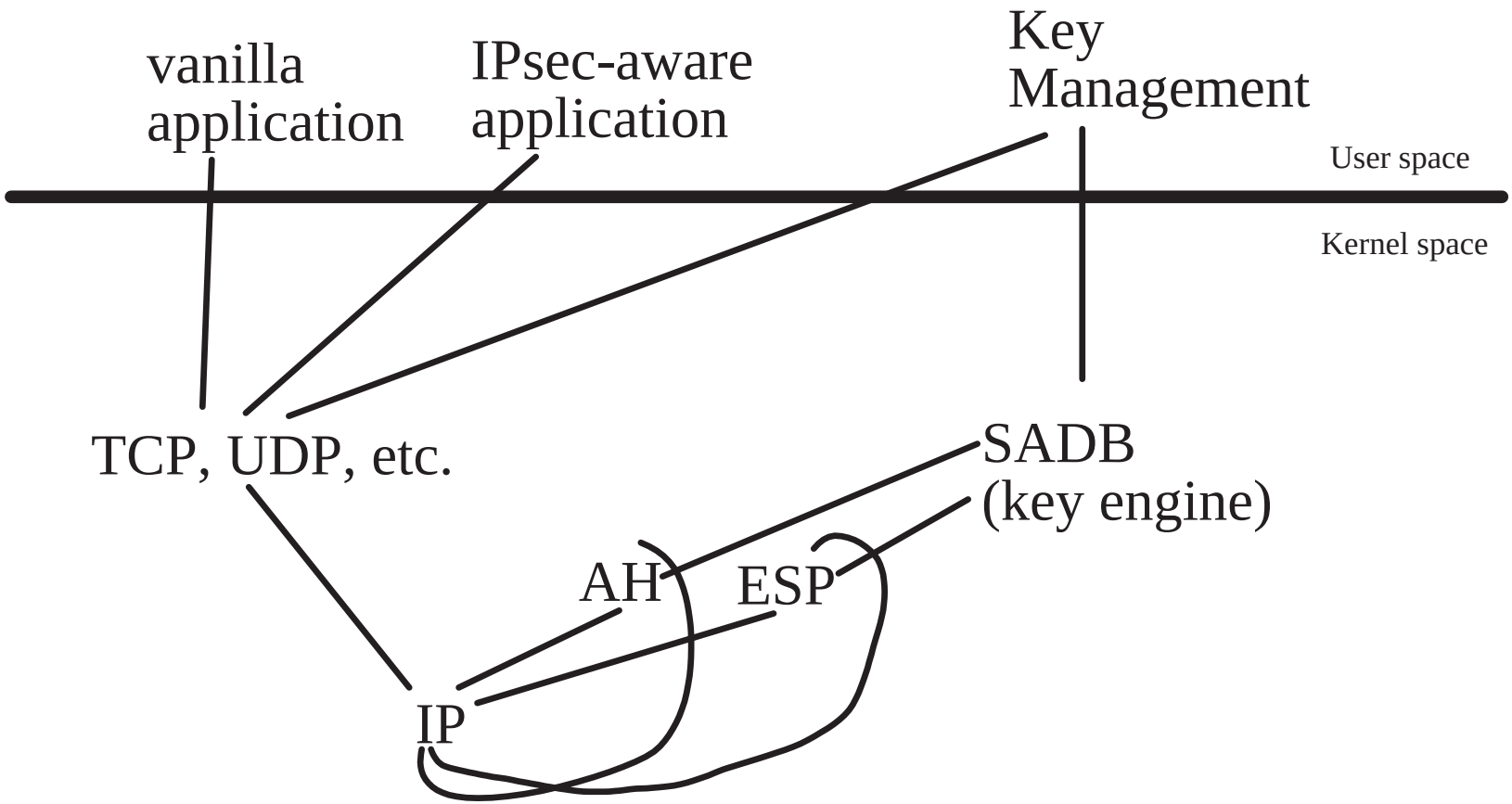
Building IP Security

Daniel L. McDonald - Solaris Internet Engineering

Introduction

- Drawing upon two platforms
  - 4.4 BSD (NRL IPv6/IPsec)
  - SunOS 5.x (IPv6/IPsec)
- General issues and considerations.
- Some nuts-and-bolts.

The Big Picture



Security Association Table (SADB)

- Is to IPsec what routing and ARP tables are to IP.
- PF_KEY provides interface to key management apps in user-space.
- `getassocby*()` calls provide primary SPI support.

PF_KEY Key Management API

- Analagous to 4.4 BSD’s routing socket.
- Keeps (out of band) key management in user-space.
- Messages originate both in kernel-space and in user-space.

