Practical use of Linux capabilities

Gerlof Langeveld
AT Computing
The Netherlands

CentOS Dojo
Oak Ridge National Laboratory
April 16, 2019

Superuser privileges and capabilities

Traditional UNIX privilege scheme

- process running with EUID 0 (superuser): *all* privileged actions allowed
  process running with EUID ≠ 0: *no* privileged actions allowed

Linux privilege scheme

- *capabilities*
  - collection of distinct privileges that can be enabled per process (thread)
  - examples: **`CAP_SYS_BOOT`** **`CAP_SYS_ADMIN`** **`CAP_CHOWN`**
    `CAP_SYS_TIME` **`CAP_SYS_NICE`** **`CAP_KILL`** …

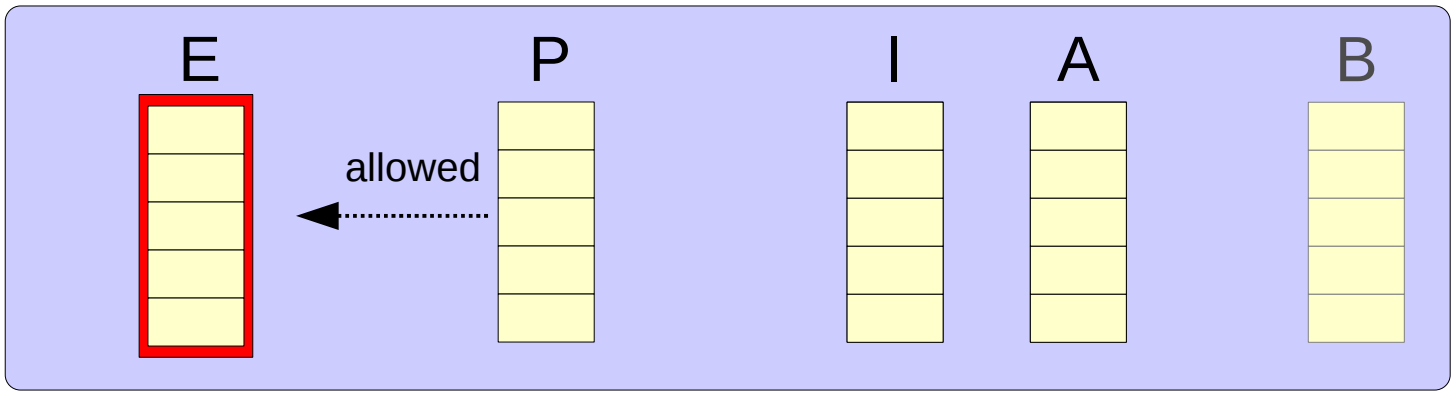
    see man page **`capabilities(7)`**
  - kernel code always checks on single capabilities, not on EUID 0
  - thread running with EUID 0: initially all capabilities enabled

Capability sets – thread

Each thread has five capability sets

- *effective:* verified for each privileged action
- *permitted:* capabilities that can be enabled in effective or in inheritable set
- *inheritable:* passed during program load as permitted set
- *ambient:* preserved during program load to pass capabilities (EUID ≠ 0)
- *bounding:* limiting superset

```
$ cat /proc/$$/task/$$/status
....
CapEff: 0000000000000000
CapPrm: 0000000000000000
CapInh: 0000000000000000
CapAmb: 0000000000000000
CapBnd: 0000001fffffffff
```



