Kurs administrowania systemem Linux

Zajęcia nr 12: Wirtualizacja

Instytut Informatyki Uniwersytetu Wrocławskiego

22 maja 2025

Wirtualizacja = Udawanie

Co udawać?

- procesor i/lub inne rodzaje hardware'u,
- system operacyjny.

Jak udawać?

- *Symulacja*: szczegółowe odtwarzanie *działania* systemu (np. procesora na poziomie mikrokodu, *cycle-accurate*). Ważna jest wierność.
- *Emulacja*: szczegółowe odtworzenie *efektów* wykonania programu. Ważna jest efektywność i wierność efektów.
- *Wirtualizacja*: zwielokrotnienie platformy — udawanie wyłączności.

Symulacja

Procesory

- Oprogramowanie EDA (*Electronic design automation*): Icarus Verilog, Verilator, QUCS

Inny hardware

- Sieci: GNS3

Systemy operacyjne

- Cisco Packet Tracer (Cisco IOS)

Emulacja

Procesory

- Qemu (Quick Emulator; uwaga: w połączeniu z KVM także hypervisor!)
- Procesory MIPS: SPIM, MARS, Dynamips
- x86 real mode: DOSBox, DOSEmu
- Mikrokontrolery

Inny hardware

- terminal emulator
- printer emulator (ghostscript)

Systemy operacyjne

- Wine
- Cygwin, Windows Subsystem for Linux v. 1
- Ale nie: WSL v. 2 lub BSD Linux compatibility layer

Emulacja hardware'owa

- FPGA

Emulacja a wirtualizacja

Qemu jako emulator

- Tiny Code Generator (TCG)
- Przekłada *basic blocks* z jednej architektury na drugą (pośrednio poprzez TCG *ops*).

KVM — wirtualizator

- Usługa jądra OS-a (Linux, FreeBSD, illumos/OpenIndiana) na wielu architekturach (x86, ARM, PowerPC, S/390, IA-64).
- Jądro jest *hypervisorem*.
- Pozwala uruchomić w trybie użytkownika program oczekujący wykonania w trybie jądra, napisany na tę samą architekturę.
- Natywnie wykonują się większe fragmenty programu, niż tylko *basic blocks*.
- Wsparcie sprzętowe umożliwia łatwe „zapanowanie" nad uruchomionym programem.
- Ktoś musi dostarczyć emulację peryferiów.

Qemu/KVM

- Qemu emuluje peryferia (karta graficzna, dyski itd.).
- KVM wirtualizuje procesor.

Wirtualizacja hardware'u

Hypervisor

Hypervisor (*Virtual Machine Monitor*): oprogramowanie, które pozwala tworzyć i uruchamiać *maszyny wirtualne*. (OS nazywano kiedyś *supervisorem* lub *monitorem*.)

Rodzaje (Popek, Goldberg, 1974)

- Typu 1 (*bare metal*, *native*): działa wprost na fizycznej maszynie. Jest bardzo uproszczonym systemem operacyjnym, zwykle o architekturze mikrojądra (Xen, VMWare ESXi, Microsoft Hyper-V).
- Typu 2 (*hosted*): jest programem uruchamianym pod kontrolą systemu operacyjnego, tzw. *hosta* (Qemu, Oracle Virtualbox, VMware Workstation).
- Hybrydowy: Kompletny system operacyjny, który ma funkcjonalność hypervisora (KVM, FreeBSD bhyve, UML).

Pierwszy hypervisor: CP/CMS na IBM/360-67 (1965)

Gerald J. Popek, Robert P. Goldberg, *Formal Requirements for Virtualizable Third Generation Architectures*, CACM **17**(7):412–421 Jul. 1974.

For any conventional third generation computer, a virtual machine monitor may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.

Instrukcje

- *wrażliwe* — zmieniają konfiguarację zasobów systemu,
- *uprzywilejowane* — wywołane w trybie użytkownika generują wyjątek procesora.

Pierwszy hypervisor: CP/CMS na IBM/360-67 (1965)

Gerald J. Popek, Robert P. Goldberg, *Formal Requirements for Virtualizable Third Generation Architectures*, CACM **17**(7):412–421 Jul. 1974.

For any conventional third generation computer, a virtual machine monitor may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.

Instrukcje

- *wrażliwe* — zmieniają konfiguarację zasobów systemu,
- *uprzywilejowane* — wywołane w trybie użytkownika generują wyjątek procesora.

Generacje komputerów

1. lampowe 1942 (John A. Fleming, Lee De Forest, 1906),
2. tranzystorowe 1953 (William B. Shockley Jr. *et al.* 1947),
3. układy scalone 1958 (Jack Kilby, Texas Instruments, 1958),
4. mikroprocesorowe 1971 (Intel 4004, 1971),
5. 1982 (Japonia, silnie zrównoleglone, porażka); dalej więc zaprzestano liczenia.

