



Manuel Schuller
Janos Pasztor

~~DevSecOps - Build it and Secure It~~
What's under the hood of Docker?
Process separation in the Linux kernel

What I do



www.a1.digital



www.exoscale.com

What I do



pasztor.at



@janosze
n

More content

Today

Today

1. What are containers?

Today

1. What are containers?
2. Containerization history

Today

1. What are containers?
2. Containerization history
- 3. Capabilities**

Today

1. What are containers?
2. Containerization history
3. Capabilities
4. **Linux namespaces**

Today

1. What are containers?
2. Containerization history
3. Capabilities
4. Linux namespaces
5. **Cgroups**

Today

1. What are containers?
2. Containerization history
3. Capabilities
4. Linux namespaces
5. Cgroups
6. **Seccomp**

Today

1. What are containers?
2. Containerization history
3. Capabilities
4. Linux namespaces
5. Cgroups
6. Seccomp
- 7. AppArmor**

Today

1. What are containers?
2. Containerization history
3. Capabilities
4. Linux namespaces
5. Cgroups
6. Seccomp
7. AppArmor
8. **Demo!**

Today

A few caveats!

1. What are containers?

1. What are containers?



Hardware

1. What are containers?

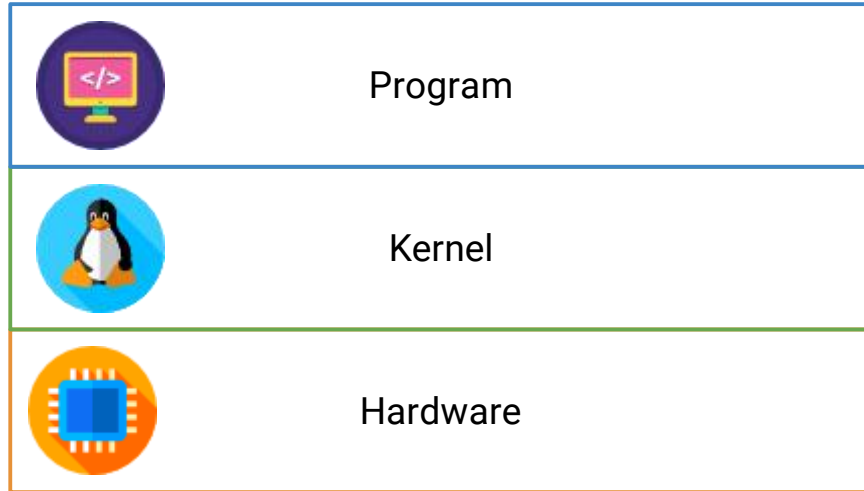


Program

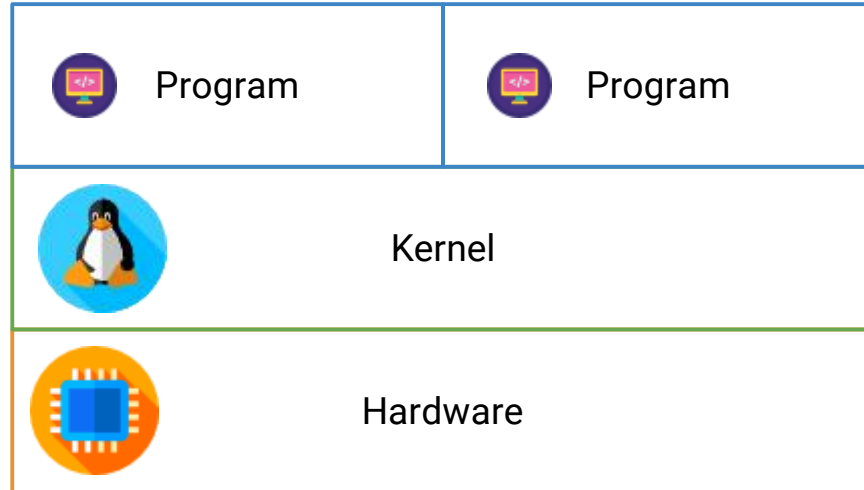


Hardware

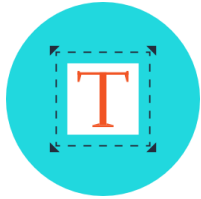
1. What are containers?



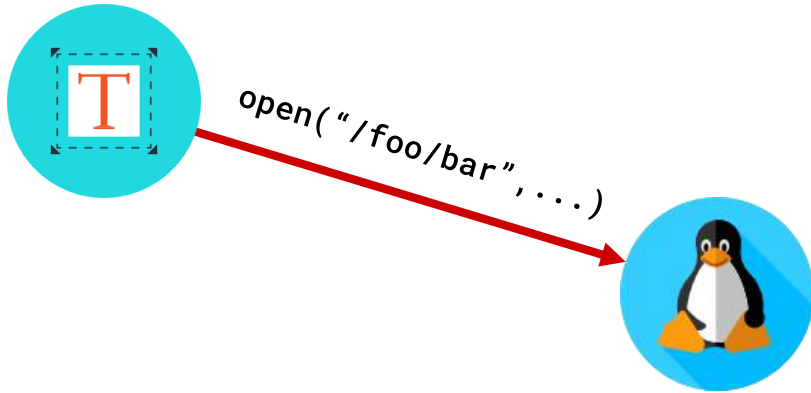
1. What are containers?



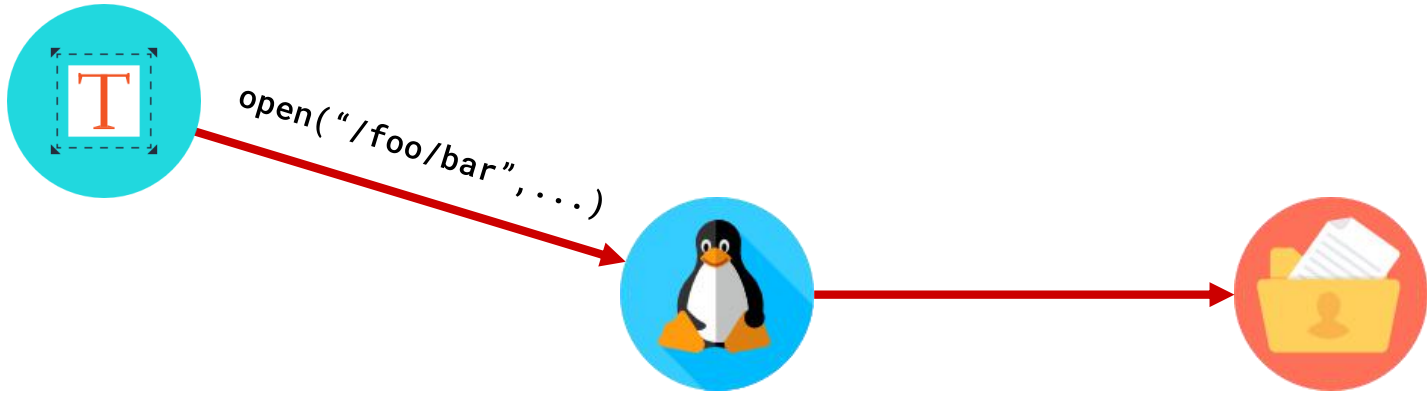
1. What are containers?



