iOS Inactivity Reboot

Jiska Classen

@naehrdine.bsky.social

Troopers

Rumors on Inactivity Reboot

- iPhones on iOS 18 will reboot, even when completely isolated from wireless networks.
- iPhones on iOS 18 will tell other iPhones on lower iOS versions to reboot – wirelessly!

Weird story?!
Let me check diffs in the latest release and prove it's fake!

ipsw diffs

Third hit: A feature called “Inactivity Reboot” 🤔

18_1_22B5045h__vs_18_1_22B5054e/MACHOS/keybagd.md · Markdown · main

```
52  + "inactivity_reboot"

54  …SSING)lu, activation_status: %!l(MISSING)lu, inactivity_reboot: %!l(MISSING)lu,
    hours_since_locked: %!l(MISSING)lu, ve…
```

Spoiler: While this match is part of the inactivity reboot feature, it has been introduced earlier. Here, Apple just adapted some diagnostics.

https://github.com/blacktop/ipsw-diffs – thank you for maintaning these 🤩

Before First Unlock

- User data is encrypted, with keys secured by the **Secure Enclave Processor** (SEP).
- Unlocking requires passcode and is rate limited.
- Reduced attack surface:
  No connection to Wi-Fis,
  no preview of contact information upon calls, no message previews,
  …

After First Unlock

- While iPhone is locked, selected encryption keys are temporarily erased (effaced), e.g., health data.
- Lots of data not effaced: Wi-Fi passcodes, caller previews, message previews, …
- Larger **Remote Code Execution** (RCE) attack surface through more available services.

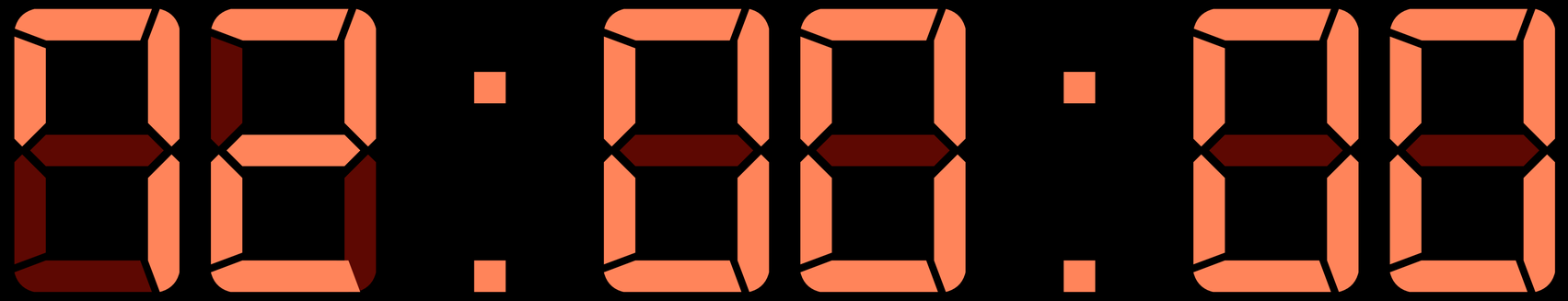
Remote Code Execution

Proximity-based Attacks

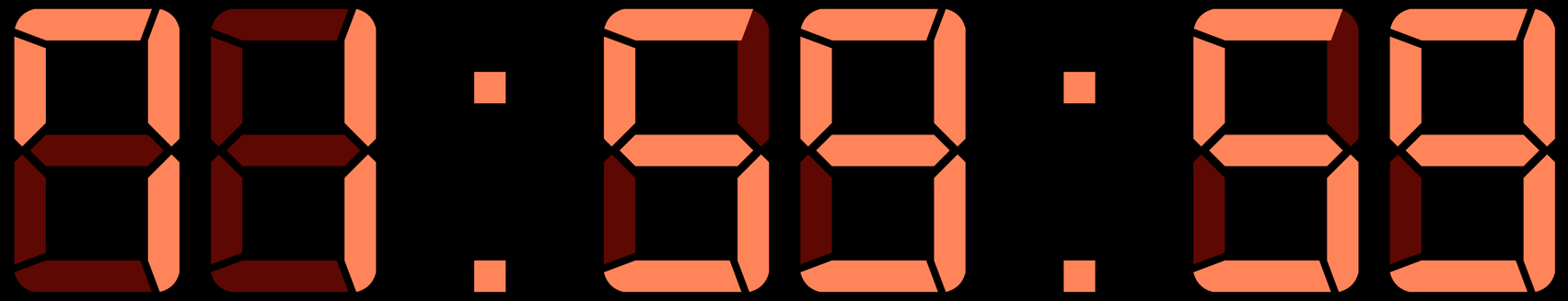
When to reboot your phone?

