Security features
in the OpenBSD operating system

Matthieu Herrb & other OpenBSD developers

LAAS
CNRS

INSTITUT
CARNOT
LAAS CNRS

Université
de Toulouse

Min2rien 10 novembre 2015

Agenda

1 Introduction

2 Random numbers

3 Increasing resilience

4 Network level protection

5 LibreSSL

6 Conclusion

Agenda

1 Introduction

2 Random numbers

3 Increasing resilience

4 Network level protection

5 LibreSSL

6 Conclusion

About myself

- PhD in Robotics, Toulouse University, 1991
- Research Engineer at CNRS/LAAS
- System software
  - system administration and security officer for LAAS
  - robots systems software integration
- OpenBSD and X.Org contributor
- Member of the tetaneutral.net associcative local ISP

OpenBSD...

- Unix-like, multi-platform operating system
- Derived from BSD 4.4
- Kernel + userland + documentation maintained together
- 3rd party applications available via the ports system
- One release every 6 months
- Hardware architectures: i386, amd64, alpha, arm, macppc, sparc, sparc64, sgi, vax...

`http://www.openbsd.org/`

“Secure by default”

- Leitmotiv since 1995
- Adopted by most other systems
- Non required services are not activated in a default installation.
- The default configuration of services is secure
- Activating services requires a manual action of the administrator
- Keep a working (ie. functional, useful) system

→ only a few remote vulnerabilities in 20 years !

Objectives

- Provide free code (BSD license...)
- Quality
- Correctness
- Adhering to standards (POSIX, ANSI)
- Providing good crypto tools (SSH, SSL, IPSEC,...)

→ better security.

Current version

OpenBSD 5.8 released Oct. 19, 2015.

Recent changes :

- doas(1) replacement for sudo(1)
- LibreSSL, OpenSSL fork
- PIE by default on more architectures
- W^X in the kernel on some architectures
- OpenSMTPd, a privilege separated SMTP daemon is now the default
- removed unsafe algorithms from OpenSSH protocol negociation
- lots of unsafe code removal (Kerberos, sendmail,...)
- signify tool to sign releases and ports

Increasing resilience to attacks

- Provide an unpredictable resource base with minimum permissions
  - Random stack gap
  - Program segments mappings randomization
    - shared libraries ASLR, random ordering
    - PIE
    - mmap ASLR
  - increased use of the ELF `.rodata` section
  - malloc randomizations
- Where it is possible to spot damage, fail hard
  - stack protector
  - stack ghost
  - atexit/ctor protection

Agenda

1 Introduction

2 Random numbers

3 Increasing resilience

4 Network level protection

5 LibreSSL

6 Conclusion

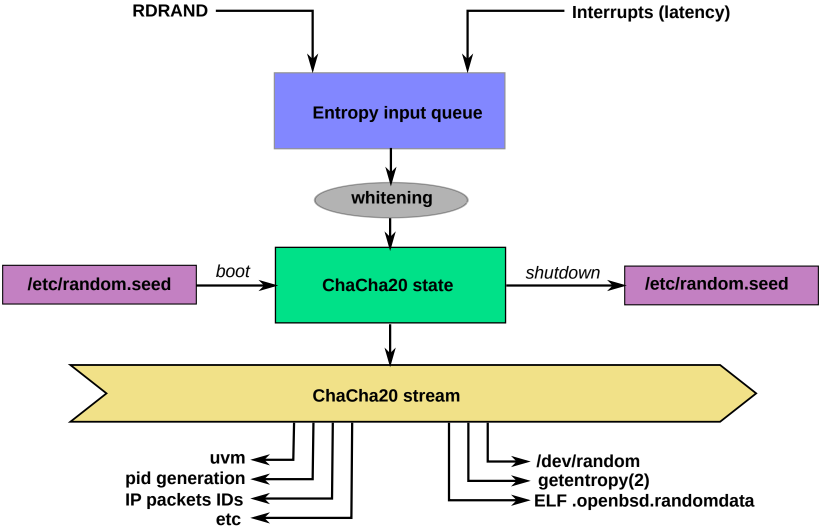
Random numbers in OpenBSD

“libc needs high quality random numbers
available under any circumstances” – Theo de Raadt

- in the kernel
- in threads
- in chroot environments
- in ENFILE/EMFILE situations
- in performance critical code

Most important characteristic : **Ease of use**

Random numbers in OpenBSD: kernel



Use of random numbers in the kernel

- random PIDs
- VM mappings (including userland malloc/free requests)
- network packets creation (sequence numbers)
- pf NAT and other operations
- port allocation
- scheduler decisions
- userland arc4random() reseeding via getentropy(2)

Slicing the random stream between many users:
→ resistance to backtracking and prediction.

