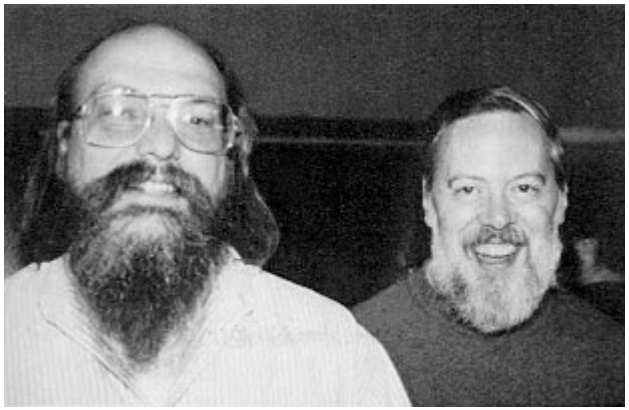
OpenBSD: seguridad industrial

Ciberdefensa para entornos críticos

by Anatoli
BSDar @ CSA LATAM FORUM
Buenos Aires, 2019

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)

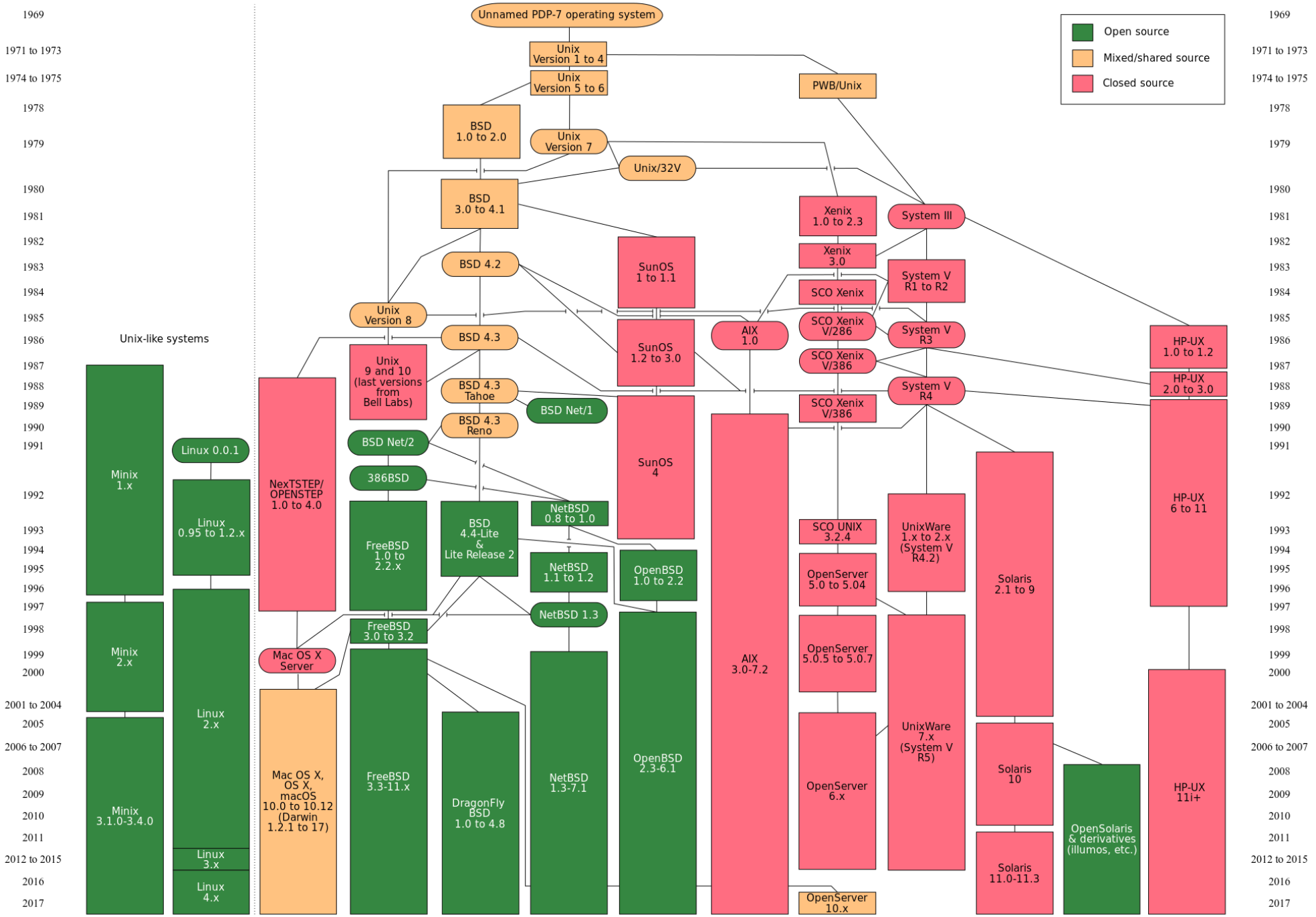
Licencia BSD (copycenter)