- It has been lost 🔍
- It has been stolen 😈
- It has been taken by the police 👮‍♀️
- Before crossing a border 🌍

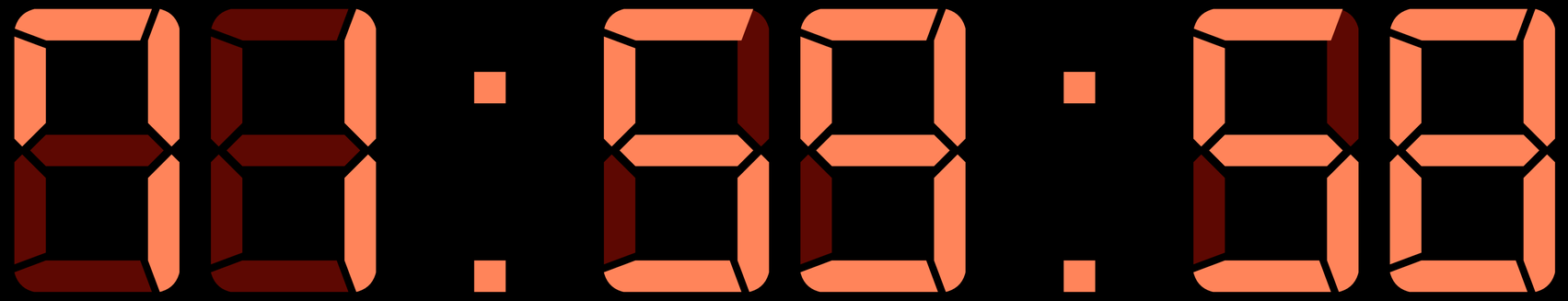
You're no longer able to reboot your phone 🤦‍♀️



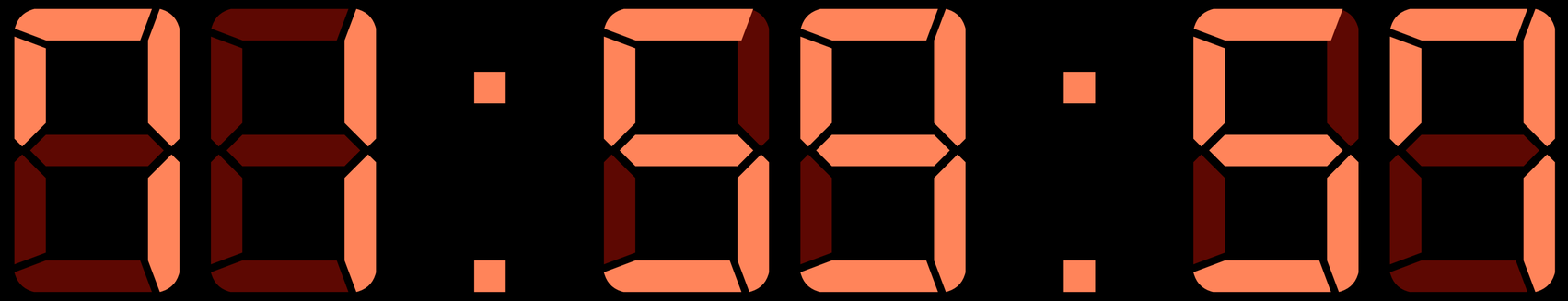
You have 72 hours to run
a lockscreen bypass!



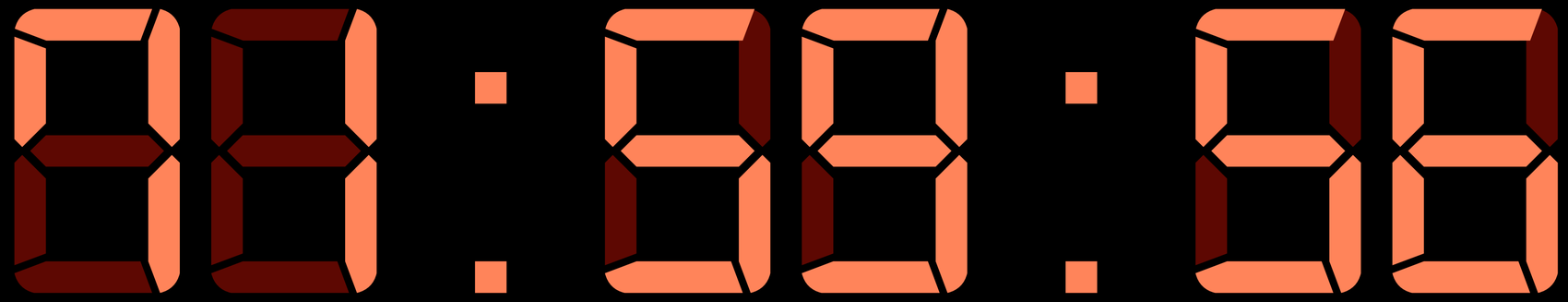
You have 72 hours to run
a lockscreen bypass!