Wirtualizowalność

Popek and Goldberg virtualization

- System/370 (1970) — tak.
- MC68000 (1979) — pojedyncza zła instrukcja MOVE from SR, poprawione w MC68010.
- IA-32 / x86 (1985) — nie, 18 złych instrukcji.
- SPARC, PowerPC — tak.

Hardware-assisted virtualization

- Intel virtualization (VT-x), 2005
- AMD virtualization (AMD-V), 2006
- ARM Hyp mode (ARMv7 Virtualization Extensions, ARMv8 EL2)

Wirtualizacja hardware'u

Architektura maszyny fizycznej i oprogramowania

- Instrukcje uprzywilejowane: tryb nadzorcy, jądro systemu operacyjnego
- Instrukcje przestrzeni użytkownika

Praca jądra na maszynie wirtualnej

- Wirtualizacja częściowa
- Wirtualizacja pełna (*trap and emulate virtualisation*, nieefektywne, nie zawsze możliwe)
- Parawirtualizacja (wsparcie ze strony gościa)
- Wsparcie sprzętowe (*hardware assisted*) dla wirtualizacji procesora: VTx, AMD-V

Wsparcie sprzętowe dla wirtualizacji innych urządzeń

- IOMMU: VTd, AMD-Vi
- PCI express: SR-IOV
- Karty graficzne: GVT-d, GVT-g and GVT-s

Najpopularniejsze hypervisory

Stacje robocze (przeważnie typu 2)

- Qemu/KVM (+ aqemu itp.)
- Sun / Oracle Virtualbox (fork Qemu)
- VMware Workstation Player, VMware Workstation Pro

Serwery (przeważnie typu 1)

- Xen — dawniej popularny w usługach hostingowych
- KVM (+ libvirt, oVirt, virt-manager itp.) — używają AWS, GCP, Azure
- VMware Server
- Microsoft Hyper-V

Zarządzanie

- libvirt
- Vagrant

Mój ulubiony eksploit

CVE-2015-4495

The PDF reader in Mozilla Firefox [...] allows remote attackers to bypass the Same Origin Policy, and read arbitrary files or gain privileges, via vectors involving crafted JavaScript code and a native setter, as exploited in the wild in August 2015.

Daniel Veditz, Mozilla Security Blog, August 6, 2015

Yesterday morning, August 5, a Firefox user informed us that an advertisement on a news site in Russia was serving a Firefox exploit that searched for sensitive files and uploaded them to a server that appears to be in Ukraine. [...] The vulnerability comes from the interaction of the mechanism that enforces JavaScript context separation (the "same origin policy") and Firefox's PDF Viewer. [...] The files it was looking for were surprisingly developer focused for an exploit launched on a general audience news site. [...] On Linux the exploit goes after the usual global configuration files like `/etc/passwd`, and then in all the user directories it can access it looks for `.bash_history`, `.mysql_history`, `.pgsql_history`, `.ssh` configuration files and keys, [...] text files with "pass" and "access" in the names, and any shell scripts.

Separation anxiety

Wieloprocesowy system operacyjny

- Podział czasu — wirtualizacja procesora.
- Ochrona pamięci — procesy mają dostęp do rozłącznych fragmentów pamięci fizycznej:
  - segmentacja,
  - adresy wirtualne.
- Wirtualizacja pozostałego hardware'u: obsługa magistral, urządzeń itd. tylko poprzez jądro.

Co poszło źle?

- Zbyt słaba separacja procesów.

Glauber Costa, LinuxCon Europe 2012

I once heard that hypervisors are the living proof of operating system's incompetence.

- Trzeba ulepszyć separację procesów.

chroot(2) i chroot(8)

Separacja systemów plików

- Każdy proces ma *Root Directory* i *Current Working Directory*.
- Syscall `chroot(2)` pojawił się w wersji AT&T Version 7 w 1979 roku.
- Był w SUSv1, w SUSv2 oznaczony jako *legacy*, usunięty w późniejszych wersjach.
- Polecenie `chroot(8)` pojawiło się w AT&T System III i w 4.3BSD-Reno.
- To samo przeznaczenie, co obecnie: wykonanie procesu w kontekście innego rootfs.
- Polecenie `chroot(8)` wykonuje `chdir(2)`, syscall `chroot(2)` — nie, trzeba wykonać samemu.

Zastosowania *off-label* `chroot`-a

Honeypots

- Idea: książka Clifforda Stolla *The Cuckoo's Egg* (1989).
- Bill Cheswick, Bell Labs.
- An Evening with Berferd: In Which a Cracker is Lured, Endured, and Studied (1991).
- Tworzy *honeypot* używając `chroot(8)`.