- 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
SCO Xenix V/386
HP-UX 1.0 to 1.2
Unix 9 and 10 (last versions from Bell Labs)
BSD 4.3 Tahoe
BSD Net/1
HP-UX 2.0 to 3.0
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
Solaris 2.1 to 9
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
Minix 2.x
NetBSD 1.3
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)
Solaris 10
HP-UX 11i+
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
OpenSolaris & derivatives (illumos, etc.)
Minix 3.1.0-3.4.0
Linux 3.x
Linux 4.x
Solaris 11.0-11.3
OpenServer 10.x

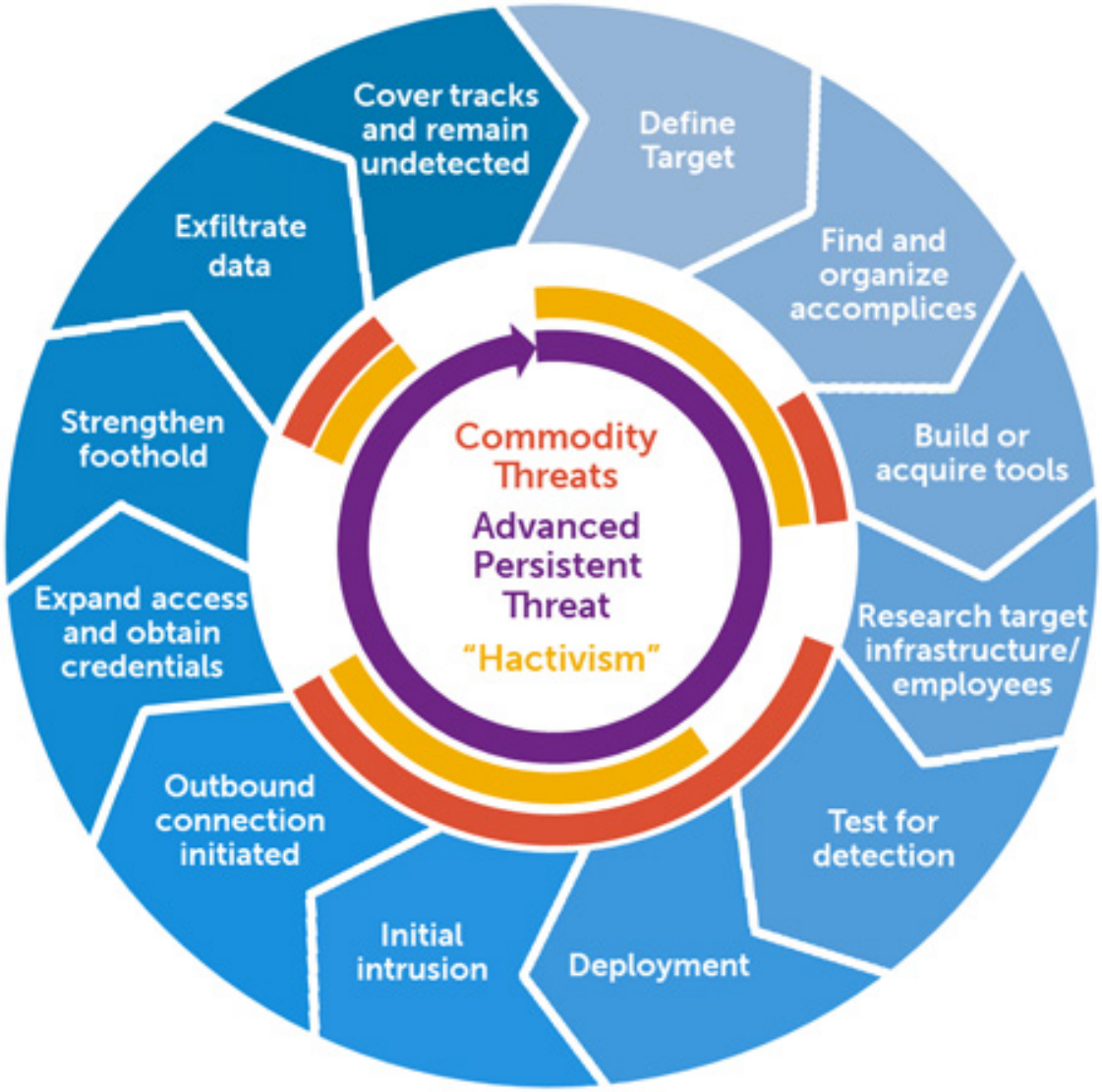
BSDar – Comunidad *BSD Argentina

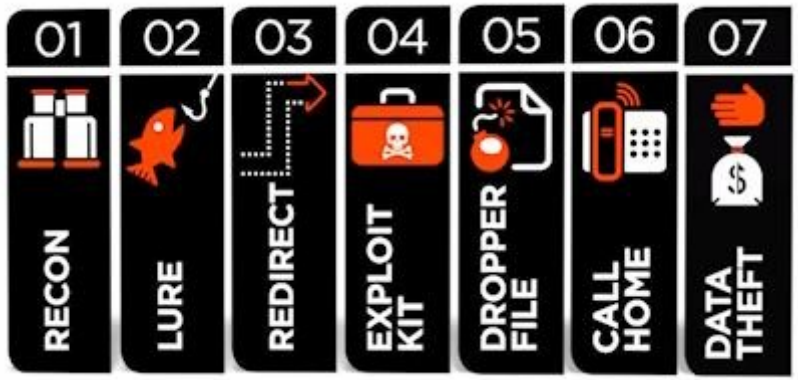
- Comunidad abierta, sin fines de lucro, todos los miembros son iguales, coordinación transparente, manda el voto de la mayoría
- Hackatons el primer domingo de cada mes
- Fundada en mayo 2019
- Meetups el tercer jueves de cada mes
- Principal grupo en Telegram: https://t.me/BSDar (presentarse!)
- Repo: https://github.com/BSDar

Fuentes de amenaza

1. Civiles amateur (hackers y "script kiddies", hacktivists)
   - individuos aislados o pequeños grupos con estructura horizontal
   - nivel bajo – intermedio del know-how técnico
   - muy bajos recursos
   - escaso tiempo: días – semanas
   - motivación débil: sensación de superioridad, causa social, ganancia económica baja
   - blancos: al azar, sistemas mal configurados o sin actualizaciones
2. Operadores profesionales (APT)
   - grupos organizados: espionaje industrial, crimen organizado, grupos clandestinos estatales
   - alto nivel del know-how técnico, especialización
   - suficientes recursos (se maneja como un negocio: planificación, inversión, ganancia, riesgos, plazos, RRHH, gestión, etc.)
   - tiempo extendido: meses – años
   - motivación fuerte: económica, política, ventaja competitiva, daño reputacional
   - blancos: sistemas vulnerables en masa o ataques dirigidos a blancos específicos

Advanced Persistent Threat (APT)





Fuentes de amenaza (cont.)

3. Estados
   - servicios de inteligencia y subversión, ejércitos
   - muy alto nivel del know-how técnico
   - recursos ilimitados
   - tiempo muy extendido: años
   - motivación fuerte: inteligencia, influencia/coerción política, ataques económicos, ciber-guerra
   - blancos nacionales: periodistas, oposición, investigadores, levantamientos sociales (cualquier amenaza doméstica al poder, real o potencial)
   - blancos internacionales: organismos estatales, movimientos políticos, sector privado (grandes empresas), periodistas, etc. de los estados vecinos o los estados que compiten en los mismos rubros (intereses geopolíticos)