You have 72 hours to run
a lockscreen bypass!



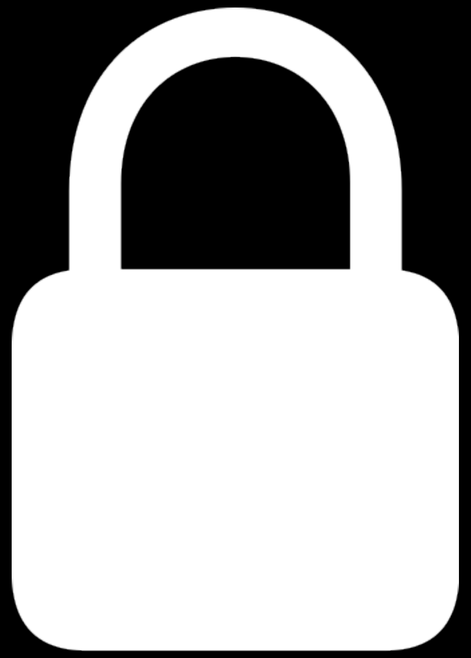
You have 72 hours to run
a lockscreen bypass!



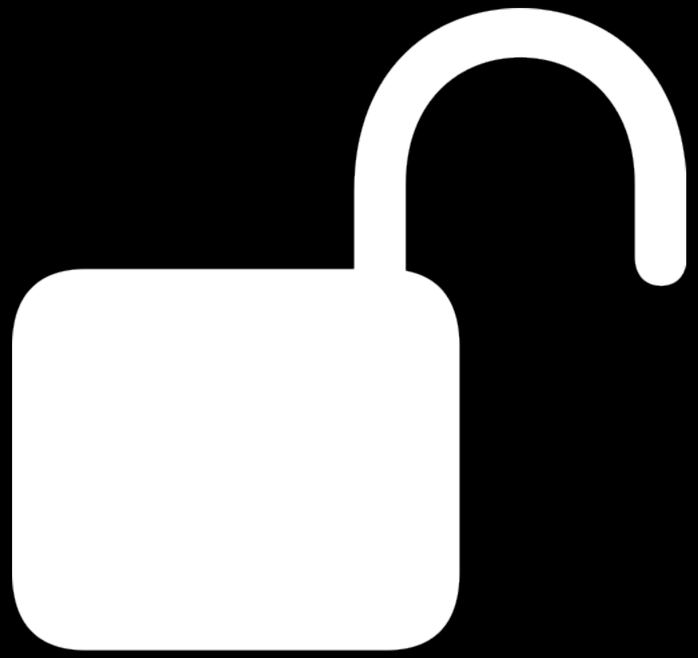
You have 72 hours to run
a lockscreen bypass!


