NAME

request-key.conf - Instantiation handler configuration file

DESCRIPTION

This file and its associated key-type specific variants are used by the /sbin/request-key program to determine which program it should run to instantiate a key.

request-key looks first in /etc/request-key.d/ for a file of the key type name plus ".conf" that it can use. If that is not found, it will fall back to /etc/request-key.conf.

request-key scans through the chosen file one line at a time until it finds a match, which it will then use. If it doesn't find a match, it'll return an error and the kernel will automatically negate the key.

Any blank line or line beginning with a hash mark '#' is considered to be a comment and ignored.

All other lines are assumed to be command lines with a number of white space separated fields:

<op> <type> <description> <callout-info> <prog> <arg1> <arg2> ...

The first four fields are used to match the parameters passed to request-key by the kernel. *op* is the operation type; currently the only supported operation is "create".

type, *description* and *callout-info* match the three parameters passed to **keyctl request2** or the **request_key()** system call. Each of these may contain one or more asterisk '*' characters as wildcards anywhere within the string.

Should a match be made, the program specified by <prog> will be exec'd. This must have a fully qualified path name. argv[0] will be set from the part of the program name that follows the last slash '/' character.

If the program name is prefixed with a pipe bar character '|', then the program will be forked and exec'd attached to three pipes. The callout information will be piped to it on it's stdin and the intended payload data will be retrieved from its stdout. Anything sent to stderr will be posted in syslog. If the program exits 0, then /sbin/request-key will attempt to instantiate the key with the data read from stdout. If it fails in any other way, then request-key will attempt to execute the appropriate 'negate' operation command.

The program arguments can be substituted with various macros. Only complete argument substitution is supported - macro substitutions can't be embedded. All macros begin with a percent character '%'. An argument beginning with two percent characters will have one of them discarded.

The following macros are supported:

%o Operation type
%k Key ID
%t Key type
%d Key description
%c Callout information
%u Key UID
%g Key GID
%T Requestor's thread keyring
%P Requestor's process keyring
%S Requestor's session keyring

There's another macro substitution too that permits the interpolation of the contents of a key:

%{<type>:<description>}

This performs a lookup for a key of the given type and description on the requestor's keyrings, and if found, substitutes the contents for the macro. If not found an error will be logged and the key under construction will be negated.

EXAMPLE

A basic file will be installed in the /etc. This will contain two debugging lines that can be used to test the installation:

```
create user debug:* negate /bin/keyctl negate %k 30 %S
```

```
create user debug:loop:* * |/bin/cat
create user debug:* * /usr/share/keyutils/request-key-debug.sh %k %d %c %S
negate * * * /bin/keyctl negate %k 30 %S
```

This is set up so that something like:

```
keyctl request2 user debug:xxxx negate
```

will create a negative user-defined key, something like:

```
keyctl request2 user debug:yyyy spoon
```

will create an instantiated user-defined key with "Debug spoon" as the payload, and something like:

```
keyctl request2 user debug:loop:zzzz abcdefghijkl
```

will create an instantiated user-defined key with the callout information as the payload.

FILES

/etc/request-key.conf
/etc/request-key.d/<keytype>.conf

SEE ALSO

keyctl(1), **request-key.conf**(5)