SYNACKTIV

Inside Apple Secure Enclave Processor in 2025

Hexacon 2025

whoami

SYNACKTIV

- **Quentin Salingue**
  - Security expert @ Synacktiv
- **Synacktiv**
  - Offensive security company
  - ~200 ninjas
  - We are hiring!

Agenda

SYNACKTIV

- Overview of the Secure Enclave Processor
- Pointer Authentication
- SEP patches
- Trusted Boot Monitor

Overview of the Secure Enclave Processor

Overview of the Secure Enclave Processor

SYNACKTIV

- **Cryptographic coprocessor since iPhone 5S (2013)**
  - Dedicated AES engine
  - Encrypted memory
  - Communication with the AP through a mailbox
- **Handles**
  - User data encryption
  - Biometrics (Unlocks and Apple Pay)
  - HDCP

Overview of the Secure Enclave Processor

SYNACKTIV

- **Public information**
  - Demystifying the Secure Enclave Processor (2016)
  - SEPOS: A Guided Tour (2018)
  - Attack Secure Boot of SEP (2020)
  - Apple platform security guide
- **Blackbird vulnerability in 2020**
  - Unpatchable vulnerability in SEPROM
  - Integer overflow leading to access to encrypted memory
  - Allows arbitrary firmware upload on A8 to A10

Overview of the Secure Enclave Processor

Getting a look at the code

SYNACKTIV

- **Leaked SEPROMs available on https://securerom.fun/**
- **SEPOS firmware in IPSW**
  - Still encrypted
  - Most keys are leaked: https://theapplewiki.com/wiki/Firmware_Keys/26.x
    - Thanks to people who leak keys <3
  - IDA loader: https://github.com/Proteas/sep-fw-dyld-cache-loader

Overview of the Secure Enclave Processor

Tooling

SYNACKTIV

- **Using the awesome `ipsw` by blacktop:**

```
$ ipsw extract -p sep -r https://updates.cdn-apple.com/2025SpringFCS/fullrestores/082-45632/B8A6EB88-60C3-4236-B8D8-D99CE1F98E34/iPhone16,2_18.5_22F76_Restore.ipsw
   • Extracting files matching pattern "sep"
        6.77 MiB / 6.77 MiB [=============================================================| ✅  ] 6.16 MiB/s
        2.82 KiB / 2.82 KiB [=============================================================| ✅  ]
        20.64 KiB / 20.64 KiB [===========================================================| ✅  ]
        1.13 KiB / 1.13 KiB [=============================================================| ✅  ]
      • Created 22F76__iPhone16,2/22F76__iPhone16,2/Firmware/all_flash/sep-firmware.d84.RELEASE.im4p
      • Created 22F76__iPhone16,2/22F76__iPhone16,2/Firmware/all_flash/sep-firmware.d84.RELEASE.im4p.plist
      • Created 22F76__iPhone16,2/22F76__iPhone16,2/Firmware/all_flash/sep-patches.d84.im4p
      • Created 22F76__iPhone16,2/22F76__iPhone16,2/Firmware/all_flash/sep-patches.d84.im4p.plist

$ ipsw img4 im4p extract 22F76__iPhone16,2/22F76__iPhone16,2/Firmware/all_flash/sep-patches.d84.im4p --iv bb434e0ac4acfb132ce4f24d6a3ae487
    --key 7514150492aa82602ae64cba9e7250dc84d88088fa191892463d5f1120891541
   • Decrypting Payload        path=22F76__iPhone16,2/22F76__iPhone16,2/Firmware/all_flash/sep-patches.d84.dec
```

Overview of the Secure Enclave Processor

Boot process

SYNACKTIV

- **SEP starts executing SEPROM code**
  - SEPROM is written once in factory
  - Read-only afterwards
- **AP sends signed SEP firmware in Image4 format**
  - DER-encoded format used by Apple for firmwares
  - Two main components
    - IM4P: firmware
    - IM4M: manifest (metadata + signature)

Overview of the Secure Enclave Processor

SYNACKTIV

Boot process

- **SEPROM validates the firmware**
  - RSA 4096 with hardcoded CA certificate
  - Checks manifest metadata matches current phone

```
  155:d=9  hl=2 l=  11 cons: SEQUENCE
  157:d=10 hl=2 l=   4 prim: IA5STRING         :CHIP
  163:d=10 hl=2 l=   3 prim: INTEGER           :8140
  211:d=9  hl=2 l=  15 cons: SEQUENCE
  213:d=10 hl=2 l=   4 prim: IA5STRING         :ECID
  219:d=10 hl=2 l=   7 prim: INTEGER           :123456789ABCDE
 7799:d=7  hl=2 l=   4 prim: IA5STRING         :sepi
 7805:d=7  hl=4 l= 444 cons: SET
 7809:d=8  hl=7 l=  58 cons: priv [ 1145525076 ]
 7816:d=9  hl=2 l=  56 cons: SEQUENCE
 7818:d=10 hl=2 l=   4 prim: IA5STRING         :DGST
 7824:d=10 hl=2 l=  48 prim: OCTET STRING      [HEX DUMP]:EEA317AD6643BEC10A60157B0ED2F3929888652EBAEF6193163DB674346C503FFBFE929934D22078959F5918E3EE2CB7
```