Swipe up for Face ID or
Enter Passcode

Inactivity Reboot

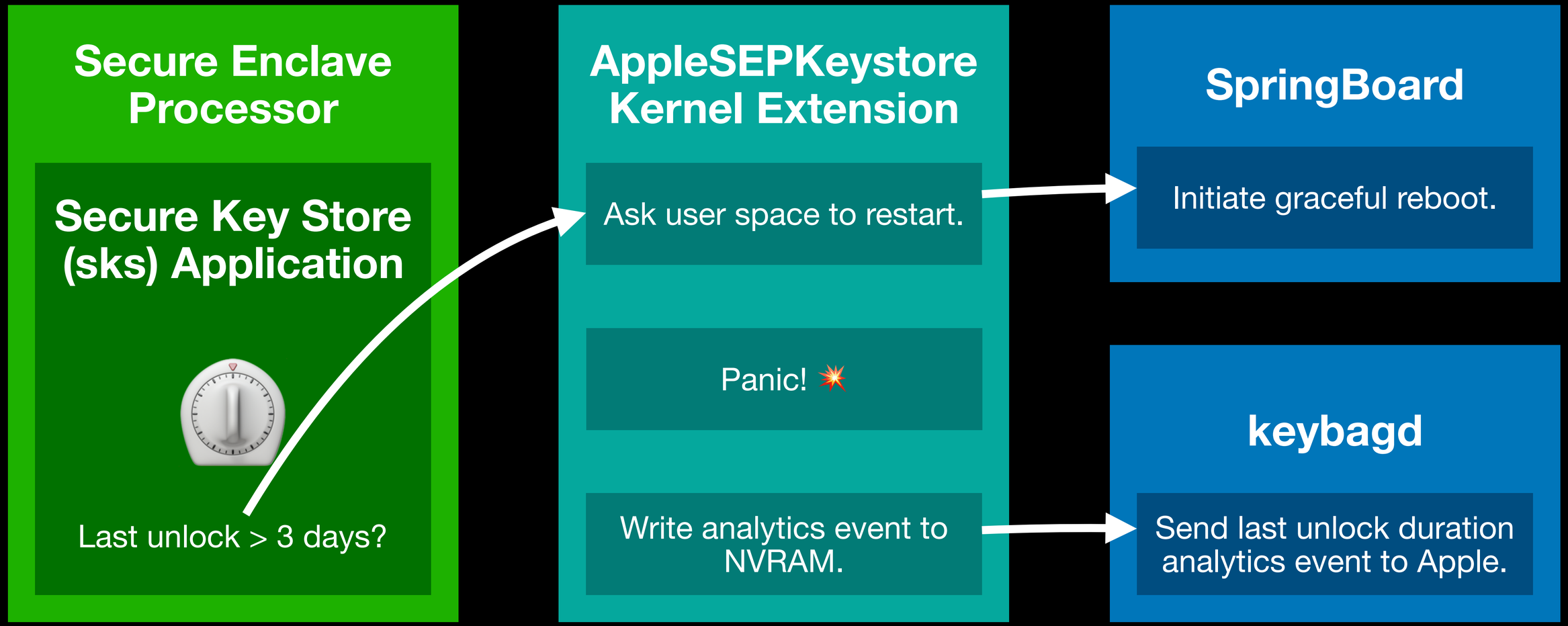


3-Day Inactivity Timer



Timer Reset with
SEP Unlock

Anatomy of Inactivity Reboot



Reverse engineer and hope Apple left enough strings everywhere to easily guess what’s going on.

Give an iPhone a well-deserved rest. When would it reboot?

Sysdiagnose

```
01:35:14.341314 kernelvoid AppleSEPManager::_notifyOSActiveGated(): SEP/OS is alive
01:35:14.341336 kernelSEP EP 18 enabled
01:35:14.341339 kernelSEP EP 10 enabled
01:35:14.341341 kernelSEP EP 9 enabled
01:35:14.341341 kernelAppleCredentialManager: getSEPEndpoint: SEPEndpoint enabled.
…
01:35:14.341697 kernel"AppleSEPKeyStore":3846:0: notifying user space of inactivity reboot
01:35:14.341757 chronod          Acquiring keep-alive with reason: Work scheduling after nonwake
01:35:14.341766 kernel"AppleSEPKeyStore":12598:31: operation failed (sel: 35 ret: e00002f0)
01:35:14.341846 SpringBoard      Received device inactivity notification. Rebooting…
```

All processes are terminated gracefully, as in a regular reboot.

AppleSEPKeyStore Kernel Extension

- No symbols in the iOS 18 kernel.
- macOS KDK might not implement what we're looking for.
- iOS 16 betas had kernel symbols* – time to diff!

** Symbols are part of the embedded kernel extensions and not the main MachO. They might not appear in some reverse engineering tools, but they are there!*

com.apple.driver.AppleSEPKeyStore — Binary Ninja 5.1.7295-dev

com.apple.dr...eSEPKeyStore | kernelcache....one15.e.bndb

Mach-O | Linear | High Level IL

```
int64_t AppleKeyStore::backup_data(uint64_t arg1, int32_t arg2, uint32_t arg3, aks_fs_context_s* arg4, void const* arg5, uint32_t arg6, void* arg7, uint32_t* arg8)

ffffffff00929a7cc    int64_t AppleKeyStore::backup_data(uint64_t arg1, int32_t arg2, uint32_t arg3, aks_fs_context_s* arg4, void const* arg5, uint32_t arg6, void* arg7,
ffffffff00929a7cc      uint32_t* arg8)

ffffffff00929a808      uint32_t* var_58 = arg8
ffffffff00929a810      int32_t x19_1
ffffffff00929a810      int64_t x21_1
ffffffff00929a810      x19_1, x21_1 = AppleKeyStore::effaceable_init(arg1.b)
ffffffff00929a870      int32_t* arg_0
ffffffff00929a870      return _err_sks_to_aks(__ipc_backup_pfk_data(
ffffffff00929a870          AppleKeyStore::sep_deliver_msg, nullptr, arg2, arg3, arg4.d, x21_1,
ffffffff00929a870          0x1c, arg6, x19_1, &var_58, arg_0))

ffffffff00929a874    int64_t AppleKeyStore::unwrap_data(uint64_t arg1, key_request_t arg2, int32_t arg3, void const* arg4, uint32_t arg5, uint32_t arg6, void const* arg7,
ffffffff00929a874      uint32_t arg8, void const* arg9, uint32_t arg10, void const* arg11, uint32_t arg12, void* arg13, uint32_t* arg14, uint32_t* arg15, void* arg16,
ffffffff00929a874      uint32_t* arg17, void* arg18, uint32_t* arg19, ipc_callback_ctx_t* arg20)

ffffffff00929a898      int32_t var_88 = arg7.d
ffffffff00929a89c      int64_t var_a0 = arg5
ffffffff00929a8a0      int64_t x28 = arg3
ffffffff00929a8b8      int32_t x24 = arg13.d
ffffffff00929a8bc      int64_t x20 = arg12.q
ffffffff00929a8c0      int32_t x27 = arg11.d
ffffffff00929a8c4      int32_t* x19 = arg10.q
ffffffff00929a8d4      int64_t x26 = 0xe00002c2
ffffffff00929a8d8      int32_t var_6c = 0
```