Sandboksy

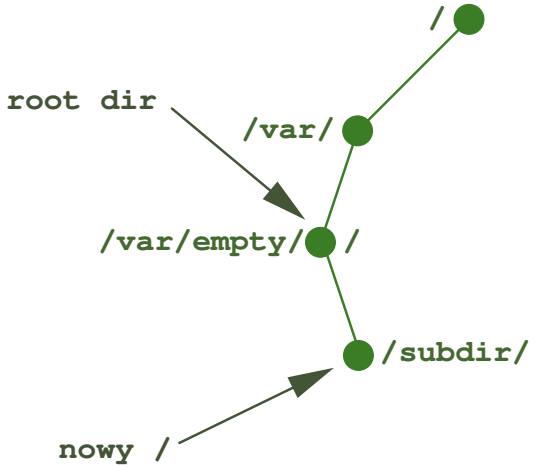
- Serwery ftp i sendmail, potem www.
- Liczne dziury, niedostateczna separacja.

Chroot escape

- Pierwsza idea: Carole Fennely, Sun World Online 1 i 8/1999.
- Potem liczne wariacje.

Classic chroot escape

```
// sandboksowanie:
chroot("/var/empty");
chdir("/var/empty");
...
// chroot escape:
mkdir("/subdir");
chroot("/subdir");
chdir("../..");
```



Balázs Bucsay: Chw00t (2015)

	Debian 7.8;2.6.32/Kali 3.12	Ubuntu 14.04.1;3.13.0-32-generic	DragonFlyBSD 4.0.5 x86_64	FreeBSD 10.-RELEASE amd64	NetBSD 6.1.4 amd64	OpenBSD 5.5 amd64	Solaris 5.11 11.1 i386	Mac OS X
Classic	YES	YES	DoS	NO	NO	NO	YES	YES
Classic FD	YES	YES	NO	NO	NO	NO	YES	YES
Unix Domain Sockets	YES	YES	DoS	PARTIALLY	NO	PARTIALLY?	YES	YES
/proc	YES	YES	NO	NO	NO	NO	YES	NO
Mount	YES	YES	NO	NO	NO	NO	NO	NO
move out of chroot	YES	YES	DoS	PARTIALLY	NO	YES	YES	YES
Ptrace	YES	PARTIALLY	NO?	YES	NO	YES	N/A	N/A

Kontenery — bezpieczny (?) chroot na resorach

Unix

- FreeBSD Jails (2000). Syscall `jail(2)` we FreeBSD 4.0.
- Solaris Zones (2005).
- Dają możliwość tworzenia bezpiecznego *sandboksa* dla grupy procesów.
- Oddzielają systemy plików, komunikację międzyprocesową, dostęp do interfejsów sieciowych itd.
- Dają możliwość tworzenia kompletnych przestrzeni użytkownika — kontenerów.

Linux

- Virtuozzo OpenVZ (2005).
- Obecnie oparte na namespace'ach i cgroups (LXC, LXD, Docker, Snap, Flatpak, AppImage).

Linux namespaces — lepsza izolacja procesów

- Wirtualizacja przestrzeni nazw.
- Inspirowane przez (ogólniejsze rozwiązanie w) Plan 9: Rob Pike, Dave Presotto, Ken Thompson, Howard Trickey, Phil Winterbottom, *The use of namespaces in Plan 9*, ACM SIGOPS 1992, także: SIGOPS **27**(2):72–76, Apr. 1993.
- Po uruchomieniu jądra: po jednej przestrzeni nazw każdego rodzaju.
- Każdy proces należy do dokładnie jednej przestrzeni nazw każdego rodzaju.
- Drzewiasta hierarchia przestrzeni nazw.
- Niektóre są zagnieżdżone, niektóre — rozłączne.

Rodzaje *namespace'ów*

- `mnt` — punkty montażowe
- `pid` — numery procesów
- `net` — interfejsy sieciowe, stosy protokołów IPv4 i IPv6, iptables, gniazda
- `ipc` — komunikacja międzyprocesowa: kolejki wiadomości, semafory, pamięć dzielona
- `uts` — *hostname* i *domainname*
- `user` — nazwy grup i użytkowników
- `cgroup` — cgrupy

Control groups (cgroups)

Nice na resorach

- Rozszerzenie idei grup procesów.
- Zaimplementowane przez Paula Menage (Google) w 2007.
  Od 2.6.24 oficjalnie w Linuksie.
- Niskopoziomowe narzędzia jądra. Można wykorzystać do zarządzania pojedynczymi procesami, sandboksowania procesów lub budowy kontenerów.
- Po uruchomieniu jądra: po jednej grupie każdego rodzaju.

Regulowanie dostępu procesu do zasobów

- ograniczanie,
- priorytetyzowanie,
- księgowanie,
- sterowanie.

