Financira
Evropska unija
NextGenerationEU

THE RECOVERY AND RESILIENCE PLAN

VARNOST PROGRAMOV

Predavanja #9

Matevž Pesek

Teme!

- Rok je ~~15. maj~~ -> 20. maj!

Od prejšnjič

- Defenzivno programiranje – kaj je namen?
- Kateri so najpogostejši vzorci (v programski kodi)?

DANAŠNJE TEME

- Linux:
- SECCOMP
- Landlock
- Control groups
- AppArmor
- Kontejnerji in izolacija

OMEJITEV DOSTOPA

Trije nivoji zaščite – obramba

- Izogibanje ranljivostim
  - Preverba kode, uporaba specifičnih jezikov, izogibanje vzorcem, rokovanje z izjemami, …
- Preventiva
  - Uporaba IDE (vzorci), monitoriranje procesov, požarni zid (in vzorci), …
- **Omejevanje privilegijev**
  - Virtualizacija, omejitve pri dostopu (človek), omejitve na nivoju procesov, …

Nivoji omejevanja

- Peskovnik znotraj procesa
  - V8 – Javascript
- Peskovnik za process
  - Seccomp
  - Landlock
- Virtualizacija
  - Control group
  - Namespace
- Kontejnerji
  - Docker

Peskovnik znotraj procesa

- Aplikacija omejuje kodo uporabnika
  - Kodo interpretiramo in zgolj prevajamo v vmesno kodo (bytecode)
    - Javascript (v8 engine)
- Peskovnik postavlja omejitve za interakcijo s sistemom
  - Napaka v peskovniku vodi k dostopu do sistema
- Interpretacija je pogosto počasna
  - Optimizacije velikokrat vodijo do novih napak

Peskovnik za process

- Postavimo peskovnik okoli procesa, da mu omejimo interakcijo s sistemom
  - Seccomp
  - Landlock
  - AppArmor
- Omejimo:
  - Sistemske klice
  - Dostop do mreže
  - Dostop do podatkov

https://selectel.ru/blog/en/2017/07/25/containers-security-seccomp/

PRIMERI

Sistemski klici

```
execve("./example", ["./example"], 0x7ffc5f5e2560 /* 71 vars */) = 0
brk(NULL)                               = 0x5d9e400a0000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=268847, ...}) = 0
mmap(NULL, 268847, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7b7cd03ec000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 _\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
fstat(3, {st_mode=S_IFREG|0755, st_size=1961272, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7b7cd03ea000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 1981296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7b7cd0206000
mmap(0x7b7cd022a000, 1458176, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x24000) = 0x7b7cd022a000
mmap(0x7b7cd038e000, 319488, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x188000) = 0x7b7cd038e000
mmap(0x7b7cd03dc000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d6000) = 0x7b7cd03dc000
mmap(0x7b7cd03e2000, 31600, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7b7cd03e2000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7b7cd0203000
arch_prctl(ARCH_SET_FS, 0x7b7cd0203740) = 0
set_tid_address(0x7b7cd0203a10)         = 39297
set_robust_list(0x7b7cd0203a20, 24)     = 0
rseq(0x7b7cd0204060, 0x20, 0, 0x53053053) = 0
mprotect(0x7b7cd03dc000, 16384, PROT_READ) = 0
mprotect(0x5d9e40012000, 4096, PROT_READ) = 0
mprotect(0x7b7cd0460000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7b7cd03ec000, 268847)          = 0
exit_group(0)                           = ?
```

```
int main()
{
        return 0;
}
```

```
execve("./example", ["./example"], 0x7fff893e41a0 /* 71 vars */) = 0
brk(NULL)                               = 0x58b263592000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=268847, ...}) = 0
mmap(NULL, 268847, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7c1b75dcf000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 _\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
fstat(3, {st_mode=S_IFREG|0755, st_size=1961272, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7c1b75dcd000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 1981296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7c1b75be9000
mmap(0x7c1b75c0d000, 1458176, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x24000) = 0x7c1b75c0d000
mmap(0x7c1b75d71000, 319488, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x188000) = 0x7c1b75d71000
mmap(0x7c1b75dbf000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d6000) = 0x7c1b75dbf000
mmap(0x7c1b75dc5000, 31600, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7c1b75dc5000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7c1b75be6000
arch_prctl(ARCH_SET_FS, 0x7c1b75be6740) = 0
set_tid_address(0x7c1b75be6a10)         = 39059
set_robust_list(0x7c1b75be6a20, 24)     = 0
rseq(0x7c1b75be7060, 0x20, 0, 0x53053053) = 0
mprotect(0x7c1b75dbf000, 16384, PROT_READ) = 0
mprotect(0x58b26300c000, 4096, PROT_READ) = 0
mprotect(0x7c1b75e43000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7c1b75dcf000, 268847)          = 0
fstat(1, {st_mode=S_IFCHR|0600, st_rdev=makedev(0x88, 0x7), ...}) = 0
getrandom("\xd3\x06\x87\x2c\xc0\xd7\x14\x1a", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x58b263592000
brk(0x58b2635b3000)                     = 0x58b2635b3000
write(1, "Hello, World!\n", 14Hello, World!
)         = 14
exit_group(0)                           = ?
```

```
int main()
{
        printf("Hello world\n"); return 0;
}
```

