address routing (started with -bt), the uid and gid are changed to the Exim user and group. This means that Exim always runs under its own uid and gid when receiving messages. This also applies when testing address verification (the -bv option) and testing incoming message policy controls (the -bh option).
-

For a daemon, queue runner, delivery, or address testing process, the uid remains as root at this stage, but the gid is changed to the Exim group.

The processes that initially retain root privilege behave as follows:
-

A daemon process changes the gid to the Exim group and the uid to the Exim user after setting up one or more listening sockets. The initgroups() function is called, so that if the Exim user is in any additional groups, they will be used during message reception.
-

A queue runner process retains root privilege throughout its execution. Its job is to fork a controlled sequence of delivery processes.
-

A delivery process retains root privilege throughout most of its execution, but any actual deliveries (that is, the transports themselves) are run in subprocesses which always change to a non-root uid and gid. For local deliveries this is typically the uid and gid of the owner of the mailbox; for remote deliveries, the Exim uid and gid are used. Once all the delivery subprocesses have been run, a delivery process changes to the Exim uid and gid while doing post-delivery tidying up such as updating the retry database and generating bounce and warning messages.

While the recipient addresses in a message are being routed, the delivery process runs as root. However, if a user’s filter file has to be processed, this is done in a subprocess that runs under the individual user’s uid and gid. A system filter is run as root unless system_filter_user is set.
-

A process that is testing addresses (the -bt option) runs as root so that the routing is done in the same environment as a message delivery. 51.3 Running Exim without privilege

Some installations like to run Exim in an unprivileged state for more of its operation, for added security. Support for this mode of operation is provided by the global option deliver_drop_privilege. When this is set, the uid and gid are changed to the Exim user and group at the start of a delivery process (and also queue runner and address testing processes). This means that address routing is no longer run as root, and the deliveries themselves cannot change to any other uid.

Leaving the binary setuid to root, but setting deliver_drop_privilege means that the daemon can still be started in the usual way, and it can respond correctly to SIGHUP because the re-invocation regains root privilege.

An alternative approach is to make Exim setuid to the Exim user and also setgid to the Exim group. If you do this, the daemon must be started from a root process. (Calling Exim from a root process makes it behave in the way it does when it is setuid root.) However, the daemon cannot restart itself after a SIGHUP signal because it cannot regain privilege.

It is still useful to set deliver_drop_privilege in this case, because it stops Exim from trying to re-invoke itself to do a delivery after a message has been received. Such a re-invocation is a waste of resources because it has no effect.

If restarting the daemon is not an issue (for example, if mua_wrapper is set, or inetd is being used instead of a daemon), having the binary setuid to the Exim user seems a clean approach, but there is one complication:

In this style of operation, Exim is running with the real uid and gid set to those of the calling process, and the effective uid/gid set to Exim’s values. Ideally, any association with the calling process’ uid/gid should be dropped, that is, the real uid/gid should be reset to the effective values so as to discard any privileges that the caller may have. While some operating systems have a function that permits this action for a non-root effective uid, quite a number of them do not. Because of this lack of standardization, Exim does not address this problem at this time.

For this reason, the recommended approach for “mostly unprivileged” running is to keep the Exim binary setuid to root, and to set deliver_drop_privilege. This also has the advantage of allowing a daemon to be used in the most straightforward way.

If you configure Exim not to run delivery processes as root, there are a number of restrictions on what you can do:
-

You can deliver only as the Exim user/group. You should explicitly use the user and group options to override routers or local transports that normally deliver as the recipient. This makes sure that configurations that work in this mode function the same way in normal mode. Any implicit or explicit specification of another user causes an error.
-

Use of .forward files is severely restricted, such that it is usually not worthwhile to include them in the configuration.
-

Users who wish to use .forward would have to make their home directory and the file itself accessible to the Exim user. Pipe and append-to-file entries, and their equivalents in Exim filters, cannot be used. While they could be enabled in the Exim user’s name, that would be insecure and not very useful.
-

Unless the local user mailboxes are all owned by the Exim user (possible in some POP3 or IMAP-only environments):
-

They must be owned by the Exim group and be writable by that group. This implies you must set mode in the appendfile configuration, as well as the mode of the mailbox files themselves.
-

You must set no_check_owner, since most or all of the files will not be owned by the Exim user.
-

You must set file_must_exist, because Exim cannot set the owner correctly on a newly created mailbox when unprivileged. This also implies that new mailboxes need to be created manually.

