OpenBSD: una workstation segura

De cero a kernel hacking en 2hs

by ToscoSys & Anatoli
BSDar @ eko15
Buenos Aires, 2019

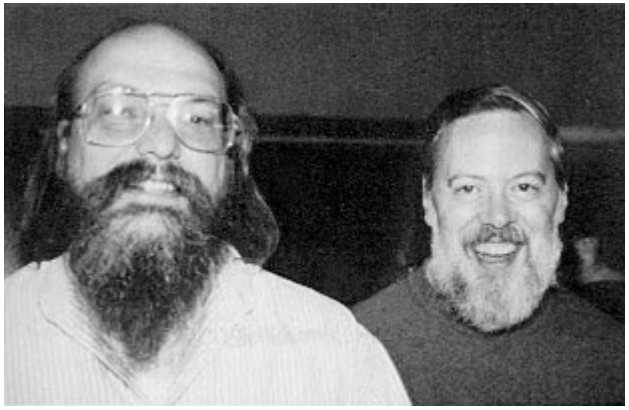
Índice

Parte 1

Parte 2

Intro: el mundo BSD – historia

Bell Labs: laboratorio I+D de AT&T



La cuna del soft moderno y tecnología relacionada

- Inventando cosas que usamos desde finales del siglo XIX
- El primer transistor, láser, celda fotovoltaica, CCD, etc.
- Teoría de información, lenguajes L2 (1956), B, C, C++, Fortran, compiladores, make, UNIX, Plan9, UTF-8, bash, awk, etc. etc.
- Ken Thompson & Dennis Ritchie, Brian Kernighan, Bjarne Stroustrup, Rob Pike, Steve Bourne (9 premios Nobel)

UNIX

- En los 1970 AT&T necesitaba software avanzado, pero tenía prohibida su comercialización
- Colaboración con las universidades, software bajo licencia muy económica y código visible
- Mejoras por los estudiantes: job control, Fast File System (FFS), TCP/IP, etc.
- Coordinación: Computer Science Research Group (**CSRG**) at the University of California, Berkeley

Intro: el mundo BSD – historia, p2

CSRG

- Centro de coordinación e intercambio de código de todas las universidades y organizaciones
- También desarrollos como TCP/IP financiados por DARPA & co
- Se conocía como **Berkeley Software Distribution (BSD)**
- De 1979 a 1994: empaquetado y distribuido gratis a todos los que poseían licencia UNIX
- Principios de proyecto de código abierto (revolucionario para aquella época)
- En 15 años casi todo el código original de UNIX fue reemplazado
- Disminución de financiamiento y problemas políticos internos alrededor de 1992
- El código es liberado al público general bajo licencia BSD (juicio por AT&T, pierden mal)

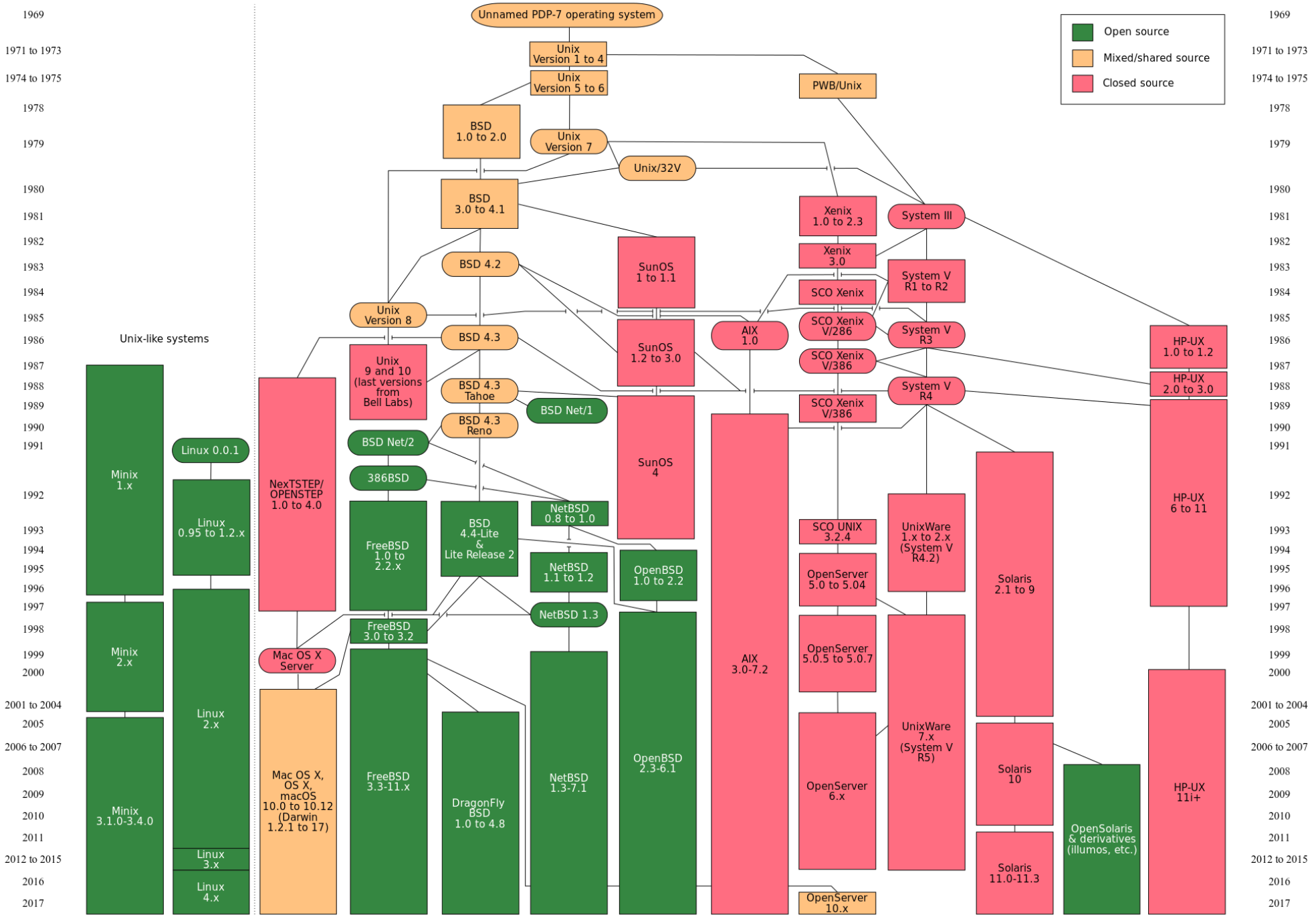
Licencia BSD (copycenter)