6 28 mac-aarch64 0xffffffff00929a7cc-0xffffffff00929a7d0 (0x4 bytes)

iOS 16 kernel with symbols, AppleSEPKeyStore split with `kextex`.

kernelcache.release.iphone15.e.bndb — Binary Ninja 5.1.7295-dev

com.apple.dr...eSEPKeyStore | kernelcache....one15.e.bndb

Mach-O · Linear · High Level IL

```
int64_t AppleKeyStore::backup_data(struct AKS* arg1, int64_t arg2, int32_t arg3, int32_t arg4, int64_t arg5, int64_t arg6, int32_t arg7, int64_t arg8, int32_t* arg9)

fffffff00933fc50    int64_t AppleKeyStore::backup_data(struct AKS* arg1, int64_t arg2, int32_t arg3, int32_t arg4, int64_t arg5, int64_t arg6, int32_t arg7, int64_t arg8,
fffffff00933fc50      int32_t* arg9)

fffffff00933fc8c        int64_t var_58 = arg8
fffffff00933fc94        AppleKeyStore::effaceable_init(aks: arg1, 0)
fffffff00933fcf4        return _err_sks_to_aks(__ipc_backup_pfk_data(
fffffff00933fcf4            AppleKeyStore::sep_deliver_msg, nullptr, arg2, arg3, arg4, arg5, 0x1c,
fffffff00933fcf4            arg6, arg7, &var_58, arg9))

fffffff00933fcf8    int64_t AppleKeyStore::unwrap_data(struct AKS* arg1, int64_t arg2, int64_t arg3, int32_t arg4, int64_t arg5, int32_t arg6, int32_t arg7, int64_t arg8,
fffffff00933fcf8      int32_t arg9, int64_t arg10, int32_t arg11, int64_t arg12, int32_t arg13, int64_t arg14, int32_t* arg15, int32_t* arg16, int64_t arg17,
fffffff00933fcf8      int32_t* arg18, int64_t arg19, int64_t* arg20, void* arg21)

fffffff00933fd20        int64_t x21 = arg19
fffffff00933fd24        int64_t x22 = arg17
fffffff00933fd30        int64_t x23 = arg14
fffffff00933fd50        int64_t x26 = 0xe00002c2
fffffff00933fd54        int32_t var_6c = 0
fffffff00933fd5c        int32_t var_bc_1
fffffff00933fd5c        int64_t var_b8_1
fffffff00933fd5c        int32_t var_ac_1
fffffff00933fd5c        int32_t* var_a8_1
fffffff00933fd5c        int64_t var_a0_1
fffffff00933fd5c        int64_t var_98_1
```

6 · 28 · mac-aarch64 · 0xfffffff00933fc50–0xfffffff00933fc54 (0x4 bytes)

Full iOS 18 kernel, diffed at places that seemed to be important.
Did this manually, as I had some ideas what I was looking for.

Address	Type	
fffffff0074f…	ASCII	"max inactivity window expired, failed to reboot the device" @%s:%d
fffffff0074f…	ASCII	aks-inactivity
fffffff0074f…	ASCII	%s%s:%s%s%s%s%u:%s%u:%s notifying user space of inactivity reboot%s…

Three strings related to inactivity reboot 🎉

notifying user space of inactivity reboot

- Already captured with the sysdiagnose, let’s start here!
- Called via the function `AppleKeyStore::handle_events`
- Looks like a function that pulls for SEP events in the background.