These restrictions severely restrict what can be done in local deliveries. However, there are no restrictions on remote deliveries. If you are running a gateway host that does no local deliveries, setting deliver_drop_privilege gives more security at essentially no cost.

If you are using the mua_wrapper facility (see chapter 47), deliver_drop_privilege is forced to be true. 51.4 Delivering to local files

Full details of the checks applied by appendfile before it writes to a file are given in chapter 26. 51.5 IPv4 source routing

Many operating systems suppress IP source-routed packets in the kernel, but some cannot be made to do this, so Exim does its own check. It logs incoming IPv4 source-routed TCP calls, and then drops them. Things are all different in IPv6. No special checking is currently done. 51.6 The VRFY, EXPN, and ETRN commands in SMTP

Support for these SMTP commands is disabled by default. If required, they can be enabled by defining suitable ACLs. 51.7 Privileged users

Exim recognises two sets of users with special privileges. Trusted users are able to submit new messages to Exim locally, but supply their own sender addresses and information about a sending host. For other users submitting local messages, Exim sets up the sender address from the uid, and doesn’t permit a remote host to be specified.

However, an untrusted user is permitted to use the -f command line option in the special form -f <> to indicate that a delivery failure for the message should not cause an error report. This affects the message’s envelope, but it does not affect the Sender: header. Untrusted users may also be permitted to use specific forms of address with the -f option by setting the untrusted_set_sender option.

Trusted users are used to run processes that receive mail messages from some other mail domain and pass them on to Exim for delivery either locally, or over the Internet. Exim trusts a caller that is running as root, as the Exim user, as any user listed in the trusted_users configuration option, or under any group listed in the trusted_groups option.

Admin users are permitted to do things to the messages on Exim’s queue. They can freeze or thaw messages, cause them to be returned to their senders, remove them entirely, or modify them in various ways. In addition, admin users can run the Exim monitor and see all the information it is capable of providing, which includes the contents of files on the spool.

By default, the use of the -M and -q options to cause Exim to attempt delivery of messages on its queue is restricted to admin users. This restriction can be relaxed by setting the no_prod_requires_admin option. Similarly, the use of -bp (and its variants) to list the contents of the queue is also restricted to admin users. This restriction can be relaxed by setting no_queue_list_requires_admin.

Exim recognises an admin user if the calling process is running as root or as the Exim user or if any of the groups associated with the calling process is the Exim group. It is not necessary actually to be running under the Exim group. However, if admin users who are not root or the Exim user are to access the contents of files on the spool via the Exim monitor (which runs unprivileged), Exim must be built to allow group read access to its spool files. 51.8 Spool files

Exim’s spool directory and everything it contains is owned by the Exim user and set to the Exim group. The mode for spool files is defined in the Local/Makefile configuration file, and defaults to 0640. This means that any user who is a member of the Exim group can access these files. 51.9 Use of argv[0]

Exim examines the last component of argv[0], and if it matches one of a set of specific strings, Exim assumes certain options. For example, calling Exim with the last component of argv[0] set to “rsmtp” is exactly equivalent to calling it with the option -bS. There are no security implications in this. 51.10 Use of %f formatting

The only use made of “%f” by Exim is in formatting load average values. These are actually stored in integer variables as 1000 times the load average. Consequently, their range is limited and so therefore is the length of the converted output. 51.11 Embedded Exim path

Exim uses its own path name, which is embedded in the code, only when it needs to re-exec in order to regain root privilege. Therefore, it is not root when it does so. If some bug allowed the path to get overwritten, it would lead to an arbitrary program’s being run as exim, not as root. 51.12 Use of sprintf()

A large number of occurrences of “sprintf” in the code are actually calls to string_sprintf(), a function that returns the result in malloc’d store. The intermediate formatting is done into a large fixed buffer by a function that runs through the format string itself, and checks the length of each conversion before performing it, thus preventing buffer overruns.

The remaining uses of sprintf() happen in controlled circumstances where the output buffer is known to be sufficiently long to contain the converted string. 51.13 Use of debug_printf() and log_write()

Arbitrary strings are passed to both these functions, but they do their formatting by calling the function string_vformat(), which runs through the format string itself, and checks the length of each conversion. 51.14 Use of strcat() and strcpy() These are used only in cases where the output buffer is known to be large enough to hold the result.