PRIMERI

SECCOMP

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/syscall.h>
#include <linux/seccomp.h>
#include <seccomp.h>

int main()
{
 scmp_filter_ctx ctx = seccomp_init(SCMP_ACT_KILL);

 seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(exit), 0);
 seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(exit_group), 0);
 seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(brk), 0);
 seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(write), 2,
   SCMP_A0(SCMP_CMP_EQ, 1),
   SCMP_A2(SCMP_CMP_LE, 64)
 );

 seccomp_load(ctx);
 seccomp_release(ctx);

 write(1, "Hello, World!\n", 14);
 write(1, "Hello, Wooooooooooooooooooooooooorld!\n", 43);
 write(1, "Hello, Wooooooooooooooooooooooooooooooooooooooooooooooooorld!\n", 73);

 return 0;
}
```

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/syscall.h>
#include <linux/seccomp.h>
#include <seccomp.h>

int main()
{
 scmp_filter_ctx ctx = seccomp_init(SCMP_ACT_KILL);

 seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(exit), 0);
 seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(exit_group), 0);
 seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(brk), 0);
 seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(write), 2,
   SCMP_A0(SCMP_CMP_EQ, 1),
   SCMP_A1(SCMP_CMP_EQ, (long) "Hello, World!\n")
 );

 seccomp_load(ctx);
 seccomp_release(ctx);

 write(1, "Hello, World!\n", 14);
 write(1, "Hello, Wooooooooooooooooooooooooorld!\n", 43);
 write(1, "Hello, Wooooooooooooooooooooooooooooooooooooooooooooooooorld!\n", 73);

 return 0;
}
```