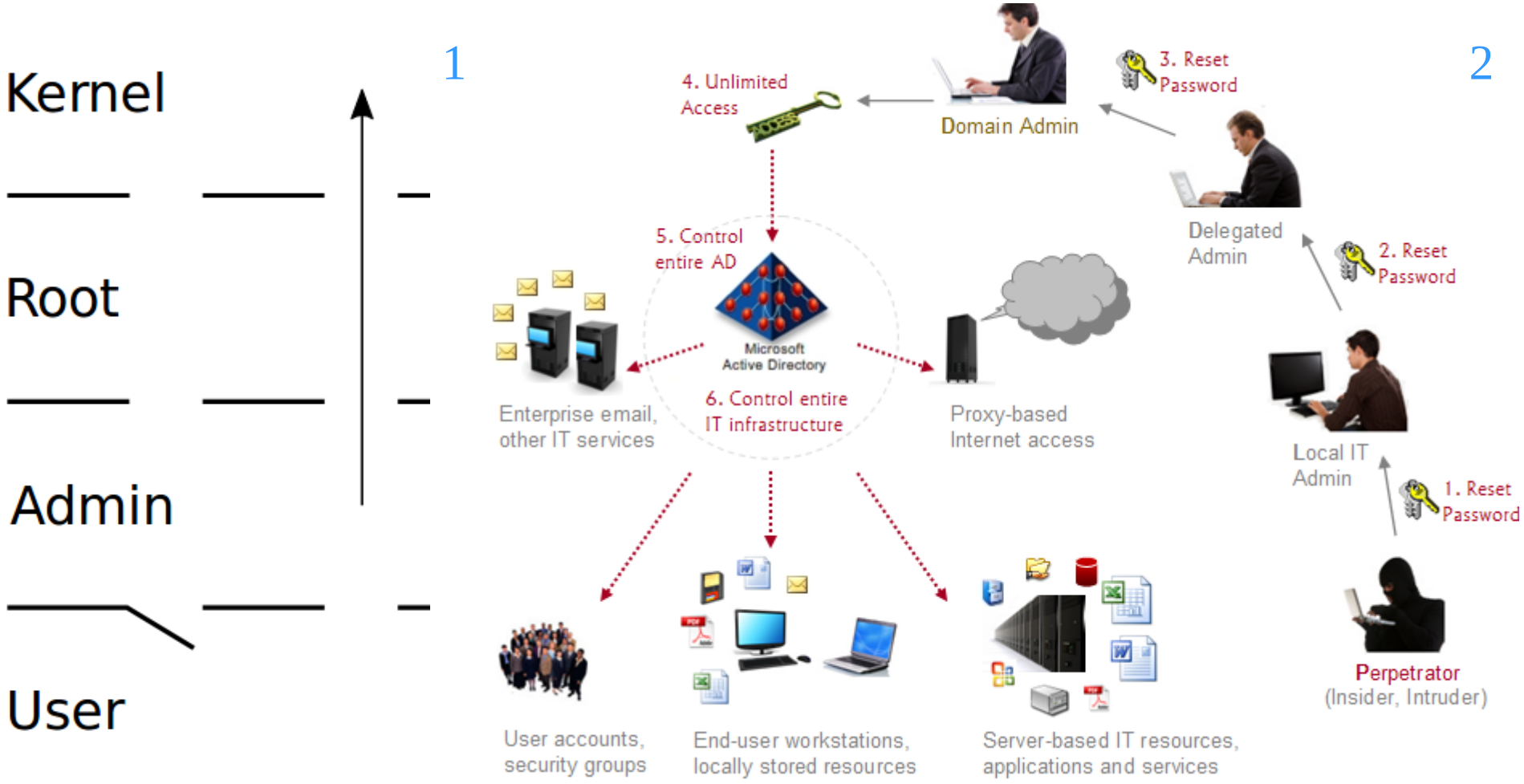
Ciber-guerra vs guerra convencional

- No-atribución, negable
- Muy bajo costo
- Muy alto impacto
- Todos están vulnerables no hay superpotencias con armamento nuclear cibernético, ni mecanismos de detección temprana del lanzamiento
- Al alcance de todos los estados
- Mecanismo de chantaje, ventaja económica, intromisión...
- No se puede tomar la postura de no-conflictividad
- Está preparada la Argentina?
- Cómo podemos defendernos de las **APTs** y los **estados**?

Métodos de intrusión y persistencia

- (Spear) Phishing: suplantación de identidad (dirigida)
- RCE: remote code execution vía un bug en Firefox, TB, AV