kernelcache.release.iphone15.e.bndb — Binary Ninja 5.1.7295-dev

com.apple.dr...eSEPKeyStore | kernelcache....one15.e.bndb

Mach-O · Linear · High Level IL

```
void AppleKeyStore::handle_events_gated__NEW_STATES_NOTIFY_REBOOT(struct AKS* aks, struct aks_event* event_1)

fffffff009344330                    char const* const x0_10
fffffff009344330                    void* const x26_3
fffffff009344330
fffffff009344330                    switch (case_that_can_reboot) {
fffffff009344334                        // lock state change
fffffff009344334                        case 0 ...
fffffff0093444c4                        // first unlock notification
fffffff0093444c4                        case 2 ...
fffffff009344540                        // invalidating background
fffffff009344540                        // policy gen
fffffff009344540                        case 3 ...
fffffff0093445cc                        // effacement notification
fffffff0093445cc                        case 4 ...
fffffff009344650                        case 5
fffffff009344650                            // notify userspace of
fffffff009344650                            // reboot
fffffff009344650                            SystemHintOp_BTI()
fffffff00934466c                            __akslog_filter
fffffff009344684                            _get_aks_log_pid()
fffffff009344688                            __akslog_filter
fffffff0093446c4                            char const* const var_150_5 =
fffffff0093446c4
fffffff0093446c8                            _IOLog("%s%s:%s%s%s%s%u:%s%u:%s notifying user space of …")
fffffff0093446cc                            setDeviceTreePropInactivity()
fffffff009344330                    }
fffffff009344310                }
```

6 28 mac-aarch64 0xfffffff0093442f4–0xfffffff0093442f8 (0x4 bytes)

`AppleKeyStore::handle_events` in turn calls what I've named `AppleKeyStore::handle_events_gated__NEW_STATES_NOTIFY_REBOOT`

aks-inactivity

- Device tree property that is written directly after notifying user space.
- This property survives the reboot.
- `keybagd` reads out this property to send an analytics event to Apple.

```
void setDeviceTreePropInactivity()

    if (zx.d(?aks_inactivity_true) == 0) {
        struct entry* entry_1 = IORegistryEntry::fromPath("/options",
            *0x10000003533798, nullptr, nullptr, nullptr)

        if (entry_1 != 0) {
            void* data = getProperty("aks-inactivity")

            if (data != 0) {
                char var_31 = 1
                int64_t* x0_1 = OSData::withBytes(&var_31, 1)

                if (x0_1 != 0) {
                    // sync
                    ?aks_inactivity_true = (&entry_1->vtbl->setProperty - 0xb8)
                        ->setProperty(entry_1, data, x0_1)
                    (*(x0_1 + 0x28))(x0_1)
                }

                (*(*data + 0x28))(data)
            }

            (&entry_1->vtbl->release - 0x28)->release(entry_1)
        }
    }
```

Why analytics?! 📈

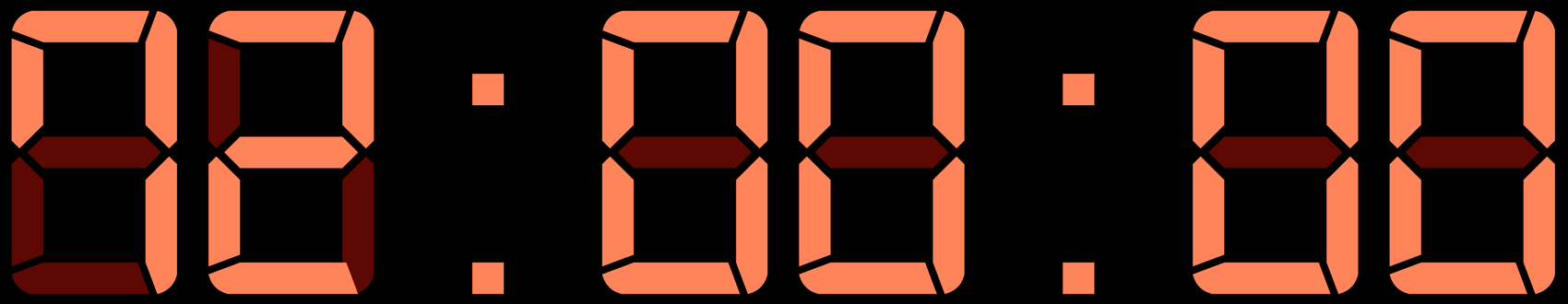
- How long should the timer ⏱️ be?
- Tradeoff between usability and security!
- 7 days in iOS 18.0, 3 days in iOS 18.1.
- User setting in GrapheneOS: As low as 10 minutes!

max inactivity window expired, failed to reboot the device

- Causes a kernel panic in case the user-space reboot failed.
- Called via `AppleKeyStore::handle_device_state_return`, which in turn can be called through various functions. State transitions again seem to be driven by SEP.

```
fffffff00936f840    void panic_max_inactivity_window_expired_force_reboot___NEW() __noreturn

fffffff00936f85c        char const* const var_20 = "AppleKeyStore.cpp"
fffffff00936f85c        int64_t var_18 = 0x3f1
fffffff00936f868        _panic(""max inactivity window expired, failed to reboot…")
fffffff00936f868        noreturn AppleKeyStore::effaceable_init.cold.1() __tailcall
```