PRIMERI

LANDLOCK

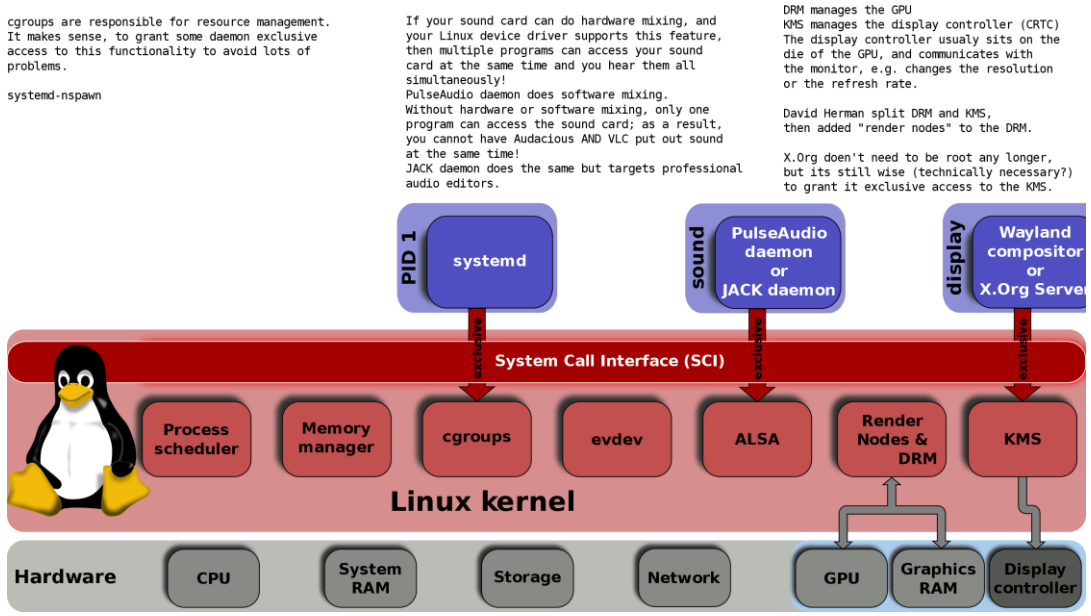
```c
#ifndef O_PATH
#define O_PATH 010000000
#endif

const char* path = "/tmp";

int main(int argc, char* argv[])
{
 if (argc < 2)
 {
  printf("Usage: %s <file>\n", argv[0]);
  return 1;
 }

 // General ruleset
 struct landlock_ruleset_attr ruleset_attr = {
  .handled_access_fs =
   LANDLOCK_ACCESS_FS_EXECUTE |
   LANDLOCK_ACCESS_FS_WRITE_FILE |
   LANDLOCK_ACCESS_FS_READ_FILE |
   LANDLOCK_ACCESS_FS_READ_DIR |
   LANDLOCK_ACCESS_FS_REMOVE_DIR |
   LANDLOCK_ACCESS_FS_REMOVE_FILE |
   LANDLOCK_ACCESS_FS_MAKE_CHAR |
   LANDLOCK_ACCESS_FS_MAKE_DIR |
   LANDLOCK_ACCESS_FS_MAKE_REG |
   LANDLOCK_ACCESS_FS_MAKE_SOCK |
   LANDLOCK_ACCESS_FS_MAKE_FIFO |
   LANDLOCK_ACCESS_FS_MAKE_BLOCK |
   LANDLOCK_ACCESS_FS_MAKE_SYM,
 };

 int ruleset =
syscall(SYS_landlock_create_ruleset,
&ruleset_attr, sizeof(ruleset_attr), 0);

 // /tmp read only ruleset
 struct landlock_path_beneath_attr attr = {
  .allowed_access =
   LANDLOCK_ACCESS_FS_READ_FILE,
  .parent_fd = open(path, O_PATH | O_CLOEXEC)
 };

 syscall(SYS_landlock_add_rule, ruleset,
LANDLOCK_RULE_PATH_BENEATH, &attr, 0);

 // Apply landlock
 prctl(PR_SET_NO_NEW_PRIVS, 1, 0, 0, 0);
 syscall(SYS_landlock_restrict_self, ruleset, 0);
 close(ruleset);

 // Read file and write to output
 FILE* file = fopen(argv[1], "r");
 char buffer[64];
 fread(buffer, 1, 64, file);
 fclose(file);

 printf("%s\n", buffer);
 return 0;
}
```

VIRTUALIZACIJA

CONTROLGROUP IN NAMESPACE

Virtualizacija

- Control group (cgroup)
  - Omejevanje porabe sistemskih virov
    - CPE
    - Pomnilnik
    - Datotečni sistem
- Namespace
  - Virtualizacija sistemskih virov
    - Omrežje
    - Priklopne točke (/dev, /sys, ...)



https://en.wikipedia.org/wiki/Cgroups

PRIMERI

CONTROLGROUP in NAMESPACE

```bash
#!/bin/bash
CPU=${1:-50}
set -xe
systemd-run --scope -p CPUQuota="${CPU}%" --user dd if=/dev/random of=/dev/null
status=progress
```

```
andraz@seth ~/varprog/intro/def_2 % ls /sys/fs/cgroup/user.slice/user-1000.slice/user@1000.service/app.s
lice/run-r20c527d59cd24ac79ecdb25717679286.scope/
cgroup.controllers      cpu.max          memory.events.local  memory.swap.max
cgroup.events           cpu.max.burst    memory.high          memory.swap.peak
cgroup.freeze           cpu.pressure     memory.low           memory.zswap.current
cgroup.kill             cpu.stat         memory.max           memory.zswap.max
cgroup.max.depth        cpu.stat.local   memory.min           memory.zswap.writeback
cgroup.max.descendants  cpu.uclamp.max   memory.oom.group     memory.numa_stat
cgroup.pressure         cpu.uclamp.min   memory.peak          pids.current
cgroup.procs            cpu.weight       memory.pressure      pids.events
cgroup.stat             cpu.weight.nice  memory.reclaim       pids.max
cgroup.threads          io.pressure      memory.stat          pids.peak
cgroup.type             irq.pressure     memory.swap.current
cgroup.subtree_control  memory.current   memory.swap.events
cpu.idle                memory.events    memory.swap.high
```

Kontejnerji

- Združimo vse do zdaj
  - Nastavitev uporabniških pravic v kontejnerju
  - Seccomp konfiguracija za izvajanje programov
  - Cgroup konfiguracija za omejitev porabe virov
  - Namespace konfiguracija za izolacijo datotečnega sistema in omrežja
- Docker/Podman

https://medium.com/@sushantkapare1717/podman-vs-docker-f7ae59b46ef1

PRIMERI

DOCKER

HVALA

Vaje

- docker