- PrivEsc: privilege escalation vía un bug en el kernel o servicio privilegiado (no es detectable por EDR, IPS, AV)
- Persistencia: infección del firmware de los dispositivos (discos rígidos, placas de red, BIOS, etc.) – imposible de eliminar
- Propagación horizontal automática mediante la explotación de las herramientas de gestión/autenticación centralizada o manualmente mediante ingeniería social o vulnerabilidades

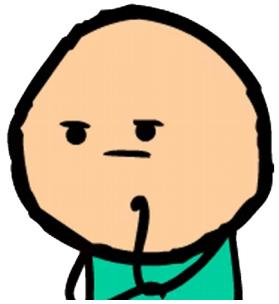
Privilege escalation



1 https://en.wikipedia.org/wiki/Privilege_escalation
2 https://www.active-directory-security.com/2013/10/5-Facts-You-Must-Know-About-Active-Directory-Privilege-Escalation.html

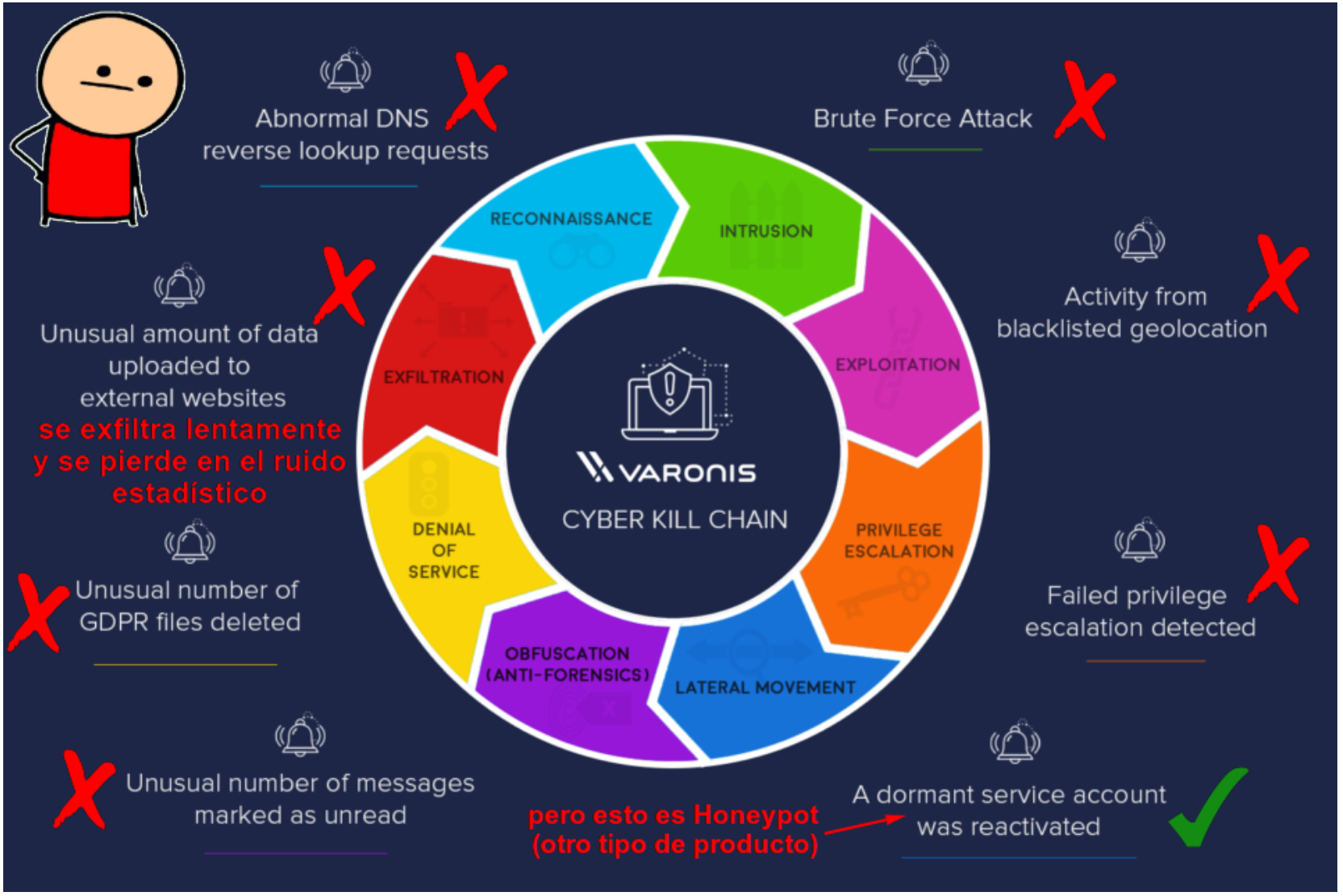
Defensa: next-gen XDR endpoint protection

