NAME

swtpm − TPM Emulator for TPM 1.2 and 2.0

SYNOPSIS

swtpm socket [OPTIONS]

swtpm chardev [OPTIONS]

swtpm cuse [OPTIONS]

DESCRIPTION

swtpm implements a TPM software emulator built on libtpms. It provides access to TPM functionality over a TCP/IP socket interface or it can listend for commands on a character device, or create a CUSE (character device in userspace) interface for receiving of TPM commands.

Unless corresponding command line parameters are used, the **swtpm** socket version requires that the environment variable *TPM_PORT* be set to the TCP/IP port the process is supposed to listen on for TPM request messages.

Similarly, the environment variable *TPM_PATH* can be set and contain the name of a directory where the TPM can store its persistent state into.

The **swtpm** process can be gracefully terminated by sending a *SIGTERM* signal to it.

The **swtpm** cuse version requires root rights to start the TPM.

Options for socket interface

The following options are supported if the *socket* interface is chosen:

−p|−−port <port>
Use the given port rather than using the environment variable TPM_PORT.

−t|−−terminate
Terminate the TPM after the client has closed the connection.

−−server [type=tcp][,port=<port>[,bindaddr=<address>[,ifname=<ifname>]]][,fd=<fd>][,disconnect]
Expect TCP connections on the given port; if a port is not provided a file descriptor must be passed with the fd parameter and the commands are read from this file descriptor then. If a port is provided the *bind address* on which to listen for TCP connections can be provided as well; the default bind address is 127.0.0.1. If a link local IPv6 address is provided, the name of the interface to bind to must be provided with *ifname*.

This parameter enables a persistent connection by default unless the disconnect option is given. This parameter should be used rather than the −p and −−fd options.

−−server type=unixio[,path=<path>][,fd=<fd>] [,mode=<0...>][,uid=<uid>][,gid=<gid>]
Expect UnixIO connections on the given path. If no path is provided, a file descriptor must be passed instead. The mode parameter allows a user to set the file mode bits of the UnixIO path. The mode bits value must be given as an octal number starting with a ’0’. The default value is 0770. uid and gid set the ownership of the UnixIO socket’s path. This operation requires root privileges.

Options for character device interface

The following options are supported if the *chardev* interface is chosen:

−c|−−chardev <device path>
Use the given device to listen for TPM commands and send response on.

−−vtpm−proxy
Create a Linux vTPM proxy device instance and read TPM commands from its backend device.

Options for the CUSE interface

The following options are supported if the *cuse* interface is chosen:

−n|−−name <NAME>
The TPM will use a device with the given name. A device with the given name will be created in /dev. This is a mandatory option.

−M|−−maj <MAJOR>
Create the device with the given major number.

−m|−−min <MINOR>
Create the device with the given minor number.

Options for socket and character device interfaces:

The following options are supported by the socket and character device interfaces:

−f|−−fd <fd>
Use the given socket file descriptor or character device file descriptor for receiving TPM commands and sending responses. For the socket interface, this option automatically assumes −t.

−d|−−daemon
Daemonize the process.

−−ctrl type=[unixio|tcp][,path=<path>] [,port=<port>[,bindaddr=<address>[,ifname=<ifname>]]] [,fd=<filedescriptor>|clientfd=<filedescriptor>] [,mode=<0...>][,uid=<uid>][,gid=<gid>]
This option adds a control channel to the TPM. The control channel can either use a UnixIO socket with a given *path* or *filedescriptor* or it can use a TCP socket on the given *port* or *filedescriptor*. If a port is provided the *bind address* on which to listen for TCP connections can be provided as well; the default bind address is 127.0.0.1. If a link local IPv6 address is provided, the name of the interface to bind to must be provided with *ifname*.

The mode parameter allows a user to set the file mode bits of the UnixIO path. The mode bits value must be given as an octal number starting with a '0'. The default value is 0770. uid and gid set the ownership of the UnixIO socket's path. This operation requires root privileges.

The control channel enables out-of-band control of the TPM, such as resetting the TPM.

Options for all interfaces

The following options are support by all interfaces:

−−tpmstate dir=<dir>[,mode=<0...>]
Use the given path rather than using the environment variable TPM_PATH.

The TPM state files will be written with the given file mode bits. This value must be given as an octal number starting with a '0'. The default value is 0640.

−−tpm2
Choose TPM 2 functionality; by default a TPM 1.2 is chosen.

−−log [fd=<fd>|file=<path>][,level=<n>] [,prefix=<prefix>][,truncate]
Enable logging to a file given its file descriptor or its path. Use '−' for path to suppress the logging.

The level parameter allows a user to choose the level of logging. Starting at log level 5, libtpms debug logging is activated.

All logged lines will be prefixed with prefix. By default no prefix is prepended.

If *truncate* is passed, the log file will be truncated.

−−locality reject−locality−4[,allow−set−locality]
The *reject−locality−4* parameter will cause TPM error messages to be returned for requests to set the TPM into locality 4.

The *allow-set-locality* parameter allows the swtpm to receive TPM/TPM2_SetLocality commands. This is parameter is useful if the Linux VTPM proxy driver access is enabled by file descriptor passing. This option is implied by the *−−vtpm−proxy* option and therefore need not be explicitly set if this option is passed. In all other cases care should be taken as to who can send the TPM/TPM2_SetLocality command.

