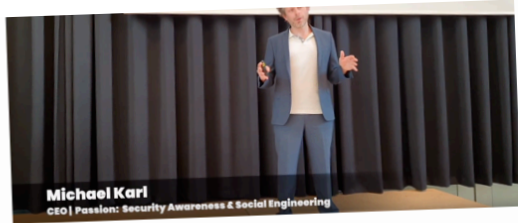


snapSEC |

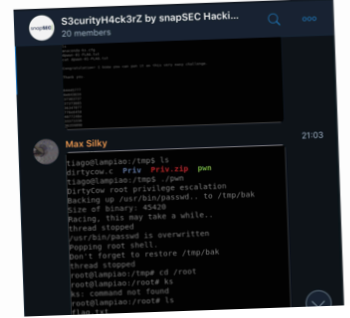
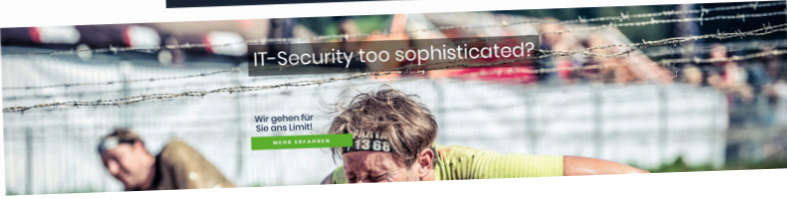
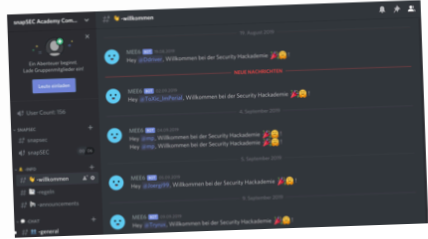
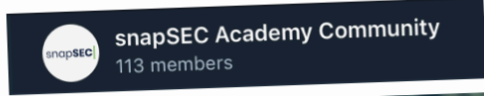


Reversing an iOS App

With Threat Modeling Insights.



snapSEC



In today's
Presentation

...

