Write a SEP app for iOS

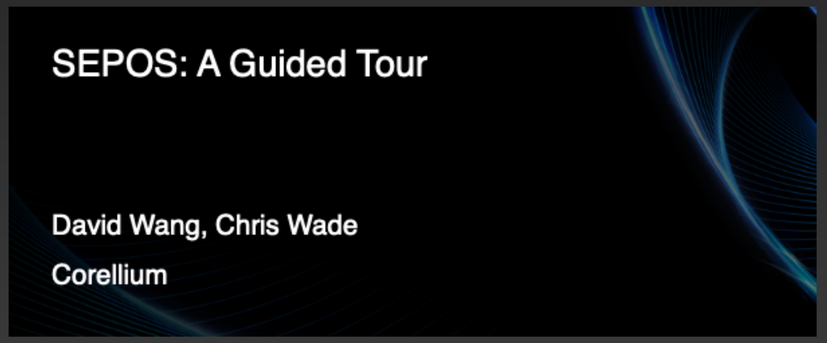
pattern-f

蚂蚁安全实验室
Ant Security Lab
蚂蚁光年
Ant Light-Year

Credit

- the content is based on
  - environment: A10, iOS 15.4
  - blackbird exploit
  - PongoOS
  - checkra1n, palera1n

- Thanks to Apple :)



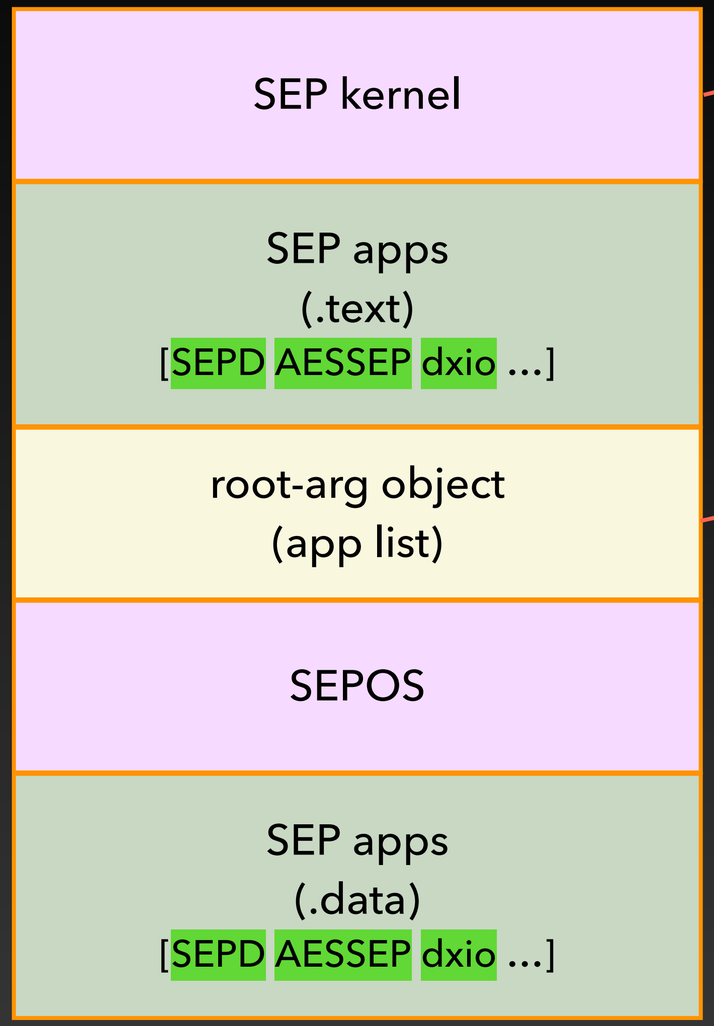
How to write a SEP app?

