Cryptography in OpenBSD: An overview

Theo de Raadt
Niklas Hallqvist
Artur Grabowski
Angelos D. Keromytis
Niels Provos

{deraadt,niklas,art,angelos,provos}@openbsd.org

Cryptography in Operating Systems: Today

Cryptography does not neccesarily provide security..

But without hard and reliable security mechanisms, crypto is often the only tool available in some fields, ie. network "security"

Crypto provides solutions for certain problems that cannot be solved in other ways.

Today, most systems ship with no cryptography

Some systems ship with very limited cryptography, ie. DES or MD5 for authentication of passwords

Why?

All major operating systems are "shipped" from USA ... except for 2 or so (OpenBSD, QNX; as far as I know)

Meanwhile non-USA Linux distributions wish to avoid greater incompatible with their USA counterparts

It's may be a world of USA software, but it is a WORLD market... so no crypto

Many groups are forced to come up with clever ways to by-pass the laws..

Those darn pesky laws...

ITAR prohibits export of cryptography from the USA, but permits export of cryptography to Canada

Canada permits export of cryptography (though a few cases require registration). However, "free" types are always permitted

France: restrictive until recently; new rules USA-like
Germany: making statements about greater freedom
UK, Russia, Sweden: waddling

We ship from Canada, and use only crypto from countries which satisfy our own rules of "free enough".

From where?

Australia
Sweden
Norway
Germany
Greece
Canada
Denmark
Finland
UK

And patents...

In particular, RSA and their nasty lawyers.

Require a licence for commercial use of the RSA algorithm within USA

We cannot use IDEA either. More nasty lawyers.

So... we use a trick to avoid the RSA licensing problem.

And the tie in to security

Our project place much emphasis on security

To improve security, we expend significant effort doing

- Efforts to repair problems as quickly as possible
- Design analysis
- Source code audits

It would be irresponsible if we did not investigate cryptographic solutions

So where does the crypto go

SSH
SSL
IPSEC
- **IKE, photuris**
- **DES, 3DES, Cast-128, Blowfish, Skipjack**
- **MD5, SHA1, RIPEMD160**

Kerberos IV
S/Key
strong and pseudo random numbers
Secure storage
bcrypt

In essence, little bits of it all over the place.

SSH

ssh v1 uses RSA code; besides... it is not really free code

ssh v2 can use other algorithms... even more restrictive licensing

No free high-quality versions available yet.

We WISH there was a good free version!

However, we include a package for each architecture on our FTP sites, since we cannot include it on the CD.

SSL (Secure Sockets Layer)

We use SSLEAY: a modern library supporting SSL2, SSL3, and TLS

OpenSSL... the same thing but maintained

Commonly used to implement the "https" service

But also can be used for other things; our IPSEC IKE daemon will link against it in the next release.

The library implements the unpatented and free DSA algorithm, but SSLEAY also does RSA -- which prevents us from entering the USA market...

So we use a shared library trick to avoid the RSA issue.

IPSEC

Network layer security mechanism which can be used in a variety of ways, ie. end-to-end, VPNs, etc.

Still being worked on at IETF: we are following all developments, and sometimes leading

We have: tunnel and transport mode, easy VPNs, photuris and isakmpd (IKE), all standard cryptographic algorithms, and more

Security policy mechanisms being worked on

IPSEC matches our security goals very closely, so our developers spend a lot of time on this area

Kerberos IV

Primarily uses DES

K4 instead of K5... ours is from Sweden, but K5 is only from USA

A K5 clone is under development in Sweden.. but not ready yet

Besides specific Kerberos tools, the following utilities use Kerberos

- login, xdm, su, rlogin[d], rsh[d], telnet[d], kx, cvs, sudo, xlock, ...

S/Key

Our S/Key has been improved to match the functionality of "opie"

Uses MD4, MD5, SHA1, or RIPEMD-160 hashes

RFC1938 compliant

S/Key is useful when other cryptographic mechanisms are not trusted or available (but... session snooping and hijacking are still threats)

Randomness

For proper operation, our system often needs random numbers of various characteristics and strengths

Kernel collects interrupt information and sustains an entropy pool, to provide data to

seed cryptographic functions,
provide numbers for use as transaction ids,
use for whatever purpose the kernel or userland may want.

A number of useful interfaces are described in the paper.

Non-Repeating Randomness

We needed to make DNS packet id's more random, due to a trivial spoofing attack (1, 2, 3 .. is bad)

16 bit space: 15 bits are non-repeating random, high bit toggles when 15 bits are exhausted and re-seeded

Makes DNS packets significantly harded to spoof

Same idea used to make IP ip_id stronger and harder to spoof

More random places to use randomness

Port numbers in the bind(2) system call

Process ID's

RPC and NFS RPC XID's

TCP ISS value

Inode generation numbers (stronger filehandles)

For stronger random names in mktemp(3)

And a whole lot more...

Secure Storage

We do not have a crypto filesystem.

CFS works, but that is not an ideal solution.

Developers are looking into it, but a quality encrypted filesystem is not as easy as simply encrypting the data (ie. issues regarding meta data, directories, fsck, etc)

Secure logging efforts also underway

bcrypt

The unix password system is hopelessly antiquated.

This problem was attacked by Niels Provos, who will describe his work in the next talk.

Conclusions

If you expect to see a particular piece of cryptographic software in an operating system, look here first.

Our cryptography efforts stress integration, not add-ons.

If you are a non-USA cryptographer who believes in integrated crypto, we want to hear from you.