

デバッグ道中記

@elliptic_shiho

自己紹介

- ・ 緑川志穂 (@elliptic_shiho)
- ・ ラボユースは5期～6期
- ・ メインは暗号技術～セキュリティ全般
- ・ ハードウェア人間でもある
 - ・ 初めてはんだごて握って20年経過
 - ・ 何ならソフトウェアより長い

macOS

- 微妙にLinuxと違う部分のせいで自作OS用ツールチェインビルダがぶっ壊れた
- 合宿中はRustを書くつもりがCとPatchfileとシェルスクリプトとflake.nixを書くことに費やされていた
- 最終的には解消したのでデバッグ時に出たあれこれのご紹介

macOS

- もともとはLinux用
 - 公式配布されているツール群を寄せ集めて一元管理できるようにしていた
- だいぶメンテナンスされていなかったらしく, macOS系のコードはかなり壊れていた

バグ

- Elf32_SymはあるのにElf64_Symがない
 - そのくせ64bit分岐は存在する
 - 対応漏れ?
- endian.hがない
 - 代用品があったのでそれを利用して対処
- strdupa(3), get_current_dir_name(3)あたりの_GNU_SOURCE系関数
 - gccにしてもうまくいかず, それぞれstrdup(3), getcwd(3)に書き換え

バグ

- コンパイルは通るけどSEGVするバグがあった
- とりあえずメモリアドレス表示は仕込めたので確認

```
Wed Aug 27 18:00:17 JST 2025 ~/prog/repo/CrinoidOS/milkv-mars-image-boilerplate/u-boot
(Milk-V Building Shell) > make
mkdir -p spl/dts/
  FDTGREP spl/dts/dt-spl.dtb
  DTB      spl/dts/dt-spl.dtb.S
  AS       spl/dts/dt-spl.dtb.o
  AR       spl/dts/built-in.o
  LD       spl/u-boot-spl
/nix/store/l8a7zq115dkdz5xnfxisv8z7797iasq2-riscv64-linux-binutils-2.44/bin/riscv64-linu
  OBJCOPY  spl/u-boot-spl-nodtb.bin
  COPY     spl/u-boot-spl.bin
  SYM      spl/u-boot-spl.sym
  MKIMAGE  u-boot.img
start = 0x10006a128
end = 0x10006a1f8
curr = 0x10006a128
make: *** [Makefile:1428: u-boot.img] エラー 139
Wed Aug 27 18:00:34 JST 2025 ~/prog/repo/CrinoidOS/milkv-mars-image-boilerplate/u-boot
(Milk-V Building Shell) > █
```


バグ

- コアダンプを採取してトレースをとる

```
(Milk-V Building Shell) > lldb
(lldb) target create tools/mkimage -c /cores/core.46103
Core file '/cores/core.46103' (arm64) was loaded.
(lldb) bt
* thread #1, stop reason = ESR_EC_DABORT_EL0 (fault address: 0x10006a128)
  * frame #0: 0x00000001021cb60c mkimage`imagetool_get_type(type=8) at imagetool.c:29:5
    frame #1: 0x00000001022053a8 mkimage`main(argc=25, argv=0x000000016dc413d8) at mkimage.c:357:12
    frame #2: 0x00000001a59bbf28 dyld`start + 2236
(lldb) f
frame #0: 0x00000001021cb60c mkimage`imagetool_get_type(type=8) at imagetool.c:29:5
  26
  27         for (curr = start; curr != end; curr++) {
  28             fprintf(stderr, "curr = %p\n", curr);
-> 29             fprintf(stderr, "(*curr) = %08x\n", *((unsigned int*) curr));
  30                 if ((*curr)->check_image_type) {
  31                     if (!(*curr)->check_image_type(type))
  32                         return *curr;
(lldb) █
```

