The basics in building secure apps on iOS

Mari-Liis Sillat

Mari-Liis Sillat
iOS developer at Mooncascade

Plan for today:

1. What does security rely on?
2. How to make apps more secure:
   - Transport Security
   - Data retention
   - Undefined behavior detection
   - Keeping up to date


[ObjC]

App Transport Security
Standards (Nov 2018)

HTTPS

TLS 1.2 (1.3 came out in August, can opt in)

Strong cryptography (SHA-1, MD5 untrusted since iOS11)

App Transport Security

Info.plist

▼ App Transport Security Settings	Dictionary	(5 items)
Allow Arbitrary Loads	Boolean	NO
NSAllowsArbitraryLoadsForMedia	Boolean	NO
Allow Arbitrary Loads in Web Content	Boolean	NO
NSAllowsLocalNetworking	Boolean	NO

App Transport Security

Info.plist

▼ App Transport Security Settings	Dictionary	(5 items)
Allow Arbitrary Loads	Boolean	YES
NSAllowsArbitraryLoadsForMedia	Boolean	NO
Allow Arbitrary Loads in Web Content	Boolean	NO
NSAllowsLocalNetworking	Boolean	NO

App Transport Security

Info.plist

▼ App Transport Security Settings	Dictionary	(5 items)
Allow Arbitrary Loads	Boolean	NO
NSAllowsArbitraryLoadsForMedia	Boolean	NO
Allow Arbitrary Loads in Web Content	Boolean	NO
NSAllowsLocalNetworking	Boolean	NO
▼ Exception Domains	Dictionary	(2 items)
▼ i-own-this.example.com	Dictionary	(2 items)
NSExceptionAllowsInsecureHTTPLoads	Boolean	NO
NSExceptionMinimumTLSVersion	String	TLSv1.0
▼ also-mine.example.com	Dictionary	(1 item)
NSExceptionAllowsInsecureHTTPLoads	String	YES

Safe data

```
{ "name": "Finding Nemo" }
```

Raw	Formatted	Primitive	Structured
`0x46 0x69`	`{...}`	`[...]`	`Movie`

```
Movie(name: "Finding Nemo")
```

Safe data

```
{
   "name": "Finding Nemo",
   "trailer": "https://example.com/nemo",
   "imdb": "imdb:///title/tt0266543/"
}
```

NSSecureCoding
Codable
+
Validation

Safe data

```
struct Movie: Decodable {

    init(from decoder: Decoder) throws {
        let container = try decoder.container(keyedBy: CodingKeys.self)

        let trailerString = try container.decode(String.self, forKey: .trailer)
        guard let trailerURL = URL(string: trailerString) else {
            throw Error.invalidURL
        }
        guard trailerURL.host == "example.com" else {
            throw Error.unexpectedHost
        }

        self.trailerURL = trailerURL

        self.name = try container.decode(String.self, forKey: .name)
    }

}
```

Data retention

Info.plist	Unencrypted
UserDefaults	Unencrypted
KeyChain	Encrypted, for passwords, encryption keys, persists on uninstall
CoreData, Realm	By default unencrypted
Secure Enclave	Hardware-based key manager

Undefined behavior

Tools for detecting Undefined behavior

Address Sanitizer - catches memory corruption

Thread Sanitizer - catches data races

Undefined behavior Sanitizer (C languages only)

LLVM compiler

Address Sanitizer

```
func createUseAfterFreeVulnerability() {
    let greeting = "Hello, TallinnSec!"
    let pointer: UnsafePointer<Int8>
    pointer = greeting.withCString { pointerToGreeting in
        return pointerToGreeting
    }

    print("Printing: \(String(cString: pointer))")
}
```