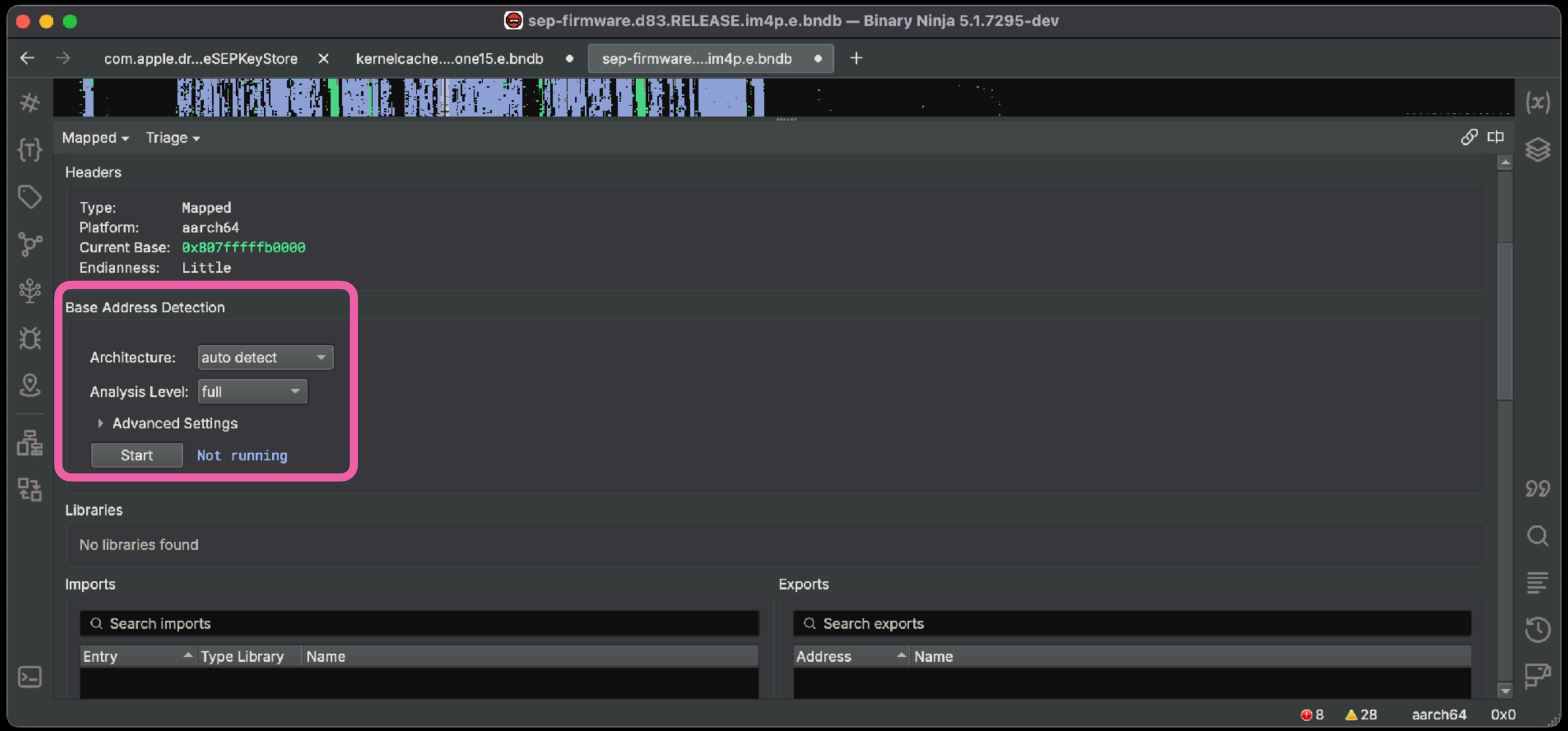


Where is the timer?

No references to any timer-related functionality that uses a 72h timer within the new kernel extension…

Analyzing SEP

- Apple encrypts SEP firmware 🤐
- @nyan_satan leaked SEP encryption keys for iOS 18.1 beta 6 🎉
- Previoius talk on SEP @ BH US 2016:
  SEP is organized in apps, the one for AppleSEPKeyStore is called `sks`
- I didn't know there was recent tooling for SEP 🤷‍♀️ (`sepsplit-rs`)
- Can't be that difficult to find a 72h timer anyway, right?



