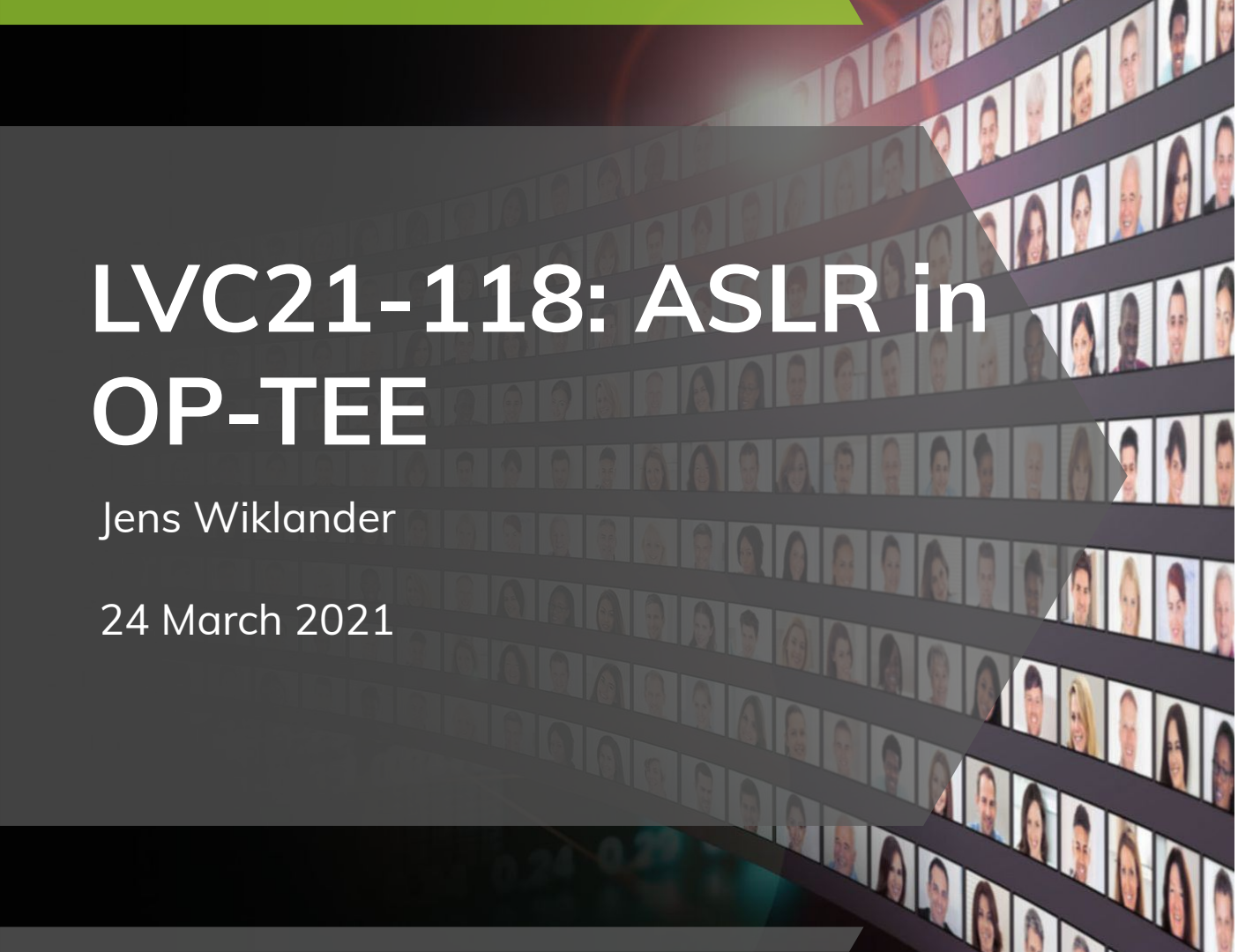


LVC21-118: ASLR in OP-TEE

Jens Wiklander

24 March 2021

Linaro Connect
VIRTUAL • SPRING 2021

Agenda

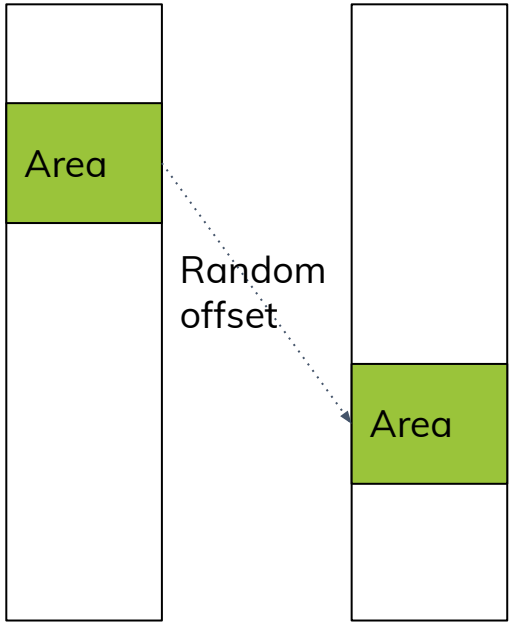
- What is ASLR?
- ASLR for OP-TEE Core
- ASLR for Trusted Applications, TAs

What is ASLR?

Address space layout randomization (ASLR) is a security technique that randomizes the virtual addresses when running code

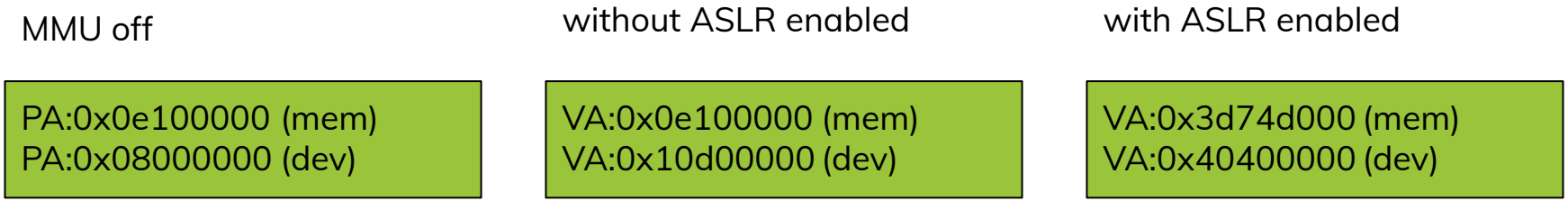
ASLR has been supported since OP-TEE version 3.8.0

The main objective is to make it harder to exploit a memory corruption vulnerability



ASLR for OP-TEE Core - overview

- The binary is linked against the same address as the physical load address
- The binary is loaded at a designated physical address in memory
- The memory is mapped at a random virtual base address



Random seed

- The random seed is used to select a random base address
- It can be retrieved from “/secure-chosen/kaslr-seed” from DT if provided
- Alternatively it can be provided by platform specific means
- Providing a seed with the value 0 disables ASLR for OP-TEE Core

```
/*
 * get_aslr_seed() - return a random
 *                   seed for core ASLR
 * @fdt: Pointer to a device tree if
 *       CFG_DT_ADDR=y
 *
 * This function has a __weak default
 * implementation.
 */
unsigned long get_aslr_seed(void *fdt);
```

Building

- Binary linked as a position independent binary to prepare for relocation
- Only relative relocations (R_ARM_RELATIVE or R_AARCH64_RELATIVE) are accepted
- The relocation entries are parsed and reduced to a list of addresses added at the end of the binary
  - Simplifies assembly code
  - Gives a build time test against unexpected relocation types

Compare:

```
typedef struct {
    Elf64_Addr    r_offset;
    Elf64_Xword   r_info;
    Elf64_Sxword  r_addend;
} Elf64_Rela;
```

with

```
uint32_t rel
```

Relocate

- Binary still linked against load address
- The loaded binary must be relocated to be able to execute from a new virtual address
- Relocation is in principle as simple as:

```
uint32_t *rel = address_of_relocations;
uintptr_t load_offset = new_address - link_address;
for (n = 0; n < rel_count; n++) {
        uintptr_t *p = (uintptr_t *)(rel[n] + load_offset);
        (*p) += load_offset;
}
```

Mappings

- A mapping is created based on the seed
- One identity mapped section is also added
- Identity mapped memory is only needed while enabling the MMU, but kept for simplicity

Example:

MMU off

PA:0x0e100000 (mem)
PA:0x08000000 (dev)

without ASLR enabled

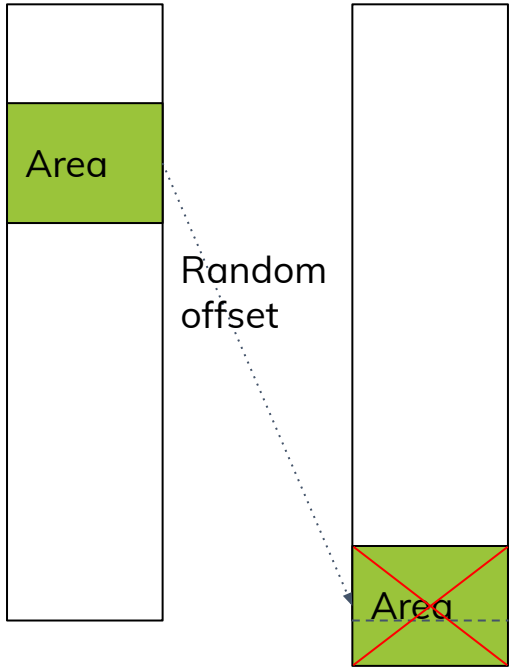
VA:0x0e100000 (mem)
VA:0x10d00000 (dev)

with ASLR enabled

VA:0x0e100000 (rx idmap)
VA:0x3d74d000 (mem)
VA:0x40400000 (dev)

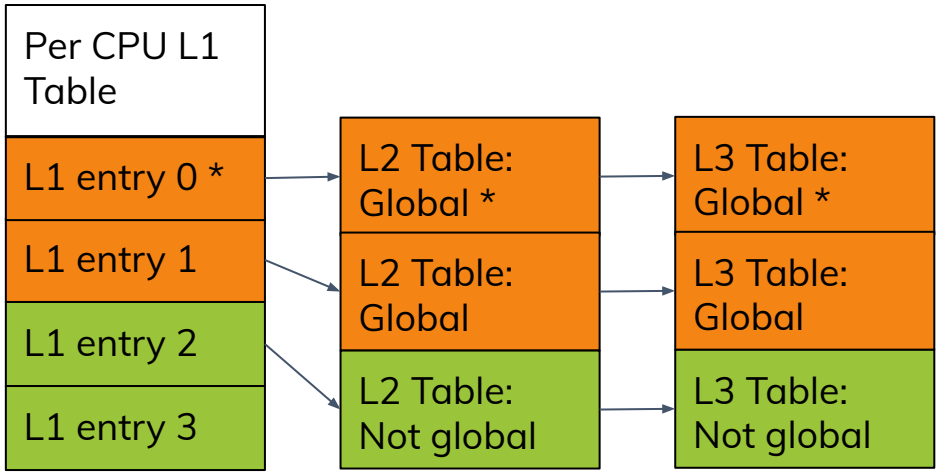
Possible locations

- Virtual position is selected as a multiple of 4KiB
- The entire virtual address range can be used
- This gives almost 20 random bits in the virtual address for a 4GiB virtual memory range
- If a generated position cannot be used a new position is generated instead of shifting
    - Avoids making some of the possible addresses more likely



Translation tables - long descriptor format

- Each CPU is assigned a unique L1 table
- Privileged mode mappings are global and share translation tables
- User mode mappings are not global and may differ from core to core
- Global mappings are established before non-global
- In this example are the two entries 0 and 1 the same in all L1 tables



