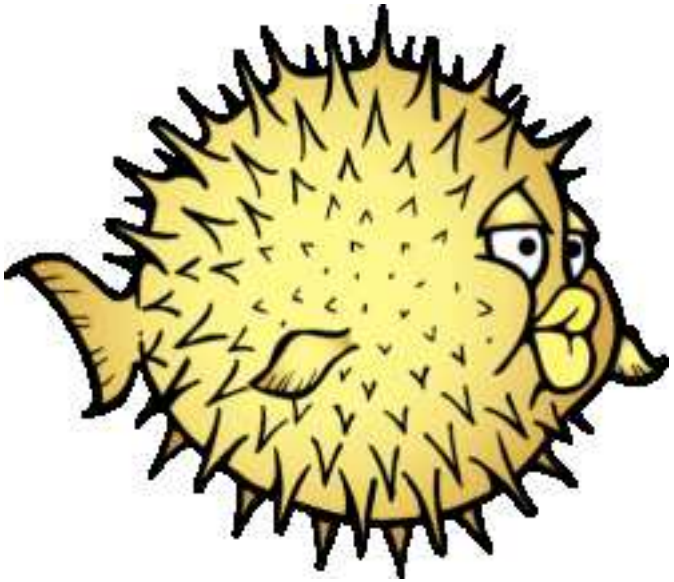
Converting OpenBSD to PIE



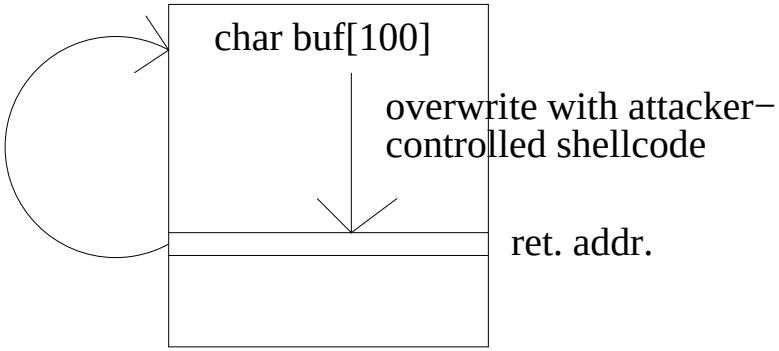
Pascal Stumpf <pascal@openbsd.org>

AsiaBSDCon 2015, March 15, Tokyo

Introduction: Return-oriented Programming

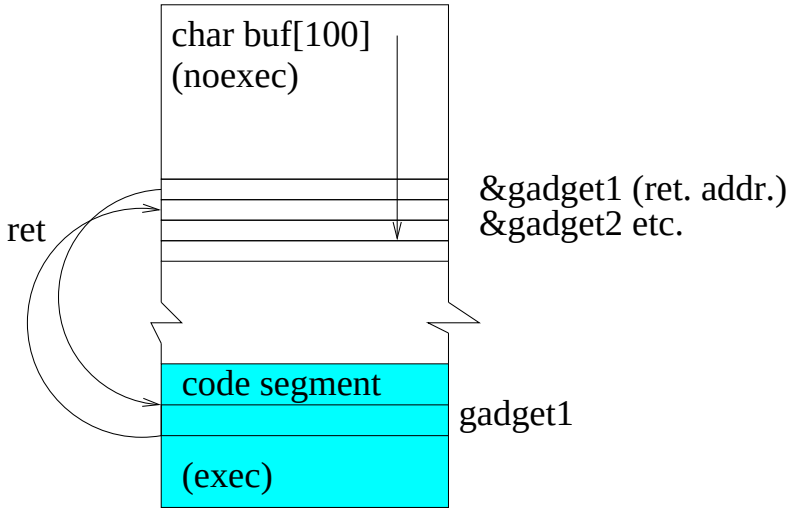
Classic buffer overflow exploits

- overwrites return address
- execute uploaded shellcode



Enter W^X

- no memory region is simultaneously writable and executable
- can no longer execute uploaded code



Enter ROP

- use code already provided by main executable/libs
- small code segments called ‘gadgets’, end in a `ret` instruction

- overwrite return adress with address of first gadget, then subsequent gadgets
- search for gadgets: look for `ret`, then search backwards for useful computations
- easier on architectures with variable instruction length (x86): `0xc3` anywhere can become an unintentional `ret`.
- not confined to `ret`, see: *Checkoway et al., 'Return-oriented Programming without returns'*.
- automated tools for 'gadget mining':
  https://github.com/JonathanSalwan/ROPgadget
  Metasploit: msfrop
- goal: **Turing-completeness** (though not even strictly necessary)
- => compiler for arbitrary code, see: *Buchanan et al., 'When Good Instructions go bad: Generalizing Return-Oriented Programming to RISC'*.