Rodzaje cgroups

- `cpuset` — przypisanie do wybranych CPU, rdzeni itp.
- `cpu` — ograniczenie zużycia czas procesora
- `cpuacct` — rejestrowanie zużycia czasu procesora
- `memory` — rejestrowanie i ograniczanie zużycia pamięci procesów, pamięci jądra dla procesów, przestrzeni wymiany itp.
- `devices` — ograniczenie użycia `mknod`
- `freezer` — zatrzymanie i późniejsze wznowienie wszystkich procesów grupy jednocześnie
- `net_cls` — class id wykorzystywany przez fw (iptables) (filtrowanie) i `tc(8)` (ograniczanie ruchu sieciowego)
- `blkio` — ograniczanie dostępu do urządzeń blokowych
- `perf_event` — monitorowanie zużycia zasobów
- `net_prio` — priorytety interfejsów sieciowych dla grupy
- `hugetlb` — ograniczanie dostępu do dużych stron
- `pids` — ograniczanie liczby procesów potomnych w grupie

Zarządzanie cgroups i namespaces

Interfejs jądra

- `/sys/fs/cgroup/`
- `/proc/$PID/ns/`
- pliki `cgroup.procs` — numery procesów należących do grup

SystemD-free userland

- Biblioteka `libcgroup`.
- Polecenia `cgcreate`, `cgconfig`, `cgset`, `cgexec`, `unshare`, `prlimit`, `ulimit`, `ip netns` itp.

SystemD userland

- Interfejs zapewnia SystemD. Uwaga: sprzeczne z `libcgroup`! Używać *tylko* SystemD!
- Polecenia: `systemd-run`, `systemd-nspawn`, `systemd-cgls`, `systemd-cgtop`.

Zastosowania cgroups i namespaces

Sandboxing

- Izolacja pojedynczej aplikacji.
- Aplikacja sama sobie to robi: Google Chrome, Firefox.
- Aplikacja uruchamiana przez nadzorcę: Firejail, Bubblewrap, nsjail.
- Główne zastosowanie: „bo to zła aplikacja była".
- Dodatkowe mechanizmy: AppArmor, SELinux, Seccomp BPF.

Kontenery

- Kontenery: maszyny wirtualne — eunuchy (tylko userspace, bez jądra).
- Implementacje: LXC, LXD (Canonical).
- Docker (+ infrastruktura: Kubernetes).
- Snap, AppImage, Flatpack — superpakiety.
- Główne zastosowanie: uruchamianie aplikacji wymagających różnych środowisk w jednym systemie.

Uwaga!

- Konteneryzacja nie zapewnia bezpieczeństwa!

Firejail

Używa:

- namespaces do stworzenia iluzji systemu operacyjnego dla aplikacji (fałszywy system plików, fałszywa lista procesów, IPC, interfejsy sieciowe itd.)
- cgroups do limitowana zużycia zasobów.
- seccomp-bpf do filtrowania wywołań systemowych.
- apparmor.
- xpra lub xephyr do izolowania dostępu do serwera okien.
- Osobne profile dla każdej aplikacji. Uwaga: wymagają edycji!

LXC

Super-chroot

- Rootfs gościa jest katalogiem gospodarza.
- Działanie gościa obsługuje demon `lxc monitor`.
- `init` gościa jest zwykłym procesem gospodarza, potomkiem `lxc monitora`.
- Każdy zasób gościa (procesy, pliki itp.) jest widoczny u gospodarza. Gość widzi tylko swoje zasoby.

Polecenia

- 25 poleceń `lxc-*`
- `lxc-start` i `lxc-stop`
- `lxc-attach` i `lxc-console`
- `lxc-create`

Unprivileged LXC containers

Zwykłe kontenery LXC

- `lxc-monitor` pracuje u gospodarza jako `root`.
- Numery użytkowników gościa i gospodarza są takie same.
- Procesy `root`-a gościa pracują jako procesy `root`-a gospodarza (choć w ograniczonych przestrzeniach nazw i cgrupach).

Nieuprzywilejowane kontenery LXC

- Korzystają z wirtualizacji przestrzeni nazw użytkowników (*user namespaces*).
- U gospodarza: pula *subuid* i *subgid* przypisana użytkownikowi.
- Numery użytkowników i grup gościa są tłumaczone na *subuid* i *subgid* gospodarza.
- `lxc-monitor` pracuje u gospodarza jako zwykły użytkownik.

Inne systemy

Izolacja aplikacji

- AppArmor
- SELinux

Pakiety (osobne środowiska aplikacji)

- Snap i Snapcraft (Canonical)
- Flatpack i Flathub (community, „the future of apps on Linux")
- AppImage (community, „Linux apps that run anywhere").

Kontenery: Docker

- Kontenery mobilne — wirtualizacja instalacji.
- UnionFS pozwala dystrybuować tylko nakładki.
- Docker hub.
- Docker Swarm, Kubernetes.