- **SEPROM loads the firmware in memory and jumps to it**

Overview of the Secure Enclave Processor

Kernel

SYNACKTIV

- **L4-based microkernel**
  - Handles hardware privileged operations
    - Memory mapping
    - Interrupt handling
    - Thread scheduling
- **Syscalls used to be easy to reverse**
  - 1 ID maps to 1 syscall
  - Now each object type has its own dispatch table

Overview of the Secure Enclave Processor

SYNACKTIV

Kernel objects

Type	Name
2	Physical address
3	CNode (Object storage)
4	Thread
5	Endpoint address (IPC)
6	Unknown
7	Notify
8	Interrupt
9	Debug
10	Unused

Type	Name
11	Process
12	Address space
13	Page Table Level 1
14	Page Table Level 2
15	Page Table Level 3
16	Process info
17	Control
18	Power management
19	Timer

Overview of the Secure Enclave Processor

SEPOS

SYNACKTIV

- **Userland "kernel" (SEPOS)**
- **Drivers to interact with peripherals**
  - PRNG, AES, …
- **Applications to interact with AP**
  - Keystore
- **Communications between apps using IPC**

Pointer Authentication

Pointer Authentication

Quick recap

SYNACKTIV

- **ARM instruction set extension**
  - Cryptographic signature of pointers
    - Stored in the high bits of the pointer
  - Prevents overwriting by an attacker
  - Introduced in A12 (2018)
- **2 set of keys: A and B**
  - Instruction (IA and IB)
  - Data (DA and DB)
  - General (GA)
- **Dedicated instructions to sign and use the pointers**
  - `PAC{I,D}{A,B}` : sign a pointer
  - `BLRA{A,B}` : verify pointer signature and branch

Pointer Authentication

Quick recap

SYNACKTIV

- **Signed pointer recipe:**
  - the pointer
  - an optional context
    - eg: where the pointer is stored
  - an optional diversifier
- **Example virtual function call:**

```
                                    ; X8 contains the vtable address
LDR             X9, [X8,#8]!        ; Load function pointer from vtable
MOV             X17, X8             ; Use vtable address as the context
MOVK            X17, #0xBBBF,LSL#48 ; Add 0xBBBF diversifier
BLRAA           X9, X17             ; Authenticate pointer using context and diversifier, branch
```

Pointer Authentication

SEPROM

SYNACKTIV

- **Only key IB is set on A14+ (2020)**
  - Used to sign return addresses
  - Compiler still emits BLRAA
    - Authentication is ignored
- **Key IA is set on A18 (2024)**
  - Used to sign function pointers
  - Uses chained fixups to provide signature information
  - Signing done in early boot
- **Keys are generated using the hardware RNG**

Pointer Authentication

SEPOS

SYNACKTIV

- **Kernel generates the keys for SEPOS**
  - Address of the fixup chain is hardcoded
- **SEPOS generates random keys for each process**
- **At startup, each app will:**
  - Look up the segment command `LC_SEP_CHAINED_FIXUPS` (`0x8000002`)
  - Iterate over the chain and signs the pointers
  - Look up the segment command `LC_SEP_SHLIB_CHAIN` (`0x8000001`)
    - Points to the header of the shared library
  - Apply the fixups for the shared library
  - Ask SEPOS to be marked read-only

Pointer Authentication

Chained fixups

SYNACKTIV

- **From the dyld sources:**

```
// DYLD_CHAINED_PTR_ARM64E
struct dyld_chained_ptr_arm64e_auth_rebase
{
    uint64_t    target    : 32,   // runtimeOffset
                diversity : 16,
                addrDiv   :  1,
                key       :  2,
                next      : 11,    // 4 or 8-byte stide
                bind      :  1,    // == 0
                auth      :  1;    // == 1
};
```

Pointer Authentication

Chained fixups example

- `0x8011BBBF000036EC`
  - `0x8011`
    - auth=1
    - bind=0
    - next=2
    - key=0 (IA)
    - addrDiv=1
  - `0xBBBF` PAC diversifier
  - `0x000036EC` offset from the base address

```
:00000004000AC038        DCQ 0x8011BBBF000036EC
:00000004000AC040        DCQ 0
:00000004000AC048        DCQ 0x8009D3A1000035DC
```