PF_KEY Messages

- SADB_REGISTER
- SADB_ACQUIRE
- SADB_GETSPI
- SADB_UPDATE
- SADB_ADD
- SADB_DELETE
- SADB_EXPIRE
- SADB_GET
- SADB_FLUSH

Key Management

- Manual Keying
- Out-of-band key management
  - ISAKMP/Oakley
  - Photuris
  - Needham-Schroeder scheme
- In-band key management requires more kernel support.

`getassocby*()` calls.

- Used by IPsec to get security associations.
- There are other SADB kernel interfaces, but those are largely in support of PF_KEY, or other maintenance.

`getassocbyspi()`

- Arguments include SPI, IP source, IP destination, type.
- Used for inbound packets.
- Usually packet is dropped if this fails.

`getassocby`*`endpoint`*`()`

- *`endpoint`* is usually replaced with system term (`socket`, `pcb`).
- Arguments include endpoint ID, IP addresses, type, and others.
- Used for outbound packets.

- May cause SADB_ACQUIRE messages.
- What else may need to be passed here?
  - Certificate ID
  - Algorithm Preference
  - Keying Properties

Modifications to Existing Code

- Modifications include:
  - Datagram tagging
  - Policy checking per endpoint (and API to set it)
  - Global policy

Datagram Tagging

- Incoming data
  - AH, or ESP done?
  - What SAs were used?
- Outgoing data
  - Endpoint ID, and progress.

On Policy, and What Is Implemented

- Categories
  - AH
  - ESP Transport
  - ESP Network
- Levels per category
  - Default/None (and Bypass)
  - Use if available
  - Use
  - Require
  - Require Unique

Per-Endpoint Policy

- Each endpoint should use its own SAs.
- Categories are socket options, levels are values.
- Enforced by tagging outbound, and by checking tags inbound.

Global Policy

- A system may enforce global IPsec policy.
- Basically, more paranoid of global policy and endpoint policy wins.
- Finer granularity may be needed.
  - Per-route is one idea.

Policy Enforcement

- Inbound packets
  - While IPsec work is done, packet is marked.
  - When endpoint is determined, compare markings and endpoint’s expectations.

Policy Enforcement (cont.)

- Outbound Packets
  - Mark packet with endpoint's expectations.
  - Before fragmentation, perform necessary IPsec processing based on packet's marks.

Other IP Concerns

- ICMP message policy?
- Resource allocation. (This is true in every part, actually.)
- Slowing down the non-IPsec cases.

Common IPsec Inbound Processing

- Demux on next-header (protocol).
- Call `getassocbyspi()`.
  - If failed, drop packet and log.
- Perform AH/ESP specific tasks.
- Perform replay functions.
- Strip headers and continue.

AH-specific Inbound Processing

- Perform authentication calculation.
- Compare with data, if no match, drop and log.
- Mark packet as authentic.

Sun microsystems

ESP-specific Inbound Processing

- Decrypt packet. (Beware garbage.)
- If needed, authenticate data.
- If authentication fails, drop and log.
- If inner packet is IP, compare inner and outer headers. If not the same pretend packet is fresh off the wire.
- Mark packet appropriately.

Common IPsec Outbound Processing

- Must perform before fragmentation.
- Mark packet with IPsec it needs.
- Apply in order: ESP transport, AH, ESP network.
  - In each case, call `getassocby`*`endpoint`*`()`.

- If key mgmt. is invoked, queue up and wait for result. (Like ARP.)
- Otherwise proceed.

- Then fragment. (NOTE: TCP might want to know the impact of IPsec overhead.)

AH-specific Outbound Processing

- Compute authentication calculation.
- Bump replay counter (if used).
- Insert AH and update pre-AH header.
- Continue

ESP-specific Outbound Processing

- Create ESP appendage and replay counter (if using replay protection).
- Compute authentication (if needed).
- Append authentication result.
- Encrypt ESP portion of datagram.
- NOTE: If using ESP Network Mode end-to-end, prepend IP before start.

Socket Enhancements for IPsec

- Each category is a socket option.
  - `IPSEC_AUTH_LEVEL`
  - `IPSEC_ESP_TRANS_LEVEL`
  - `IPSEC_ESP_NETWORK_LEVEL`
- Each level is a value.
  - `IPSEC_LEVEL_BYPASS`
  - `IPSEC_LEVEL_{NONE,DEFAULT}`
  - `IPSEC_LEVEL_AVAIL`
  - `IPSEC_LEVEL_USE`
  - `IPSEC_LEVEL_REQUIRE`

- `IPSEC_LEVEL_UNIQUE`

- Will also eventually need other settings:
  - Algorithm preferences.
  - Certificate IDs.

Miscellaneous Issues

- Finer grained policy.
- Modifying applications to use IPsec
  - Inetd (inetd.conf settings)
  - Rcmds (with cert. IDs?)
- Tunnelling abstraction
  - Virtual interface?
  - Special "secure routes"?

Conclusion

- Many considerations when building IPsec.
- We need significant implementation experience like we've had for TCP.