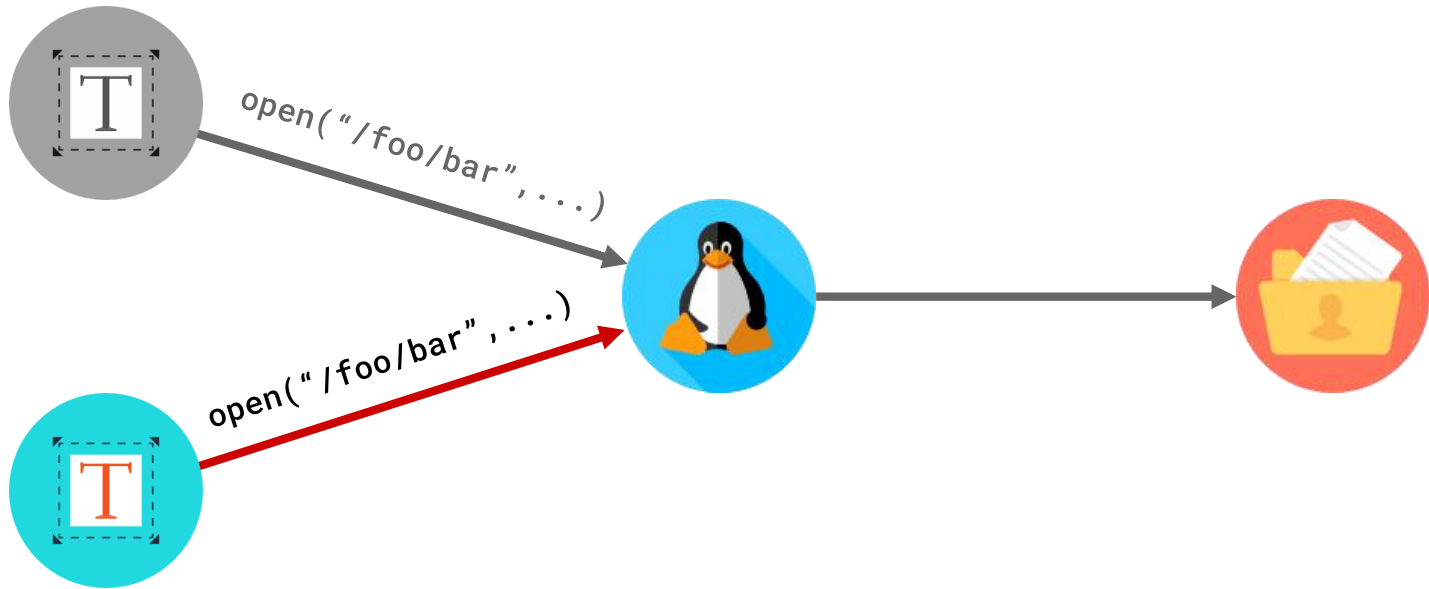
1. What are containers?



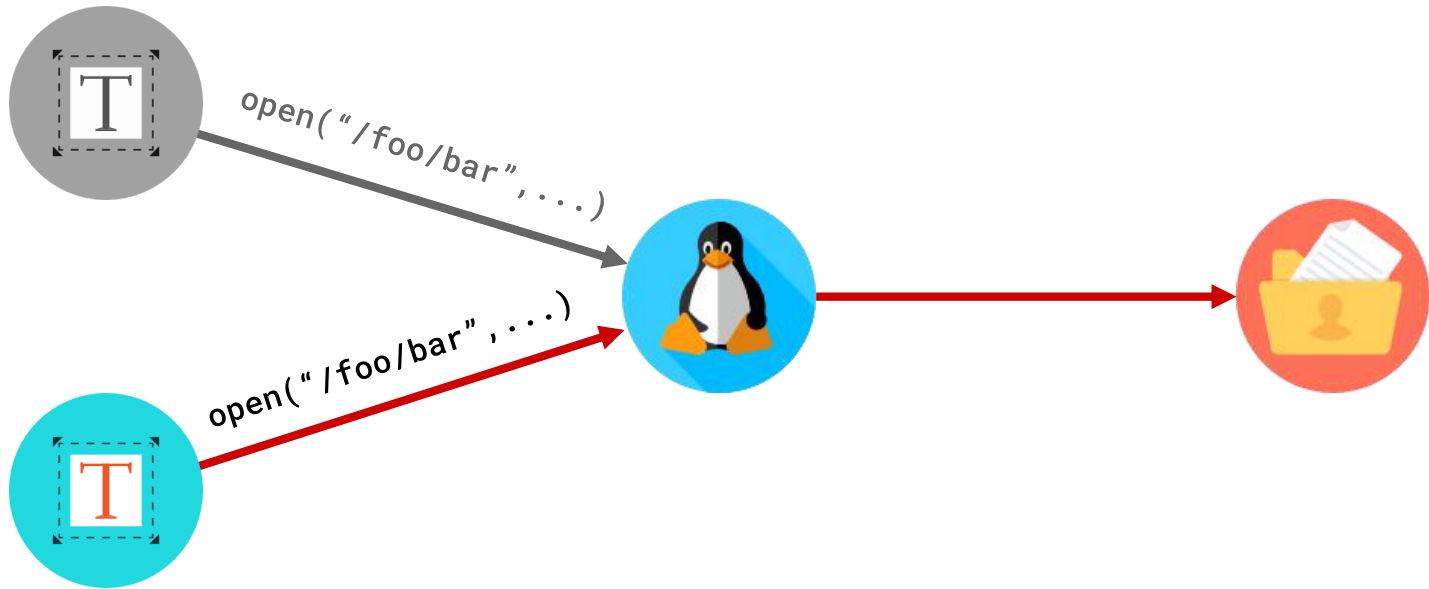
1. What are containers?