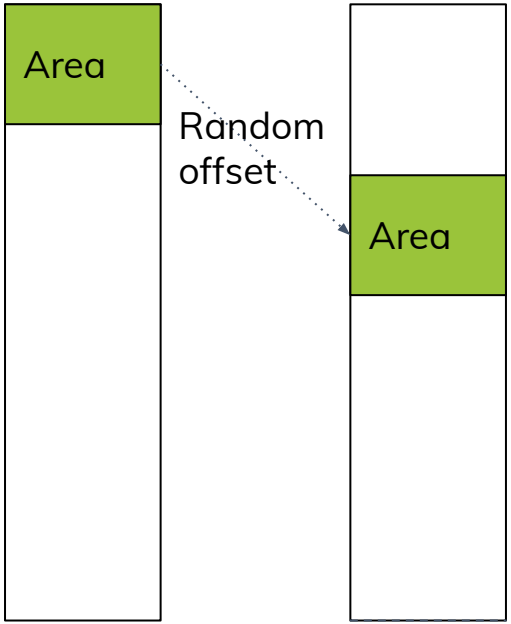
* identity map

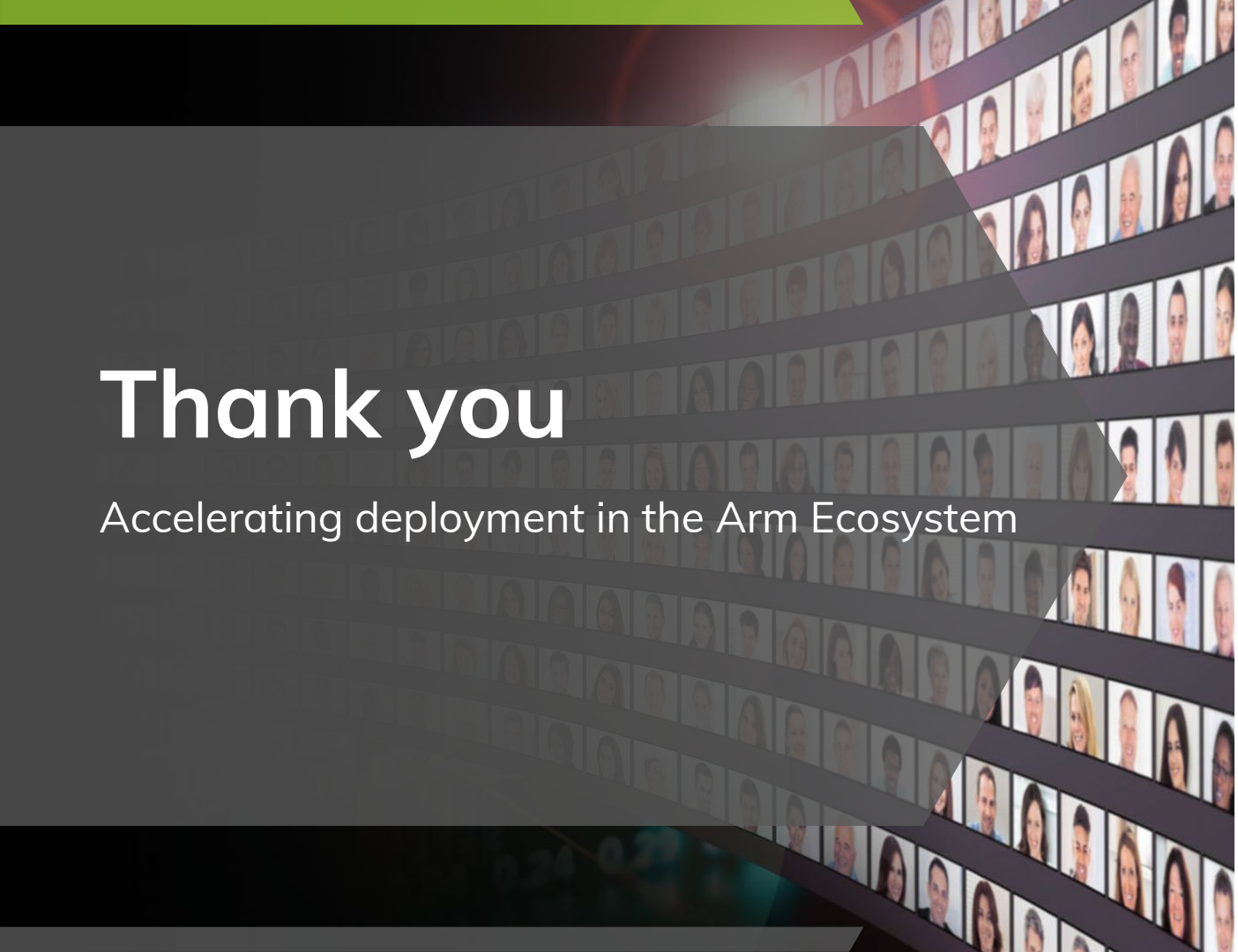
TA loading

- TAs have always been compiled and linked as position independent executables
- OP-TEE Core selects a suitable virtual memory range for TAs at boot time
  - Depending on present memory map used by OP-TEE Core
- TAs are linked against address 0
- TAs are loaded with ldelf, the ELF loader
- Ldelf relocates the TA to the address where it has been loaded before being entered for the first time
- Ldelf shares address space with the TA

ASLR for TAs

- Address randomization is handled by ldelf
- OP-TEE Core provides APIs
- The load address is randomized for each ELF being loaded
- Effective address range is limited by the fact that translation tables are assigned to holes in the mapping, room for improvement





Thank you

Accelerating deployment in the Arm Ecosystem

Linaro Connect
VIRTUAL • SPRING 2021