SEP consists of multiple parts. Without splitting it, some references will be inaccurate, but the automatic base detection worked for me.

sep-firmware.d83.RELEASE.im4p.e.bndb — Binary Ninja 5.1.7295-dev

com.apple.dr...eSEPKeyStore | kernelcache....one15.e.bndb | sep-firmware....im4p.e.bndb

Mapped · Linear · High Level IL

```
void InitializeSEPKeyStore(int64_t arg1, int64_t arg2, int64_t arg3, int64_t arg4) __noreturn

80090000ffe94ee4            noreturn
80090000ffe94e6c        }
80090000ffe94e6c
80090000ffe94e78        *(x0_2 + 0x220) = sub_80090000ffe959b8(4)
80090000ffe94e7c        int64_t* x0_14 = sub_80090000ffe95b48()
80090000ffe94e88        *x0_14 = (arg3 & 0xfffffffffffffffe) | (*x0_14 & 1)
80090000ffe94d4c    } else {
80090000ffe94d54        sub_80090000ffe95988()
80090000ffe94d58        sub_80090000ffe95b98(0)
80090000ffe94d4c    }
80090000ffe94d4c
80090000ffe94eb0    other_sks_state = &macho_header_sth - vmbase_sth(&macho_header_sth)
80090000ffe94eb4    sks_state = arg4
80090000ffe94eb8
80090000ffe94ebc    run_macho_constructors_and_init()
80090000ffe94ec0    AppleSEPKeyStore_main_service_workloop()
80090000ffe94ec0    noreturn
```

Strings: sks

Address	Type	Value
80090000ffe9…	ASCII	skssks acsssscasimp
80090000ffea…	ASCII	/Library/Caches/com.apple.xbs/Sources AppleKeyStore_SEP/sks.c:830
80090000fff1…	ASCII	osks
80090000fff4…	ASCII	osks

8 28 aarch64 0x80090000ffe94d2c–0x80090000ffe94d30 (0x4 bytes)

The sks app is still there, does some setup, and then enters a workloop.
Looks all right, where's the timer?

Compilers…

```c
int check_72h_timer(int timestamp) {
    return 3*24*60*60 > timestamp;
}
```

Must be `0x3f480` somewhere in the binary!

Or, if not in seconds, maybe factor 1000 etc.?

```arm64
check_72h_timer(int):
 sub    sp, sp, #0x10
 str    w0, [sp, #12]
 ldr    w9, [sp, #12]
 mov    w8, #0xf480
 movk   w8, #0x3, lsl #16
 subs   w8, w8, w9
 cset   w0, gt
 add    sp, sp, #0x10
 ret
```

sep-firmware.d83.RELEASE.im4p.e.bndb — Binary Ninja 5.1.7295-dev

com.apple.dr...eSEPKeyStore | kernelcache....one15.e.bndb | sep-firmware....im4p.e.bndb

Mapped | Linear | High Level IL

```
int64_t check_device_lock_times_3_days(int64_t last_unlock, int64_t now)

80090000ffe3873c        result = 0xffffffff
80090000ffe38644      else {
80090000ffe38648        uint32_t x8_1 = zx.d(var_30:1.b)
80090000ffe38648
80090000ffe38650        if (x8_1 u<= 0x3f)
80090000ffe38658            var_30:1.b = x8_1.b + 1
80090000ffe38658
80090000ffe3865c        int64_t time_difference = now - last_unlock
80090000ffe38668        int64_t time_based_state
80090000ffe38668
80090000ffe38668        if (now u< last_unlock || time_difference u< 0x4b1)
80090000ffe386ac            time_based_state = 0
80090000ffe38668         else if (time_difference u< _2.5h)
80090000ffe386b4            time_based_state = 1
80090000ffe38670        // 1 day + 1 second
80090000ffe38670         else if (time_difference u< _1_days)
80090000ffe386bc            time_based_state = 2
80090000ffe38680        // 2 days + 1 second
80090000ffe38680         else if (time_difference u< _2_days)
80090000ffe386c4            time_based_state = 3
80090000ffe38690         else if (time_difference u> _3_days)  // 3 days
80090000ffe386a4            // highest state entered when 3 days
80090000ffe386a4            time_based_state = 5
80090000ffe386a4         else
80090000ffe386a4            time_based_state = 4
80090000ffe386a4
```

```
enum times : uint32_t
{
    _3_days = 0x3f480,
    _2_days = 0x2a301,
    _1_days = 0x15181,
    _2.5h = 0xe11
};
```

8 | 28 | aarch64 | 0x80090000ffe38748–0x80090000ffe3874c (0x4 bytes)

…finally! 🎉

Consequences for Law Enforcement 👮‍♀️

- The first who discovered this phenomenon and motivated my research.
- Must act faster to search phones, but it won’t get impossible.
- Maybe law changes towards techniques that help stopping the timer.
- Interesting times combined with Apple also killing other bugs exploitable for forensic analysis…

Consequences for Thieves 😈

- Data about you and access to your bank accounts is worth much more than a stolen phone!
- Thieves can no longer use cheap, outdated tooling originally made for law enforcement.
- Yes, you can buy this equipment on eBay 🤦

Thanks!

Blog post on how I reverse engineered inactivity reboot:

https://naehrdine.blogspot.com/2024/11/reverse-engineering-ios-18-inactivity.html

@naehrdine.bsky.social

@jiska@chaos.social