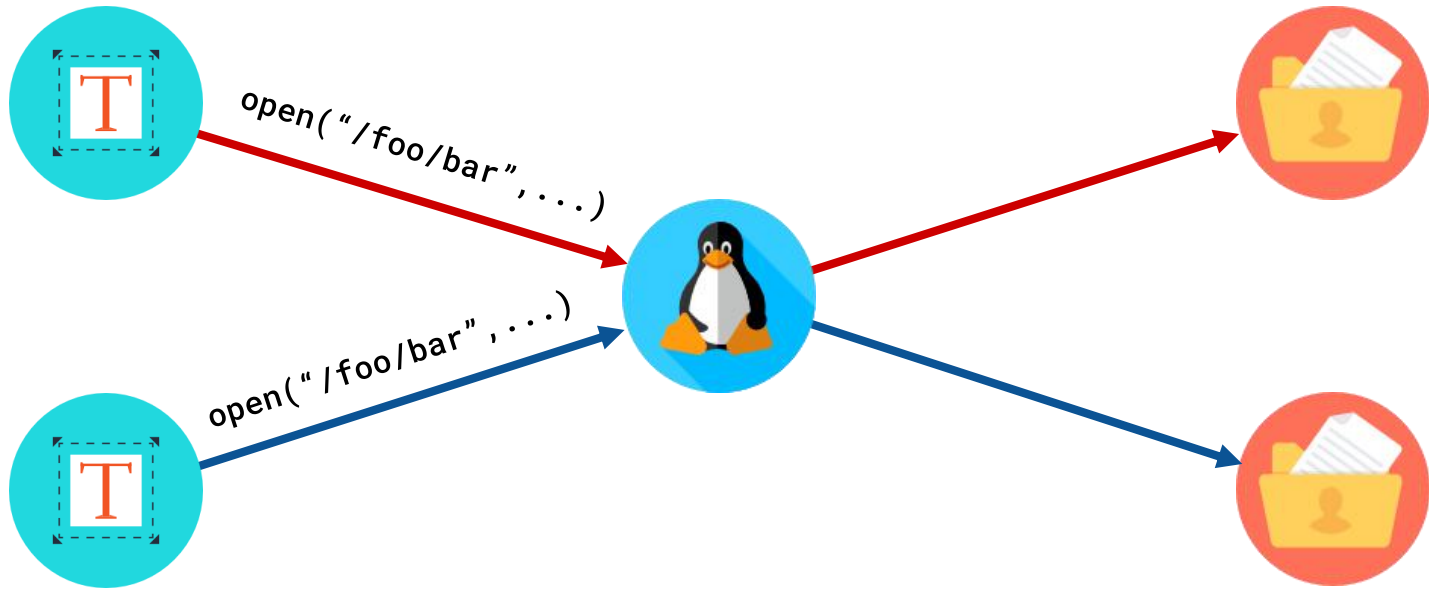
1. What are containers?