Random numbers in userland

Per-process stream, with re-seeding:

- too much volume of data has moved
- too much time elapsed
- when a fork() is detected

Slicing between several users occurs too :

- malloc(3)
- DNS
- ld.so
- crypto

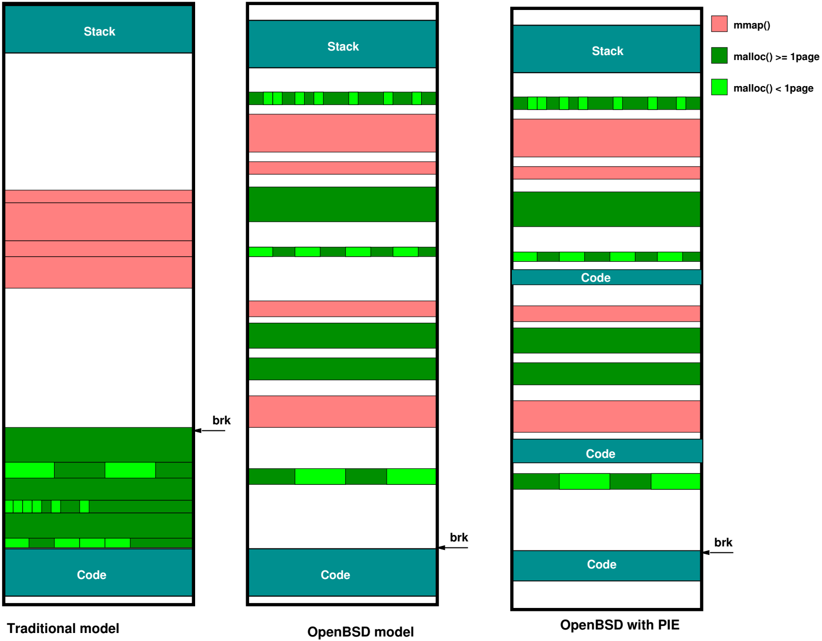
More than 1000 call points in the libraries and system utilities.

Agenda

1. Introduction
2. Random numbers
3. Increasing resilience
4. Network level protection
5. LibreSSL
6. Conclusion

ASLR

- stackgap: random offset in stack placement
- mmap()
- shared libraries
- PIE executables by default, including static binaries on most architectures


Stack
brk
Code
Traditional model
Stack
brk
Code
OpenBSD model
Stack
Code
Code
brk
Code
OpenBSD with PIE
🟥 mmap()
🟩 malloc() >= 1page
🟩 malloc() < 1page

Randomness in mmap()

Address returned by `mmap()`:

If `MAP_FIXED` is not specified: returns a random address.

(traditional behaviour: 1st free page after a base starting address)

Randomness in malloc()

- $\geqslant 1$ page allocations: mmap() $\rightarrow$ random addresses.
- < 1 page allocations: classical fixed block allocator, but random selection of the block in the free list.

$\Rightarrow$ heap attacks more difficult.

Protecting dynamically allocated memory

[Moerbeek 2009]

- Activated by `/etc/malloc.conf` → `G`
- Each bigger than one page allocation is followed by a guard page ⇒ segmentation fault if overflow.
- Smaller allocations are randomly placed inside one page.

Propolice / SSP

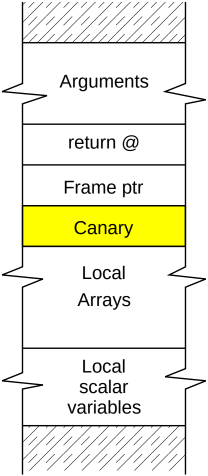
gcc patches initially developped by IBM Tokyo Research Labs (2002).