SEP Patches

SEP patches

SYNACKTIV

- **New firmware file introduced in A17 (iPhone 15 Pro)**
  - Also sent by AP
- **Allows for "patching" of the immutable ROM**
- **File format is no longer an IMG4**
  - Small header with offset/size to IM4P/IM4M
- **Validated like the firmware**
- **Loaded at a fixed address**

SEP patches

Patch usage

- **Used like a hook at various points of the boot process :**

```
      copyin_firmware(fw, v7, v6);
      dprintf_trace(0x20006LL);
      call_sep_patches_with_context(8LL);
      bzero(&v30, 0x240uLL);
      sep_img4_validate(a2, v32, v31, 2u, &v30);
      if ( v31 < v30.payload.length ) { panic(0x123); }
      call_sep_patches_with_context(9LL);
```

- **Context :**

```
struct sep_patches_ctx
{
  uint64_t field_0;
  void (__fastcall *dtrace)(uint64_t);
  uint64_t field_10;
  void (__fastcall __noreturn *panic)(uint64_t);
  uint32_t (__fastcall *get_chip_revision)();
  uint64_t (__fastcall *get_board_id)();
};
```

SEP patches

Patch code as of iOS 26.0

SYNACKTIV

- **Just a big switch-case with tracing logic:**

```
void __fastcall sub_4000(__int64 id, patches_ctx *ctx)
{
  switch ( (int)id )
  {
    /* more cases */
    case 8:
      if ( !ctx )
        patch_panic(id);
      ctx->dtrace(0x101011LL);
      break;
    case 9:
      if ( !ctx )
        patch_panic(id);
      ctx->dtrace(0x101013LL);
      break;
    /* more cases */
    default:
      if ( !ctx )
        patch_panic(id);
      ctx->panic(0x124);
      break;
  }
}
```

Trusted Boot Monitor

Trusted Boot Monitor

Version 1

SYNACKTIV

- **ARM coprocessor**
  - Added to A13+ devices (2019)
- **Monitors the execution of the SEPROM**
  - Marks SEPROM pages as executable (and read-only)
  - Resets SEP to start of the SEPROM
- **SEP can no longer jump to firmware directly:**
  - SEP sends the address of SEPOS to TBM
  - TBM marks pages as executable
  - TBM resets SEP to start of SEPOS

Trusted Boot Monitor

Version 2

SYNACKTIV

- **Version 2 added to A17+ devices (2023)**
  - Also added on the AP side for A16+ devices (2022)
- **Requires a manifest sent by the AP**
  - Found in the SHSH under `SEP-TBM`
- **Contains 2 structs:**
  - `ucer` : `uCertificate`
  - `ucon` : `uContainer`

Trusted Boot Monitor

ucer

SYNACKTIV

- **uCertificate:**
  - ECDSA P384 key used to sign `ucon`
  - Signed by an hardcoded key in the TBM
    - Same key for all SEP
    - AP key is different
  - Contains properties related to the platform:
    - `CHIP`, `SDOM`, `SCEP`

Trusted Boot Monitor

ucer

- Glitch-resistant booleans
  - TMM:
    - 0: `0x365A356C`
    - 1: `0xC9A5CA93`
  - AP fuses
    - 0: `0xA050C030`
    - 1: `0xA55AC33C`

Trusted Boot Monitor

ucon

SYNACKTIV

- **uContainer:**
  - Like IMG4 Manifest properties
  - Contains informations related a specific device and firmware version:
    - `CHIP`, `SDOM`, `SCEP`
    - `ECID`, `BORD`, `CPRO`, `snon`
    - hash of the firmware
    - hash of the IMG4 manifest
  - Signed with the `ucer` key

Trusted Boot Monitor

Boot process

SYNACKTIV

- **AP sends the TBM manifest (ucer+ucon) to SEPROM**
- **SEPROM sends it to TBM**
- **TBM checks:**
  - signatures of `ucer` and `ucon`
  - the properties of `ucer` and `ucon` match
    - `CHIP`, `SDOM`, `SCEP`

Trusted Boot Monitor

Boot process

SYNACKTIV

- **AP then sends patches**
- **SEPROM validates patches and sends them to TBM**
- **TBM checks:**
  - the properties of the TBM manifest against the fuses
  - hash of firmware is the same as in TBM manifest
  - hash of the IMG4 manifest is the same as in TBM manifest
- **TBM marks the patches as executable**

Conclusion

SYNACKTIV

- **Currently, SEP patches are a no-op**
- **Trusted Boot Monitor adds yet another layer of security**
  - Need a vulnerability in TBM to get code execution on SEP
- **Good example of defense in depth by Apple**
  - locking_apple_logo.gif

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://x.com/synacktiv

https://bsky.app/profile/synacktiv.com

https://synacktiv.com