1. What are containers?



1. What are containers?



1. What are containers?



Mount

1. What are containers?



Mount



User

1. What are containers?



Mount



User



UTS

1. What are containers?



Mount



Network



User



UTS

1. What are containers?



Mount



Network



User



Process
ID



UTS

1. What are containers?



Mount



User



UTS



Network

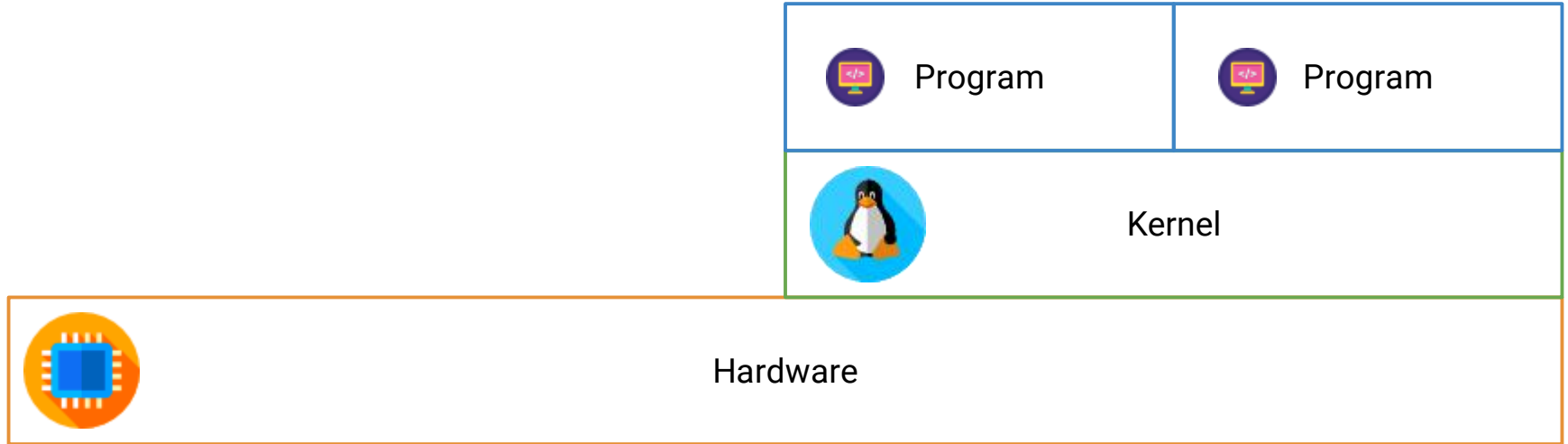


Process

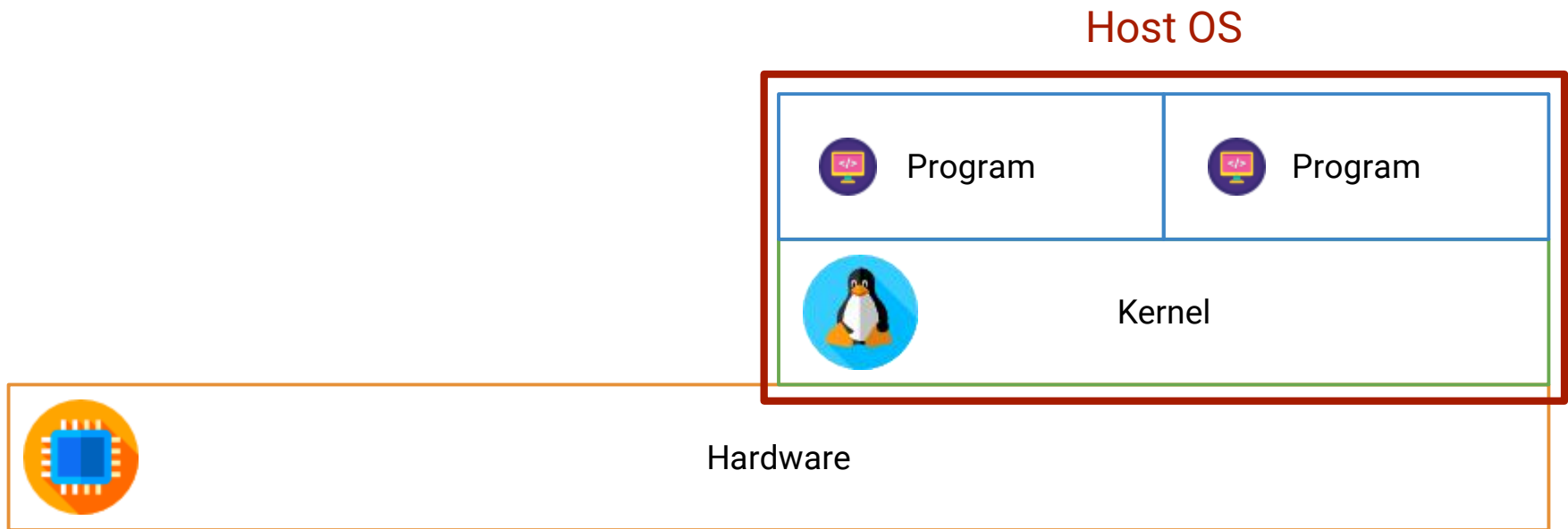


ID

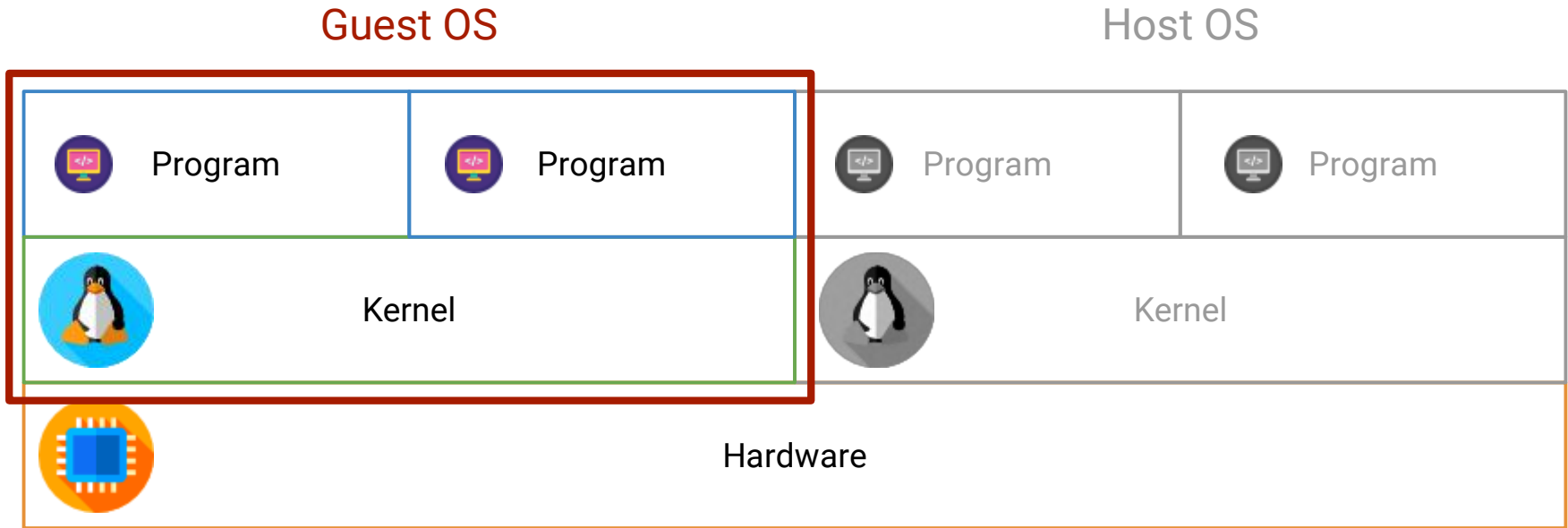
1. What are containers?



1. What are containers?

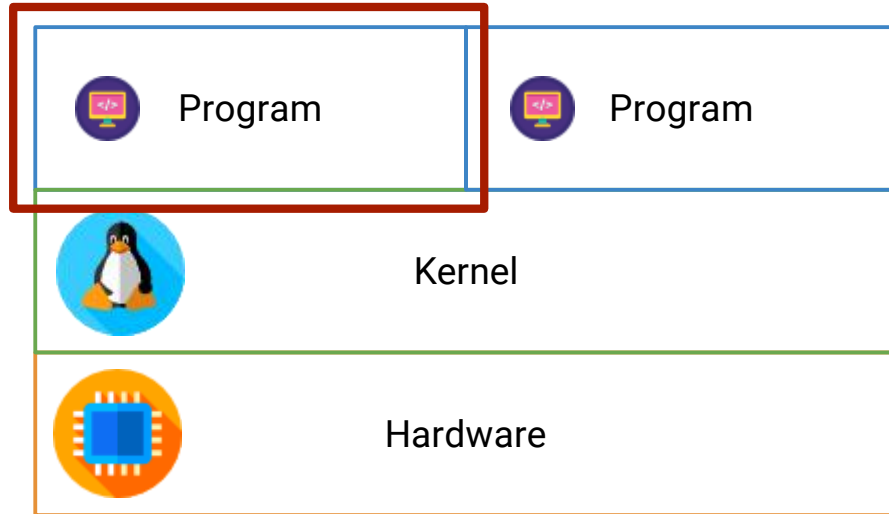


1. What are containers?



1. What are containers?

Container



1. What are containers?

`docker run traefik`

1. What are containers?

ps auwfx

1. What are containers?

Host:

— `/usr/bin/dockerd`

1. What are containers?

Host:

- /usr/bin/dockerd

 - └─ docker-containerd

1. What are containers?

Host:

- /usr/bin/dockerd

 - └─ docker-containerd

 - └─ **docker-containerd-shim**

1. What are containers?

Host:

- /usr/bin/dockerd
 - └─ docker-containerd
 - └─ docker-containerd-shim
 - └─ /traefik

1. What are containers?

Guest:

— /traefik

1. What are containers?

1. Capabilities

1. What are containers?

1. Capabilities

2. Linux namespaces

1. What are containers?

1. Capabilities
2. Linux namespaces
3. **Seccomp**

1. What are containers?

1. Capabilities
2. Linux namespaces
3. Seccomp
4. AppArmor, SELinux, etc

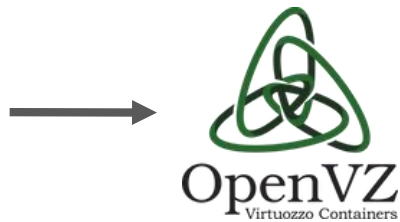
1. What are containers?

