Hardening pkgsrc

Securing packages, 17.000 at a time

BSDCan 2017

June 9-10 2017, Ottawa, Canada

Pierre Pronchery
<khorben@NetBSD.org>

About myself

- Pierre Pronchery, planet Earth
- DeforaOS Project since 2004
- IT-Security consultant since 2006
- NetBSD developer since May 2012
- Working on NetBSD with Git through the EdgeBSD community since August 2013
- Co-founder of Defora Networks since July 2016: https://www.defora.net/

Introduction

- pkgsrc is a multi-platform:
  - Software distribution
  - Build framework
  - Package manager
- Default source for packaged software on NetBSD, SmartOS, Minix...
- Supports many more!
  - Over 17.000 packages on 17+ platforms

NetBSD®

Motivation

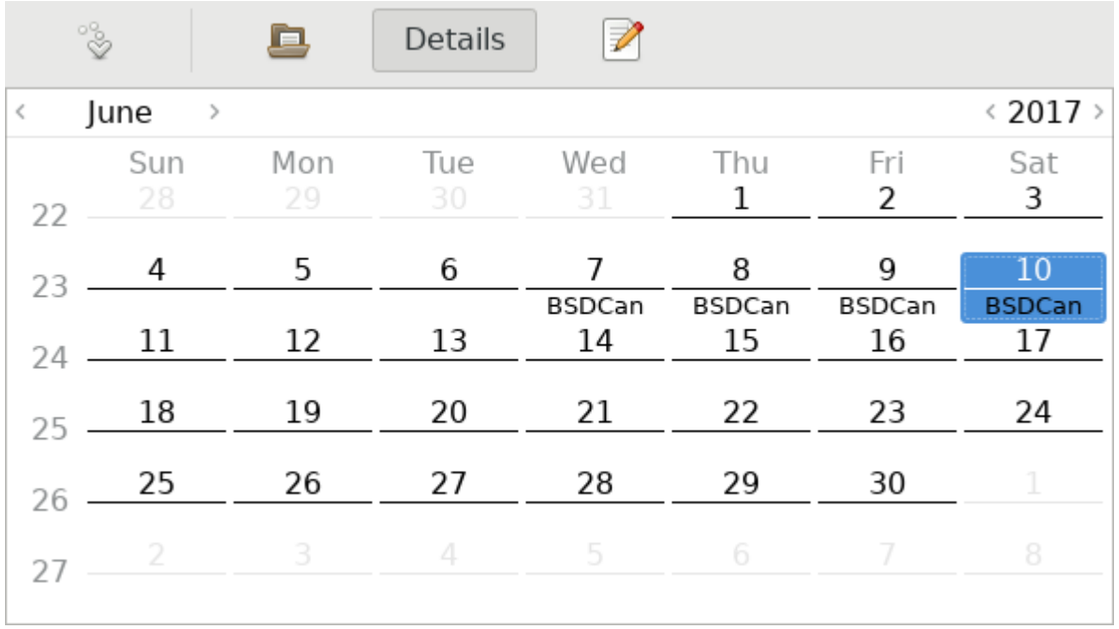
- As illustrated again in the news this week, a “**cyber-war**” is raging *right now*
- We have a responsibility towards our users
- pkgsrc offers a great opportunity for hardening a complete software setup

NetBSD®

Agenda

1. Security management
   Processes in place
2. Hardening features
   Technical measures
3. Future work
   Perspectives for improvement

   Questions & Answers



NetBSD®

1. Security management

1. Teams in charge
   - Security Team
   - Release Engineering Group
2. Vulnerability assessment database
   - Usage from source
   - Auditing binary packages
3. Maintenance of the stable release
   - Security patches
   - Long-Term Support (LTS)

NetBSD®

pkgsrc Security Team

- List of duties:
  - Handles security issues relevant to pkgsrc:
    pkgsrc-security@NetBSD.org
    http://pkgsrc.org/pkgsrc-security_pgp_key.asc
  - Maintains the vulnerability database:
    http://cdn.netbsd.org/pub/NetBSD/packages/vulns/pkg-vulnerabilities.bz2

