Passing and retrieving the logs from the bootloader

Daniel Kiper

Oracle, Software Developer, GRUB upstream maintainer

Linux Plumbers Conference 2020, August 24-28, 2020

Safe harbor statement

Why do we need more information from the bootloader?

- We need the information how the platform was configured to start TrenchBoot
- ...but this can be useful in other use cases too, e.g. firmware or shim logs...

The bootloader log structure

Pseudocode

```
struct bootloader_log
{
  uint32_t version;
  uint32_t producer;
  uint32_t size;
  uint32_t next_off;
  bootloader_log_msg msgs[];
}

struct bootloader_log_msg
{
  uint32_t level;
  uint32_t facility;
  char type[];
  char msg[];
}
```

The bootloader log structure
Description

```
struct bootloader_log
{
  uint32_t version;
  uint32_t producer; or char producer[16];
  uint32_t size;
  uint32_t next_off;
  bootloader_log_msg msgs[];
}
```

- version: the bootloader log format version number, 1 for now,
- producer: the producer/bootloader type; we can steal some values from linux/Documentation/x86/boot.rst:type_of_loader,
- size: the total size of the log buffer including the bootloader_log struct,
- next_off: the offset in bytes, from start of the bootloader_log struct, of the next byte after the last log message in the msgs[]; i.e. the offset of the next available log message slot,
- msgs: the array of log messages,
- Should we add crc32 here?

The bootloader log structure

Description - Continuation

```
struct bootloader_log_msg
{
  uint32_t level;
  uint32_t facility;
  char type[];
  char msg[];
}
```

- level: similar to syslog meaning; can be used to differentiate normal messages from debug messages; the exact interpretation depends on the current producer/bootloader type specified in the bootloader_log.producer,
- facility: similar to syslog meaning; can be used to differentiate the sources of the messages, e.g. message produced by networking module; the exact interpretation depends on the current producer/bootloader type specified in the bootloader_log.producer,
- type: similar to the facility member but NUL terminated string instead of integer; this will be used by the GRUB2 for messages printed using grub_dprintf(),
- msg: the bootloader log message, NUL terminated string,
- There was also a proposal to add a timestamp here; probably it should be a delta since startup, like dmesg does, but maybe interpretation should depend on the type of the bootloader which produces it.

How the GRUB2 logging works

- The GRUB2 collects log messages into the temporary buffer, dynamically (re)allocated, via logging calls placed in the grub_*printf*() functions,
- Before passing the control to the Linux kernel, the bootloader log is copied to the final resting place (rellocator); since that moment the log cannot be updated,
- The GRUB2 logging is controlled via `grub_log`, `grub_log_debug` and `grub_log_debug_fl` environment variables.

Linux kernel boot_params/zero_page for legacy BIOS and TrenchBoot

```
diff --git a/arch/x86/include/uapi/asm/bootparam.h b/arch/x86/include/uapi/asm/bootparam.h
index 13093c7..278c947 100644
--- a/arch/x86/include/uapi/asm/bootparam.h
+++ b/arch/x86/include/uapi/asm/bootparam.h
@@ -142,7 +142,9 @@ struct boot_params {
        __u32 ext_ramdisk_image;                                /* 0x0c0 */
        __u32 ext_ramdisk_size;                                 /* 0x0c4 */
        __u32 ext_cmd_line_ptr;                                 /* 0x0c8 */
-       __u8  _pad4[116];                                       /* 0x0cc */
+       __u64 bootloader_log_addr;                              /* 0x0cc */
+       __u32 bootloader_log_size;                              /* 0x0d4 */
+       __u8  _pad4[104];                                       /* 0x0d8 */
        struct edid_info edid_info;                             /* 0x140 */
        struct efi_info efi_info;                               /* 0x1c0 */
        __u32 alt_mem_k;                                        /* 0x1e0 */
```

Linux kernel bootloader log info for UEFI platforms

- Bootloader → `EFI_BOOT_SERVICES.InstallConfigurationTable(EFI_GUID *bl_guid, void *data)`
- The Linux kernel looks for `bl_guid` in the list of configuration tables.
- The kernel exposes the bootloader log through `/sys/firmware/efi/bootlogs/<producer>` (the path can be different for non-UEFI platforms, e.g. `/sys/kernel/boot_params/bootloader_log`).
- User space tools parse log as needed.

Code - WIP

- Most of the initial implementation work was done by Oracle intern Alec Brown
- The initial bootloader log specification:
  - https://lkml.org/lkml/2020/5/29/428
- The GRUB2 patch:
  - https://github.com/rossphilipson/travail/blob/master/misc/grub-booloader-log-support.patch
- The Linux kernel patch:
  - https://github.com/rossphilipson/travail/blob/master/misc/0001-Bootloader-log-support.patch
- sl-stat – the booloader log parser:
  - https://github.com/TrenchBoot/sltools/tree/master/sl-stat

Discussion

- Is this feature interesting for you?
  - Yes or No
- Is there any chance to use it in other cases?
  - Yes or no

Whiteboard

Thank You

Daniel Kiper

daniel.kiper@oracle.com