- cc test.c -o test
- ./test

```c
#include <stdio.h>
int main()
{
    printf("Hello, world.\n");
    return 0;
}
```

- Obviously this won't work for SEP
- SEP app is also a macho, but SEP has no file system

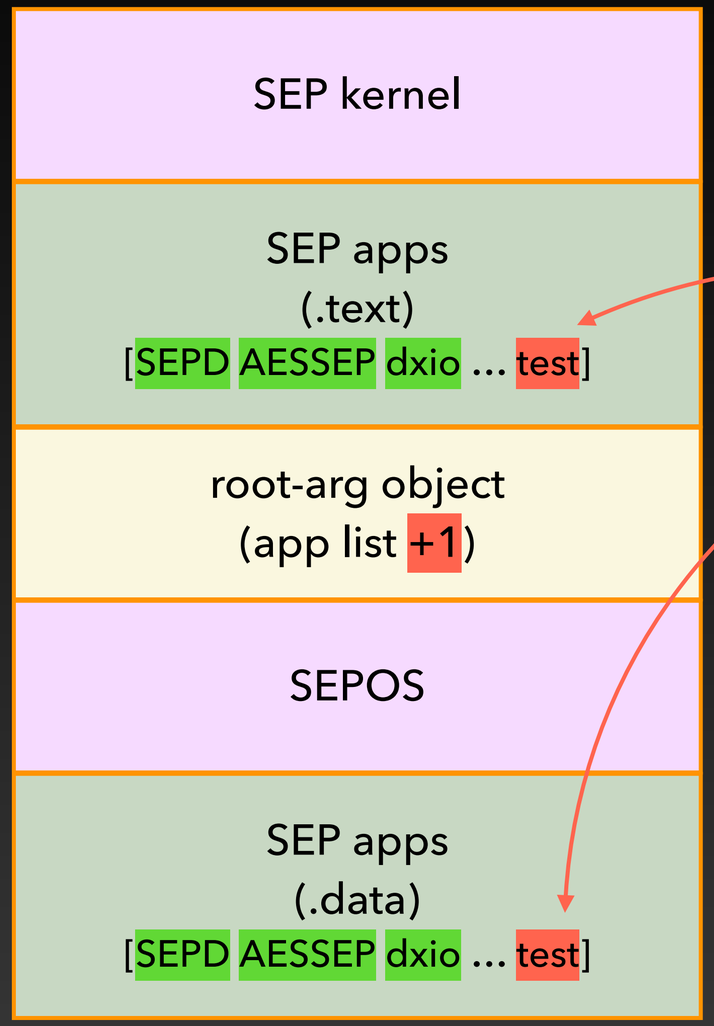
SEP fw structure (iOS 15)




```
00000800  02 00 00 00 00 00 00 f0  00 00 00 00 00 a0 08 00
00000810  00 90 25 00 70 9e 00 00  83 4e f1 ee cf 8c 3b fe
00000820  a8 5d b1 fa 60 dc 43 b6  00 00 00 00 00 00 00 00
00000830  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

```
SEPOS text 0x259000 data 0x261000 entry 0x009e70 base 0x007000
SEPD  text 0x09d000 data 0x278000 entry 0x00bcc0 base 0x008000
AESSE text 0x0a9000 data 0x280000 entry 0x00b2d0 base 0x008000
dxio  text 0x0b1000 data 0x283000 entry 0x00b5c0 base 0x008000
entit text 0x0b8000 data 0x285000 entry 0x019be0 base 0x008000
skg   text 0x0cd000 data 0x286000 entry 0x00c6b0 base 0x008000
```

Put macho into SEP fw




```
$ z sep32 -detail app-fixed
arm64? False cputype 0xc cpusubtype 0x9
----
          __TEXT ( 4) file: 0x00000000 - 0x00005000, vm: 0x0 - 0x5000
          __DATA ( 5) file: 0x00005000 - 0x00006000, vm: 0x5000 - 0x6000
LC_UNIXTHREAD        entry 0x1000
```

- macho must follow a certain format
  - 2 segments: TEXT & DATA (page aligned & page sized)
  - PIE flag
  - Mach-o header must in .text section
  - etc.
- SEPOS crc check patch, update offsets, etc.

Make a hole in SEP

- SEP is “totally” a black box
- no input, output
- Should find a way to check if my code was successfully executed
- SEP kernel: syscall: a panic can be triggered directly by userspace