Vulnerability database

- Assembled from:
  - Release notes from upstream packages
  - Security Advisories from vendors (Secunia...)
  - Announcements on public mailing-lists (OSS-Security...)
  - Erratas or advisories from other distributions, governmental or technical organisations (MITRE, CERT...)
- Cryptographically signed (PGP)

Vulnerability assessment

- Configure updates in `/etc/daily.conf`:

  ```
  fetch_pkg_vulnerabilities=YES
  ```

- To fetch manually:

  ```
  # pkg_admin fetch-pkg-vulnerabilities -s
  ```

- To audit the packages installed:

  ```
  # pkg_admin audit
  ```

Vulnerability assessment (from sources)

```
sysutils/xenkernel45$ make install
=> Bootstrap dependency digest>=20010302:
found digest-20160304
===> Checking for vulnerabilities in
xenkernel45-4.5.5nb1
Package xenkernel45-4.5.5nb1 has a information-leak
vulnerability, see
http://xenbits.xen.org/xsa/advisory-200.html
[…]
ERROR: Define ALLOW_VULNERABLE_PACKAGES in
/etc/mk.conf or IGNORE_URL in pkg_install.conf(5) if
this package is absolutely essential.
*** Error code 1
```

Vulnerability assessment (binary packages)

```
# pkg_add wireshark-2.2.1.tgz
Package wireshark-2.2.1 has a denial-
of-service vulnerability, see
https://www.wireshark.org/security/wn
pa-sec-2016-58.html
[…]
pkg_add: 1 package addition failed
```

Vulnerability assessment (binary packages)

- In `/etc/pkg_install.conf`:

  ```
  CHECK_VULNERABILITIES=always
  ```

- Alternatively, set to `interactive` to be prompted:

  ```
  […]
  Do you want to proceed with the
  installation of wireshark-2.2.1 [y/n]?
  n
  Cancelling installation
  pkg_add: 1 package addition failed
  ```

