NAME

ipsec_set_policy, **ipsec_get_policylen**, **ipsec_dump_policy** **-** create an IPsec policy structure from a human readable string

LIBRARY

IPsec Policy Control Library (libipsec, -lipsec)

SYNOPSIS

#include <netipsec/ipsec.h>

*char **
ipsec_set_policy(*char *policy*, *int len*);

int
ipsec_get_policylen(*char *buf*);

*char **
ipsec_dump_policy(*char *buf*, *char *delim*);

DESCRIPTION

The **ipsec_set_policy**() function generates an IPsec policy specification structure, struct sadb_x_policy and/or struct sadb_x_ipsecrequest from a human-readable policy specification. The policy specification must be given as a C string, passed in the *policy* argument and the length of the string, given as *len*. The **ipsec_set_policy**() function returns pointer to a buffer which contains a properly formed IPsec policy specification structure. The buffer is dynamically allocated, and must be freed by using the free(3) library function.

The **ipsec_get_policylen**() function returns the length of the buffer which is needed when passing the specification structure to the setsockopt(2) system call.

The **ipsec_dump_policy**() function converts an IPsec policy structure into a human readable form. The *buf* argument points to an IPsec policy structure, struct sadb_x_policy. *delim* is a delimiter string, which is usually a blank character. If you set *delim* to NULL, a single white space is assumed. The **ipsec_dump_policy**() function returns a pointer to dynamically allocated string. It is the caller’s responsibility to free the returned pointer using the free(3) library call.

A *policy* is given in the following way:

direction discard

The *direction* must be in or out and specifies which direction the policy needs to be applied, either on inbound or outbound packets. When the discard policy is selected, packets will be dropped if they match the policy.

direction entrust

entrust means to consult the security policy database (SPD) in the kernel, as controlled by setkey(8).

direction bypass

A direction of bypass indicates that IPsec processing should not occur and that the packet will be transmitted in clear. The bypass option is only available to privileged sockets.

direction ipsec *request* ...

A direction of ipsec means that matching packets are processed by IPsec. ipsec can be followed by one or more *request* string, which is formatted as:

protocol / *mode* / *src* - *dst* [*/level*]

The *protocol* is one of: ah, esp or ipcomp indicating Authentication Header, Encapsulating Security Protocol or IP Compression protocol is used.

The *mode* is either transport or tunnel the meanings of both modes are described in ipsec(4).

The *src* and *dst* specify the IP address, either v4 or v6, of the source and destination systems. The *src* always stands for the "sending node" and *dst* always stands for the "receiving node". When *direction* is in, *dst* is this local node and *src* is the remote node or peer. If *mode* is transport, both *src* and *dst* can be omitted.

The *level* must be set to one of the following: default, use, require or unique. default means that the kernel should consult the default security policies as defined by a set of sysctl(8), variables. The relevant sysctl(8) variables are described in ipsec(4).

When use is selected a relevant security association (SA) can be used when available but is not necessary. If the SA is available then packets will be handled by IPsec, i.e., encrypted and/or authenticated but if an SA is not available then packets will be transmitted in the clear. The use option is not recommended because it allows for accidental mis-configurations where encrypted or authenticated link becomes unencrypted or unauthenticated, the require keyword is recommended instead of use where possible. Using the require keyword means that a relevant SA is required, and that the kernel must perform IPsec processing on all matching packets.

The unique keyword has the same effect as require, but adds the restriction that the SA for outbound traffic is used only for this policy. You may need the identifier in order to relate the policy and the SA when you define the SA by manual keying using setkey(8). Put the decimal number as the identifier after the unique keyword in this way: unique: number, where number must be between 1 and 32767.

If the *request* string is kept unambiguous, *level* and the slash prior to *level* can be omitted but you are encouraged to specify them explicitly to avoid unintended behaviors. If *level* is omitted, it will be interpreted as default.

Note that there is a difference between the specification allowed here and in setkey(8). When specifying security policies with setkey(8), neither entrust nor bypass are used. Refer to setkey(8) for details.

RETURN VALUES

The **ipsec_set_policy**() function returns a pointer to the allocated buffer containing a the policy specification if successful; otherwise a NULL pointer is returned.

The **ipsec_get_policylen**() function returns a positive value, indicating the buffer size, on success, and a negative value on error.

The **ipsec_dump_policy**() function returns a pointer to a dynamically allocated region containing a human readable security policy on success, and NULL on error.

EXAMPLES

Set a policy that all inbound packets are discarded.

```
in discard
```

All outbound packets are required to be processed by IPsec and transported using ESP.

```
out ipsec esp/transport//require
```

All inbound packets are required to be authenticated using the AH protocol.

```
in ipsec ah/transport//require
```

Tunnel packets outbound through the endpoints at 10.1.1.2 and 10.1.1.1.

```
out ipsec esp/tunnel/10.1.1.2-10.1.1.1/require
```

SEE ALSO

ipsec_strerror(3), ipsec(4), setkey(8)

HISTORY

These functions first appeared in WIDE/KAME IPv6 protocol stack kit.

IPv6 and IPsec support based on the KAME Project (http://www.kame.net/) stack was initially integrated into FreeBSD 4.0.