thread
admi

Capability sets – awareness

Application program might be

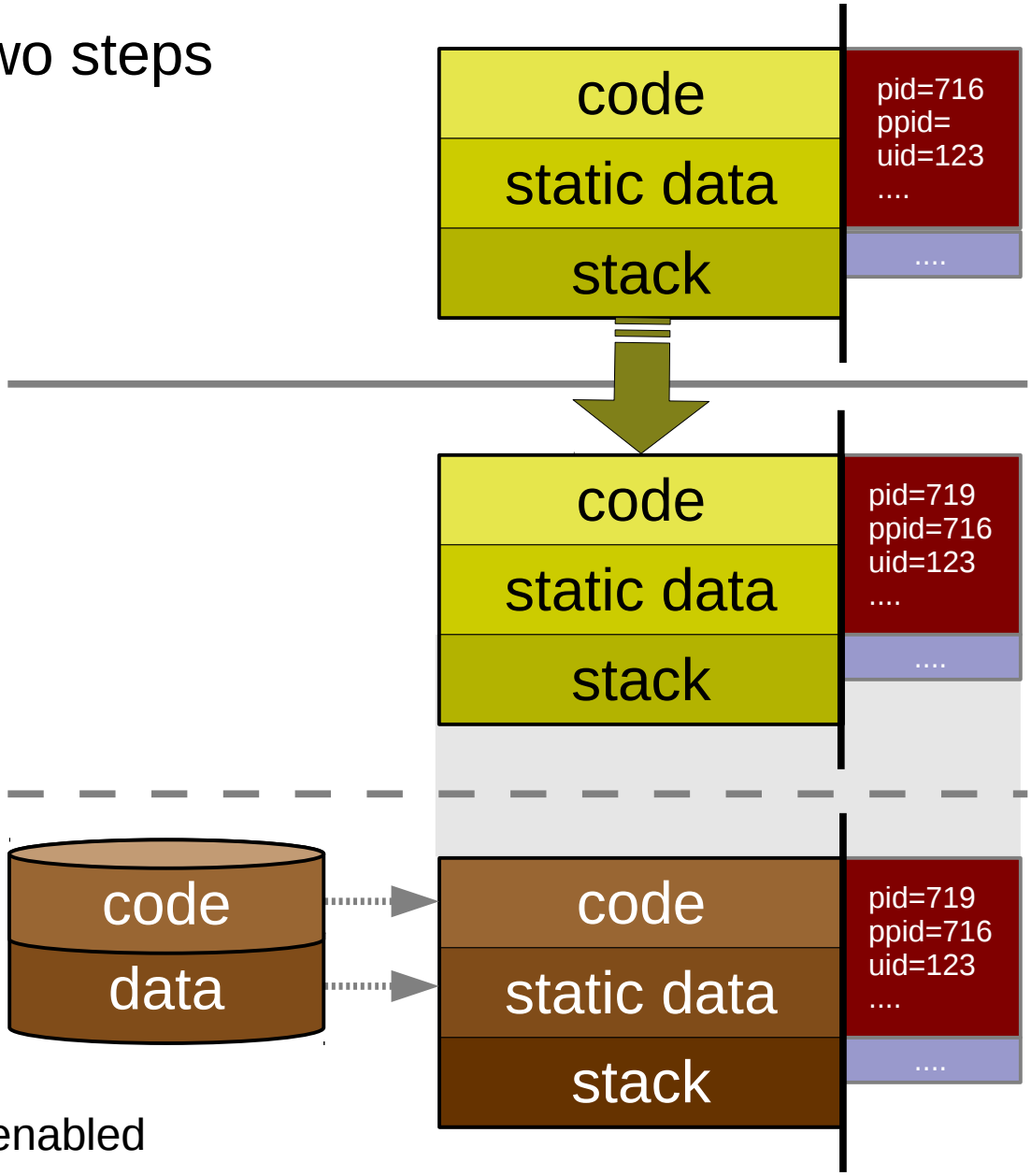
- capability-aware
  - actively manipulates its capability sets with system calls (capset, capget, prctl)

- capability-unaware
  - depends on the capability sets that are
    - inherited from parent
    - manipulated during program load