1. Capabilities
2. Linux namespaces
3. Seccomp
4. AppArmor, SELinux, etc
5. ...

2. Containerization history

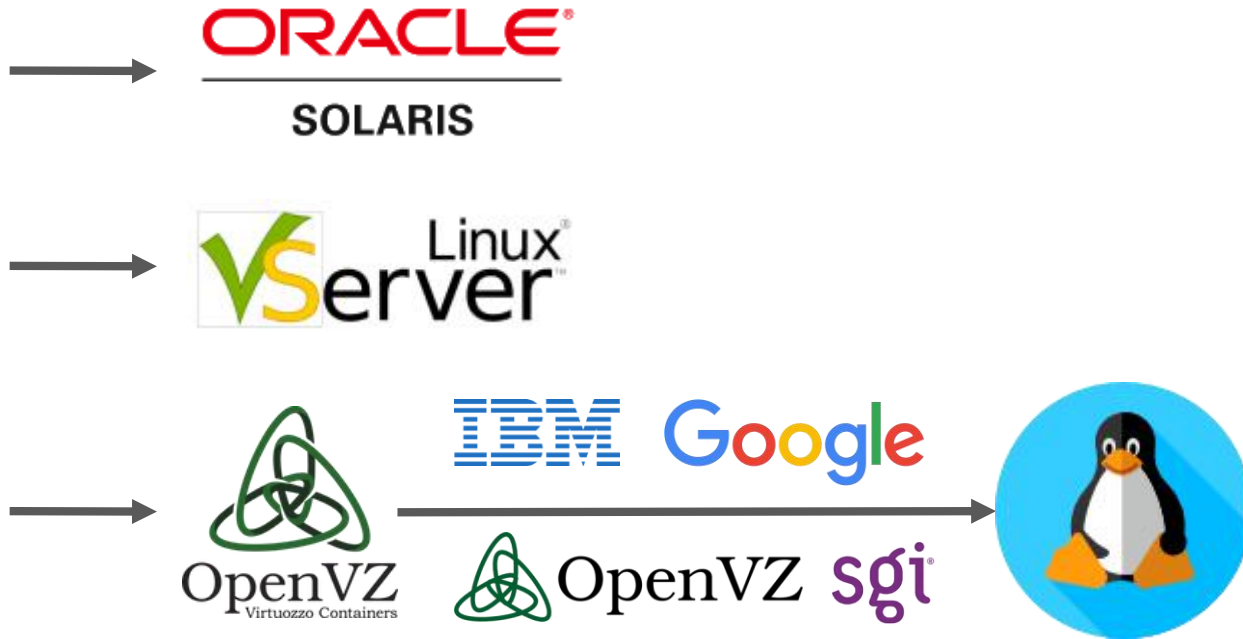
Thank you to Kir Kolyshkin (@kolyshkin) for helping me put this together.

2. Containerization history



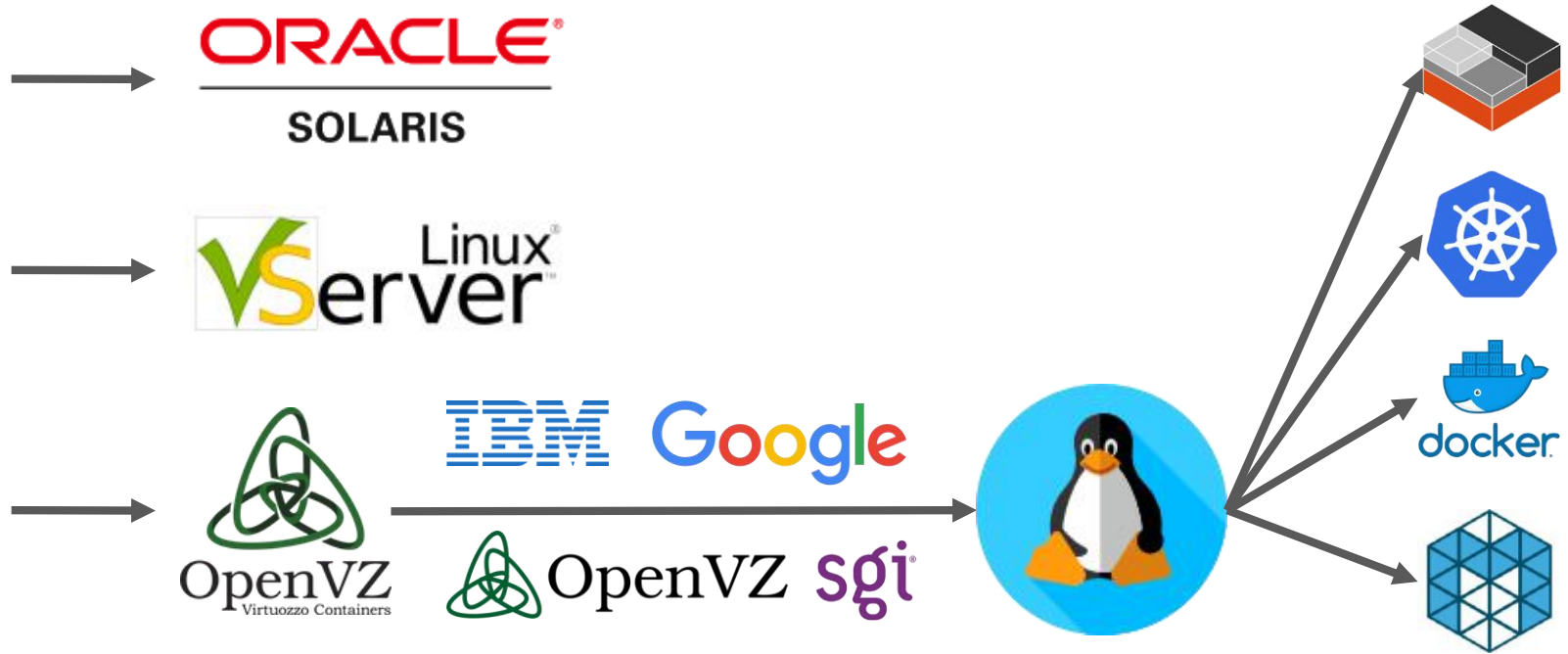
Thank you to Kir Kolyshkin (@kolyshkin) for helping me put this together.

2. Containerization history



Thank you to Kir Kolyshkin (@kolyshkin) for helping me put this together.