Security Team members

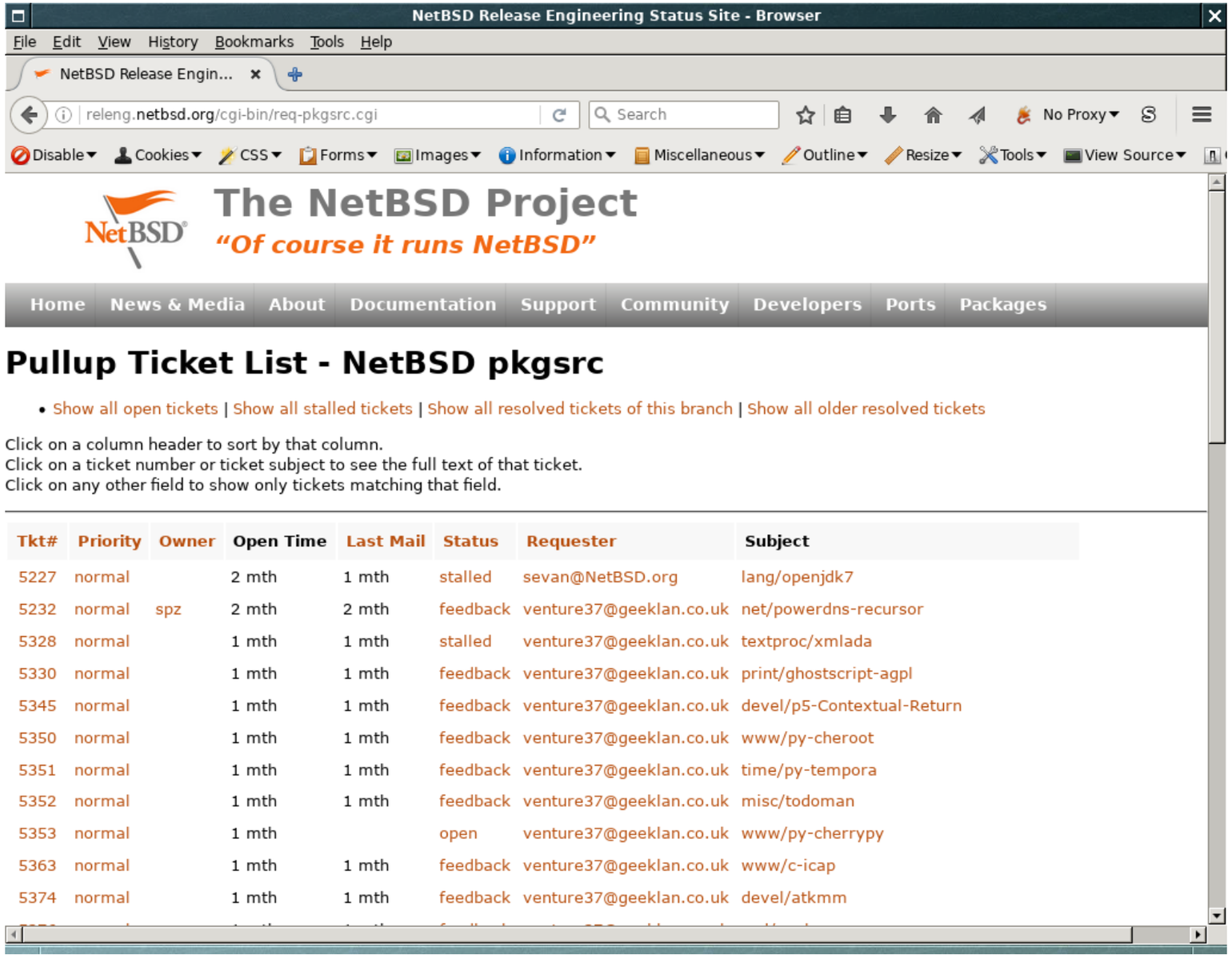
- Alistair G. Crooks <agc@>
- Daniel Horecki <morr@>
- Sevan Janiyan <sevan@>
- Thomas Klausner <wiz@>
- Tobias Nygren <tnn@>
- Ryo Onodera <ryoon@>
- Fredrik Pettai <pettai@>
- Jörg Sonnenberger <joerg@>
- Tim Zingelman <tez@>

NetBSD®

Release Engineering Group

- List of duties:
  - Manage stable branches
    https://releng.netbsd.org/cgi-bin/req-pkgsrc.cgi
  - Process pullup requests
    Including security issues
    https://www.netbsd.org/developers/releng/pullups.html#pkgsrc-releng
  - Schedule freeze periods
    https://www.pkgsrc.org/is-a-freeze-on/

Release Engineering Group



Tkt#	Priority	Owner	Open Time	Last Mail	Status	Requester	Subject
5227	normal		2 mth	1 mth	stalled	sevan@NetBSD.org	lang/openjdk7
5232	normal	spz	2 mth	2 mth	feedback	venture37@geeklan.co.uk	net/powerdns-recursor
5328	normal		1 mth	1 mth	stalled	venture37@geeklan.co.uk	textproc/xmlada
5330	normal		1 mth	1 mth	feedback	venture37@geeklan.co.uk	print/ghostscript-agpl
5345	normal		1 mth	1 mth	feedback	venture37@geeklan.co.uk	devel/p5-Contextual-Return
5350	normal		1 mth	1 mth	feedback	venture37@geeklan.co.uk	www/py-cheroot
5351	normal		1 mth	1 mth	feedback	venture37@geeklan.co.uk	time/py-tempora
5352	normal		1 mth	1 mth	feedback	venture37@geeklan.co.uk	misc/todoman
5353	normal		1 mth		open	venture37@geeklan.co.uk	www/py-cherrypy
5363	normal		1 mth	1 mth	feedback	venture37@geeklan.co.uk	www/c-icap
5374	normal		1 mth	1 mth	feedback	venture37@geeklan.co.uk	devel/atkmm