Process creation and transformation

New program (usually) executed in two steps

- running process

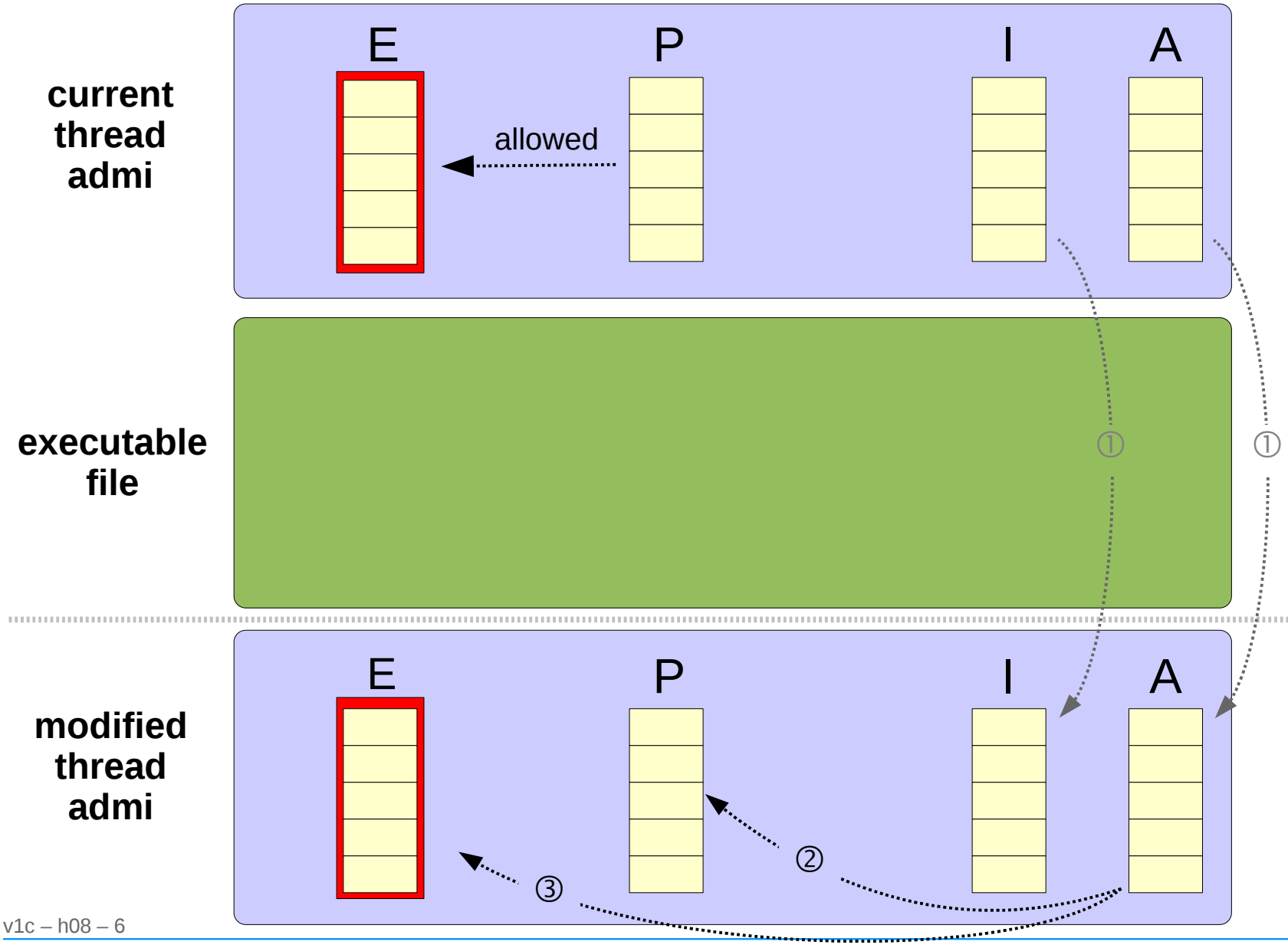


- system call **`fork()`**
  - create new process as clone of existing process
  - capability sets copied

- system call **`execve()`**
  - load new code and data from program file
  - when process' EUID is 0 or program file is setuid root
    – all permitted and effective capabilities enabled

System call **execve**: load executable file

Transition of capabilities: *unprivileged* program file



① $A_t = A_{t(cur)}$
$I_t = I_{t(cur)}$

② $P_t = A_t$

③ $E_t = A_t$

Capabilities — use cases

Capability sets manipulated by

- starting services – **`systemd`**
- starting containers – **`dockerd`**
- loading programs – **`execve()`** system call
- ....