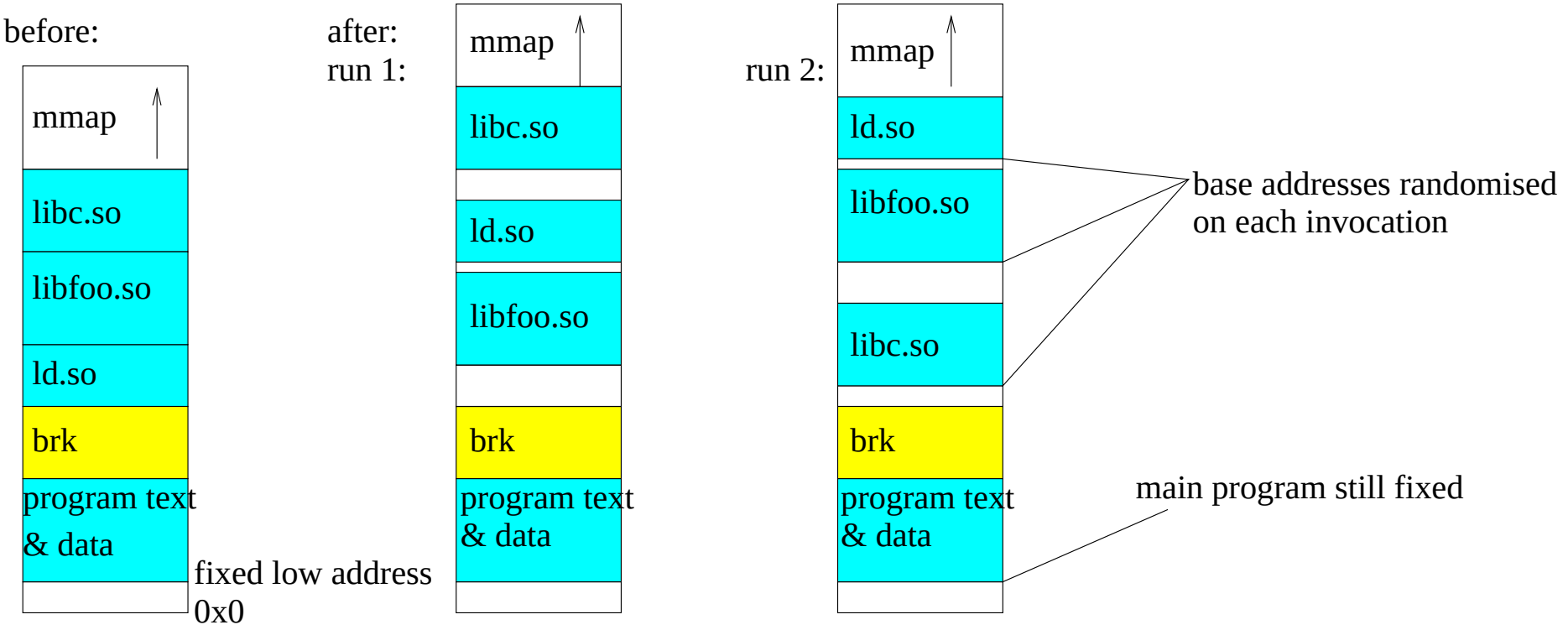
Conclusion: ROP is a powerful, pervasive and highly automated exploit technique that has to be taken seriously by OS vendors and application writers.

It is not a theoretical threat.

Fortunately, there's a decent mitigation: Address space layout randomisation (ASLR).