バグ

- ・ セクションアドレス自体はLOADセグメントだからメモリ上にあるはず
- ・ なぜアクセスできないのか？

バグ

- image dump sections コマンドでメモリマップ確認

```
0x00000400 container [0x0000000102220000-0x00000001023c0000) rw- 0x00068000 0x00004000 0x00000000 mkimage.__DATA
0x00000009 data-ptrs [0x0000000102220000-0x0000000102220560) rw- 0x00068000 0x00000560 0x00000007 mkimage.__DATA.__la_symbol_ptr
0x0000000a data [0x0000000102220560-0x0000000102222128) rw- 0x00068560 0x00001bc8 0x00000000 mkimage.__DATA.__data
0x0000000b regular [0x0000000102222128-0x00000001022221f8) rw- 0x0006a128 0x000000d0 0x00000000 mkimage.__DATA.image_type
0x0000000c zero-fill [0x00000001022221f8-0x00000001023bcebc) rw- 0x00000000 0x00000000 0x00000001 mkimage.__DATA.__bss
```

バグ

- ・ アクセス先アドレス確認

```
(lldb) x/8i $pc-16
0x1021cb5fc: 0xb0000280    adrp    x0, 81
0x1021cb600: 0xf9400400    ldr     x0, [x0, #0x8]
0x1021cb604: 0xf9400002    ldr     x2, [x0]
0x1021cb608: 0xf9401be0    ldr     x0, [sp, #0x30]
-> 0x1021cb60c: 0xb9400000    ldr     w0, [x0]
0x1021cb610: 0xb90003e0    str     w0, [sp]
0x1021cb614: 0xf00001e0    adrp    x0, 63
0x1021cb618: 0x91234001    add     x1, x0, #0x8d0           ; "(*curr) = %08x\n"
(lldb) print/x $x0
(unsigned long) $3 = 0x000000010006a128
(lldb) █
```

バグ

- ・ アクセス先アドレス確認
 - ・ ズれている

```
(lldb) print/x $x0
(unsigned long) $3 = 0x000000010006a128
(lldb) █
```

0x00000400	container	[0x0000000102220000-0x00000001023c0000)	rw-	0x00068000	0x00004000	0x00000000	mkimage.__DATA
0x00000009	data-ptrs	[0x0000000102220000-0x0000000102220560)	rw-	0x00068000	0x00000560	0x00000007	mkimage.__DATA.__la_symbol_ptr
0x0000000a	data	[0x0000000102220560-0x0000000102222128)	rw-	0x00068560	0x00001bc8	0x00000000	mkimage.__DATA.__data
0x0000000b	regular	[0x0000000102222128-0x00000001022221f8)	rw-	0x0006a128	0x000000d0	0x00000000	mkimage.__DATA.image_type
0x0000000c	zero-fill	[0x00000001022221f8-0x00000001023bcebc)	rw-	0x00000000	0x00000000	0x00000001	mkimage.__DATA.__bss

バグ

- デバッガ経由で実行
 - ズれてない

```
(lldb) target create tools/mkimage
Current executable set to '/Users/.../prog/repo/Crinoid0S/milkv-mars-image-bo
(lldb) run -f auto -A riscv -T firmware -C none -O u-boot -a 0x40200000 -e 0x4020
t-nodtb.bin u-boot.img
Process 46196 launched: '/Users/.../prog/repo/Crinoid0S/milkv-mars-image-boi
start = 0x10006a128
end = 0x10006a1f8
curr = 0x10006a128
(*curr) = 00068858
curr = 0x10006a130
(*curr) = 000688b0
curr = 0x10006a138
(*curr) = 00068908
FIT description: Firmware image with one or more FDT blobs
Created: Tue Jan 1 09:00:00 1980
Image 0 (firmware-1)
Description: U-Boot 2021.10-dirty for visionfive2 board
Created: Tue Jan 1 09:00:00 1980
Type: Firmware
Compression: uncompressed
Data Size: 910112 Bytes = 888.78 KiB = 0.87 MiB
Architecture: RISC-V
OS: U-Boot
Load Address: 0x40200000
Hash algo: crc32
Hash value: 121fbaa2
```