Capabilities — use case **`systemd`** (1)

Process 1 – **`systemd`**

- runs with EUID 0 and all effective capabilities
  - by default inherited by child processes (services) and their descendents

- keywords in service file

	Effective	***Permitted***	***Inheritable***	***Ambient***	***Bounding***
Default	all	all	none	none	all
User=....	none	none	none	none	all
AmbientCapabilities=X	X	X	X	X	all
CapabilityBoundingSet=Y	max(Y)	max(Y)	max(Y)	max(Y)	Y

Capabilities — use case **systemd** (2)

Examples with service

- default service

```
$ capshow -l
  PID     TID COMMAND         EFFECTIVE   PERMITTED   INHERITABL  AMBIENT     BOUND
31744   31744 myserv          1fffffffff  1fffffffff  0000000000  0000000000  1fffffffff
```

- service with **`User=...`**

```
$ capshow -l
  PID     TID COMMAND         EFFECTIVE   PERMITTED   INHERITABL  AMBIENT     BOUND
31745   31745 myserv          0000000000  0000000000  0000000000  0000000000  1fffffffff
```

- service with **`User=...`** and **`AmbientCapabilities=CAP_SYS_NICE`**

```
$ capshow -l
  PID     TID COMMAND         EFFECTIVE   PERMITTED   INHERITABL  AMBIENT     BOUND
31746   31746 myserv          0000800000  0000800000  0000800000  0000800000  1fffffffff
```

- and with **`CapabilityBoundingSet=CAP_SYS_NICE`**

```
$ capshow -l
  PID     TID COMMAND         EFFECTIVE   PERMITTED   INHERITABL  AMBIENT     BOUND
31747   31747 myserv          0000800000  0000800000  0000800000  0000800000  0000800000
```

Capabilities — use case **`dockerd`** (1)

Run command in container: **`docker run [`*`options`*`] `*`image`*` [`*`cmd`*`]`**

- activated command
  - runs as native process, by default under EUID 0
  - runs with selected set of capabilities

- capability-related arguments
  - add capabilities **`--cap-add `*`list`*** (not for normal user)
  - drop capabilities **`--cap-drop `*`list`***
  - user **`--user `*`name`*`|`*`uid`*** (no capabilities)

Capabilities — use case **dockerd** (2)

Example: start **bash** in container

- default capabilities

```
$ docker run -it --rm  centos
[root@1e5aa7d06fd7 /]# nice -n -20  sleep 10
nice: cannot set niceness: Permission denied

[root@1e5aa7d06fd7 /]# chown 1234 /etc/services
[root@1e5aa7d06fd7 /]#
```

```
$ capshow -lp 11652
  PID     TID COMMAND         EFFECTIVE   PERMITTED   INHERITABL  AMBIENT     BOUND
11652   11652 bash            00a80425fb  00a80425fb  00a80425fb  0000000000  00a80425fb

$ capshow -Hp 11652
    PID     TID COMMAND
pid: 11652   tid: 11652   cmd: bash
    CAPEFF: chown dac_override fowner fsetid kill setgid setuid setpcap
            net_bind_service net_raw sys_chroot mknod audit_write setfcap
    CAPPRM: chown dac_override fowner fsetid kill setgid setuid setpcap
            net_bind_service net_raw sys_chroot mknod audit_write setfcap
    CAPINH: chown dac_override fowner fsetid kill setgid setuid setpcap
            net_bind_service net_raw sys_chroot mknod audit_write setfcap
    CAPAMB: NONE
    CAPBND: .....
```