- No digas que escribiste esto
- No nos culpes si se rompe
- No uses nuestro nombre para promocionar tu producto

Intro: el mundo BSD – historia, p3

Código BSD: todos buscan aprovechar calidad con licencia permisiva

- Código cerrado: System V R4, SunOS, AIX, Solaris, HP-UX, NexTSTEP/Darwin/Mac OS X
- Código abierto: BSD 4.3 Reno → BSD Net/2 (1991 sin código de AT&T) → 386BSD + *BSD 4.4-Lite*
  - 386BSD + *BSD 4.4-Lite* → **FreeBSD** (1993) por un grupo de usuarios de 386BSD
  - 386BSD + *BSD 4.4-Lite* → **NetBSD** (1993) por Chris Demetriou, Theo de Raadt, Adam Glass, y Charles Hannum
  - NetBSD (Theo removed in 1994) → **OpenBSD** (1995) by Theo de Raadt
  - FreeBSD 4.8 → DragonFly BSD (2003) por Matthew Dillon
- Logo: Beastie (B-S-D), The BSD Daemon por John Lasseter (Toy Story, Cars, etc.), copyright de Marshall Kirk McKusick
- **BSDar** (2019)


Open source
Mixed/shared source
Closed source
Unix-like systems
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2015
2016
2017
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
Unix Version 7
PWB/Unix
BSD 1.0 to 2.0
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
BSD 4.2
SunOS 1 to 1.1
System V R1 to R2
SCO Xenix
Unix Version 8
SCO Xenix V/286
AIX 1.0
System V R3
BSD 4.3
SunOS 1.2 to 3.0
HP-UX 1.0 to 1.2
SCO Xenix V/386
HP-UX 2.0 to 3.0
Unix 9 and 10 (last versions from Bell Labs)
BSD 4.3 Tahoe
BSD Net/1
System V R4
BSD 4.3 Reno
SCO Xenix V/386
Minix 1.x
Linux 0.0.1
NeXTSTEP/ OPENSTEP 1.0 to 4.0
BSD Net/2
SunOS 4
386BSD
HP-UX 6 to 11
NetBSD 0.8 to 1.0
SCO UNIX 3.2.4
UnixWare 1.x to 2.x (System V R4.2)
Linux 0.95 to 1.2.x
FreeBSD 1.0 to 2.2.x
BSD 4.4-Lite & Lite Release 2
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
Solaris 2.1 to 9
NetBSD 1.3
Minix 2.x
FreeBSD 3.0 to 3.2
OpenServer 5.0.5 to 5.0.7
Linux 2.x
Mac OS X Server
AIX 3.0-7.2
UnixWare 7.x (System V R5)
Mac OS X, OS X, macOS 10.0 to 10.12 (Darwin 1.2.1 to 17)
FreeBSD 3.3-11.x
DragonFly BSD 1.0 to 4.8
NetBSD 1.3-7.1
OpenBSD 2.3-6.1
OpenServer 6.x
Solaris 10
OpenSolaris & derivatives (illumos, etc.)
HP-UX 11i+
Minix 3.1.0-3.4.0
Linux 3.x
Linux 4.x
Solaris 11.0-11.3
OpenServer 10.x

Obra de **Phil Foglio** para el décimo aniversario de USENIX (1985)

https://www.mckusick.com/beastie/shirts/usenix.html

Intro: BSDar – Comunidad *BSD Argentina

Una vez en la juntada AdminBirras (SysArmy)

- Y por qué no armamos una juntada BSD?

- Y dale!..

BSDar: https://github.com/bsdar/community

- Comunidad abierta, sin fines de lucro, todos los miembros son iguales, coordinación transparente, manda el voto de la mayoría
- Fundada en mayo 2019 por Pablo Carboni, ToscoSys, Anatoli, Vampii, Motor, Franco, Jedux, Wen, Марсело, Thonhy y otros
- Meetups el tercer jueves de cada mes (+ próximamente hackatons)
- Principal grupo en Telegram: https://t.me/BSDar (presentarse!)

BSDar

Comunidad *BSD Argentina

OpenBSD

Qué es?

- Un SO *entero*, creado en 1995 por Theo de Raadt como fork de NetBSD
- Foco: seguro, correcto, usable, minimalista y libre
- Acceso público de sólo lectura al repositorio CVS & commit logs
- Desarrollo abierto, comunicación vía diffs en mailinglists

Por qué?

- OpenBSD rocks! Facts: https://why-openbsd.rocks/fact/
- What is you motivational to use OpenBSD? https://www.mail-archive.com/misc@openbsd.org/msg169012.html

+ 64bit Time
+ ASLR
+ Anti-ROP
+ KARL
+ LibreSSL
+ License
+ base system concept
+ No *-dev packages
+ OpenSMTPD
+ OpenSSH
+ PID randomization
+ PIE
+ Privilege Separation
+ Reliable release model
+ Sane & secure defaults
+ UTF-8 only
+ W^X Memory
+ Xserver without root permissions
+ acme-client(1)
+ afterboot(8)
+ arc4random(3)
+ autoinstall(8)
+ Memory allocation randomization
+ carp(4)
+ doas(1)
+ file(1)
+ freezero(3)
+ httpd(8)
+ mandoc(1)
+ ntpd(8)
+ openrsync
+ pf(4)
+ ping(8) randomness
+ pledge(2)
+ rcctl(8)
+ rdist(1)
+ recallocarray(3)
+ relayd(8)
+ securelevel(7)
+ sensorsd(8)
+ signify(1)
+ slowcgi(8)
+ spamd(8)
+ stack-register checking
+ strlcpy(3), strlcat(3)
+ sysmerge(8)
+ syspatch(8)
+ sysupgrade(8)
+ tmux(1)
+ unveil(2)
+ unwind(8)
+ vmd(8)
+ xenodm: the secure X Display manager

- I'm a "linux guy" who wants a little bit more security…

- I use OpenBSD because it can do everything I want it to do and it’s easy to use.

- This is one of the ardently appealing factors of OpenBSD; technical quality isn't compromised or superseded by other arbitrary and subjective measures. The singular focus on technical quality and correctness is reassuring and consistently produces a reliable, performant product. It's certainly a quality that I find appealing and keeps me looking to contribute however I can to the project.

- I use OpenBSD because there are no surprises when you install a service. The service is not started until you start it. Even if it started inadvertently, the config will have 'sane' defaults and not get you breached.

