

C-Bindings for BTNS APIs

Miika Komu <miika@iki.fi>

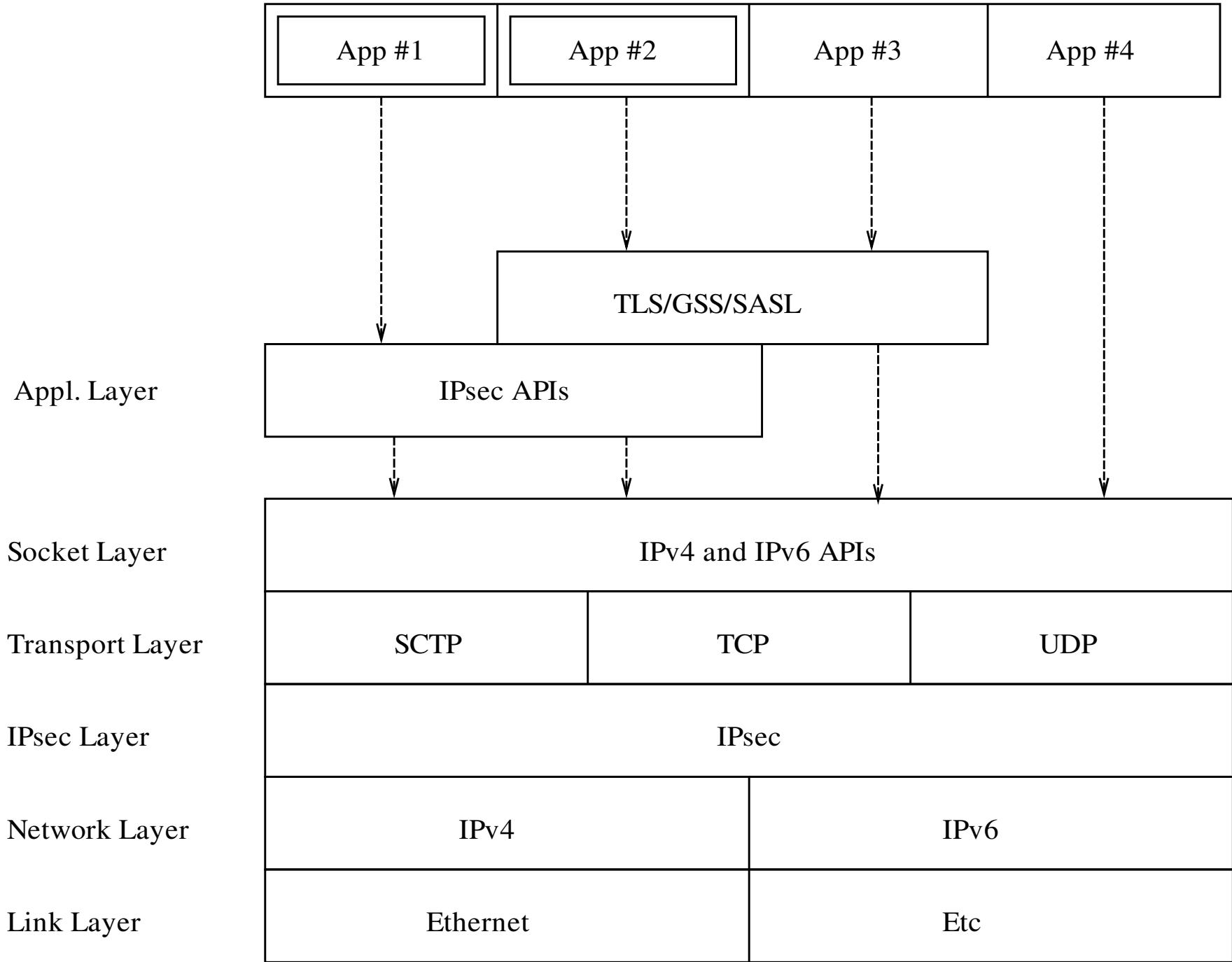
Sasu Tarkoma <sasu.tarkoma@hiit.fi>

Nicolas Williams <
nicolas.williams@sun.com>

Michael Richardson
<mcr@sandelman.ottawa.on.ca>

What Problem Are We Solving?

