The IPSec Architecture in OpenBSD

Niels Provos
CITI, University of Michigan
provos@citi.umich.edu

Overview

- Introduction
- History
- Kernel Details
- IKE
- Policy using Keynote
- Crypto Acceleration Framework
- Bridging with IPSec
- Conclusion

Introduction

- OpenBSD is a free UNIX-like operating system with a strong emphasis on security.
- We realized early that many security problems can be solved only by using cryptography.
- Cryptography is integrated seamlessly into the system and is an essential component:
  - IPSec, IPv6, key management daemons,
  - Cryptographic Acceleration Framework
  - Kerberos, AFS, OpenSSH, etc...

History

- July 1997: OpenBSD 2.1 with IPSec support
  - Based on BSD/OS code by John Ioannidis,
  - Ported to OpenBSD by Angelos Keromytis,
  - no automatic keying, only manual setup.
- December 1997: OpenBSD 2.2
  - Photuris key management daemon,
  - new ESP and AH support according to RFC 2406 + 2402.

History

- May 1998: OpenBSD 2.3
  - more encryption and authentication transforms, e.g. Blowfish,
  - `PF_ENCAP` notifications for key management daemon.
- May 1999: OpenBSD 2.5
  - `isakmpd` as IKE daemon,
  - `PFKEYv2` as generic key management API.

History

- December 1999: OpenBSD 2.6
  - support kernel acquire messages.
- June 2000: OpenBSD 2.7
  - support IPSec tunnels for bridging,
  - hardware acceleration for IPSec,
  - ACLs for ingress flows,
  - IPv6 IPSec support.

History

- December 2000: OpenBSD 2.8
  - AES (Rijndael) support,
  - Support for automatic IKE setup (no configuration).
- December 2001: OpenBSD 3.0
  - Path MTU discovery for IPSec,
  - IPCOMP (Payload compression) support.

Kernel Details

- Incoming IPSec packets are decapsulated and processed in ipsec_common_input(), and passed pack into the IP input queue.
- Outgoing packets are IPSec processed and encapsulated in ipsp_process_packet().
- SPD is stored in the BSD routing radix tree:
  - SA selection can be based on protocol and port numbers.
  - Supported attributes (ingress and egress):
    - acquire, require, dontacq, permit, and bypass.

Kernel Details

- Prevent source address spoofing by specifying permitted source addresses for specific SAs.
- Cryptographic processing of packets is done asynchronously, so that network stack processing does not block:
  - just by itself caused a performance improvement.
- Hard- and soft-expirations based on SA life time, processed bytes, etc...

Kernel Details

- Flows specify IPSec policy:
  - acquire/require flows send `PFKEY_ACQUIRE` messages to registered key management daemons,
  - permit/bypass either make use of existing IPSec SAs or bypass IPSec completely.
- Multiple key management daemons can run at the same time, e.g. photuris and isakmpd.

IKE

- isakmpd, new implementation started in 1998 since our IKE daemon had to be
  - scalable:
    - needs to support thousands of SAs.
  - flexible:
    - the standard was still evolving.
- Protocol complexity mandates modular design.
- isakmpd is an event driven UNIX daemon using `select(2)`

IKE

- Features
  - Main Mode, Aggressive Mode.
  - Diffie-Hellman groups 1 - 5, including elliptic curve cryptography.
  - Authentication via pre-shared pass phrases and RSA signatures with X509 certificates.
- Main abstraction is the exchange:
  - Provides the context for negotiations as script-driven state machine.
  - Enables high-level syntax checking of messages.

IKE

- Configuration:
  - Very flexible, allows for good interoperability, but too complex.
  - PFKEY extensions permit mode without configuration file:
    - Identities and policies are communicated via PFKEY.
- Portable with system dependency layer. Runs on *BSD and Linux.
- About 42 KLOC code size (12% platform dependent, 4% regression tests).

Policy using Keynote

- Uses the Keynote trust management system for policy decisions.
- Keynote determines who can negotiate what kind of SAs.
- Allows very fine grained access controls:

```
KeyNote-Version: 2
Authorizer: "POLICY"
Licensees: "passphrase:blubberdubber"
Conditions: app_domain == "IPsec policy" &&
            pfs == "yes" && esp_present == "yes" &&
            esp_enc_alg != "null" -> "true";
```

  - Requires the use of PFS and ESP, where the ESP algorithm is not null.

Crypto Acceleration Framework

- Cryptographic framework manages available hardware accelerators:
  - Asynchronous interface,
  - Callbacks on completion of cryptographic operation.
- Software cryptographic engine as fallback.
- Currently supports:
  - HiFn 7751 (e.g. Powercrypt),
  - Broadcom BCM5805 (ubsec).

Crypto Acceleration Framework

- Performance:

Engine	Transform	Spee
software (P3/550)	3DES	11 MBit/
software (P3/550)	Blowfish	34 MBit/
HiFn	3DES	70 MBit/
BCM5805	3DES	122 MBit/

  - Broadcom card has no problems dealing with 100 MBit ethernet.

Crypto Acceleration Framework

- `/dev/crypto` allows userland to access cryptographic framework:
  - Supports currently only symmetric ciphers and message authentication codes.
  - Acceleration for asymmetric operations planned:
    - Speedup for Diffie-Hellman and RSA.
- OpenSSL abstracts access to `/dev/crypto`:
  - Standard applications will automatically use hardware acceleration
    - OpenSSH, isakmpd, etc...

Bridging with IPSec

- In OpenBSD, a bridge device creates a logical link between a set of network interfaces.
- The bridge copies frames read from one interface to the other interfaces associated with the bridge device.
- Supports filtering on Layer-2 and Layer-3 -> transparent firewall.

Bridging with IPSec

- Supports extended LAN over the Internet.
  - Ease of administration, no routing required.
- Uses IPSec as encapsulation mechanism:
  - Set up a tunnel via generic tunnel interface `gif`:
    - IPv4 or IPv6.
  - Set up IPSec flows between tunnel endpoints.
  - `gif` interface may become a bridge member.
  - Ethernet in IP over IPSec.

Conclusion

- OpenBSD has state-of-the-art IPSec architecture:
  - IPSec is part of every OpenBSD installation,
  - Very fast with hardware acceleration,
  - Bridge and IPSec allow easy to administrate VPNs.
- OpenBSD code and information can be found at `http://www.openbsd.org/`.
- Further reading: `http://www.openbsd.org/crypto.html`.