- Simplicity. Clean. Lean and Slim. Work as advertised. Secure.

- The vastly superior mascot and soundtrack.

- I enjoy using it because of it's clean design. It's a fairly simple system, with sane default configuration and it "just works" on most laptops that I've used it on.

- The objective of bug free code is nearly unheard of in the industry and in fact where it is paid lip-service it is not sufficiently served. The objective and means of obtaining security is among the most accurate, most logical, and least impulsive strategies in the industry. Stability, fitness for service and security combine to form a practical morality in applied software engineering that I find compelling.

- What I really like in the OpenBSD team is the ability to take correct decisions and not trying to be consumer friendly or following a trend.

- My number 1 motivation for moving to BSDs in particular is the system(d) cancer in linux, what they forced down on our throats. It's a free world so you can use whatever OS you want.

- You'll never know how good OpenBSD is until you try it. That's what I did.

OpenBSD: Filosofía

OpenBSD: defensa en profundidad

https://www.openbsd.org/security.html y https://www.openbsd.org/innovations.html

Randomización

arc4random

OpenBSD has its own cryptographic random number generator. Wherever random input is needed, arc4random is used. arc4random is an abstraction layer for currently considered as safe ciphers and produces ChaCha20 ciphers at the moment. Arc4random is "A Replacement Call For Random", to generate very quickly high quality 32-bit pseudo-random numbers. It's a syscall, i.e. no need to have access to /dev/random.

ASLR

Address Space Layout Randomization places code, data and stack in randomly selected location in the memory of the OpenBSD Operating System. As a result every execution of a binary ends up in a different layout. This makes it hard for an attacker to predict memory addresses and process behavior.

PIE (Position Independent Code)

A PIE binary and all of its dependencies are loaded into random locations within virtual memory every time the application is executed. This heavy use of randomization makes it hard for an attacker to predict the binary’s behavior.

KARL

At every install, upgrade, and boot a new kernel is generated with randomized addresses. A unique and unpredictable kernel is a huge security improvement. This technique is called Kernel Address Randomized Link. A unique kernel is linked such that the startup assembly code is kept in the same place, followed by randomly-sized gapping, followed by all the other .o files randomly re-organized. As a result the distances between functions and variables are entirely new. An info leak of a pointer will not disclose other pointers or objects.

Anti-ROP

The order of symbols in libc.so are randomized at boot time to prevent “Return oriented programming”. An attacker gains control of the call stack to hijack program control flow and then executes carefully chosen machine instruction sequences that are already present in the machine’s memory. With randomized symbols, this is not an attack vector anymore.

Memory allocation randomization

malloc allocates areas of memory that programs have requested using system calls. It randomizes memory allocations over the entire address space. This makes attacks harder because each run has a different memory layout. It traps bugs (allocations are surrounded by unmapped memory) and allows realloc to grow an allocation without copying in most cases.

PID randomization

Each new process has a random, unused PID. This protects the user from attacks that predict new PIDs.

Eliminación/desactivación de lo innecesario

securelevel

1 (Secure mode) the raw memory devices can not be written to, the raw devices of mounted file systems can not be written to, important kernel variables are locked down. Actually, this is the mode by default.

2 (Highly secure mode) raw disk devices can not be written to, certain time related functions are locked down so the time cannot be set in the past and pf rules may not be altered. This ultimate mode is a last line of defence if the superuser account is compromised or limit potential damage.

No audio/video capture by default

For privacy reasons, the OpenBSD team disabled audio recording for all devices by default in the kernel. This can be toggled on/off with a simple sysctl change, without rebooting.

Non-root Xserver

OpenBSD developers have worked hard to enable the non-root execution of an Xserver. Since 2014, X no longer requires special privileges and can be run as a regular user rather than root.

No bluetooth, no linux-compat layer

Too complex to implement it right.

Servicios/librerías/herramientas fortificadas

LibreSSL

The unmaintainable and bloated OpenSSL Codebase was forked after Heartbleed was revealed. The code was thoroughly cleaned up, improved and documented. Besides new modern ciphers FRP256v1, RFC 5639 EC Brainpool, ChaCha20, Poly1305, LibreSSL is API compatible with OpenSSL but without the mess. It is actively developed.

Chrooted daemons

By default, HTTP and other daemons are chrooted in /var/www. As an OpenBSD system administrator you don’t need to configure anything to have a secured webserver installation running.

doas

doas replaced sudo because of the former’s security flaws and large, complex codebase. doas is easy to configure and use and suits most use cases. Its source code is small and elegant too.

xenodm: the secure X Display manager

xenodm is the X Display Manager, since OpenBSD 6.1! xenodm is a simplified fork of xdm, lightweight, more secure, rid of XDMCP support, because of many security vulnerabilities. It support only the BSDauth code used in OpenBSD. On OpenBSD 6.5, xenodm is absolutly necessary to start the X server, because it no longer has setuid rights enabled by default.

Mitigaciones misceláneas

Meltdown & Spectre prevention

To mitigate the Meltdown and Spectre vulnerabilities, OpenBSD holds separate page tables for the kernel and userland. The next generation of Meltdown was mitigated in this way. OpenBSD was no longer affected as memory could not be inappropriately accessed.

Sane & secure defaults

OpenBSD has sane and secure defaults set in daemons and configurations. The system is intended to be secure by default, and many of its security features are either missing or optional in other operating systems. This means you don’t have to tweak your freshly installed operating system to get services running. There is no hardening process required when you setup sshd(8), for example. Just as for every other daemon or component in the base system.

signify

signify is a small and elegant tool to cryptographically sign and verify files. It was created to sign OpenBSD releases, since OpenBSD 5.5, and Binary Patches for syspatch. It uses only the Ed25519 algorithm. Think of it as a simple, easy replacement for PGP signing.

Needed functionality inside chroots vía syscalls

syslog and arc4random are syscalls eliminating the need to have access to special files inside chroots.