2. Containerization history



Thank you to Kir Kolyshkin (@kolyshkin) for helping me put this together.

2. Containerization history

A Brief History of Containers
Jeff Victor & Kir Kolyshkin

3. Capabilities

3. Capabilities

Can you run ping as a normal user?

(Shout the answer.)

3. Capabilities

```
$ ls -lah /bin/ping
```

```
-rwsr-xr-x 1 root root 44K May 7  
2014 /bin/ping
```

3. Capabilities

SUID bit

```
$ ls -lah /bin/ping
```

```
-rwsr-xr-x 1 root root 44K May 7  
2014 /bin/ping
```



3. Capabilities

CAP_NET_RAW

3. Capabilities

CAP_NET_ADMIN

3. Capabilities

Example: OpenVPN in Docker

3. Capabilities

Example: OpenVPN in Docker

```
docker run \  
  --cap-add NET_RAW \  
  --cap-add NET_ADMIN \  
  ...
```

3. Capabilities

CAP_CHOWN

3. Capabilities

CAP_KILL

3. Capabilities

CAP_NET_BIND_SERVICE

3. Capabilities

CAP_SETUID and CAP_SETGID

3. Capabilities

CAP_SYS_CHROOT

3. Capabilities

CAP_SYS_NICE

3. Capabilities

libcap

libcap-ng

prctl

3. Capabilities

```
int main() {
```

3. Capabilities

```
int main() {  
    capng_clear(CAPNG_SELECT_BOTH);  
}
```

3. Capabilities

```
int main() {  
    capng_clear(CAPNG_SELECT_BOTH);  
    capng_apply(CAPNG_SELECT_BOTH);  
}
```

3. Capabilities

```
int main() {  
    capng_clear(CAPNG_SELECT_BOTH);  
    capng_apply(CAPNG_SELECT_BOTH);  
    //Start bash here  
}
```

3. Capabilities

Actually, it's quite hard...

4. Linux namespaces

4. Linux Namespaces

```
int main() {
```

```
}
```

4. Linux Namespaces

```
int main() {  
    //stack magic  
  
}
```

4. Linux Namespaces

```
int main() {  
    //stack magic  
    pid_t childPid = clone(  
  
    );  
}
```

4. Linux Namespaces

```
int main() {  
    //stack magic  
    pid_t childPid = clone(  
        childFunction,  
  
    );  
}
```

4. Linux Namespaces

```
int main() {  
    //stack magic  
    pid_t childPid = clone(  
        childFunction,  
        stackTop,  
  
    );  
}
```

4. Linux Namespaces

```
int main() {  
    //stack magic  
    pid_t childPid = clone(  
        childFunction,  
        stackTop,  
        CLONE_NEWNS | CLONE_NEWNET | SIGCHLD,  
  
    );  
  
}
```

4. Linux Namespaces

```
int main() {  
    //stack magic  
    pid_t childPid = clone(  
        childFunction,  
        stackTop,  
        CLONE_NEWNS | CLONE_NEWNET | SIGCHLD,  
        {}  
    );  
}
```

4. Linux Namespaces

```
int main() {  
    //stack magic  
    pid_t childPid = clone(  
        childFunction,  
        stackTop,  
        CLONE_NEWNS | CLONE_NEWNET | SIGCHLD,  
        {}  
    );  
    //wait for child here  
}
```


4. Linux Namespaces

```
int main() {  
    //stack magic  
    pid_t childPid = clone(  
        childFunction,  
        stackTop,  
        CLONE_NEWNS | CLONE_NEWNET | SIGCHLD,  
        {}  
    );  
    //wait for child here  
}
```

4. Linux Namespaces

```
static int childFunction(void *arg) {  
    //Start bash here  
}
```

4. Linux Namespaces

`clone()`: New process in NS

4. Linux Namespaces

`clone()` : New process in NS

`unshare()` : Existing process in NS

4. Linux Namespaces

`clone()`: New process in NS

`unshare()`: Existing process in NS

`setns()`: Join NS

4. Linux Namespaces

```
TARGET=my_nginx_container
```

4. Linux Namespaces

```
TARGET=my_nginx_container
```

```
docker run \
```

```
-ti
```

```
janoszen/debug
```

4. Linux Namespaces

```
TARGET=my_nginx_container
```

```
docker run \
```

```
  --pid container:$TARGET \
```

```
  --net container:$TARGET \
```

```
-ti
```

```
janoszen/debug
```


4. Linux Namespaces

```
TARGET=my_nginx_container
```

```
docker run \
```

```
  --pid container:$TARGET \
```

```
  --net container:$TARGET \
```

```
  --cap-add NET_RAW \
```

```
  --cap-add NET_ADMIN \
```

```
  --cap-add SYS_PTRACE \
```

```
  -ti
```

```
  janoszen/debug
```

5. Cgroups

5. Cgroups

```
cgroup on /sys/fs/cgroup/cpuset type cgroup  
(rw,nosuid,nodev,noexec,relatime,cpuset)
```

5. Cgroups

```
cgcreate -g cpu,cpuacct:/my_group
```

5. Cgroups

```
cgcreate -g cpu,cpuacct:/my_group
```

```
cgclassify -g cpu,cpuacct:/my_group 26432
```

5. Cgroups

```
cgcreate -g cpu,cpuacct:/my_group
```

```
cgclassify -g cpu,cpuacct:/my_group 26432
```

```
echo "2" > /sys/fs/cgroup/cpuset/my_group/cpuset.cpus
```

5. Cgroups

cpuset.cpus

5. Cgroups

cpu.shares

5. Cgroups

`cpu.rt_runtime_us`

`cpu.rt_period_us`

5. Cgroups

`memory.limit_in_bytes`

5. Cgroups

`memory.memsw.limit_in_bytes`

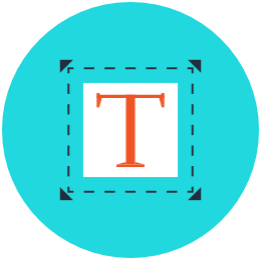
5. Cgroups

`blkio.throttle.read_bps_device`

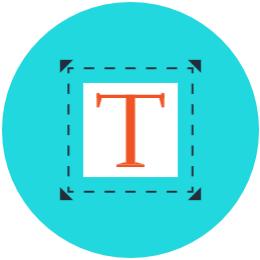
`blkio.throttle.write_bps_device`

6. Seccomp

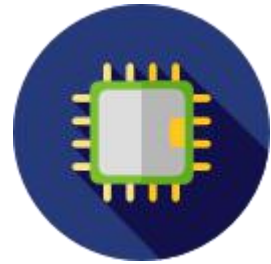
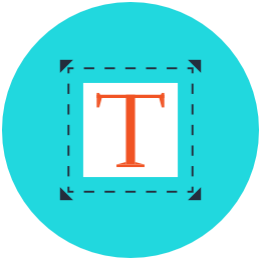
6. Seccomp