Stable releases

- Stable releases happening every quarter:
  - 2016Q4 no longer maintained
  - 2017Q1 latest stable
  - 2017Q2 in progress (HEAD)
- Joyent provides Long-Term Support (LTS)
  - joyent/feature/backports/20XXQ4
    https://github.com/joyent/pkgsrc
  - Focus on SmartOS

Release Engineering Group members

- Ryo Onodera <ryoon@>
- Fredrik Pettai <pettai@>
- Eric Schnoebelen <schnoebe@>
- Benny Siegert <bsiegert@>
- S.P. Zeidler <spz@>

2. Hardening features

1. Package signatures
2. Stack Smashing Protection (SSP)
3. Fortify
4. PIE (for ASLR)
5. RELRO and BIND_NOW

NetBSD®

Package signatures

- Support introduced initially in 2001:
  - Based on X.509 certificates or GnuPG
- Ensures authenticity and integrity:
  - Critical when installing binaries over HTTP or FTP
- Used by Joyent on SmartOS since 2014Q4:
  - Patch to use libnetpgpverify instead of GnuPG
- Still using GnuPG to generate packages

Package signatures

- Chicken and egg problem with GnuPG:
  - Not available in base
  - Needs to be installed as a package to verify itself
- Adding support for netpgp instead:
  - Available in NetBSD’s base system
  - Command line wrapper available (`gpg2netpgp`)
  - Still requires some patches (work in progress)
  - Security issue remaining with detached signatures

Package signatures (creation)

- Generate a key for the user building packages:
  `$ gpg --gen-key`
- In `/etc/mk.conf`:
  `SIGN_PACKAGES=gpg`
- Optionally, in `/etc/pkg_install.conf`:
  ```
  GPG=/usr/pkg/bin/gpg
  #GPG=/usr/local/bin/gpg2netpgp
  GPG_SIGN_AS=DEADBEEF
  ```
- Then use pkgsrc from source normally

Package signatures (installation)

- Import the key for the user installing packages:
  `# gpg --import`
- In `/etc/pkg_install.conf`:
  `VERIFIED_INSTALLATION=always`
- Then use pkgsrc normally:

```
# pkg_add socat
gpg: Signature made Thu Nov  3 14:44:06 2016 CET
using RSA key ID CC245448
gpg: Good signature from "EdgeBSD test packages
(khorben) <root@edgebsd.org>"
Primary key fingerprint: 968C 30DE B3C9 C147 203A
2E6E 5FFC 2014 CC24 5448
```

Stack Smashing Protection (SSP)

- Mitigation: reduce the impact and exploitation of Buffer Overflow vulnerabilities
- Different memory layout (stack variables)
- Addition of a « canary » value
  - Marker to detect memory corruption
  - Slight performance penalty
  - Controlled crashes instead of Code Execution

Stack Smashing Protection (SSP)

- Supported in pkgsrc for Linux (x86), FreeBSD (x86), and NetBSD
- Enabled in `/etc/mk.conf`:
  `PKGSRC_USE_SSP=yes`
- Sets a compilation flag, in the case of GCC and clang:
  `-fstack-protector`
  (protects only some functions)
- **Requires the package to support `CFLAGS`**
  Some packages still do not ☹

Stack Smashing Protection (challenges)

- Only protects C/C++ programs and interpreters
  - JIT compilation is not protected
- Supporting more flags:
  `-fstack-protector-all`
  (protects every function, now supported)
  `-fstack-protector-strong`
  (balanced, requires patch from Google)
- Add support for more compilers and platforms

Stack Smashing Protection (validation)

- To confirm a binary was successfully compiled with SSP:

```
$ nm hello
[...]
         U __stack_chk_fail
00600f00 B __stack_chk_guard
```

This is specific to GCC on NetBSD

- Enabled by default in OpenBSD (2003), Fedora and Ubuntu Linux (2006), DragonFlyBSD (2013)

Fortify

- Automatically adds boundary checks: `sprintf()`, `strncat()`, `memmove()`...
- Completely mitigates some Buffer Overflows
- Involves support from the libc (system headers)
  - Negligible performance impact
  - Controlled crashes instead of memory corruption