Memoria

W^X Memory

Since 2003, memory on OpenBSD can be written to or executed, but not both. This is a major security feature that prevents malicious code from producing buffer overflows and executing what has been inserted.

freezero

The libc function freezero(3) allows programs to free memory that holds sensitive data, and to overwrite it with zeros.

recallocarray

recallocarray is a libc function that ensures data is discarded before allocating new memory and checks for integer overflow from multiplication.

stack-register checking

A memory object should have the fewest permissions possible: typically read, write and execute. OpenBSD introduced a new permission flag known as stack. If you want to use memory as a stack, you must mmap it with that flag bit. When a system call happens, we check to see if the stack-pointer register points to such a page. If not, the program is killed. The ABI is tighter as a result. You may no longer point your stack register at non-stack memory, or your program will die.

strlcpy, strlcat

This kernel function will safely copy/concat strings, an improvement over strncpy(3), strncat(3).

malloc options

Upon the first call to the malloc() family of functions, an initialization sequence inspects the value of the vm.malloc_conf sysctl(2), next checks the environment for a variable called MALLOC_OPTIONS, and finally looks at the global variable malloc_options in the program.

C “Canaries”. Add canaries at the end of allocations in order to detect heap overflows. The canary's content is checked when free is called. If it has been corrupted, the process is aborted.

F “Freecheck”. Enable more extensive double free and use after free detection. All chunks in the delayed free list will be checked for double frees. Unused pages on the freelist are read and write protected to cause a segmentation fault upon access.

G “Guard”. Enable guard pages. Each page size or larger allocation is followed by a guard page that will cause a segmentation fault upon any access.

J “More junking”. Increase the junk level by one if it is smaller than 2.

j “Less junking”. Decrease the junk level by one if it is larger than 0. Junking writes some junk bytes into the area allocated. Junk is bytes of 0xdb when allocating; freed chunks are filled with 0xdf. By default the junk level is 1: after free, small chunks are completely junked; for pages the first part is junked.

R “realloc”. Always reallocate when realloc() is called, even if the initial allocation was big enough.

U “Free unmap”. Enable use after free protection for larger allocations. Unused pages on the freelist are read and write protected to cause a segmentation fault upon access.

S Enable all options suitable for security auditing.

OpenBSD: defensa en profundidad (cont.)

pledge

Linux (2005): **seccomp** (secure computing mode), a computer security facility in the Linux kernel intended as a means of safely running untrusted compute-bound programs. seccomp allows a process to make a one-way transition into a "secure" state where it cannot make any system calls except exit(), sigreturn(), read() and write() to already-open file descriptors.

FreeBSD (2012): **capsicum**, a lightweight OS capability and sandbox framework

OpenBSD (2016): **pledge**, a syscall to force the current process into a restricted-service operating mode.

pledge() allows you to limit a program’s access to system calls very easily. This is a huge improvement in security: why should cut(1) ever need to open a socket? Just deny it the ability to do so. Even if a binary is compromised, its chances to misbehave are greatly reduced.

Within only two releases, the OpenBSD Developers managed to introduce pledge to most of the binaries in the base system, unlike other OS that had similar capabilities for years and had almost zero adoption.

Pledge: realistic subsets of POSIX functionality (https://man.openbsd.org/pledge.2):

- Named subsets "dns", "stdio", "wpath", "rpath" → easy to learn to add to programs
- No subtle behaviour changes
- No error returns → illegal operations crash the program (SIGABRT)

unveil

Unveil parts of a restricted filesystem view. The unveil system call limits the filesystem open call to a given set of paths. The first call to unveil() removes visibility of the entire filesystem from all other filesystem-related system calls (such as open(2), chmod(2) and rename(2)), except for the specified path and permissions.

The unveil() system call remains capable of traversing to any path in the filesystem, so additional calls can set permissions at other points in the filesystem hierarchy. https://man.openbsd.org/unveil.2

After establishing a collection of path and permissions rules, future calls to unveil() can be disabled by passing two NULL arguments. Alternatively, pledge(2) may be used to remove the “unveil” promise.

It extends the idea of pledge: simply limiting programs to open is insufficient, because open is valid for the whole filesystem. For example, why should a program like passwd have access to your file system beyond /etc/passwd and /etc/shadow? If there is a security bug in passwd then with unveil its effects would be quite limited.

Firefox case (https://bugzilla.mozilla.org/show_bug.cgi?id=1580271 **paths for rw access)**:

```
unveil("/dev/video rw,
        /tmp rwc,
        /var/run rw,
        ~/.mozilla rwc,
        ~/.pki rwc,
        ~/.sndio rwc,
        $XDG_CACHE_HOME/dconf rwc,
        $XDG_CACHE_HOME/thumbnails rwc,
        $XDG_CONFIG_HOME/mozilla rwc,
        $XDG_DATA_HOME/applications rwc,
        ~/Downloads rwc")
```

OpenBSD: desventajas

- Performance
  - kernel no SMP: pf, interrupts y kernel en general single core
  - FFS (UFS), el único FS soportado para base, fue creado pre-OpenBSD
- Soporte incompleto de hardware/features (802.11ac)
- La frase: *"Sólo dos agujeros remotos en la instalación base en mucho tiempo!"* – conflicto de intereses

  BTW, exploit para el 2do agujero (OpenBSD 4.0) fue anunciado en 2007 por Alfredo Ortega & Gerardo Richarte de Core Security, Argentina[1]

1 https://www.coresecurity.com/sites/default/private-files/publications/2016/05/OpenBSD-remote-exploit.pdf

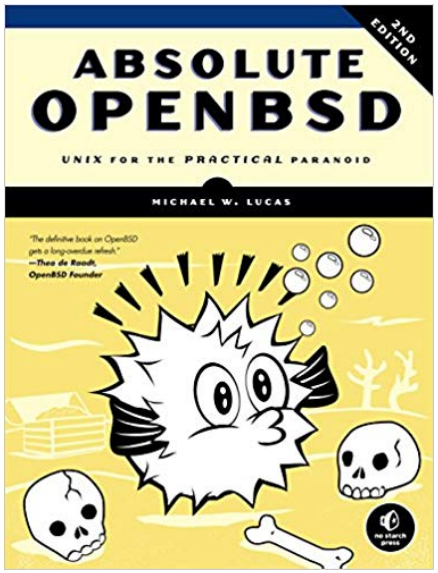
OpenBSD: desventajas, p2

- Pobre análisis/revelación(?) de las vulnerabilidades

OpenBSD: 001: RELIABILITY FIX All architectures. If a userland program sets the IPv6 checksum offset on a raw socket, an incoming packet could crash the kernel. ospf6d is such a program.
→ **El mismo reporte ("008: RELIABILITY FIX") en el mismo subsistema (IPv6) fue convertido en un (2do) exploit remoto en 2007**

Xen: XSA-287 "x86: steal_page violates page_struct access discipline": Xen's reference counting rules were designed to allow pages to change owner and state without requiring a global lock. Each page has a page structure, and a very specific set of access disciplines must be observed to ensure that pages are freed properly, and that no writable mappings exist for PV pagetable pages.
Unfortunately, when the XENMEM_exchange hypercall was introduced, these access disciplines were violated, opening up several potential race conditions.
A single PV guest can leak arbitrary amounts of memory, leading to a denial of service.
A cooperating pair of PV and HVM/PVH guests can get a writable pagetable entry, leading to information disclosure or privilege escalation.
Privilege escalation attacks using only a single PV guest or a pair of PV guests have not been ruled out.
Note that both of these attacks require very precise timing, which may be difficult to exploit in practice.