ASLR in OpenBSD

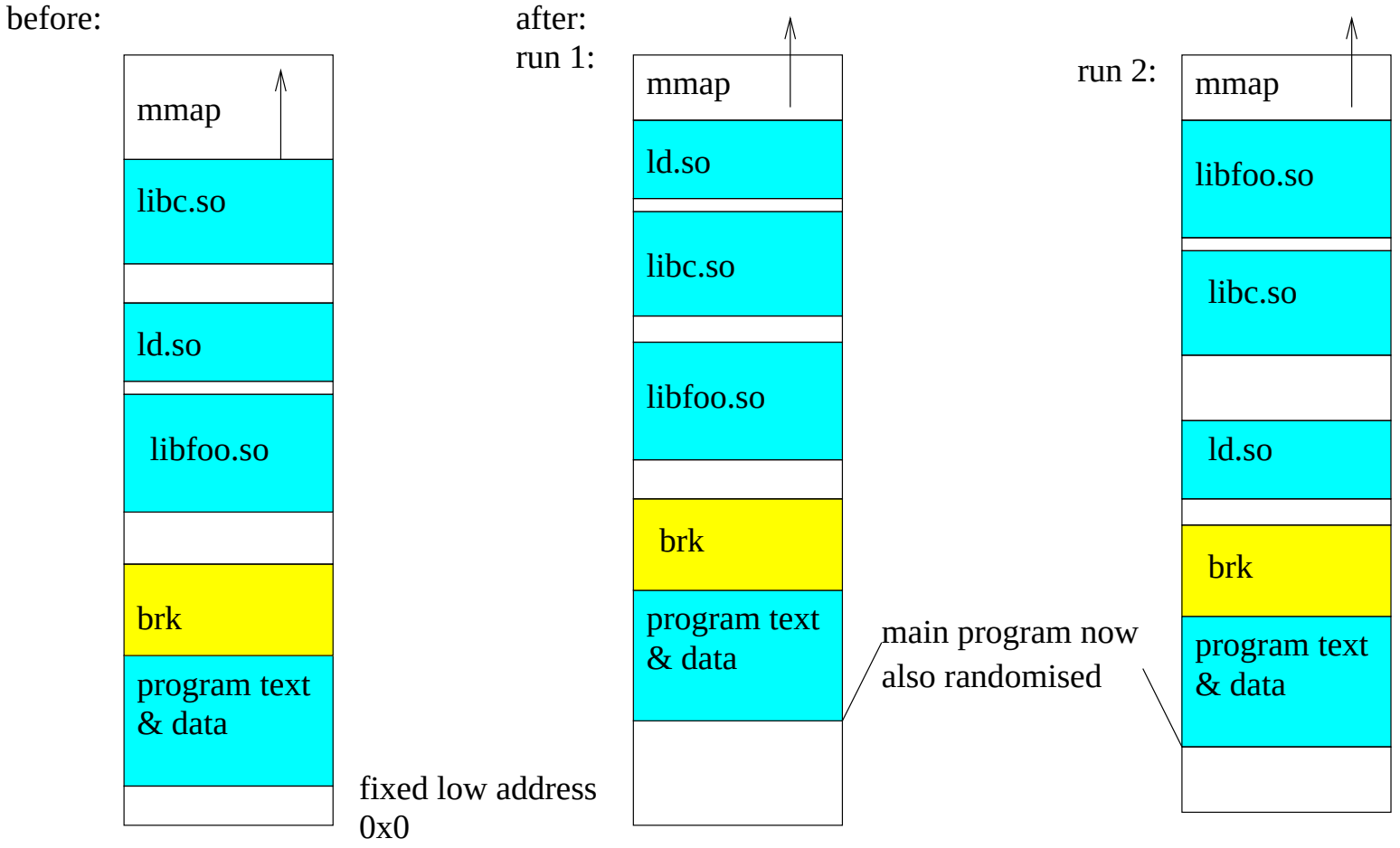
- OpenBSD has randomised the location and order of libraries since 3.4 (Nov 2003).
- each library has its own base address (also, random StackGap and mmap(2))



Introducing PIE

- PIC model allows shared libraries to be loaded anywhere in memory
- references to functions and global data go through tables of indirection: GOT (global offset table) and PLT (procedure linkage table)
- ‘PIC for executables’: PIE
- local global variables/functions can be optimised in PIE
- support added to GCC and GNU binutils in 2003
- complete support implemented in OpenBSD 4.5 by kurt@
- adjustments to kernel, runtime linker, debugger etc.
- `cc -fpie -pie foo.c` to produce a PIE

PIE address space



ROP against main program using absolute addresses impossible

OpenBSD: Secure by default

- knobs are for knobs
- security: not hidden behind a sysctl, not a 66k line kernel patch, not a compiler flag
- we try to turn on as many mitigations as possible **by default** and see what breaks
- with the goal of keeping Firefox, LibreOffice, GNOME, KDE4 etc. working
- examples: ProPolice, random mmap, malloc junking, increased StackGap size, fail on overlapping `memcpy(3)`
- all default, all across the system, all have found bugs in upstream software
- knobs provided to turn **on** security features can be abused by an attacker to turn them **off**.
- therefore: plan was all along to deploy PIE on a large scale, **compiler default**
- work started by kurt@, taken up again at g2k12
- finalised during 5.3 release cycle