Fortify

- Supported in pkgsrc for Linux and NetBSD (GCC)
- Enabled in `/etc/mk.conf`:
  `PKGSRC_USE_FORTIFY=yes`
- Sets a pre-processing flag, in the case of GCC:
  `-D_FORTIFY_SOURCE=2`
- **Requires the package to support `CFLAGS`**
  Just like SSP ☹

Fortify (challenges)

- Only protects C/C++ programs and interpreters
  - Again JIT compilation is not protected
  - Requires an optimization level of 1 or more (e.g. `-O2`)
- Supporting more levels now possible in pkgsrc:
  `-D_FORTIFY_SOURCE=1`
  (protects fewer cases)
  `-D_FORTIFY_SOURCE=2`
  (some conforming programs might fail)
- Add support for more compilers and platforms

Fortify (validation)

- To confirm a binary was successfully compiled with Fortify:

```
$ nm hello
[…]
                 U __sprintf_chk
```

This is specific to GCC on NetBSD

- Enabled by default in Ubuntu Linux and Android

Position-Independent Executables (PIE)

- Necessary companion to PaX ASLR (Address Space Layout Randomization)
- PaX ASLR enabled by default in NetBSD 8 (incoming!)
- Allow compiled binaries to be re-positioned dynamically in memory
- Makes exploitation more difficult (requires a memory leak including pointer values)
- Involves compilation **and linking** phases

Position-Independent Executables

- Supported in pkgsrc for NetBSD and GCC
- Enabled in `/etc/mk.conf`:
  `PKGSRC_MKPIE=yes`
- Sets a compilation flag, in the case of GCC:
  `-fPIC`
- Requires the package to support both `CFLAGS` **and `LDFLAGS` as well (with a caveat)**
  Even stricter than SSP and Fortify ☹

Position-Independent Executables (challenges)

- The compilation flag should really be `-fPIE` for executables
- The linking phase must be completed with `-pie` but **only for executables** so **not directly through `LDFLAGS`**
- Currently implemented in the GCC wrapper
- **Not supported in cwrappers** yet (patch in review)

Position-Independent Executables (advantages)

- Packages linked but not compiled correctly will **fail to build**
- Great way to know which packages do not implement flags as they should
- Program crashes usually reveal silent bugs
- Can be combined with `paxctl` otherwise:
  `NOT_PAX_ASLR_SAFE`
  `NOT_PAX_MPROTECT_SAFE`
  (see `mk/pax.mk`)

Position-Independent Executables (validation)

- To confirm an executable binary is a PIE:

```
$ file hello-pie
ELF 64-bit LSB shared object, x86-64,
version 1 (SYSV), dynamically linked (uses
shared libs), for NetBSD 7.0, not stripped

$ file hello-nopie
ELF 64-bit LSB executable, x86-64, version
1 (SYSV), dynamically linked (uses shared
libs), for NetBSD 7.0, not stripped
```

RELRO and BIND_NOW

- RELRO protects ELF executable programs from tampering at run-time
- Makes exploitation harder by reducing the attack surface through relocations
- Benefits from immediate binding with BIND_NOW
- Performance penalty when starting big programs
- Involves the **linking** phase

RELRO and BIND_NOW

- Supported in pkgsrc for Linux and NetBSD (GCC)
- Enabled in `/etc/mk.conf`:
  `PKGSRC_USE_RELRO=yes`
- Sets two linking flags, in the case of GCC:
  `-Wl,-z,relro -Wl,-z,now`
- Requires the package to support `LDFLAGS`

RELRO and BIND_NOW (challenges)

- More granularity is now supported:
  - Full, or
  - Partial (without BIND_NOW)
- Some packages break at run-time with full RELRO (e.g. Xorg)
- Could be adapted to more platforms
- Same issue as before with support from packages ☹

RELRO and BIND_NOW (validation)