OpenBSD: cómo resolver problemas



- https://www.openbsd.org/faq/index.html
- Absolute OpenBSD, 2nd Edition (2013)
- man
- https://www.reddit.com/r/openbsd/
- https://unix.stackexchange.com/questions/tagged/openbsd
- https://t.me/BSDar
- https://www.openbsd.org/mail.html
  - announce: anuncios (new releases, erratas, etc.) – suscribirse!
  - misc: para hacer preguntas bien preparadas (sólo-lectura para principiantes)
  - ports: temas altamente técnicos sobre los ports (ahí están los maintainers)
  - bugs: reporte de bugs
  - tech: kernel & base (sólo para devs y usuarios avanzados)

OpenBSD flavors

-release

- congelado en el tiempo
- cada 6 meses
- último: 6.5 (2019-09; 6.6 planificado para 2019-10)

-stable: release + patches de erratas

- cuando publiquen erratas (errores críticos de seguridad y estabilidad)
- lo más apropiado para producción

-current (builds nocturnos del repo)

- todos los días
- para los que quieran tener lo último (soporte hardware & features)

Setup inicial

Preparación del medio de instalación

```
$ mkdir -p ~/tmp/openbsd && cd ~/tmp/openbsd
$ wget https://cdn.openbsd.org/pub/OpenBSD/6.5/amd64/install65.fs \
https://cdn.openbsd.org/pub/OpenBSD/6.5/amd64/SHA256.sig \
https://ftp.openbsd.org/pub/OpenBSD/6.5/openbsd-65-base.pub
$ sha256sum openbsd-65-base.pub
 7fcc2aec60009be208389b7f0dcff148232eb3fb3cd750b532942c7ec4dfb4fe openbsd-65-base.pub
$ sudo apt -y install signify-openbsd
$ signify-openbsd -Cp openbsd-65-base.pub -x SHA256.sig install65.fs
 Signature Verified
 install65.fs: OK
$ sudo fdisk -l /dev/sdb
$ sudo dd if=install65.fs of=/dev/sdb bs=4M status=progress
```

Setup inicial

Incorporación del firmware al medio de instalación

```
$ echo -e "n\np\n2\n\n+500M\nt\n2\nb\nw\n" | sudo fdisk /dev/sdb
$ sudo fdisk -l /dev/sdb
 Disk /dev/sdb: 14,4 GiB, 15502147584 bytes, 30277632 sectors
 Disklabel type: dos

 Device     Boot  Start     End Sectors   Size Id Type
 /dev/sdb1           64    1023     960   480K ef EFI (FAT-12/16/32)
 /dev/sdb2       921600 1945599 1024000   500M  b W95 FAT32
 /dev/sdb4  *      1024  921535  920512 449,5M a6 OpenBSD

$ sudo mkfs.msdos /dev/sdb2
$ sudo mount /dev/sdb2 /mnt
$ wget --recursive --no-parent http://firmware.openbsd.org/firmware/6.5/
$ sudo cp firmware.openbsd.org/firmware /mnt/ && sudo umount /mnt
```

Setup inicial

Instalación de OpenBSD con FDE (full disk encryption)

```
probing: pc0 mem[640K 1005M 16M 536K 64K]
disk: hd0 hd1 sr0*
>> OpenBSD/amd64 BOOT 3.43
boot>
...
root on rd0a swap on rd0b dump on rd0b
erase ^?, werase ^W, kill ^U, intr ^C, status ^T

Welcome to the OpenBSD/amd64 6.5 installation program.
(I)nstall, (U)pgrade, (A)utoinstall or (S)hell? s
```

```
# dmesg | grep "^[sw]d"      # identificar el disco para la instalación (sd0)
# dd if=/dev/zero of=/dev/rsd0c bs=1m count=1 # borrar estructura previa
    # r = raw device (o sea, character; sino sería block device), c = disco entero
# fdisk -y -ig -b 960 sd0    # escribir nueva estructura GPT
```

Setup inicial

```
# disklabel -E sd0
 Label editor (enter '?' for help at any prompt)
 sd0> a a
 offset: [1024] ← [enter] para aceptar el valor por defecto
 size: [39825135] *
 FS type: [4.2BSD] RAID
 sd0> w
 sd0> q
 No label changes.
bioctl -l sd0a -c C -r auto softraid0  # -r auto = retraso de 1 segundo
New passphrase:
Re-type passphrase:
sd2 at scsibus1 targ 1 lun 0: <OPENBSD, SR CRYPTO, 006> SCSI2 0/direct fixed
sd2: 16383MB, 512 bytes/sector, 33552817 sectors
softraid0: CRYPTO volume attached as sd2 ← es el CRYPTO volume que vamos a usar
```

```
# cd /dev && sh MAKEDEV sd2 # hay que crear el device node para el disco
# exit
 Welcome to the OpenBSD/amd64 6.5 installation program.
 (I)nstall, (U)pgrade, (A)utoinstall or (S)hell? i
 ...
 Start sshd(8) by default? [yes] no
 Do you want the X Window System to be started by xenodm(1)? [no] yes
 Setup a user? [no] user1
 ...
 Available disks are: sd0 sd1 sd2.
 Which disk is the root disk? ('?' for details) [sd0] sd2 # acá va el
                                                          # CRYPTO volume
 No valid MBR or GPT.
 Use (W)hole disk MBR, whole disk (G)PT or (E)dit? [gpt] G
 Setting OpenBSD GPT partition to whole sd2...done.
 The auto-allocated layout for sd2 is:
```