Capabilities — use case **dockerd** (3)

Example: start **bash** in container

- modified capabilities

```
$ docker run -it --rm  --cap-add sys_nice --cap-drop chown  centos
[root@fc08a155bfc0 /]# chown 1234 /etc/services
chown: changing ownership of '/etc/services': Operation not permitted

[root@fc08a155bfc0 /]# nice -n -20  sleep 1000
```

```
$ capshow -Hp 13069
pid: 13069   tid: 13069   cmd: bash
    CAPEFF: dac_override fowner fsetid kill setgid setuid setpcap
            net_bind_service net_raw sys_chroot sys_nice mknod audit_write
            setfcap
    CAPPRM: dac_override fowner fsetid kill setgid setuid setpcap
            net_bind_service net_raw sys_chroot sys_nice mknod audit_write
            setfcap
    CAPINH: dac_override fowner fsetid kill setgid setuid setpcap
            net_bind_service net_raw sys_chroot sys_nice mknod audit_write
            setfcap
    CAPAMB: NONE
    CAPBND: ....

$ ps -lp 13428
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0 13428 13069  0  60 -20 -  1090 hrtime pts/3    00:00:00 sleep
```

Capabilities – use case executable file

Executable file (program) might provide privileges to normal user

- traditional
  - programs with owner **`root`** and *setuid* bit
    - *all* privileged actions allowed, while often only one privilege required

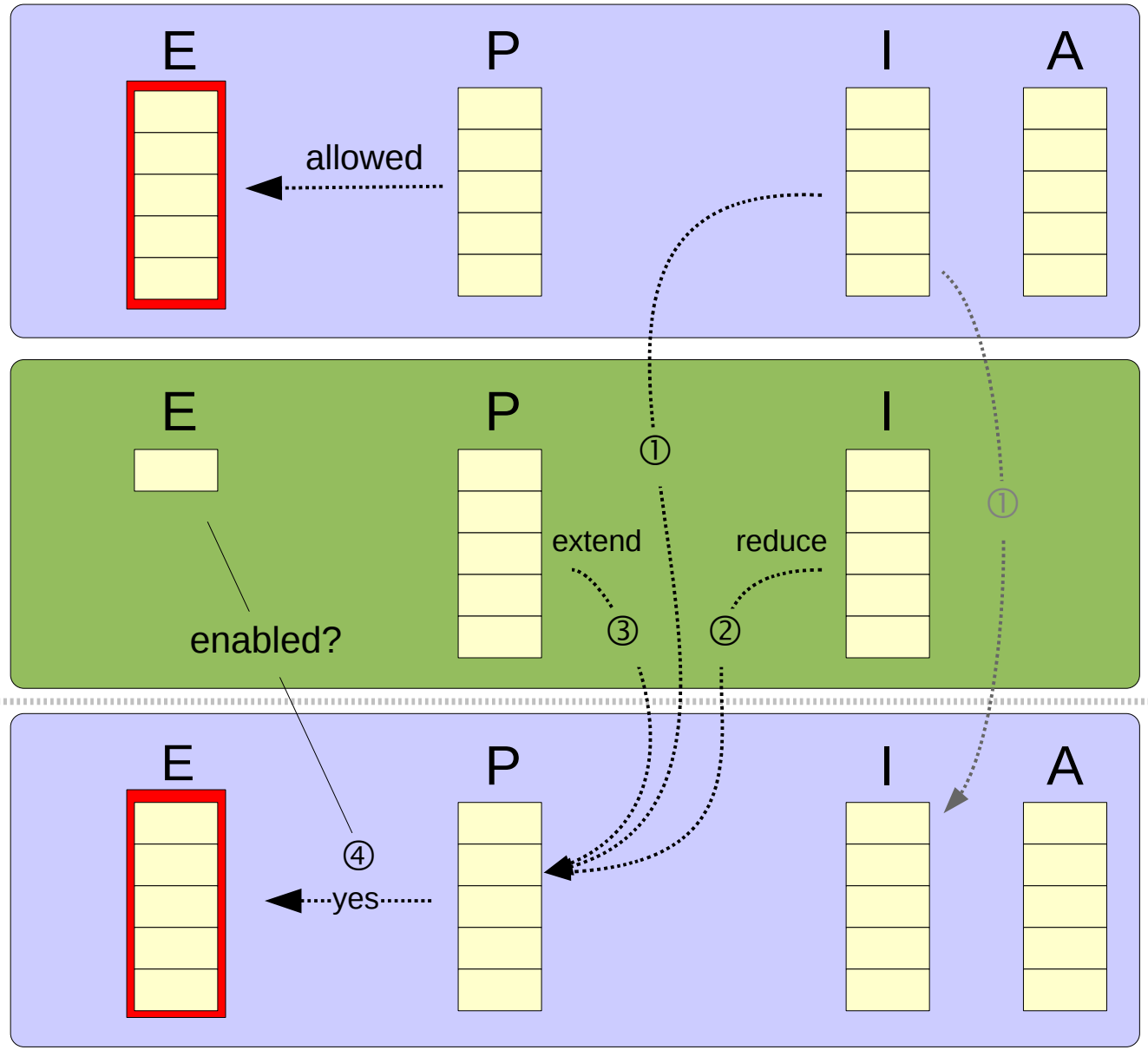
```
$ ls -l /bin/ping
-rwsr-xr-x. 1  root  root  40760  Sep 26 2013 /bin/ping
```

