NAME

mixminion − Type III anonymity server

SYNOPSIS

mixminiond start [**−-h** | **−-help**] [**−-Q** | **−-quiet**] [**−-f** *file* | **−-config=***file*]
[**−-daemon** | **−-nodaemon**] [**−-echo**] [**−-severity=***level*]

mixminion stop [**−-h** | **−-help**] [**−-f** *file* | **−-config=***file*]
mixminion reload [**−-h** | **−-help**] [**−-f** *file* | **−-config=***file*]
mixminion republish [**−-h** | **−-help**] [**−-f** *file* | **−-config=***file*]
mixminion DELKEYS [**−-h** | **−-help**] [**−-f** *file* | **−-config=***file*]
mixminion stats [**−-h** | **−-help**] [**−-f** *file* | **−-config=***file*]
mixminion upgrade [**−-h** | **−-help**] [**−-f** *file* | **−-config=***file*]

Deprecated:
mixminion { **server-start** | **server-stop** | **server-reload** | **server-republish**
server-DELKEYS | **server-stats** | **server-upgrade** }

DESCRIPTION

Mixminion is software suite that lets you send and receive very anonymous mail via the "Type III" remailer protocol. `mixminiond`(8) is the standard interface for running a Mixminion server.

To configure a Mixminion server, follow these steps.

1. Optionally, create a separate account on your system for the mixminiond user. This step is recommended.

2. Create a configuration file. The easiest way to do this is by editing the file `etc/mixminiond.conf` from the Mixminion distribution. See `mixminiond.conf`(5) for more information on configuration options.

3. Install the configuration file in one of: `~/.mixminiond.conf ~/etc/mixminiond.conf, or /etc/mixminiond.conf`. (You may store it elsewhere, but you will need to specify the location on the command line when you start mixminion.)

4. To start your server, run: **mixminiond start** [**−f** *path to mixminiond.conf*]

   (The **−f** option is only necessary if you placed the file somewhere besides one of the default locations.)

5. To try out your server, clients will need a copy of your server descriptor, whose location is stored in a file named `current-desc` under your server's base directory.

   For example, if your mixminiond.conf file contains the line: "Homedir: /home/mixminion/spool", then if you read the contents of `/home/mixminion/spool/current-desc`, you will file a filename like "/home/mixminion/spool/keys/key_0001_ServerDesc". This file is your current server descriptor.

   To try using this server descriptor, send messages using the filename as part of your path:
   mixminion send -t <addr> -P '<filename>,*2'

6. When you're ready to advertise your server, edit 'mixminiond.conf' and set the 'Publish' option to 'yes'. When you restart your server, it will advertise itself to the central directory.

   The first time you do this, your server will not be inserted automatically; the directory server holds your server's information separately until I confirm it (to prevent pseudospoofing). Once your server is listed, future updates will be get into the directory automatically.

WARNING: We don't have statistics yet, so the system isn't robust in the presence of unreliable servers in the directory. Please don't publish a server if you don't think you can keep it up for a good while.

Once invoked, the **`mixminiond`** process tries to perform all the tasks necessary to implement the Type III anonymous remailer protocol correctly. These include

Listening on a network port and accepting incoming Type III packets via the "Mixminion Transfer Protocol" (MMTP).

Decrypting, storing, re-ordering, delaying, and scheduling outgoing packets for delivery.

Delivering outgoing packets via MMTP.

Delivering outgoing messages via email (SMTP).

Discarding invalid packets.

Reassembling fragmented messages before delivery.

Advertising its presence to the directory server(s).

Periodically downloading fresh directories.

Generating new keys as its old ones expire.

INVOKING MIXMINIOND

Like `mixminion`(1), **`mixminiond`** expects as its first argument a command name, and expects options for that command as subsequent arguments. To invoke a specific command, call **`mixminiond`** *`command_name`*. The supported commands are:

`start` Begin running a mixminiond process. Depending on the value of the *Daemon* variable in the configuration file, the process will run in the foreground, or the background.

`stop` Safely shutdown a mixminiond process. You can also do this by sending a KILL signal to the process (on Unix).

`reload` Tell a mixminiond process to reload its configuration data. You can also do this by sending a HUP signal to the process (on Unix). (This isn't implemented yet; right now, **`mixminiond .Cm reload`** only closes and re-opens the log files.)

`republish` Tell a mixminiond process to re-publish all of its server descriptors to the directory servers, whether it has already done so or not. This is almost never necessary anymore.

`DELKEYS` Delete keys from the server's directory. This can be handy for some forms of disaster recovery, but is almost never necessary anymore.

`stats` Dump statistics for the server's current time period. (Old statistics are stored a file, configurable with the *StatsFile* option in `mixminiond.conf`(5)).

`upgrade` Upgrade an older server's file formats. The last forward-incompatible format change was between 0.0.4 and 0.0.5, but future incompatible changes are possible. (Backward-incompatible format changes are a matter of course, and will be for as long as the software is in alpha.)

Every command can take takes one or more options. The supported options are listed below, along with a summary of which command support them:

`−−daemon`
{**`mixminiond start`**} Run the server in the background, no matter what the configuration file requests. (Unix only.)

-f *filename* | **--config=***filename*
: {all} Load the configuration file from the provided filename, instead of searching in the usual places.

--echo
: {**mixminiond start**} Print log messages to standard output, even if the configuration file requests otherwise. For debugging.

-h | **--help**
: {all} Print a help message and exit.

--nodaemon
: {**mixminiond start**} Run the server in the foreground, no matter what the configuration file requests. For debugging. (Unix only.)

-Q | **--quiet**
: {**mixminiond start**} Don't print non-error messages to standard output.

--severity=*level*
: Log at the requested severity level, no matter what the configuration file requests.