Setup inicial

WTF is disklabel?

```
#                size           offset  fstype [fsize bsize   cpg]
  a:          450592.0K             1024  4.2BSD   2048 16384  7012 # /
  b:          675905.0K           902208    swap                    # none
  c:        16776408.5K                0  unused   # disco entero, su tamaño también
  d:          598048.0K          2254048  4.2BSD   2048 16384  9307 # /tmp
  e:          854096.0K          3450144  4.2BSD   2048 16384 12958 # /var
  f:         1628192.0K          5158336  4.2BSD   2048 16384 12958 # /usr
  g:          571408.0K          8414720  4.2BSD   2048 16384  8892 # /usr/X11R6
  h:         1939552.0K          9557536  4.2BSD   2048 16384 12958 # /usr/local
  i:             480.0K               64   MSDOS   # particiones del MBR fuera del
                                          # layout de disklabel
  j:         1449984.0K         13436640  4.2BSD   2048 16384 12958 # /usr/src
  k:         5480464.0K         16336608  4.2BSD   2048 16384 12958 # /usr/obj
  l:         3127600.0K         27297536  4.2BSD   2048 16384 12958 # /home
```

Setup inicial

WTF is disklabel? (continuación)

`$ man 5 disklabel`: Disk pack labels (disklabels) are used to manage OpenBSD filesystem partitions. They contain certain details about your disk, such as drive geometry and filesystem information. This can help overcome some architectures' disk partitioning limitations. For example, on i386, there are only **four primary MBR partitions** available. With disk labels, one of these primary partitions contains all your OpenBSD partitions, while the other three are still available for other operating systems.

It should be initialized when the disk is formatted, and may be changed later with the disklabel(8) program. This information is used by the system disk driver and by the bootstrap program to determine how to program the drive and where to find the filesystems on the disk partitions.

Relacionado:

- DUID (como blkid en Linux): `sysctl hw.disknames`
- Layout of filesystems: `man hier` (https://man.openbsd.org/hier)

Setup inicial

Instalación de OpenBSD con FDE (continuación)

```
Use (A)uto layout, (E)dit auto layout, or create (C)ustom layout? [a] ← [enter]
Which disk do you wish to initialize? (or 'done') [done] ← [enter]
Location of sets? (disk http or 'done') [http] disk  # puede también tener cdrom, que
        sería la opción a elegir si la instalación se hace desde un install65.iso
Is the disk partition already mounted? [yes] no
Available disks are: sd0 sd1 sd2.  # sd0 = SSD, sd1 = pendrive, sd2 = el nuevo CRYPTO
# si ctrler fuese IDE podría ser así: wd0 = HDD, sd0 = pendrive, sd1 = el nuevo CRYPTO
Which disk contains the install media? (or 'done') [sd0] sd1
  a:          920512             1024  4.2BSD   2048 16384 16142
  i:             960               64  MSDOS
Which sd1 partition has the install sets? (or 'done') [a] ← [enter]
Pathname to sets? (or 'done') [6.5/amd64] ← [enter]
Set name(s)? (or 'abort' or 'done') [done] -game*
Set name(s)? (or 'abort' or 'done') [done] -x*
```

Install sets

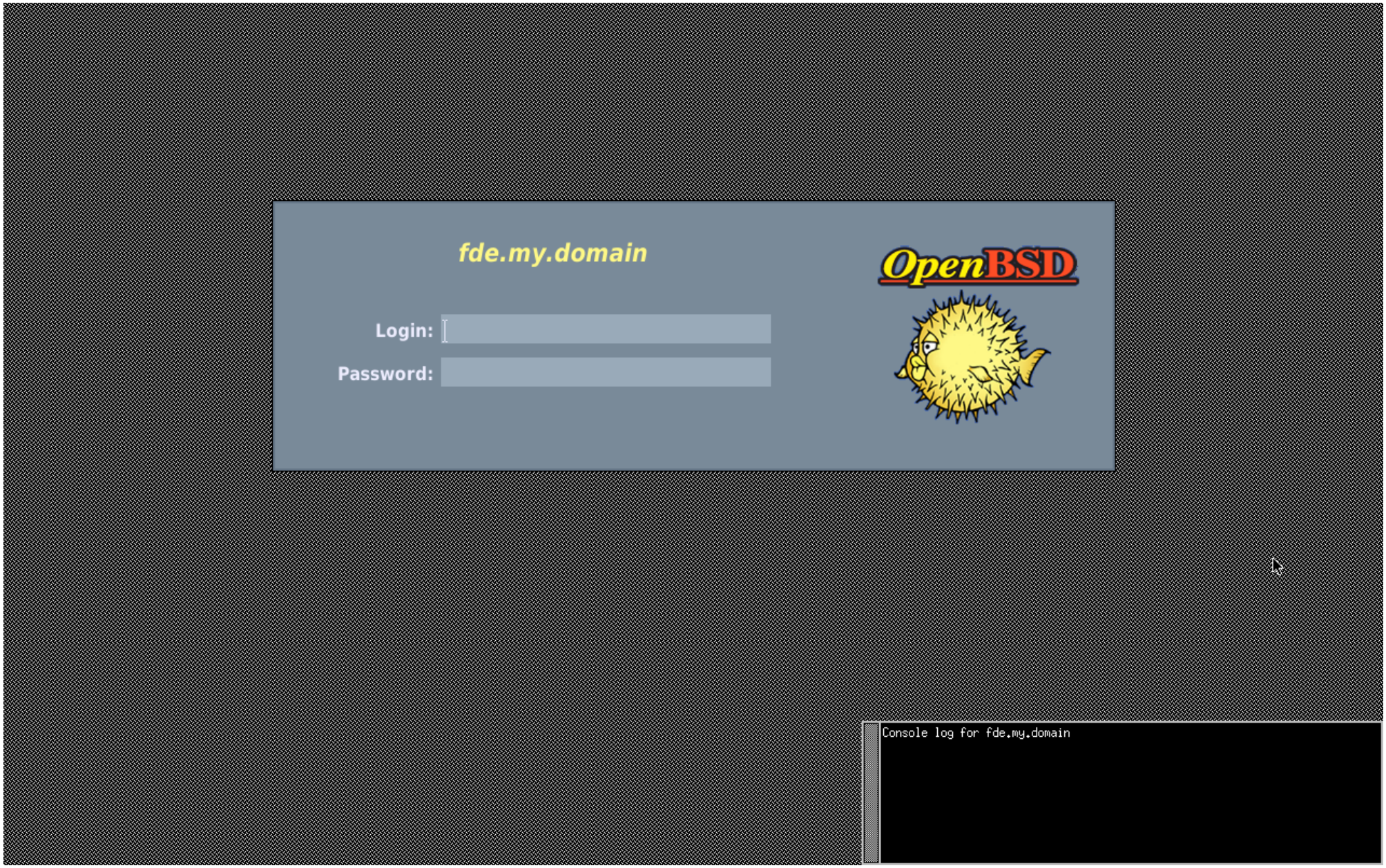
The complete OpenBSD installation is broken up into a number of file sets:

- **bsd** – The kernel (required)
- **bsd.mp –** The multi-processor kernel (only on some platforms)
- **bsd.rd –** The ramdisk kernel
- **baseXX.tgz –** The base system (required)
- **compXX.tgz –** The compiler collection, headers and libraries
- **manXX.tgz –** Manual pages
- **gameXX.tgz –** Text-based games
- **xbaseXX.tgz –** Base libraries and utilities for X11 (requires xshareXX.tgz)
- **xfontXX.tgz –** Fonts used by X11
- **xservXX.tgz –** X11's X servers
- **xshareXX.tgz –** X11's man pages, locale settings and includes

Setup inicial

Instalación de OpenBSD con FDE (continuación)

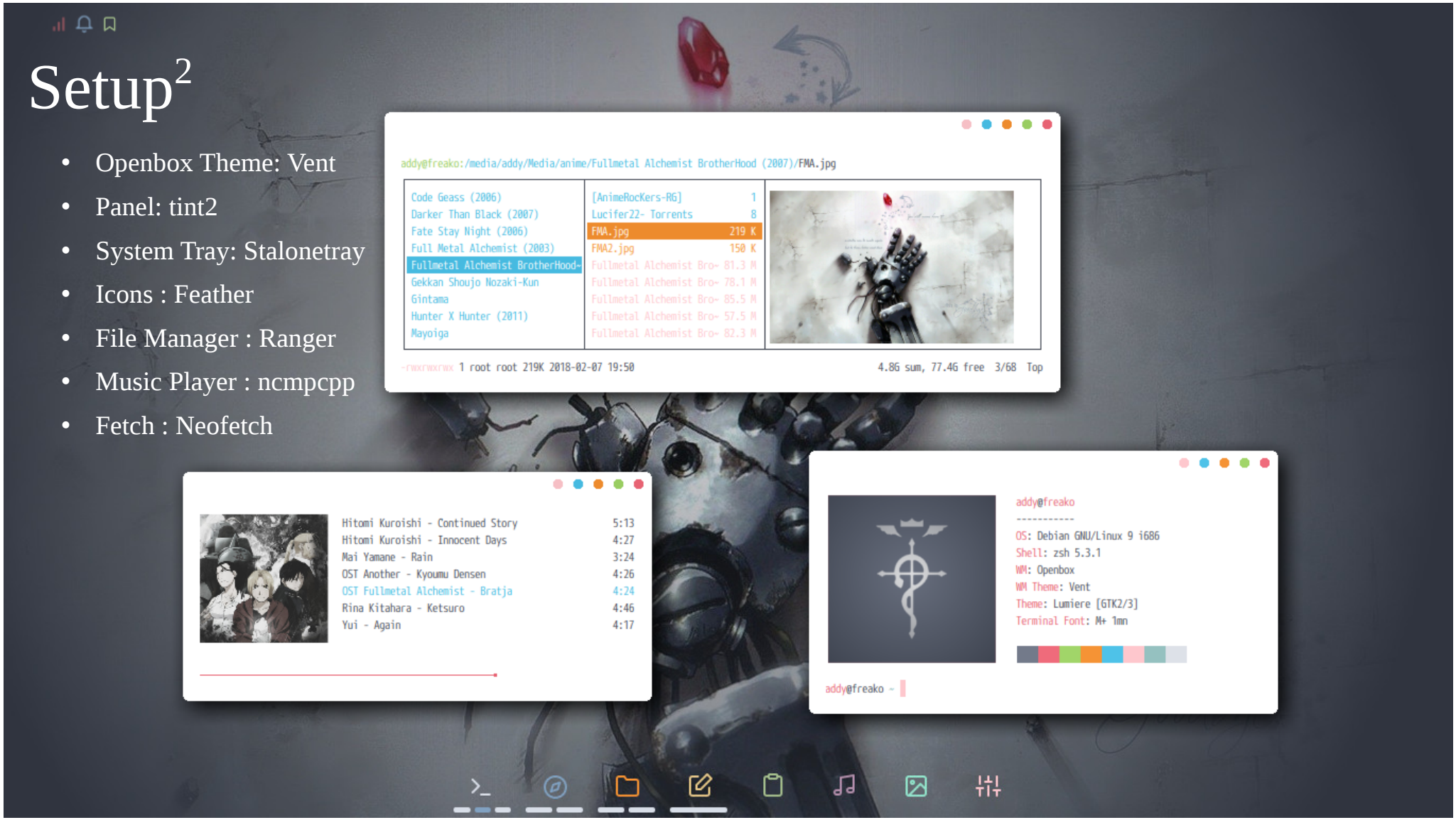
```
Directory does not contain SHA256.sig. Continue without verification? [no] yes
Installing bsd                    80% |****************************        | 15163 KB
...
Location of sets? (disk http or 'done') [done] ← [enter]
What timezone are you in? ('?' for list) [...] America/Argentina/Buenos_Aires
...
Multiprocessor machine; using bsd.mp instead of bsd.
Relinking to create unique kernel... done.

CONGRATULATIONS!
--- desactiven la red en la VM para simular sin WiFi firmware ---
[reboot]
```