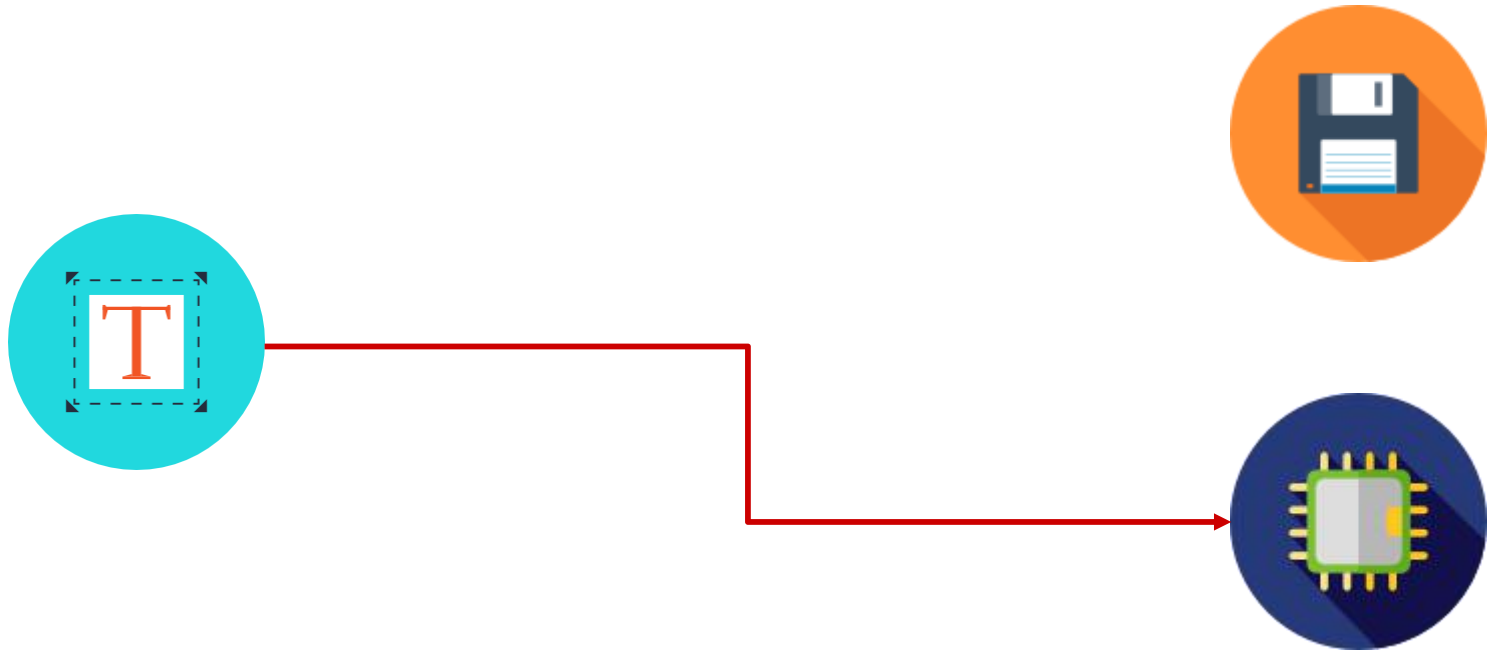
6. Seccomp



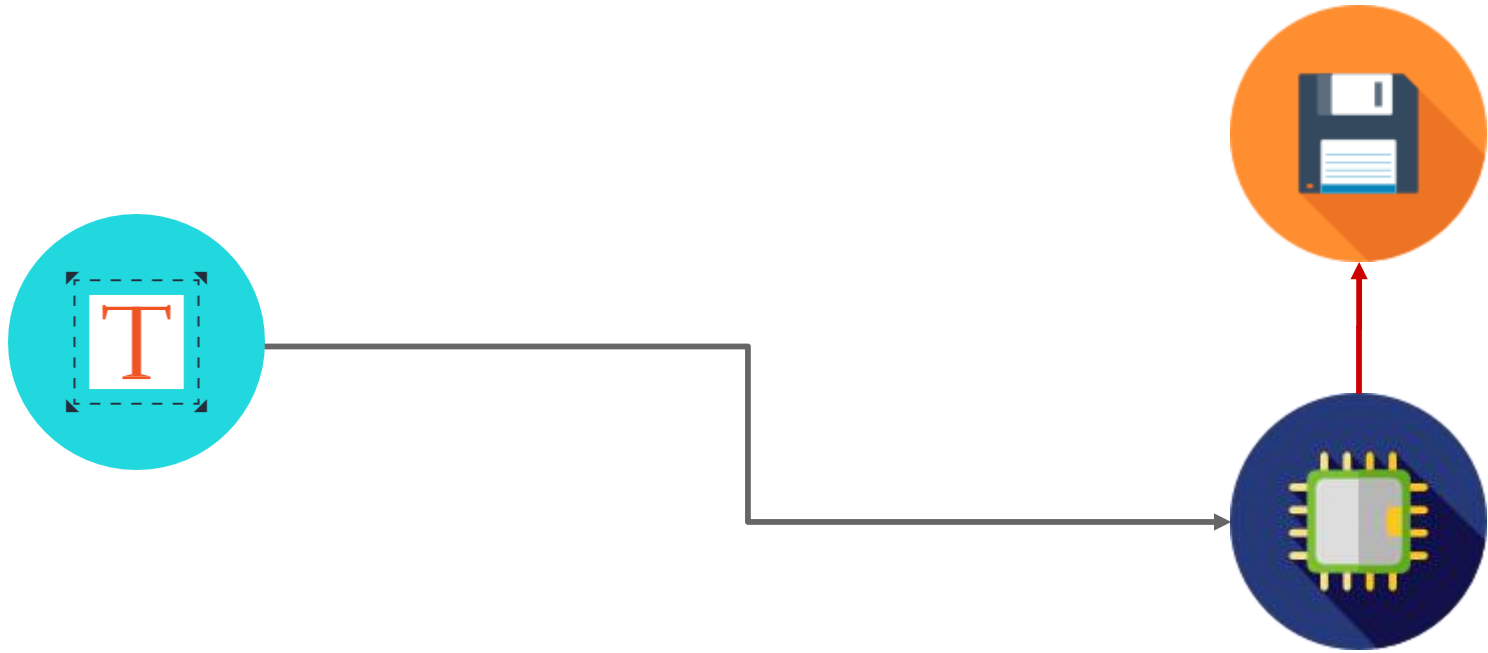
6. Seccomp



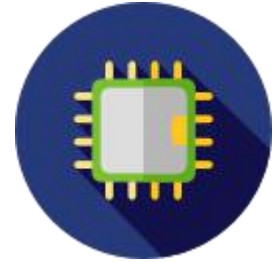
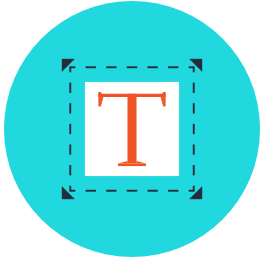
6. Seccomp