- Foco: **lucro, espionaje**
- "Next-gen" Firewall: es el mismo "next-gen" de hace 10 años?
- Antivirus: muy baja eficacia[3], alto riesgo en sí[4]
- EDR/IDS/IPS: lindas imágenes, muy baja eficacia[5]
- Threat intelligence: sólo sirve para ataques masivos
- Leyes FVEY[6]: Patriot Act[7], National Security Letters, gag orders
- **Resultado**: despliegue voluntario del spyware muy costoso



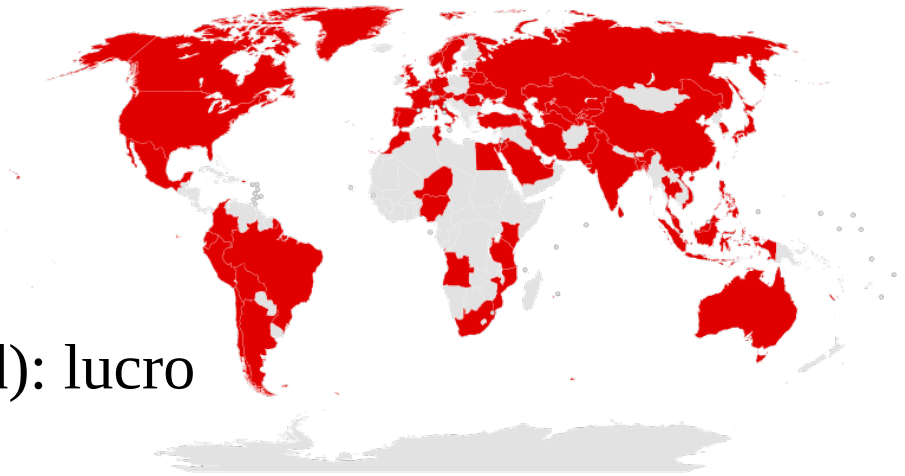
3 https://www.theguardian.com/technology/2014/may/06/antivirus-software-fails-catch-attacks-security-expert-symantec
4 https://arstechnica.com/information-technology/2017/01/antivirus-is-bad/
5 https://www.exploit-db.com/papers/41914
6 https://en.wikipedia.org/wiki/Five_Eyes
7 https://www.aclu.org/issues/national-security/privacy-and-surveillance/surveillance-under-patriot-act

Defensa: Endpoint Detection and Response (EDR)[8]



8 https://www.varonis.com/blog/cyber-kill-chain/

Defensa: **SO propietarios**



- Foco: **lucro, recolección de datos**
- Windows: lucro, recolección de datos
- Android-based: recolección de datos
- OS X, AIX, HP-UX, Solaris (Unix-based): lucro
- Imposibilidad de analizar el código
- Imposibilidad de arreglar el código
- Dependencia, backdoors
- Leyes FVEY: Patriot Act, National Security Letters, gag orders
- **Resultado**: ciberseguridad ~= hacer que la infraestructura ande
- Palabras de terror: CryptoLocker, CryptoWall, TorrentLocker, Locky, Petya, WannaCry[9], NotPetya[10]

9 https://en.wikipedia.org/wiki/WannaCry_ransomware_attack
10 https://techtalk.gfi.com/the-10-worst-ransomware-attacks-that-ever-happened/