Principle : put a “canary” on the stack, in front of local variables

- check it before return.
- if still alive: no overflow
- if dead (overwritten): overflow → `abort()`

Only when there are arrays in local variables

Adopted by gcc since version 4.1.
Enabled by default in OpenBSD..



W^X

Principle of least privilege.

Write **exclusive or** execution right granted on a page..

- easy on some architectures (x86_64, sparc, alpha): per page 'X' bit
- harder or others (x86, powerpc): per memory segment 'X' bit
- impossible in some cases (vax, m68k, mips)

In OpenBSD 5.7: WˆX inside the kernel for x86_64

(PaX on Linux...)

Privileges reduction

- Completely revoke privileges from privileged (setuid) commands, or commands launched with privileges, once every operation requiring a privilege are done.
- Group those operations as early as possible after start-up. Examples:
  - ping
  - named

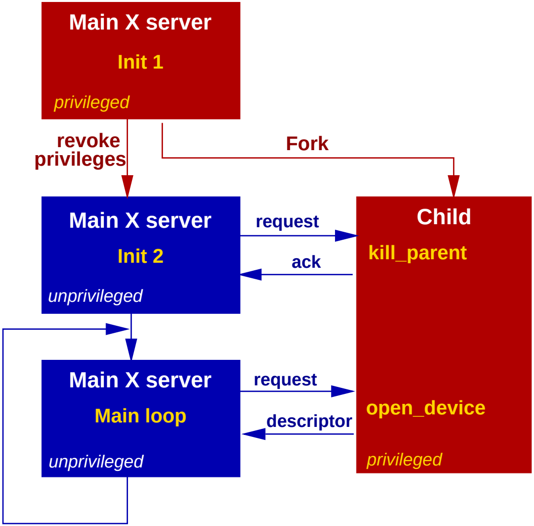
Privileges separation

[Provos 2003]

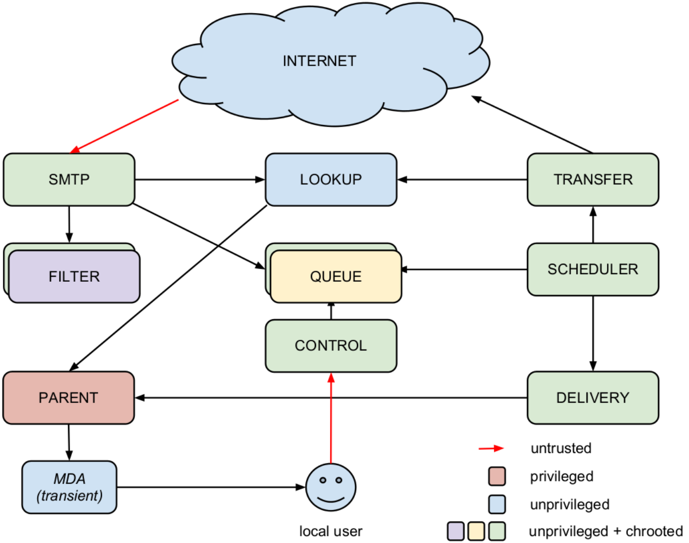
- Run system daemons:
  - with an uid $\neq 0$
  - in a chroot(2) jail
- additional helper process keeps the privileges but do paranoid checks on all his actions.

A dozen of daemons are protected this way.

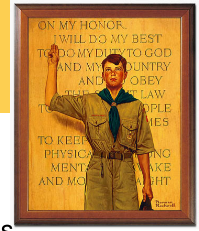
Example: X server



Example: OpenSMTPd



New : pledge(2)



Selectively limits system calls that can be used by a process.

```
int
pledge(const char *request, const char *paths[]);
```

request is a list of keywords representing the behaviour of the process and the authorized system calls.