ENVIRONMENT

Mixminion servers recognize the following environment variables:

`http_proxy` If you use a proxy to access the web, you should set this variable so that mixminion can use HTTP to download its directory.

`MM_NO_FILE_PARANOIA`
If set, don't check file permissions on private files.

FILES

The mixminion server stores its files in configurable locations, as configured in `mixminiond.conf`(5). In the list of files below, file locations are given relative to configuration variables. For example, if a file is named `fname` and is stored in a directory configured with the *SomeDir* variable, we describe its location as: `${SomeDir}/fname`.

`mixminiond.conf` Configuration file. When **mixminiond** starts a new server, it checks in a list of standard file locations in order, unless you use the **-f** option to provide a different filename on the command line. See `mixminiond.conf`(5) for information on the file format. The default search path is
1. `$HOME/mixminiond.conf`
2. `$HOME/etc/mixminiond.conf`
3. `/etc/mixminiond.conf`
4. `/etc/mixminion/mixminiond.conf`

`${BaseDir}/current-desc`
A file containing the name of the file holding the current server descriptor.

`${BaseDir}/version` The version of the current file format used by this server. Mixminion 0.0.7 uses "1001"; older software does not use a version at all.

`${WorkDir}` Directory holding volatile non-key data. This defaults to `${BaseDir}/work` the *WorkDir* variable is not set.

`${WorkDir}/tls/dhparameters`
Diffie-Hellman parameters used for MMTP key exchange.

`${WorkDir}/hashlogs/hash_*`
: Logs of packet hashes, used to prevent replay attacks. These files may be stored as Berkeley DB files, as GDBM files, as DBM files, or as flat text files, depending on your Python configuration. Each one corresponds to a separate key set in `${KeyDir}`.

`${WorkDir}/stats.tmp`
: Cache of server statistics from latest period, stored as a Python object. Use the **`mixminiond stats`** command to see the contents of this file.

`${WorkDir}/dir/*`
: Latest server directory, downloaded from the directory server. Currently, this is used to print useful nicknames for other servers.

`${QueueDir}`
: Directory used to hold packets and messages. Defaults to `${WorkDir}/queues`. See "Pool Directories" below for information about files under this directory.

`${QueueDir}/incoming/`
: A pool directory holding packets that have been received via MMTP, but not yet processed.

`${QueueDir}/mix/`
: A pool directory holding packets that have been received and decrypted. Packets are delayed in this directory for a while after receipt in order to prevent blending attacks.

`${QueueDir}/outgoing/`
: A pool directory holding packets for delivery via MMTP.

`${QueueDir}/deliver/`
: A directory holding messages for file outgoing delivery, and files used by various delivery modules to deliver those files.

`${KeyDir}`
: A directory holding private key information. Defaults to `${BaseDir}/keys`. Every subdirectory of ${KeyDir} corresponds to a separate set of keys, with its own lifetime. The **`mixminiond`** server automatically generates new keys as necessary, and deletes them as they expire.

`${KeyDir}/identity.key`
: This server's long-term signing private key.

`${KeyDir}/key_*/ServerDesc`
: A server descriptor corresponding to a single key set.

`${KeyDir}/key_*/mix.key`
: A private key used to decrypt mix packets.

`${KeyDir}/key_*/mmtp.key`
: A private key used for on-the-wire encryption.

`${KeyDir}/key_*/mmtp.cert`
: An X.509 certificate chain used for on-the-wire encryption.

`${KeyDir}/key_*/published`
: This file is present only if the corresponding server descriptor has been published to a directory server.

`${LogFile}`
: A file holding log messages generated by the **`mixminiond`** process. The location defaults to `${BaseDir}/log`.

`${PidFile}`	A file holding the numeric process ID for the current **`mixminiond`** process. While the server is running, this file is locked to prevent multiple servers from running with the same configuration. The location defaults to `${BaseDir}/pid`.
`${StatsFile}`	A file holding a record of packet statistics for the server. The location defaults to `${BaseDir}/stats`.

Note: the only one of these files you should ordinarily be modifying is `.mixminiond.conf`.

Pool Directories

Most of the directories under `${QueueDir}` store messages or packets with a standardized naming format. Each file begins with a prefix, followed by an underline, followed by a random string of characters. All file transitions are performed via the (atomic) `rename`(2) operation, to prevent race conditions or data loss in the event of a crash. The recognized prefixes are:

inp	A message or packet being written to the filesystem. If any of these are found when the server starts, they are assumed to be incomplete messages from a previous run and deleted.
msg	A message or packet. These can either be stored as a raw file, or as a "pickled" Python object, depending on the pool. These formats are not frozen yet.
rmv	A message or packet that has been scheduled for deletion.
crp	A corrupted file that, for some reason, could not be read. These files are not deleted automatically, since their presence implies a bug that needs to be addressed. If you find any of these, please report a bug.
inpm	Metadata being written; Corresponds to "inp".
meta	A metadata file for a given message. These files are usually "pickled" Python objects of some kind. These formats are not frozen yet.
rmvm	Metadata being removed; Corresponds to "rmv".
crpm	Corrupted metadata; Corresponds to "crp".

SEE ALSO

`mixminion`(1), `mixminiod.conf`(5)

AUTHORS

See the AUTHORS section in `mixminion`(1)

ACKNOWLEDGMENTS

The Mixminion software is by Nick Mathewson, with contributions by Roger Dingledine, Brian Fordham, Lucky Green, Peter Palfrader, Robyn Wagner, Brian Warner, and Bryce "Zooko" Wilcox-O'Hearn.

BUGS

Future releases will probably break backward compatibility with this release at least once or twice.

See the manpage for `mixminion`(1) for information on other bugs, and instructions for reporting bugs.