Defensa: **Linux**

- Foco del proyecto: **usuarios, funcionalidad, performance**
- Torvalds: *“the number one rule of kernel development is that we don't break users“*[11]
- Torvalds: *“As long as you see your hardening efforts as a 'let me kill the machine/process on bad behavior', I will stop taking those shit patches.” “...security people are f*cking morons“*[12]
- Resultado: **44 vulnerabilidades** con puntaje >= 7.5 **(10 con puntaje >=** 9.0 **)** en el Linux *kernel* **desde el comienzo del año**[13]
- Pobre prevención de explotación en user-mode
- Ventanas de vulnerabilidades para 10.0 son de semanas a años[14]

11 https://lkml.org/lkml/2017/11/21/356
12 http://lkml.iu.edu/hypermail/linux/kernel/1711.2/01701.html
13 https://www.cvedetails.com/vulnerability-list.php?vendor_id=33&product_id=47&cvssscoremin=7.5&year=2019&order=3, en 4 años: 279-104 vulnerabilidades, promedio 70-26
14 https://github.com/torvalds/linux/commit/7fafcfdf6377b18b2a726ea554d6e593ba44349f vs https://www.cvedetails.com/cve/CVE-2018-20961/

Defensa: **FreeBSD**

- Foco del proyecto: **performance, estabilidad, compatibilidad**
- La seguridad es considerada, pero...
- Muchas cosas legacy, compatibilidad, componentes complejos
- Opciones por defecto inadecuadas[15]
- Problemas administrativos[16], liderazgo basado en el consenso, equipo de seguridad sobrecargado; aún así hacen su trabajo[17]
- Resultado: **8 vulnerabilidades** con puntaje >= 7.5 **(1 con puntaje >=** 9.0 **)** en FreeBSD **desde el comienzo del año**[18]
- Pobre prevención de explotación en user-mode

15 https://vez.mrsk.me/freebsd-defaults.html
16 https://www.youtube.com/watch?v=Ib7tFvw34DM
17 https://www.slideshare.net/apnic/improving-the-freebsd-security-advisory-process
18 https://www.cvedetails.com/vulnerability-list.php?vendor_id=6&product_id=7&cvssscoremin=7.5&year=2019&order=3

Defensa: **OpenBSD**

- Foco del proyecto: **seguridad, implementación correcta**
- Buen diseño (5 años para pledge & unveil), riguroso proceso de desarrollo, minimalista, mejora continua, exclusión de lo inseguro
- No se molestan con los ~~caprichos~~ necesidades de los *consumers*
- SO entero: OpenBSD base = kernel + user-mode + services
- Hardened ports (nginx with chroot, firefox with pledge & unveil, etc.)
- Extremadamente buena prevención de explotación en user-mode
- Resultado: **0 vulnerabilidades** con puntaje >= 7.5 en OpenBSD **base desde el comienzo del año**[19]

19 https://www.cvedetails.com/vulnerability-list.php?vendor_id=97&product_id=163&cvssscoremin=7.5&order=1 (ver las versiones de OpenBSD afectadas, la versión de hace 10 años fue 4.5)

Defensa: **OpenBSD en 10 años**

OpenBSD base: vulnerabilidades en los últimos 10 años

0 vulnerabilidades críticas, sólo 4 vulnerabilidades serias[20]:

- 7.8 CVE-2009-0687 (2009-04-09) – DoS en ipv6 pf code
- 7.8 CVE-2016-6244 (2016-07-17) – DoS en mmap
- 7.8 CVE-2017-5850 (2017-02-07) – DoS en httpd
- 7.5 CVE-2017-1000372 (2017-06-19) – local bypass de un mecanismo de prevención (stack guard page)[21]

20 https://media.ccc.de/v/34c3-8968-are_all_bsds_created_equally
21 https://www.qualys.com/2017/06/19/stack-clash/stack-clash.txt

OpenBSD

Qué es?

- Un SO *entero* (base + ports con mejoras), creado en 1995 por Theo de Raadt como fork de NetBSD
- Foco: seguro, correcto, usable, minimalista y libre
- Acceso público al repositorio CVS & commit logs desde el 1995
- 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

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.