The gory details

GCC implementation

- similar to `-fstack-protector`, but controlled via `bsd.own.mk` on an arch-dependent basis: default value of GCC's `flag_pie` variable is passed directly (1 for small `-fpie`, 2 for big `-fPIE`)
- exception: profiling code `-p` and `-pg` (profiling stack does not support PIC)

Binutils

- `/usr/bin/ld` likewise defaults to `-pie`
- flag added to turn off PIE: `-nopie`
- problem: `cc -static foo.o -o foo` produces binary with static libs, but depending on `ld.so` => make `-static` imply `-nopie`

```
PIE_ARCH=alpha amd64 hppa i386 mips64 mips64el powerpc sh
sparc64
```

Exceptions to default PIE

- new knob `NOPIE=` to turn off PIE selectively
- for now, make `LDSTATIC=` imply `NOPIE=` (no static PIE yet)
- bootloaders & kernel: add -fno-pie/-nopie as unconditional flags
- GCC's PCH implementation breaks when brk/sbrk is at different address => `NOPIE=` (for details on the format, read gcc/libcpp/pch.c)

ramdisks

- adding `NOPIE=` to ramdisk build system is easy, but: still uses system libc.a (with PIE objects)
- PIE code is bigger => overflow on i386
- guenther@'s solution: use a linkmap (`ld -M`) to recompile only those objects with `-fno-pie` that are needed

=> src is now ready!

Problems in Xenocara

(intentionally left blank)

Problems in the ports tree

- over 9000 ports (7800 as of 5.3), very very few have had issues

Compilers

- compilers that use the system linker are bitten by the `-pie` switch
- some have support for PIE and can be converted similarly to the base compiler: `lang/gcc/*`, `lang/gfortran`, `lang/g77`, `devel/llvm`
- some do not and have to pass `-nopie` on every invocation: `lang/fpc`, `lang/ghc`, `lang/gprolog`, `lang/sbcl`

Bootloaders and the like

- `sysutils/grub`, `sysutils/memtest86+` need PIE turned off

Assembler

- assembler that tries to access a global symbol without GOT/PLT or clobbers PIC register `%ebx` (i386)
- non-PIC-safe assembler should be marked as such (builtin define `__PIC__`). DO NOT USE `#ifdef __OpenBSD__` FOR THIS
- examples for `__PIC__`: `emulators/xnp2`, `multimedia/avidemux`, `security/aircrack-ng`
- some ports already have PIC-safe versions that just needed to be enabled: `emulators/dosbox`
- sometimes, it's easy to do yourself: `games/0ad`, `games/megaglest` (cpuid)
- PIE worsens register pressure on i386: some constraints imposed by inline asm can no longer be fulfilled (hello `-fPIC` ffmpeg)
- need to free up a register with `-fomit-frame-pointer`: `x11/mplayer`, `emulators/mupen64plus/video-glide64`, `emulators/openmsx`, `graphics/rawstudio`

Ports (continued)

Emacs

- some software makes assumptions about the address space incompatible with PIE (in this case, dump/undump at build time) => big hammer: disable PIE

- … and that's it!
- everything else just works™
- the upstream ecosystem is ready

Performance

- yes, there is some overhead, and more on i386 because of register pressure, but:
- performance hit is never more than with PIC
- benchmarks often fail to take the reality of the software ecosystem into account: most code is already outsourced to shared libraries, without performance complaints
- case in point: `bzip2(1)`: horrible measurements of 20% performance loss on i386, but:

```
/usr/local/bin/bzip2:
Start            End              Type Open Ref GrpRef Name
000014512e900000 000014512ed0a000 exe  1     0      0 /usr/local/bin/bzip2
00001453df9db000 00001453dfdeb000 rlib 0     1      0 /usr/local/lib/libbz2.so.10.4
00001453f7a79000 00001453f7f65000 rlib 0     1      0 /usr/lib/libc.so.78.1
00001453ce900000 00001453ce900000 rtld 0     1      0 /usr/libexec/ld.so
```

- all of the (de)compression code is already in a shared library.
- PIE has absolutely no effect on the real-world `bzip2(1)`