- How does an network application know that a connection is secured by IPsec?
- How can the application tell explicitly that the use of BTNS extensions is ok?
- How to verify channel bindings?
- How to do this in a portable way in the sockets API in C-language?



Relationship to GSS/SASL

- GSS/SASL APIs deal with upper layer security
 - GSS/SASL APIs are not based on socket descriptors
- IPsec APIs deal with lower layer security
 - IPsec can be used without any changes in the application
 - IPsec APIs are based on socket descriptors
- IPsec APIs can be used in an application simultaneously with GSS or SASL

API Overview

- Applications set their IPsec connection preferences and requirements using “attributes”
 - Applications can get and set attributes (with some limitations)
- Attributes are bundled into two different groups
 - Policies (protection tokens) describe **how** IPsec protection should be done (BTNS_OK, IPsec algo profiles)
 - Identities describe **who** we are and who the peer is. Used for channel bindings.
- Relationship:
 - Identity token -> protection token -> socket

Data structures

- Based on opaque data structures
 - Dynamically allocated
 - Application is not supposed to peek inside
 - Instead, apps use accessors
 - Benefits
 - Easier to extend and add new information
 - Application binaries remain backwards compatible
- Attribute names should be strings, not numbers
 - Users want to use strings: e.g. telnet -btns myserver
 - Developers don't have to worry about numeric conversion

Policies (protection tokens)

```
typedef ipsec_policy_t struct ipsec_policy;

ipsec_policy_t *ipsec_create_policy(uint32_t type);
int ipsec_free_policy(ipsec_policy_t *policy);

int ipsec_get_policy_attr(const ipsec_policy_t *policy,
                        uint32_t attr_type,
                        uint32_t *attr_len,
                        void **attr_val);
int ipsec_set_policy_attr(ipsec_policy_t *policy,
                        uint32_t attr_type,
                        uint32_t attr_len,
                        const void *attr_val);

int ipsec_set_socket_policy(int fd, const ipsec_policy_t *policy);
int ipsec_get_socket_policy(int fd, ipsec_policy_t **policy);

int ipsec_get_policy_attr(const ipsec_policy_t *policy,
                        uint32_t attr_type,
                        uint32_t *attr_len,
                        void **attr_val);
int ipsec_set_policy_attr(ipsec_policy_t *policy,
                        uint32_t attr_type,
                        uint32_t attr_len,
                        const void *attr_val);
```


What You Can Do With the API

- Compare and verify that two connections or datagrams share similar protection
- Compare and verify that two connections or datagrams share same identities
- Set the local/peer identity before connection or datagram recv/delivery
- Application can explicitly say that it is ok to use BTNS
- Reuse upper layer security credentials for IPsec

ChangeLog from 00 to 01

- 00 included only ideas, but 01 contains concrete API definitions
 - Hello world applications in the appendix
- Based on comments from Nicolas Williams, Michael Richardson, Love Åstrand and Julien Laganier

Todo list 1/3

- SASL/GSS code examples and more use cases
- Storing of channel bindings to disk
 - Binary descriptions (similar to GSS_export)
- Channel binding is not a settable thing
- Associate channel bindings with protection tokens, not with sockets
- Remove “prevent IKE authentication attribute”, replace with BTNS OK
- Error values

Todo list 2/3

- rename: ipsec_policy -> protection token
 - contains information on e.g. IPsec algos
- add: identity token (itoken)
 - local or peer identities
- handle credentials
- constants should be strings in the API
 - channel bindings should be octet strings
- typedefs should be pointers
- common accessors for attributes

Todo list 3/3

- Define attributes for protection tokens
 - KE protocol: EAP, PKIX, BTNS, HIP
 - algo profile names, lifetimes
- Define attributes for identity tokens
 - ID type, name, certificates
- Channel bindings should be properties of identity tokens
- Conversion functions: human readable/loggable protection and identity tokens
- Editorial corrections