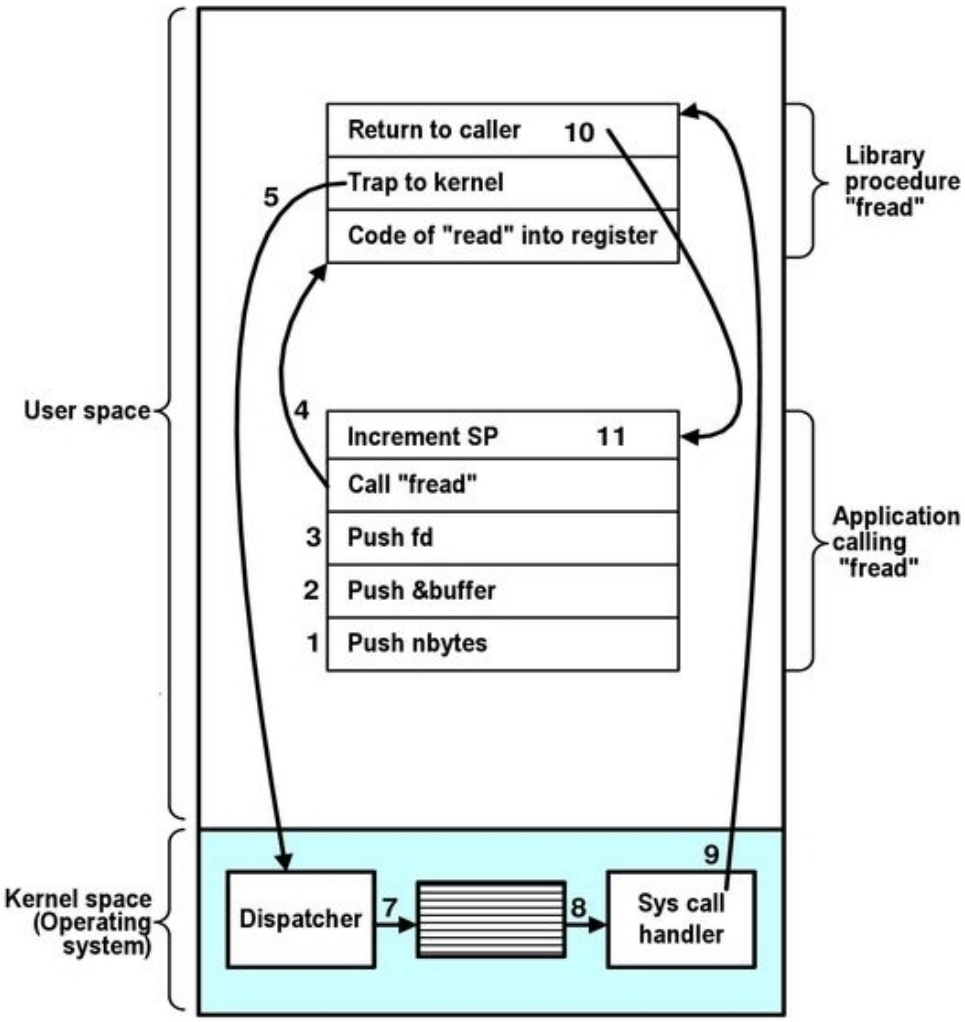
R “realloc”. Always reallocate when realloc() is called, even if the initial allocation was big enough.

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.

Qué es un syscall?

- **The application program makes a System Call:**
  - A system library routine is called first
  - It transforms the call to the system standard (*native API*) and traps to the kernel
  - Control is taken by the kernel running in the system mode
  - According to the service “code”, the *Call dispatcher* invokes the responsible part of the Kernel
  - Depending on the nature of the required service, the kernel may block the calling process
  - After the call is finished, the calling process execution resumes obtaining the result (success/failure) as if an ordinary function was called