fde.my.domain
OpenBSD
Login:
Password:
Console log for fde.my.domain


xterm
fde#
xconsole
OpenBSD 6.5

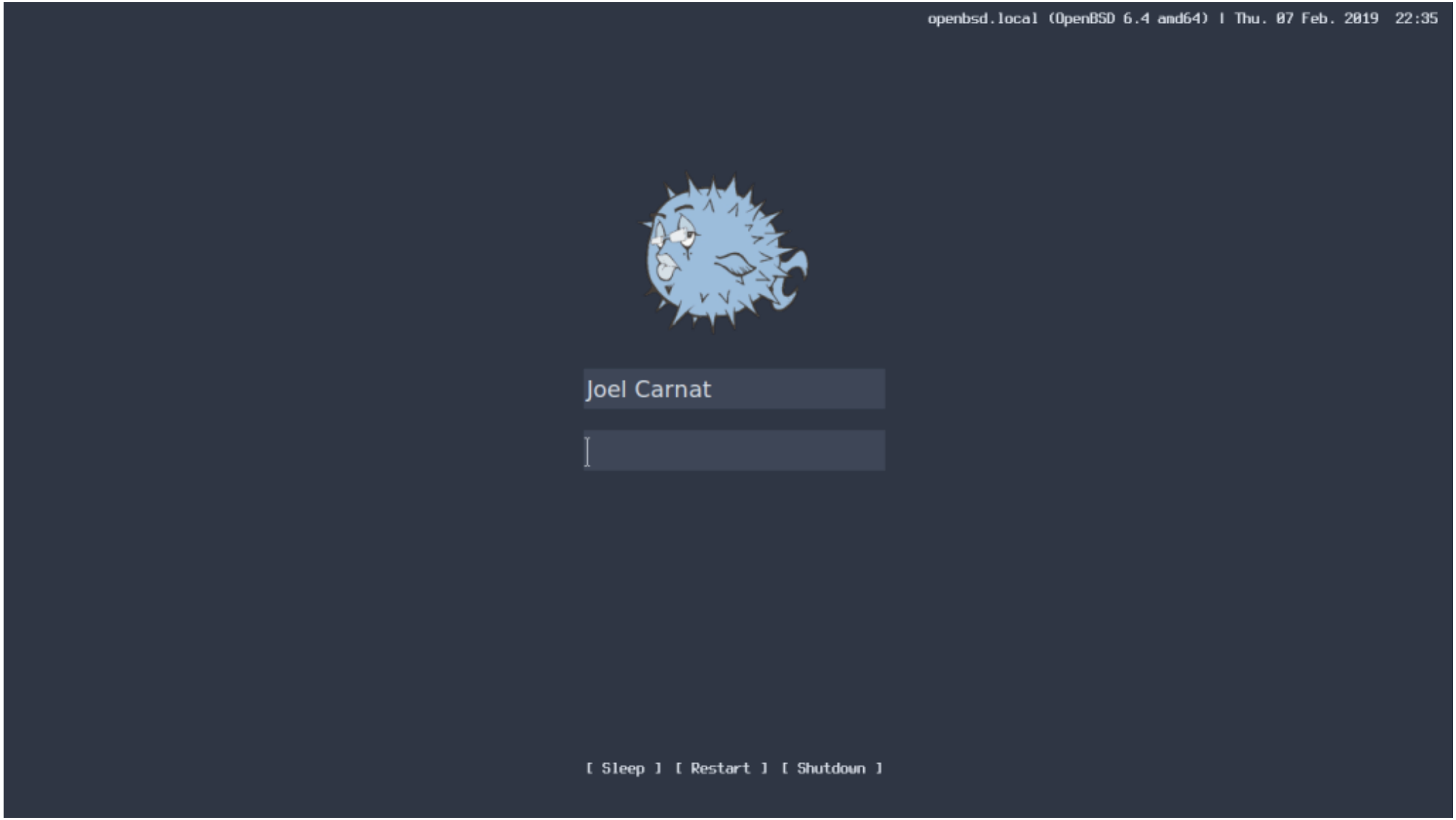
Setup[2]

- Openbox Theme: Vent
- Panel: tint2
- System Tray: Stalonetray
- Icons : Feather
- File Manager : Ranger
- Music Player : ncmpcpp
- Fetch : Neofetch



2 https://forum.archlabslinux.com/t/march-2018-screenshot-thread/318/20

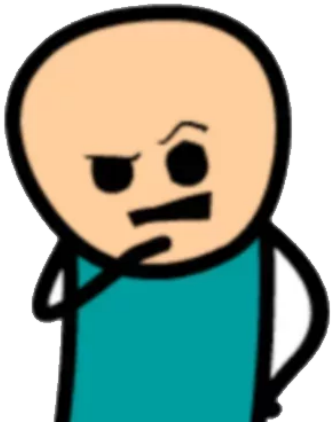
Login screen (xenodm) customizable[3] también!



3 https://www.tumfatig.net/20190208/customizing-openbsd-xenodm/, https://www.romanzolotarev.com/openbsd/xenodm.html

Setup inicial

WTF is disklabel? (continuación)

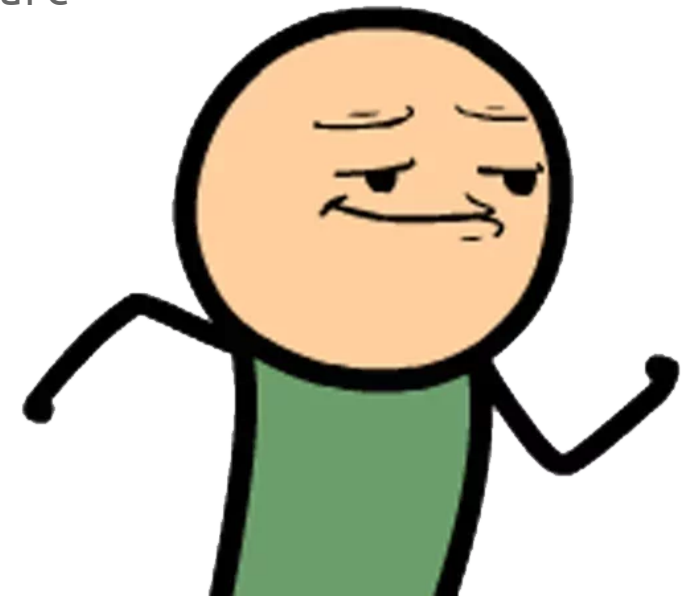


```
# dmesg | grep "^sd"
# disklabel -pm sd1
#                size           offset  fstype [fsize bsize   cpg]
  a:            449.5M             1024  4.2BSD   2048 16384 16142
  c:          14784.0M                0  unused
  i:            0.5M                64   MSDOS
# disklabel -e sd1
 > :q [enter]   # salimos del editor sin cambiar nada
# disklabel -pm sd1
#                size           offset  fstype [fsize bsize
  a:            449.5M             1024  4.2BSD   2048 16384
  c:          14784.0M                0  unused
  i:            0.5M                64   MSDOS
  j:            500.0M           921600   MSDOS
```

Setup inicial

Instalación de OpenBSD con FDE (continuación)

```
# mount /dev/sd1j /mnt
# ls -la /mnt
 total 52
 drwxr-xr-x   1 root  wheel  16384 Dec 31  1979 .
 drwxr-xr-x  13 root  wheel    512 Sep 26 22:43 ..
 drwxr-xr-x   1 root  wheel   8192 Sep 26 22:26 firmware
# fw_update -p /mnt/firmware/6.5/
 intel-firmware-20190514p0v0: ok
```



Preguntas?

Break 10 min!