Static PIE

- static binaries (`/bin`, `/sbin`, some in `/usr/bin`) used as rescue binaries, must not depend on `/usr/libexec/ld.so`
- side effect until 5.6: code segment could not be randomised **at all** because static binaries could not do relocation
- affected programs: `/bin/ksh`, `/sbin/iked`, `/usr/bin/ftp` etc. :-(
- `ld.so` itself already has code to self-relocate when loaded at a random position in memory
- bring `_dl_boot_bind()` (MI) to `src/lib/csu`, call from MD code
- compiler modifications: need to use new `rcrt0.o` when making static PIE (for now, `-static -pie`)
- linker: create static PIEs with `DYNAMIC` flag set, but no `P_INTERP` section
- kernel: needs to recognise this
- 5.7: every static binary except `/sbin/init` uses this, on every arch that supports PIE

Other operating systems

Linux

- weak form of ASLR since 2.6.12, mostly enabled by default
- PIE as compiler default: Hardened Gentoo, Alpine Linux, OpenSUSE on its way (nice! :-)); Android doesn't support non-PIE since 5.0
- others use a selective approach (Ubuntu, Fedora, Debian, Arch) for performance concerns

offset2lib

- unfortunately, the kernel implementation is lacking
- loads first object at random offset, then all other objects **in sequence**
- address leak in main executable reveals the whole address space
- details: http://cybersecurity.upv.es/attacks/offset2lib/offset2lib.html
- patch proposed by authors, but not merged: creates new zone for PIE randomisation.

‘PaX is the solution’

- distributions that use PaX: Hardened Gentoo, Alpine Linux (no wide adoption in mainstream)
- has been a patch for almost 15 years, never integrated in mainline (and never will be?)
- exact opposite of OpenBSD’s integrated security
- general knobbiness: `paxctl(1)` to modify a header in the binary that tells the kernel which protections to enable/disable
- these hacks are **required** to make applications behave: https://en.wikibooks.org/wiki/Grsecurity/Application-specific_Settings
- Chromium:

```
$ paxctl -v /opt/google/chrome/chrome
PaX control v0.5
Copyright 2004,2005,2006,2007 PaX Team <pageexec@freemail.hu>
- PaX flags: P----m-x-eR- [/opt/google/chrome/chrome]
          PAGEEXEC is enabled
          MPROTECT is disabled
          RANDEXEC is disabled
          EMUTRAMP is disabled
          RANDMMAP is enabled
```

FreeBSD

- worse than Linux: no ASLR, totally predictable address space
- nudges in the right direction: kernel ASLR patch exists, but still under review: https://reviews.freebsd.org/D473
- some effort for default PIE in base, but devil is in the details:
- not implemented as compiler default, but flags passed from `bsd.prog.mk`
- result: huge problems with dynamic programs using static libs (non-PIE)
- please rethink; imagine the mess in ports!
- W^X? ('NX support' is NOT the same thing)
- library load order? rtld self-relocation?

OSS upstream vendors

- many upstream projects want to enable PIE (GNU autoconf `--enable-gcc-hardening`): tor, qemu, pidgin …

Windows

- Microsoft has gotten the message
- DEP since XP SP3, ASLR since Vista, /DYNAMICBASE compiler default since Visual Studio 2010
- good responses to new exploits (e.g. heap guard pages, removal of address leaks)

Mac OS X

- PIE by default since 10.7
- KASLR since 10.8: kernel location randomised on each boot
- weaknesses: libraries only re-randomised when software is updated or on reboot (prebinding), incomplete NX (only stack and heap)

TODO

- binutils 2.17 for arm PIE support
- change `cc -static` default to static PIE
- `/sbin/init` static PIE :-)

Future directions in ROP mitigation

- application writers need to become defensive about ROP (see `OPENSSL_indirect_call` function)
- BROP: *Bittau et al., 'Hacking Blind'*: bruteforcing stack canary and ROP gadgets in forking daemons with ASLR and W^X enabled. remote root shell with vulnerable nginx/MySQL/yaSSL in 20min. => solution: fork + exec (OpenSSH)? too expensive for webservers? threads?
- gfree: eliminate gadgets in binaries
- shuffle around `.o` files **inside** libraries/executables?
- control-flow integrity: next generation of mitigations, existing prototype for LLVM

Conclusions

- default PIE is a necessary step in the ROP arms race
- quirks have been worked out
- now OSS vendors need to catch up

Thanks

- to kurt@ for doing most of the work beforehand :-)
- to deraadt@ for relentlessly pushing me in the right direction (and making me give this talk)
- to kettenis@, matthew@, miod@, jsg@ for giving valuable feedback on every step of the conversion in src
- to naddy@, sthen@, espie@, brad@ and many other ports developers for help with getting the ports tree ready, doing bulk builds etc.

… any questions?