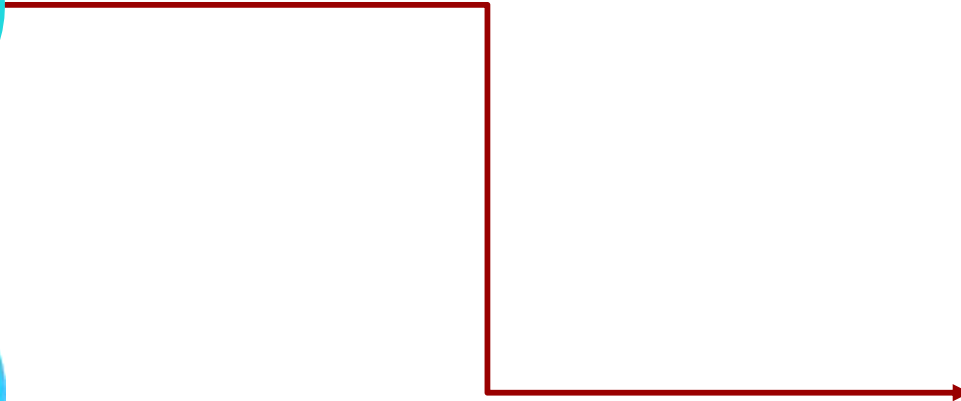
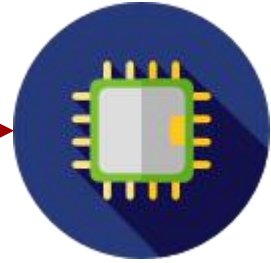
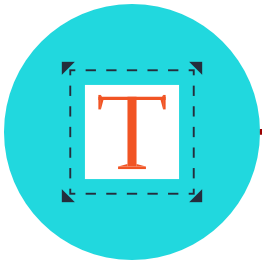
6. Seccomp



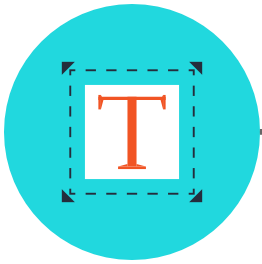
6. Seccomp



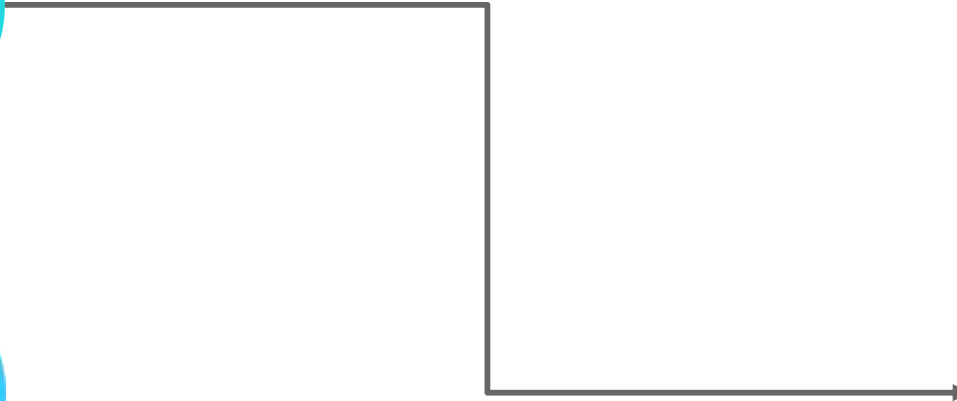
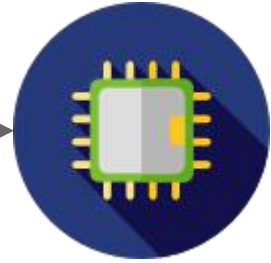
6. Seccomp



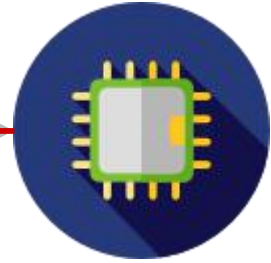
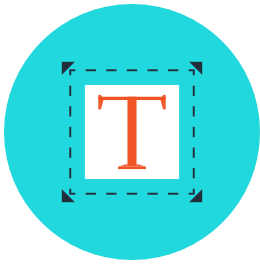
6. Seccomp



X



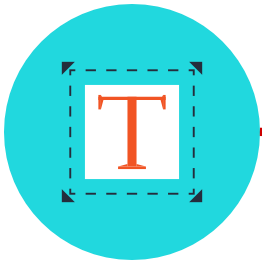
6. Seccomp



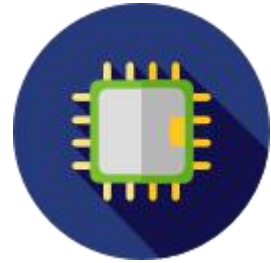
6. Seccomp



6. Seccomp



Syscall



6. Seccomp

```
ctx = seccomp_init(SCMP_ACT_ALLOW);
```

6. Seccomp

```
ctx = seccomp_init(SCMP_ACT_ALLOW);  
  
seccomp_rule_add(  
    ctx,  
    SCMP_ACT_KILL,  
    SCMP_SYS(bind),  
    0  
);
```

6. Seccomp

```
ctx = seccomp_init(SCMP_ACT_ALLOW);  
  
seccomp_rule_add(  
    ctx,  
    SCMP_ACT_KILL,  
    SCMP_SYS(bind),  
    0  
);  
  
seccomp_load(ctx)
```

7. AppArmor

7. AppArmor

deny network raw

7. AppArmor

deny mount

7. AppArmor

```
deny @{PROC}/sysrq-trigger rwk1x
```

7. AppArmor

capability setuid

8. Demo

8. Demo



8. Demo

`github.com/janoszen`



Questions?

@janoszen