- capabilities
  - program file has its own capability sets
    - *inheritable set:* *reduces* new permitted set of thread after **`execve()`**
    - *permitted set:* *extends* new permitted set of thread after **`execve()`**
    - *effective bit:* if enabled, new permitted set of thread becomes effective set after **`execve()`**
      if disabled, effective set of thread is empty after **`execve()`**

```
$ ls -l /bin/ping
-rwxr-xr-x. 1  root  root  66176  Aug 4  2017 /bin/ping
```

System call **execve**: load executable file

Transition of capabilities: *privileged* program file



① $P_t = I_{t(cur)}$
$I_t = I_{t(cur)}$
$A_t = 0$

② $P_t = P_t \;\&\; I_f$

③ $P_t = P_t \mid P_f$

④
```
if E_f
    E_t = P_t
else
    E_t = 0
```

system call **execve()**

Manipulate file capabilities — **getcap** and **setcap**

Commands **getcap** and **setcap**

- show capabilities on executable file

```
$ ls -l /bin/ping
-rwxr-xr-x. 1  root  root  66176  Aug 4  2017 /bin/ping

$ getcap /bin/ping
/bin/ping = cap_net_admin,cap_net_raw+p
```

- set capabilities on executable file

```
# cp /bin/nice /bin/notnice
# setcap cap_sys_nice=pe /bin/notnice
```

  - 'p' refers to permitted set of file and 'e' to effective bit
  - command **nice** is capability-unaware (only 'p' would be useless)

```
$ nice -n -11  sleep 10
notnice: cannot set niceness: Permission denied

$ notnice -n -11  sleep 10
```

Manipulate thread capabilities — aware programs

Capability-aware programs

- manipulate own capability sets
  - give up unnecessary capabilities
  - make permitted capabilities effective
  - .....

