AMD TrenchBoot support in GRBU2

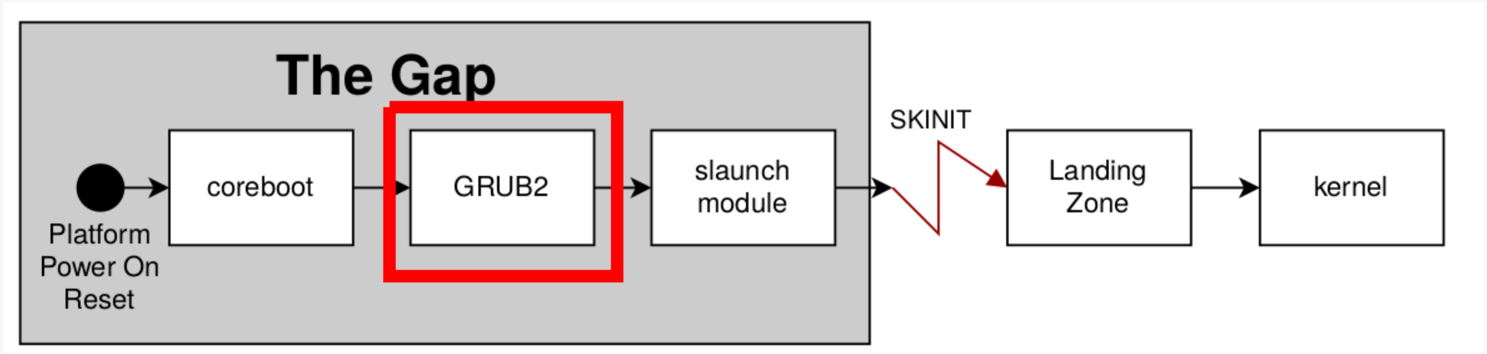
GRUB2 and 3mdeb ”minisummit” 2019

Piotr Król

Agenda

- What is this talk about
- Test environment assumptions
- Bootloader goals in TrenchBoot
- How to build recent code
- GRUB2 changes
- Q&A

What is this talk about



- GRUB2 modification to enable TrenchBoot
- Things to improve and road to mainlining

Test environment assumptions

- PC Engines apu2 as reference platform
- coreboot as firmware
- GRUB2 coreboot payload placed in SPI
- Since we don't have modern AMD platform suitable for testing yet and there is no sponsorship of UEFI work we delay it for now

Bootloader goals in TrenchBoot

- Load TB-capable Linux kernel
- Initialize secure launcher
- Load Landing Zone (aka Secure Loader)
- Boot

Load TB-capable Linux kernel

- kernel can be loaded from any GRUB2 supported source
- we assume use of cbfs at this point
- OS storage validation would be added in future
- no modifications to loading code in case of TB for AMD

Initialize secure launcher

- currently `slaunch [skinit|txt]`
- `slaunch` checks if we run on BSP and what CPU we use, it will throw error if we try `skinit` on Intel etc.
- if everything is fine correct function is set

Load Landing Zone (LZ) from cbfs

- LZ can be loaded from any GRUB2 supported source
- we assume use of cbfs at this point
- OS storage validation would be added in future
- `slaunch_module` takes LZ file location as required parameter
- there are some checks made for LZ (type, size)
- if relocator at this point doesn't exist it is created
- allocate LZ size memory with `grub_relocator_alloc_chunk_align` at `0x2000000`, 64k aligned, `GRUB_RELOCATOR_PREFERENCE_NONE`
- get virtual and physical address of allocated memory and feed `grub_slaunch_module` struct
- use newly created structure as first element of slaunch module list
- read LZ file to allocated memory

Boot

- at the end of `grub_linux_boot`, instead of `grub_relocator32_boot`, if `grub_slaunch_func` was set by `slaunch` command `slparams` structure is filled and `grub_slaunch_func` with params and relocator pointers is called
- `grub_slaunch_func` which for AMD would be `grub_slaunch_boot_skinit`
- in `grub_slaunch_func` set pointer to Linux kernel params to `GRUB_SL_ZEROPAGE_OFFSET/4` offset
- get slaunch modules list set during `slaunch_module`
- set all AP in INIT state by writing directly to APIC
- initialize TIS
- close active and requested localities
- set registers according to TrenchBoot spec (EAX: slb, ESP: real_mode_target (only for debug), EIP: skinit function)
- call `grub_relocator32_boot`

How to build recent code

```
git clone https://github.com/pcengines/coreboot.git -b pcengines_trenchboot_v4.10.0.2
git clone https://github.com/pcengines/pce-fw-builder.git -b custom_sdk_ver_support
cd coreboot
git submodule update --init --checkout
cd ../pce-fw-builder
SDK_VER=psec2019 ./build.sh dev-build ../coreboot apu2.tb
```

- Above procedure pulls various repositories from forks that contain required patches
- We should gradually get rid of forks and make transition to mainline repositories
- Above procedure is just about coreboot, further components like Xen, dom0 and VMs have to be prepared using Yocto

coreboot image built-in grub.cfg

```
linux path/to/bzImage (...)
slaunch skinit
slaunch_module (cbfsdisk)/lz
boot
```