−−key file=<keyfile>|fd=<fd> [,format=<hex|binary>][,mode=aes−cbc|aes−256−cbc], [remove[=true|false]]

Enable encryption of the state files of the TPM. The keyfile must contain an AES key of supported size; 128 bit (16 bytes) and 256 bit (32 bytes) keys are supported.

The key may be in binary format, in which case the file size must be 16 or 32 bytes. If the key is in hex format (default), the key may consist of 32 or 64 hex digits starting with an optional '0x'.

The *mode* parameter indicates which block chaining mode is to be used. Currently aes-cbc (aes−128−cbc) and aes−256−cbc are supported. The encrypted data is integrity protected using encrypt-then-mac.

The *remove* parameter will attempt to remove the given keyfile once the key has been read.

−−key pwdfile=<passphrase file>|pwdfd=<fd> [,mode=aes−cbc|aes−256−cbc][remove[=true|false]][,kdf=sha512|pbkdf2]

This variant of the key parameter allows a user to provide a passphrase in a file. The file is read and a key is derived from it using either a SHA512 hash or PBKDF2. By default PBKDF2 is used.

−−migration−key file=<keyfile>|fd=<fd> [,format=<hex|binary>][,mode=aes−cbc|aes−256−cbc] [,remove[=true|false]]

The availability of a migration key ensures that the state of the TPM will not be revealed in unencrypted form when the TPM state blobs are retrieved through the ioctl interface. The migration key is not used for encrypting TPM state written to files, this is what the *−−key* parameter is used for.

The migration key and the key used for encrypting the TPM state files may be the same.

While the key for the TPM state files needs to stay with those files it encrypts, the migration key needs to stay with the TPM state blobs. If for example the state of the TPM is migrated between hosts in a data center, then the TPM migration key must be available at all the destinations, so in effect it may have to be a key shared across all machines in the datacenter. In contrast to that, the key used for encrypting the TPM state **files** can be different for each TPM and need only be available on the host where the TPM state resides.

The migration key enables the encryption of the TPM state blobs. The keyfile must contain an AES key of supported size; 128 bit (16 bytes) and 256 bit (32 bytes) keys are supported.

The key may be in binary format, in which case the file size must be 16 or 32 bytes. If the key is in hex format (default), the key may consist of 32 or 64 hex digits starting with an optional '0x'.

The *mode* parameter indicates which block chaining mode is to be used. Currently aes-cbc (aes−128−cbc) and aes−256−cbc are supported. The encrypted data is integrity protected using encrypt-then-mac.

The *remove* parameter will attempt to remove the given keyfile once the key has been read.

−−migration−key pwdfile=<passphrase file>|pwdfd=<fd> [,mode=aes−cbc|aes−256−cbc][,remove[=true|false]][,pdf=sha512|pbkdf2]

This variant of the key parameter allows a user to provide a passphrase in a file. The file is read and a key is derived from it using either a SHA512 hash or PBKDF2. By default PBKDF2 is used.

−−pid file=<pidfile>|fd=<filedescriptor>

This options allows a user to set the name of file where the process ID (pid) of the TPM will be written into. It is also possible to pass a file descriptor to a file that has been opened for writing.

−r|−−runas <owner>

Switch to the given user. This option can only be used when swtpm is started as root.

−−seccomp action=none|log|kill (since v0.2)

This option allows a user to select the action to take by the seccomp profile when a syscall is executed that is not allowed. The default is *kill*. To disable the seccomp profile, choose *none*. The *log* action logs offending syscalls. The *log* action is only available if libseccomp supports logging.

This option is only available on Linux and only if swtpm was compiled with libseccomp support.

−−flags [not−need−init] [,startup−clear|startup−state|startup−deactivated|startup−none]
The *not-need-init* flag enables the TPM to accept TPM commands right after start without requiring an INIT to be sent to it through the command channel (see the '−i' option of swtpm_ioctl).

The *startup* options cause a TPM_Startup or TPM2_Startup command to automatically be sent. The *startup-deactivated* option is only valid for a TPM 1.2. These options imply *not-need-init*, except for the *startup-none* option, which results in no command being sent.

If *−−vtpm−proxy* is used, *startup-clear* is automatically chosen but this can be changed with this option.

−−print−capabilities (since v0.2)
Print capabilities that were added to swtpm after version 0.1. The output may contain the following:

```
{
  "type": "swtpm",
  "features": [
    "cmdarg−seccomp",
    "cmdarg−key−fd",
    "cmdarg−pwd−fd",
    "tpm−send−command−header",
    "flags−opt−startup",
    "rsa−keysize−1024",
    "rsa−keysize−2048",
    "rsa−keysize−3072"
  ]
}
```

The meaning of the feature verbs is as follows:

cmdarg-seccomp
The *−−seccomp* option is supported.

cmdarg-key-fd
The *−−key* option supports the *fd=* parameter.

cmdarg-pwd-fd
The *−−key* option supports the *pwdfd=* parameter.

tpm-send-command-header
The TPM 2 commands may be prefixed by a header that carries a 4−byte command, 1 byte for locality, and 4−byte TPM 2 command length indicator. The TPM 2 will respond by preprending a 4−byte response indicator and a 4−byte trailer. All data is sent in big endian format.

flags-opt-startup
The *−−flags* option supports the *startup−...* options.

rsa−keysize−2048
The TPM 2 supports the shown RSA key sizes. If none of the rsa-keysize verbs is shown then only RSA 2048 bit keys are supported.

−h|−−help
Display usage info.

SEE ALSO

swtpm_bios, **swtpm_cuse**