- example: command **`ping`**

```
$ getcap /bin/ping
/bin/ping = cap_net_admin,cap_net_raw+p
```

```
# strace  ping localhost
...
capset(..., {0, 1<<CAP_NET_ADMIN|1<<CAP_NET_RAW, 0}) = 0
...
capget(..., {0, 1<<CAP_NET_ADMIN|1<<CAP_NET_RAW, 0}) = 0
capset(..., {1<<CAP_NET_RAW, 1<<CAP_NET_ADMIN|1<<CAP_NET_RAW, 0}) = 0
...
socket(AF_INET, SOCK_RAW, IPPROTO_ICMP) = 3
capset(..., {0, 1<<CAP_NET_ADMIN|1<<CAP_NET_RAW, 0}) = 0
...
```

Manipulate thread capabilities – case: **atop** (1)

```
ATOP - robin          2019/03/14  14:10:36          --------------          10s elapsed
PRC | sys    2.49s  | user  10.93s  | #proc    311 |  #zombie    0 |  #exit      5 |
CPU | sys      27%  | user    113%  | irq       3% |  idle    188% |  wait     69% |
CPL | avg1    1.77  | avg5    0.80  | avg15   0.69 |  csw   135876 |  intr  109271 |
MEM | tot     7.7G  | free  126.7M  | cache   5.3G |  buff    0.0M |  slab  559.5M |
SWP | tot    10.0G  | free    9.1G  |              |  vmcom   5.4G |  vmlim  13.9G |
PAG | scan  213673  | steal 213442  | stall       0 |  swin       0 |  swout    197 |
LVM | tos_ssd-root  | busy      84% | read    23675 |  write      0 |  avio 0.35 ms |
LVM | tos_hdd-bulk  | busy       9% | read     1776 |  write     16 |  avio 0.52 ms |
LVM | tos_hdd-swap  | busy       0% | read        0 |  write    197 |  avio 0.22 ms |
DSK |          sdb  | busy      84% | read    23670 |  write      3 |  avio 0.35 ms |
DSK |          sda  | busy       9% | read     1774 |  write     17 |  avio 0.52 ms |
NET | transport     | tcpi   89665  | tcpo  170355 |  udpi       0 |  udpo       0 |
NET | network       | ipi    89667  | ipo   170355 |  ipfrw      0 |  deliv  89665 |
NET | eth0     19%  | pcki   89681  | pcko  170355 |  si 4940 Kbps |  so  195 Mbps |

  PID SYSCPU  USRCPU  VGROW   RGROW   RDDSK   WRDSK  RNET   SNET  CPU  CMD        1/5
32558  0.30s   9.44s  1312K   1212K      0K      0K     0      0  99%  ssh
32559  0.80s   0.82s     0K     -4K      0K      0K 89601  170e3  17%  ssh
32552  0.40s   0.06s     0K      0K       -       -     0      0   5%  <grep>
32557  0.33s   0.02s     0K      0K  222.1M      0K     0      0   4%  scp
 2915  0.21s   0.05s 17392K   4100K      0K      0K     0      0   3%  Xorg
 4329  0.03s   0.14s     0K    -452K      0K      0K     0      0   2%  firefox
   58  0.17s   0.00s     0K      0K      0K      0K     0      0   2%  kswapd0
27166  0.03s   0.02s     0K      0K      0K      0K     0      0   1%  atop
```

- requires setuid-root for special privileges

Manipulate thread capabilities – case: **atop** (2)

Special privileges

- process accounting
  - switch on — **cap_sys_acct**
  - accounting file in protected directory — **cap_dac_override**
- processor and memory
  - lower nice — **cap_sys_nice**
  - memory lock — **cap_ipc_lock**
    set resource limit — **cap_sys_resource**
- special metrics
  - instructions per cpu cycle — **cap_sys_admin**
  - disk read/write **/proc/pid/io** — **cap_dac_read_search**
    cap_sys_ptrace

```
# setcap  \
cap_sys_pacct,cap_dac_override,cap_sys_nice,cap_ipc_lock,cap_sys_resource,cap_sys_admin,cap_sys_ptrace=pe    /usr/bin/atop
```

Practical use of Linux capabilities

Questions?

command **capshow**:
https://old.atcomputing.nl/downloads/capshow.tar