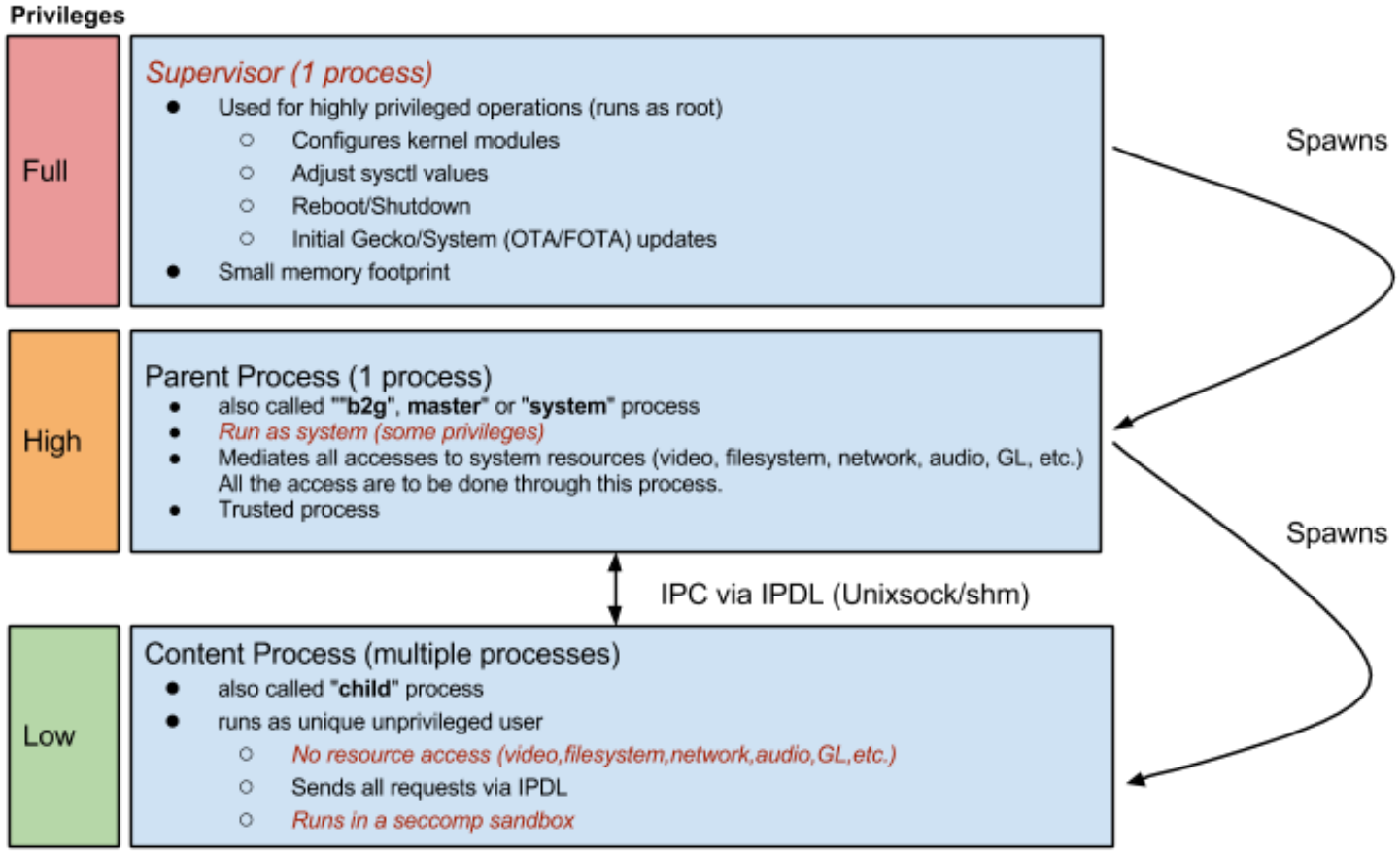


11 steps to execute the service
fread (fd, buffer, nbytes)

seccomp

A one-way transition into a "secure" state where it cannot make any syscalls except exit(), sigreturn(), read() and write() to already-open file descriptors[22]

Firefox OS[23]



22 https://terenceli.github.io/%E6%8A%80%E6%9C%AF/2019/02/04/seccomp
23 https://developer.mozilla.org/en-US/docs/Archive/B2G_OS/Security/System_security

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.

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 devs 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 behavior changes

Returns no error; in case of illegal operations kernel just kills the program (SIGABRT)

chroot

Change the apparent root directory for the current running process and its children

Orden de ejecución para *chroot tardío* (lo más eficiente)

- `init`
  - `cargar librerías` (linkear con `-z now` para deshabilitar lazy load)
  - `leer y procesar los configs`
  - `cargar plug-ins`
- `resolver user/group a uid/gid` – necesita `/etc/passwd` & `/etc/group` + librerías
- `initgroups` inicializar grupos suplementarios
- `chroot(/srv/nginx)` – sólo root (o CAP_SYS_CHROOT) puede hacerlo
- `setuid(uid), setgid(uid)` – renunciar a los privilegios de root

Es inseguro durante `init` y no siempre es posible (ej. navegadores)

OpenBSD: defensa en profundidad (cont.)

unveil – un chroot refinado[24]

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")
```

24 Es una simplificación, no son exactamente los mismos mecanismos.

OpenBSD: debilidades

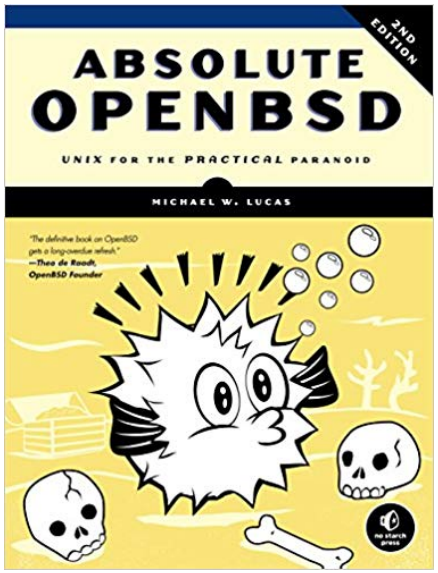
- Performance
  - El kernel no es SMP: pf, interrupts y kernel en general son single core

    - Martin Pieuchot (mpi@) está trabajando en eso[25], ayúdenle![26]
  - FFS (UFS), el único FS soportado para base, fue creado pre-OpenBSD

    - Una vez el kernel es SMP, se podría incorporar HAMMER[27] de DragonFly
- Soporte incompleto del hardware/features
  - 802.11ac (WiFi 5GHz)
  - Algunas GPUs como nvidia

25 http://www.grenadille.net/post/2019/05/09/Reducing-that-contention
26 http://www.grenadille.net/pages/Hire-me
27 https://www.dragonflybsd.org/hammer/

OpenBSD: auto-ayuda



- http://openbsdjumpstart.org - quickstart intro
- 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)

Preguntas?

Gracias!

t.me/BSDar

GitHub.com/BSDar

BSDar

Comunidad *BSD Argentina