GRUB2 changes

- currently on `scratchpad_kh` branch
- after rebase `diffstat`:

```
grub-core/Makefile.am                     |   4 ++
grub-core/Makefile.core.def               |  11 ++++
grub-core/kern/tis.c                      | 279 +++++++++++++++++++++++++++++++++++(...)
grub-core/kern/tpm.c                      |  52 ++++++++++++++++
grub-core/loader/i386/linux.c             |  29 ++++++++--
grub-core/loader/i386/slaunch.c           | 240 +++++++++++++++++++++++++++++++++++(...)
grub-core/loader/i386/slaunch_skinit.c    |  71 ++++++++++++++++++++++
grub-core/loader/i386/slaunch_txt.c       |  37 ++++++++++++
include/grub/file.h                       |   3 +
include/grub/i386/cpuid.h                 |  13 ++++
include/grub/i386/mmio.h                  | 105 ++++++++++++++++++++++++++++++++
include/grub/i386/msr.h                   |  82 +++++++++++++++++++++++++
include/grub/i386/txt.h                   | 475 +++++++++++++++++++++++++++++++++++(...)
include/grub/slaunch.h                    |  62 +++++++++++++++++++
include/grub/tis.h                        | 156 +++++++++++++++++++++++++++++++++++(...)
15 files changed, 1615 insertions(+), 4 deletions(-)
```

Modifications description

- **`grub-core/Makefile.am`**
    - Ross:
        - add tis header
        - add slaunch
- **`grub-core/Makefile.core.def`**
    - Ross:
        - add tis and tpm
        - add slauch module
    - Krystian:
        - add tis usage to slaunch
- **`grub-core/kern/tis.c`**
    - Ross:
        - `grub_{read,write}{8,32}`, `grub_io_dalay`, `grub_bust_wait`
        - `grub_tis_{init, send, recv_data, recv, request_locality}`

Modifications description

- **`grub-core/kern/tpm.c`**
    - Ross:
        - grub_tpm_pce_extend
- **`grub-core/loader/i386/linux.c`**
    - Ross:
        - grub_linux_slaunch_set: set slaunch function
        - call slaunch function
    - Krystian:
        - copy the command line to final address (not used anymore since relocator solves that)
        - add relocator as param to `*grub_slaunch_func` and `grub_linux_slaunch_set` and pass it to `grub_slaunch_func`

Modifications description

- **`grub-core/loader/i386/slaunch.c`**
  - Ross:
    - `grub_slaunch_{get_modules, add_module, free }`
    - `grub_cmd_slaunch{,_module}`
    - `GRUB_MOD_{INIT,FINI}`
  - Piotr:
    - allocate memory `grub_relocator_alloc_chunk_align`
  - Krystian:
    - address alignment
    - relocate kernel to 0x2000000 because of DEV (DEV related assumptions are no longer valid, we have to use IOMMU)

Modifications description

- **`grub-core/loader/i386/slaunch_skinit.c`**
    - Ross:
        - grub_slaunch_boot_skinits placeholder
    - Piotr:
        - call skinit in asm
    - Krystian:
        - improve skinit calling code
        - allocate_zeropage using `grub_relocator_alloc_chunk_align`
        - implement skinit function with debugging log in asm
        - use grub_relocator32_boot
        - remove allocate_zeropage and use fixed GRUB_SL_ZEROPAGE_OFFSET
        - close active and requested localities `grub_tis_request_locality(0xff)`
        - put all AP in INIT state

Modifications description

- **`grub-core/loader/i386/slaunch_txt.c`**
    - Ross:
        - `grub_slaunch_boot_txt` placeholder
    - Krystian:
        - add relocator as param
- **`include/grub/file.h`**
    - Ross:
        - add `GRUB_FILE_TYPE_SLAUNCH_MODULE`
- **`include/grub/i386/cpuid.h`**
    - Ross:
        - Intel and AMD CPUID defines
- **`include/grub/i386/mmio.h`**
    - Ross:
        - `grub_{read,write}{b,w,l,q}`

Modifications description

- **`include/grub/i386/msr.h`**
  - Ross:
    - defines for general and AMD sepcific MSRs
    - `grub_{rdmsr,wrmsr}`
- **`include/grub/i386/txt.h`**
  - Ross:
    - TXT related defines and structures
- **`include/grub/slaunch.h`**
  - Ross:
    - struct: `grub_slanuch_{info,params}`
    - func definition: `grub_slaunch_boot_{txt,skinit}`
    - struct: `grub_slaunch_module`
  - Krystian:
    - add relocator as param
- **`include/grub/tis.h`**
  - Ross:
    - defines and tpm structures

Known issues

- we should not hardcode values e.g. `0x2000000`
- KH: pointer to kernel params should be at the end of SLB and not measured by SKINIT since pointer always can change for various reasons
- `skinit` function should be cleaned
- ESP setup before `skinit` is just for debugging purposes
- after allocating space:

```
err = grub_relocator_alloc_chunk_align (relocator, &ch,
          0x1000, 0x90000,
          0x1000, 0x1000,
          GRUB_RELOCATOR_PREFERENCE_LOW,
          0);
addr = (void *)get_virtual_current_address(ch);
```

- `memmove` throws exception with `addr`
- it looks that there is some additional magic, which looking at `grub_relocator32_boot`, we reused that approach and it seem to work ;P
- question is what should be correct^{TM} allocation
- some code is cryptic, have to be rewritten to human readable

Known issues

- `slaunch_module` probably doesn't implement correct checks on provided LZ, what measures we would like to apply (except implemented type check, `size!=0`)?
- Where we should setup IOMMU?

Q&A