Address Sanitizer

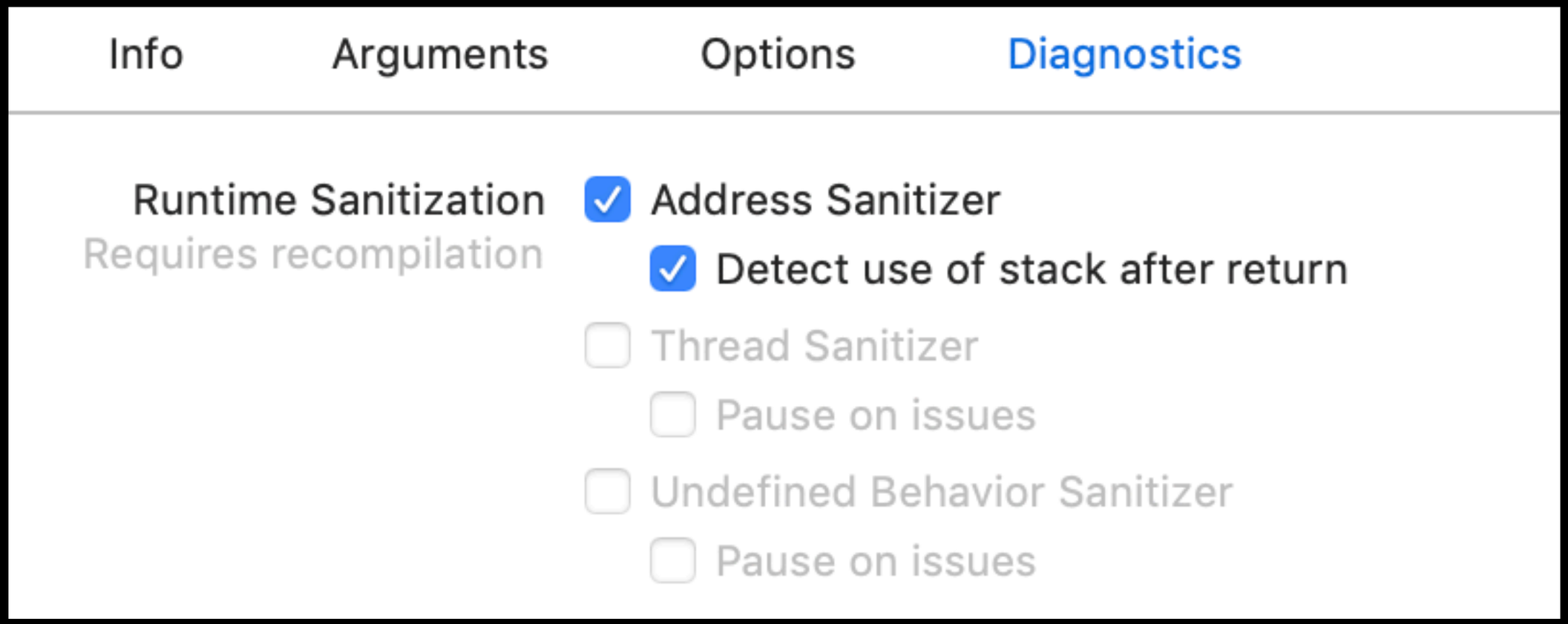
```
func createUseAfterFreeVulnerability() {
    let greeting = "Hello, TallinnSec!"
    let pointer: UnsafePointer<Int8>
    pointer = greeting.withCString { pointerToGreeting in
        return pointerToGreeting
    }

    print("Printing: \(String(cString: pointer))")
}
```

```
Printing: Hello, TallinnSec!

All Output ⌃   printing   1
```

Address Sanitizer
Edit Scheme... -> Run



Address Sanitizer

```
func createUseAfterFreeVulnerability() {
    let greeting = "Hello, TallinnSec!"
    let pointer: UnsafePointer<Int8>
    pointer = greeting.withCString { pointerToGreeting in
        return pointerToGreeting
    }

    print("Printing: \(String(cString: pointer))")
}
```

Thread 1: Use of deallocated memory

Updating

Update Swift (upgrade to Swift)

Update third party-libraries

Remove legacy, adhere to newest standards

Takeaways

🚀 Protect users' data in transit (HTTPS, TLS 1.3)

✅ Validate and sanitize all incoming data

🔑 Store only necessary data and encrypt it!

⚙️ Use the cool tools given to you (ASan, TSan etc)

⏳ Update early and often

Where to go from here

OWASP Mobile Security Testing Guide,
https://github.com/OWASP/owasp-mstg

Strategies for Securing Web Content,
WWDC 2018 Session 207

Data you can trust,
WWDC 2018 Session 222

Damn Vulnerable iOS app (2018)

Thank you!