Some system calls, when allowed, have restrictions applied to them. (for ex. `ioctl()`)

All other system calls are forbidden.

pledge(2)

A few requests:

- `""` only `_exit` is permitted
- `stdio` system calls related to basic libc functions, including memory allocation, but excluding opening new file descriptors.
- `rpath` read-only access to the filesystem
- `wpath` write access to the filesystem
- `tmppath` read/write/create files in `/tmp`
- `inet` IPv4 & IPv6 socket access
- `dns` subset of `inet` for DNS transactions
- `tty` terminal manipulation (termios)
- `proc exec` fork / exec

pledge : example wc.c

```
int
main(int argc, char *argv[])
{
        int ch;

        setlocale(LC_ALL, "");

        if (pledge("stdio rpath", NULL) == -1)
                err(1, "pledge");

        while ((ch = getopt(argc, argv, "lwchm")) != -1)
        ...
```

Agenda

1 Introduction

2 Random numbers

3 Increasing resilience

4 **Network level protection**

5 LibreSSL

6 Conclusion

Threats on protocols

Internet: favours working stuff over security.

- easy to guess values
- forged packets accepted as valid
- information leaks
- use of time as a secret ??

Protection Principle

Use data that are impossible (hard) to guess wherever arbitrary data are allowed, even if no known attack exists.

- counters
- timestamps
- packet, session, host... identifiers

But respect constraints and avoid breaking protocols:

- non repetition
- minimal interval between 2 values
- avoid magic numbers

Randomness in the network stack

Use:

- IPID (16 bits, no repetition)
- DNS Queries (16 bits, no repetition)
- TCP ISN (32 bits, no repetition, steps of 2^{15} between 2 values)
- Source ports (don't re-use a still active port)
- TCP timestamps (random initial value, then increasing at constant rate)
- Id NTPd (64 bits, random) instead of current time
- RIPd MD5 auth...

PF: more than one trick in its bag

Packet Filter

- Stateful filtering and rewriting (NAT) engine
- **Scrub** to add randomness to packets:
  - TCP ISN
  - IP ID
  - TCP timestamp
  - NAT : rewriting of source ports (and possibly addresses)

Also protects non-OpenBSD machines behind a pf firewall.

Agenda

1 Introduction

2 Random numbers

3 Increasing resilience

4 Network level protection

5 LibreSSL

6 Conclusion

OpenSSL & Heartbleed

- for years no one really looked at the OpenSSL code
- those who had a glance ran away (too horrible)
- so everyone blindly trusted the OpenSSL project
- then came Heartbleed, made people look again
- OpenBSD decided that the only way out was to fork

LibreSSL - goals

- Keep the OpenSSL API
- Important : remove layers of wrappers around system primitives
- malloc wrappers were hiding bugs from valgrind/OpenBSD's malloc
- Printf-like wrappers may have hidden format string bugs
- Review the public OpenSSL bug database : dozen of valid bug reports sitting for years
- Fix random number generator → `getentropy()`
- Fix many (potential) interger overflows → `reallocarray()`
- Portable version for Linux, MacOSX, Windows,...

http://www.libressl.org/

libTLS

- new API
- hides implementation details (no ASN.1, x509,... structures)
- safe default behaviour (hostnames/certificates verification,...)
- privilege separation friendly
- example use in OpenSMTPd, relayd, httpd...
- still under active development

Agenda

1 Introduction

2 Random numbers

3 Increasing resilience

4 Network level protection

5 LibreSSL

6 Conclusion

Conclusion

- Lots of progress since the beginning.
- Contributed to fix bugs in many 3rd party applications.
- Often Copied (good).
- Still lots of issues to address...

Bibliography

http://www.openbsd.org/papers/index.html

- pledge() a new mitigation mechanism, Theo Deraadt, Hackfest 2015, Quebec City.
- arc4random - randomization for all occasions, Theo de Raadt, Hackfest 2014, Quebec City.
- LibreSSL - An OpenSSL replacement, the first 30 days and where we go from here, Bob Beck, BSDCan 2014.
- Exploit mitigation techniques - An update after 10 years, Theo de Raadt, ruBSD 2013, Moscow.
- OpenSMTPD: We deliver!, Eric Faurot, AsiaBSDCon 2013.
- Time is not a secret: Network Randomness in OpenBSD, Ryan McBride Asia BSD Conference 2007
- Security issues related to Pentium System Management Mode, Loïc Duflot, CansecWest 2006.
- Preventing Privilege Escalation, Niels Provos, Markus Friedl and Peter Honeyman, 12th USENIX Security Symposium, Washington, DC, August 2003.
- Enhancing XFree86 security, Matthieu Herrb LSM, Metz 2003.

Questions ?