```
  case 0x12:
    HIDWORD(v7) = tcb->regs_r1;
    if ( HIDWORD(v7) == 2 )
    {
      v127 = tcb->regs_r2;
      *(_DWORD *)(__mrc(15, 0, 13, 0, 4) + 428) = v127;
    }
    else if ( HIDWORD(v7) != 1 )
    {
      LODWORD(v7) = 4;
      if ( HIDWORD(v7) )
        goto LABEL_307;
      v110 = linux_eabi_syscall((int)&savedregs, "debug.c:24", v
    }
```

First line of code

```
_start:
    bl  _syscall_panic
    b   .

_syscall_panic:
    mov  r1, #0
    mov  r2, #0
    mov  r0, #0x12
    svc  0x12
    bx   lr
```

SEP app, v1

- SEP is ~~"totally" a black box~~ a gray box
- use syscall_panic as a boolean var
- panic = true

```
panic(cpu 1 caller 0xffffff00fc47980): SEP Panic: [elfour panic] [&&&&&&&&] debug.c:(*
Firmware type: UNKNOWN SEPOS
SEP state: 5
PM state: 2
Boot state: 32
Mailbox status:
IDLE_STATUS: 0x00000068
MAILBOX_SET: 0x00000110
MAILBOX_CLR: 0x00000110
INBOX_CTRL: 0x00024401
OUTBOX_CTRL: 0x00025501
```

A typical SEP panic

- When SEP detects an error: print backtrace
- idea: use this words to pass on information

```
panic(cpu 1 caller 0xffffff01615f980): SEP Panic: :INIT/BOOT: 0x0000eaf9 0x0000b7ef 0x0000c149

Panic app vers: 1869.100.98
Panic app UUID: 6DCC7EE3-6BD8-31ED-97AF-E8AF65BDC159
Root task tag:  AppleSEPOS-1869.100.98
Root task vers: AppleSEPOS-1869.100.98
Root task UUID: 6DCC7EE3-6BD8-31ED-97AF-E8AF65BDC159

Firmware type: UNKNOWN SEPOS
```

Next step

- panic is commonly used in SEP apps to check params, copy its code

SEPOS: System Methods

Class	Id	Method	Description
0	0	sepos_proc_getpid()	Get the process pid
0	2000	sepos_panic()	Panic the operating system

```
void sepos_panic(uintptr_t *words)
{
    int ret;
    L4_msg msg = {};

    build_L4_message(&msg, ert_rpc_bootstrap_server(), 0, 2000, 0);
    for (int i = 0; i < 8; i++) {
        msg.args[i] = words[i];
    }
    ret = L4_Call(&msg);        // svc 0x2
    if ( ret | msg.args[0] ) {
        syscall_panic();
    }
}
```

SEP app, v2

- SEP is a friendly gray box now
- I can use this to debug SEP memory easily
- PS: SEPOS only print valid backtrace, need patch SEPOS first

```
panic(cpu 0 caller 0xfffffff015703980): SEP Panic: :test/_pf_: 0x6c6c6548 0x77202c6f 0x646c726f 0x00000a2e

Panic app vers: 1869.100.98
Panic app UUID: 81CB0A25-0080-3EA5-BC98-66C8C70D035A
Root task tag:  AppleSEPOS-1869.100.98
Root task vers: AppleSEPOS-1869.100.98
Root task UUID: 6DCC7EE3-6BD8-31ED-97AF-E8AF65BDC159
```

```
>>> struct.pack('<4I', 0x6c6c6548, 0x77202c6f, 0x646c726f, 0x00000a2e)
b'Hello, world.\n'
```

End of the story?

- I want this

```c
#include <stdio.h>
int main()
{
    printf("Hello, world.\n");
    return 0;
}
```

- I noticed this

SEP Endpoints (1/2)

Index	Name	Driver
0	AppleSEPControl	AppleSEPManager.kext
1	AppleSEPLogger	AppleSEPManager.kext

AppleSEPLogger

- ap side, XNU kernel
- XNU supports output SEP log, but SEP doesn't implement EP1