ここで落ちずに処理が続く
→ 正しく参照できている

バグ

- アドレス取得部

```
by we need a call to get the respective section symbols /
#endif
#ifdef __MACH__
#include <mach-o/getsect.h>

#define INIT_SECTION(name) do { \
    unsigned long name##_len; \
    char *__cat(pstart_, name) = getsectdata("__DATA", \
        #name, &__cat(name, _len)); \
    char *__cat(pstop_, name) = __cat(pstart_, name) + \
        __cat(name, _len); \
    __cat(__start_, name) = (void *)__cat(pstart_, name); \
    __cat(__stop_, name) = (void *)__cat(pstop_, name); \
} while (0)
#define SECTION(name) __attribute__((section("__DATA, " #name)))

struct image_type_params **__start_image_type, **__stop_image_type;
#else
#define INIT_SECTION(name) /* no-op for ELF */
#define SECTION(name) __attribute__((section(#name)))
```

バグ

- Mach-O側のセクション情報を正しく読み取っている
- しかし実行時にアドレスが合わない
- デバッガだとうまくいく

バグ

- Mach-O側のセクション情報を正しく読み取っている
- しかし実行時にアドレスが合わない
- デバッガだとうまくいく
 - → ASLR!

バグ

- 解決

```
112 @@ -271,6 +271,7 @@ int rockchip_copy_image(int fd, struct image_tool_params *mparams);
113     * b) we need a API call to get the respective section symbols */
114     #if defined(__MACH__)
115     #include <mach-o/getsect.h>
116     +#include <mach-o/dyld.h>
117
118     #define INIT_SECTION(name) do { \
119         unsigned long name##_len; \
120 @@ -278,12 +279,12 @@ int rockchip_copy_image(int fd, struct image_tool_params *mparams);
121         #name, &__cat(name, _len)); \
122         char *__cat(pstop_, name) = __cat(pstart_, name) + \
123         __cat(name, _len); \
124 -     __cat(__start_, name) = (void *)__cat(pstart_, name); \
125 -     __cat(__stop_, name) = (void *)__cat(pstop_, name); \
126 +     __cat(__start_, name) = (void *)__cat(pstart_, name) + _dyld_get_image_vmaddr_slide(0); \
127 +     __cat(__stop_, name) = (void *)__cat(pstop_, name) + _dyld_get_image_vmaddr_slide(0); \
128     } while (0)
129     #define SECTION(name) __attribute__((section("__DATA, " #name)))
130
131 -struct image_type_params *__start_image_type, *__stop_image_type;
132 +extern struct image_type_params *__start_image_type, *__stop_image_type;
133     #else
```


バグ

- 解決

```
112 @@ -271,6 +271,7 @@ int rockchip_copy_image(int fd, struct image_tool_params *mparams);
113     * b) we need a API call to get the respective section symbols */
114     #if defined(__MACH__)
115     #include <mach-o/getsect.h>
116     +#include <mach-o/dyld.h>
117
118     #define INIT_SECTION(name) do { \
119         unsigned long name##_len; \
120 @@ -278,12 +279,12 @@ int rockchip_copy_image(int fd, struct image_tool_params *mparams);
121         #name, &__cat(name, _len)); \
122         char *__cat(pstop_, name) = __cat(pstart_, name) +
123         __cat(name, _len); \
124 -     __cat(__start_, name) = (void *)__cat(pstart_, name); \
125 -     __cat(__stop_, name) = (void *)__cat(pstop_, name); \
126 +     __cat(__start_, name) = (void *)__cat(pstart_, name) + _dyld_get_image_vmaddr_slide(0); \
127 +     __cat(__stop_, name) = (void *)__cat(pstop_, name) + _dyld_get_image_vmaddr_slide(0); \
128     } while (0)
129     #define SECTION(name) __attribute__((section("__DATA, " #name)))
130
131 -struct image_type_params *__start_image_type, *__stop_image_type;
132 +extern struct image_type_params *__start_image_type, *__stop_image_type;
133     #else
```

ASLRオフセットの加算

教訓

- Macのコアダンプ取得に手間取る
 - entitlements埋め込みあたり
 - ulimit -c unlimitedを忘れてしばらく困っていた
- Mach-Oの理解不足
 - 「セクションアドレス取得してるんだからASLR適用後だろう」という思い込み
- Nixで環境固定できるのが大変役立った