iptable reference

git.pdh

iptable.h

Structures

`entry_t`

The type `entry_t` has 2 members:

- `rn[2]`, an array of two radix nodes: a leaf & an internal node.
- `void *value`, which points to user data.

The radix tree stores/retrieves pointers to `radix leaf nodes` using binary keys. So a user data structure must begin with an array of two radix nodes: a leaf node for storing the binary key, associated with the user data, and an internal node to actually insert the leaf node into the tree.

Retrieving the data is done by matching, either exact or using a longest prefix match, againts a binary key which yields a pointer to a leaf node. That pointer is then recast to `entry_t *` in order to access the user data associated with the matched binary key in the tree via the `value` pointer.

iptable.c

Structures

`all_ones`

```
uint8_t all_ones[RDX_MAX_KEYLEN] = {-1, .., -1};
```

`all_ones` serves as an AF family's default mask in case one is needed, but not supplied. Thus missing masks are interpreted as host masks.

Key functions

`key_alloc`

```
  uint8_t *key_alloc(int af);
```

Allocate space for a binary key of AF family type `af` and return its pointer. Supported `af` types include: - `AF_INET`, for ipv4 protocol family - `AF_INET6`, for the ipv6 protocol family

`key_copy`

```
   uint8_t *key_copy(uint8_t *src);
```

Copies binary key `src` into newly allocated space and returns its pointer. The src key must have a valid first length byte. $0 <= KEY\_LEN(src) <= MAX\_KEYLEN$. Returns `NULL` on failure.

`key_bystr`

```
   uint8_t *key_bystr(uint8_t *dst, int *mlen, int *af, const char *s);
```

Store string s' binary key in dst. Returns NULL on failure. Also sets mlen and af. mlen=-1 when no mask was supplied. Assumes dst size MAX_BINKEY, which fits both ipv4/ipv6.

int key_bypair(a, b, m)