```
void AppleSEPLogger::_logAction(AppleSEPLogger *this, void *a2, sep_log_message *sep_msg)
{
  if ( sep_msg->opcode == 0xB )
  {
    if ( sep_msg->log_pos >= this->ool_buffer_size ) {
      IOLog("AppleSEP:WARNING: bad buffer offset %u on EP1\n", sep_msg->log_pos);
    }
    else {
      if ( this->ool_buffer_pos != sep_msg->log_pos )
      {
        memcpy(&text[0], &this->ool_buffer[this->ool_buffer_pos]
            sep_msg->log_pos - this->ool_buffer_pos);
        IOLog("SEP: %s\n", text);       // redirect sep log to kernel log
        this->ool_buffer_pos = sep_msg->log_pos;
      }
```

Implement SEP log endpoint

- only SEPD can create ‘log ‘, patch it

```
dword_13788     DCD 'cntl'                  ; DATA XREF:
                                            ; sub_EAB4+2
                DCD 'SEPD'
                DCD 'log '
                DCD 'SEPD'   <-- test
                DCD 'arts'
                DCD 'ARTM'
```

- copy code from other SEP endpoints

```
driver_AKF = find_driver('AKF ', 0);

ool_mem_t ool_in  = { 0, 0 };
ool_mem_t ool_out = { 1, 4 }; // 4 pages for log buffer
ep_log = driver_create_endpoint(driver_AKF, 'log ', 1, &ool_in, &ool_out);

uintptr_t args[2] = { ep_log, 0 };
driver_io_control(driver_AKF, 0x412E, 2, args);
obj_handle_t ool_out_obj = args[1];

cout_buffer = sepos_object_map(bootstrap_server(), ool_out_obj, 6, &cout_size);
```

printf 🤩

- The trick is to write a mailbox message

```
int printf(const char *fmt, ...)
{
    if (cout_buffer == NULL) {
        sepos_panic("printf not intialized\n");
    }
    int len = vsprintf(cout_buffer + cout_pos, fmt, ap);
    cout_pos += len;

    sep_log_message mailbox_msg = {};
    mailbox_msg.opcode = 0x0b;
    mailbox_msg.log_pos = cout_pos;

    int ret = driver_write(driver_AKF, ep_log, &mailbox_msg, sizeof(mailbox_msg));
    if (ret != sizeof(mailbox_msg)) {
        sepos_panic("write ep_log failed\n");
    }
    return len;
}
```

SEP app, v3

- SEP is totally a white box now!

```
IOReturn AppleSEPBooter::bootSEP(AppleSEPFirmware *, bool, AppleSEPSharedMem
SEP:
SEP: Hello, SEP!
SEP:     - a SEP app built by pattern-f
SEP:
SEP: app_main 0x09c45a34 slide 0x09c43000
SEP: pid[ 0] KERN
SEP: pid[ 1] INIT
SEP:     tid[ 0] BOOT tid 0 flags 0x1
SEP: pid[ 2] SEPD
SEP:     tid[ 0] SEPD tid 0x10000 flags 0xc80001
SEP:     tid[ 1] intr tid 0x10001 flags 0xfa0000
SEP:     tid[ 2] XPRT tid 0x10002 flags 0xc80000
SEP:     tid[ 3] PMGR tid 0x10003 flags 0xc80000
SEP:     tid[ 4] AKF  tid 0x10004 flags 0xc80000
SEP:     tid[ 5] EP0D tid 0x10005 flags 0xc80000
SEP:     tid[ 6] TRNG tid 0x10006 flags 0xc80000
SEP:     tid[ 7] KEY  tid 0x10007 flags 0xc80000
SEP:     tid[ 8] shnd tid 0x10008 flags 0xc80000
SEP:     tid[ 9] ep0  tid 0x10009 flags 0xc80000
```

TODO

- SEP has an input method in EP0, actually we can interact with SEP apps. But we should implement it first.

Opcode	Name	Description
10	TTYIN	Write to SEP console

- other interesting things
- to be continued…