- To confirm a binary was built with RELRO and BIND_NOW:

```
$ objdump -x hello
[…]
Program Header: […]
    RELRO off    0x00000d68
          vaddr  0x00600d68
          paddr  0x00600d68 align 2**0
          filesz 0x00000298
          memsz  0x00000298 flags r--
[…]
Dynamic Section: […]
  BIND_NOW                 0x00000000
```

edgebsd/hardening

- Package meant to test a local pkgsrc setup:
  https://git.edgebsd.org/gitweb/?p=edgebsd.git;a=tree;f=hardening

```
$ hardening
[!] Hi! I am a library.
[!] Let's see if I am strong enough...
[+] built with -fPIC
[!] Bye! I am not a library anymore.
[!] Hi! I am an executable.
[+] built with -fPIC, good enough for full ASLR
[+] built with _FORTIFY_SOURCE 2, all good
[+] mmap() failed W|X, good
[-] mmap() gave two identical addresses :(
```

Demo

- Let us pray the demo gods?
- This presentation is the demo
- Userland with every feature mentioned so far (except Modular Xorg with partial RELRO)
- All the way to LibreOffice 5.3.0.3

3. Future work

- Reproducible Builds
- Code Flow Integrity (CFI)
- SafeStack
- Address Sanitizer

NetBSD®

Reproducible Builds

> *« Reproducible builds are a set of software development practices that create a verifiable path from human readable source code to the binary code used by computers. »*

- More at https://reproducible-builds.org/

Reproducible Builds

1. Deterministic build system:
   - Always the same result from a given source (including the current date and time, ordering of output...)
2. Pre-defined (or recorded) build environment:
   - Specific file format for build definitions
3. Let users reproduce and verify the original build

Reproducible Builds

- Already implemented in FreeBSD's ports:
  - Initial patch takes the timestamp from `distinfo`
  - Specific patches needed as well (Perl...)
- Can affect many aspects of the build process:
  - Build environment: setting `$SOURCE_DATE_EPOCH`
  - Some flags relevant for GCC:
    - `gcc -Wp,-iremap,…`
    - `gcc -fdebug-prefix-map=…`

Code Flow Integrity (CFI)

- Prevents exploits from redirecting the execution flow of programs
- Controlled crashes instead of undefined behaviour
- Again, pkgsrc should be a great test-bed for this feature

Code Flow Integrity (Clang)

- Implementation available in Clang:
  http://clang.llvm.org/docs/ControlFlowIntegrity.html
- Requires the following in `CFLAGS`:
  `-flto -fsanitize=cfi`
  (individual schemes can be selected)
  and possibly `-fvisibility=hidden`
- Additional debugging information can be obtained
- Suitable for release builds:
  - Negligible performance impact

SafeStack (Clang)

- « An instrumentation pass that protects programs against attacks based on stack buffer overflows, without introducing any measurable performance overhead. It works by separating the program stack into two distinct regions: the safe stack and the unsafe stack. The safe stack stores return addresses, register spills, and local variables that are always accessed in a safe way, while the unsafe stack stores everything else. This separation ensures that buffer overflows on the unsafe stack cannot be used to overwrite anything on the safe stack. »
  https://clang.llvm.org/docs/SafeStack.html
- Involves `CFLAGS`:
  `-fsanitize=safe-stack`

Address Sanitizer (GCC)

- A memory error detector from GCC: https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html
- Instruments memory access instructions
- Detects out-of-bounds and use-after-free bugs
- Involves `CFLAGS`:
  `-fsanitize=address`
  (more schemes are supported)

Closing words

- pkgsrc is a great project for testing security features
- Some possibilities can already be enabled *could some of them be turned on by default?*
- A lot more can still be done!

NetBSD®

Thank you!

- BSDCan 2017: http://www.bsdcan.org/2017/
- pkgsrc: https://pkgsrc.org/
  - The pkgsrc Security Team & the Release Engineering Group
- Joyent: https://pkgsrc.joyent.com/
  - Jonathan Perkin <jperkin@>
- Devio.us, EdgeBSD, HardenedBSD, OpenBSD...
- Contact me at khorben@NetBSD.org
- Time for questions?

NetBSD®