Mobile Penetration Testing Basics

Mobile OWASP Top 10 2016

Software Engineering of an iOS App

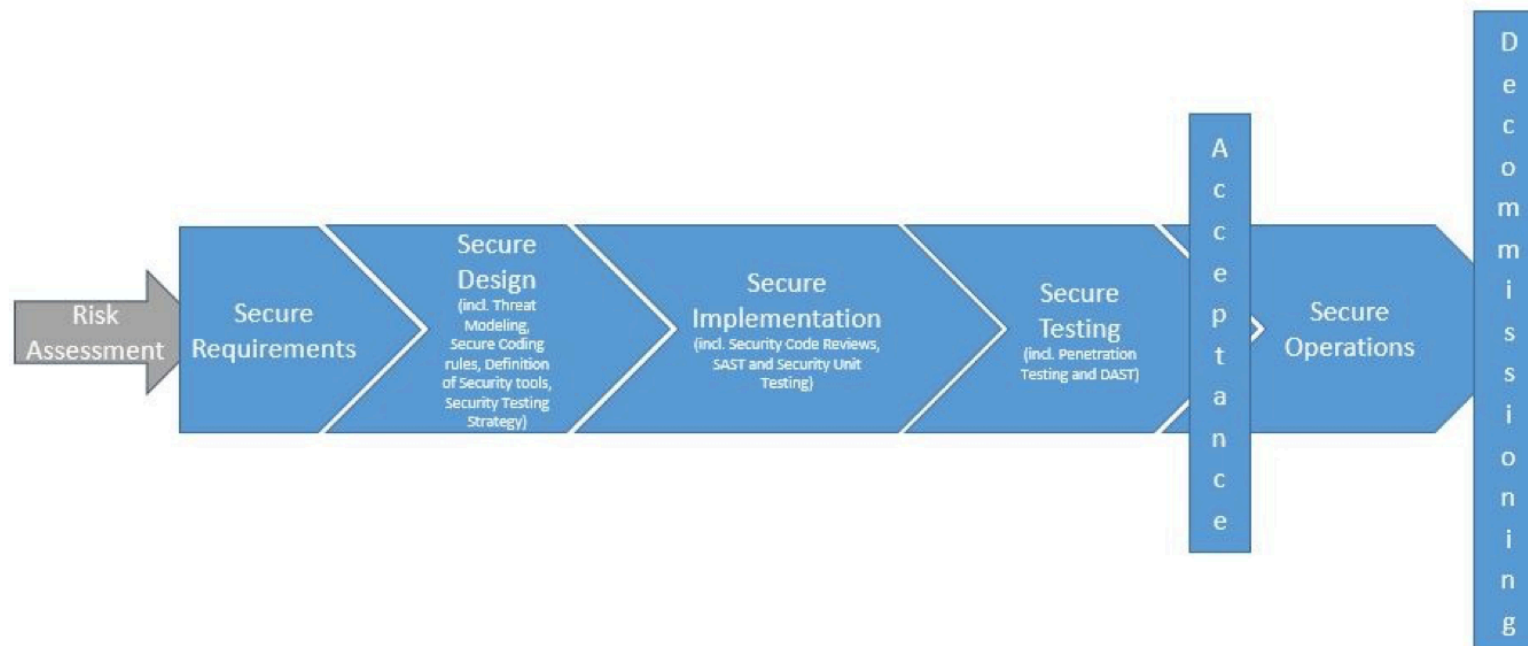
Reversing the iOS App

How to prevent Security Issues with Threat
Modeling

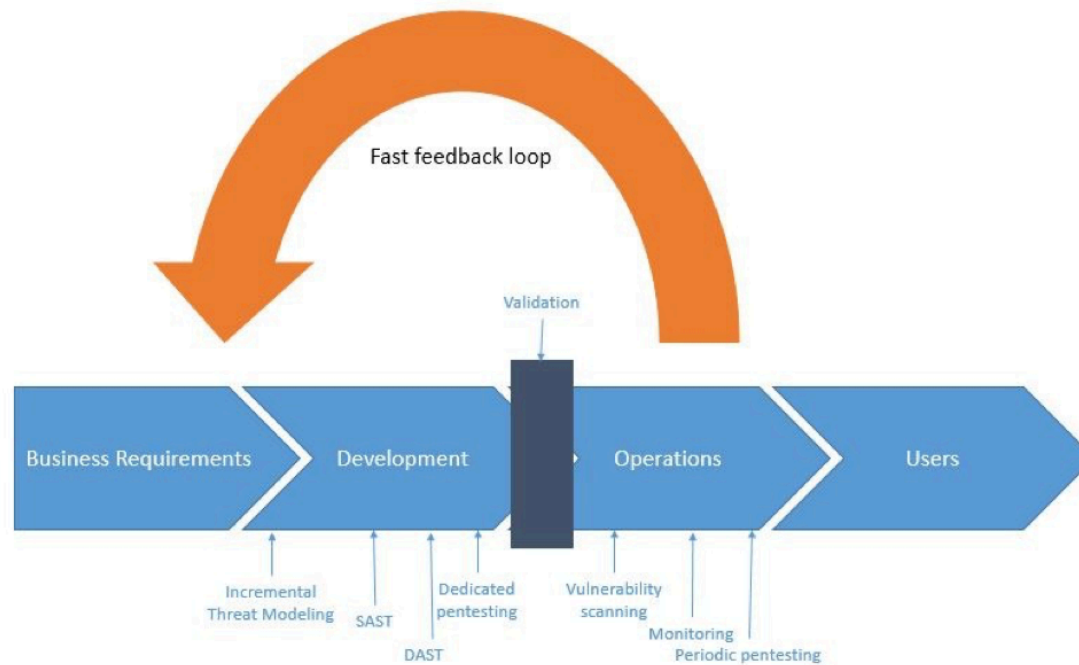


Mobile Penetration Testing Basics

(SECURE) Software Development Life Cycle



DevSecOps



Phases of Mobile Penetration Tests

Preparation

Intelligence Gathering

Mapping the Application

Exploitation

Reporting

Testing Principles

White-Box Testing

Black-Box Testing

Gray-Box Testing

SAST VS. DAST

Static Source Code Analysis

- Manual Code Review
- Automated Source Code Analysis

Dynamic Source Code Analysis

- Automated Scanning Tools → FALSE-POSITIVES!
- Clipboard
- Fuzzing → “Spray and Pray!”
- Penetration Testing

What Tools and Gear do you need?

macOS

Xcode and its simulator

Some jailbroken iPhones...

USB Cable for Connection

Cydia, Needle, Frida and a Reverse Engineering Tool (e.g. Hopper), Burp Suite (Pro), Wireshark

A dark blue ink splatter graphic on a white background, with the text centered within the splatter.

OWASP Mobile Top 10 2016

OWASP Mobile Top 10 – Part 1

M1 – Improper Plattform Usage

M2 – Insecure Data Storage

M3 – Insecure Communication

M4 – Insecure Authentication

M5– Insecure Cryptography

OWASP Mobile Top 10 – Part 2

M6 – Insecure Authorization

M7 – Client Code Quality

M8 – Code Tampering

**M9 – Reverse Engineering → The all
time winner**

M10 – Extraneous Functionality



Software Engineering of an iOS App

Types of Apps

Native Apps

Web Apps

Hybrid Apps

Progressiv Web Apps

What you need to develop an iOS App

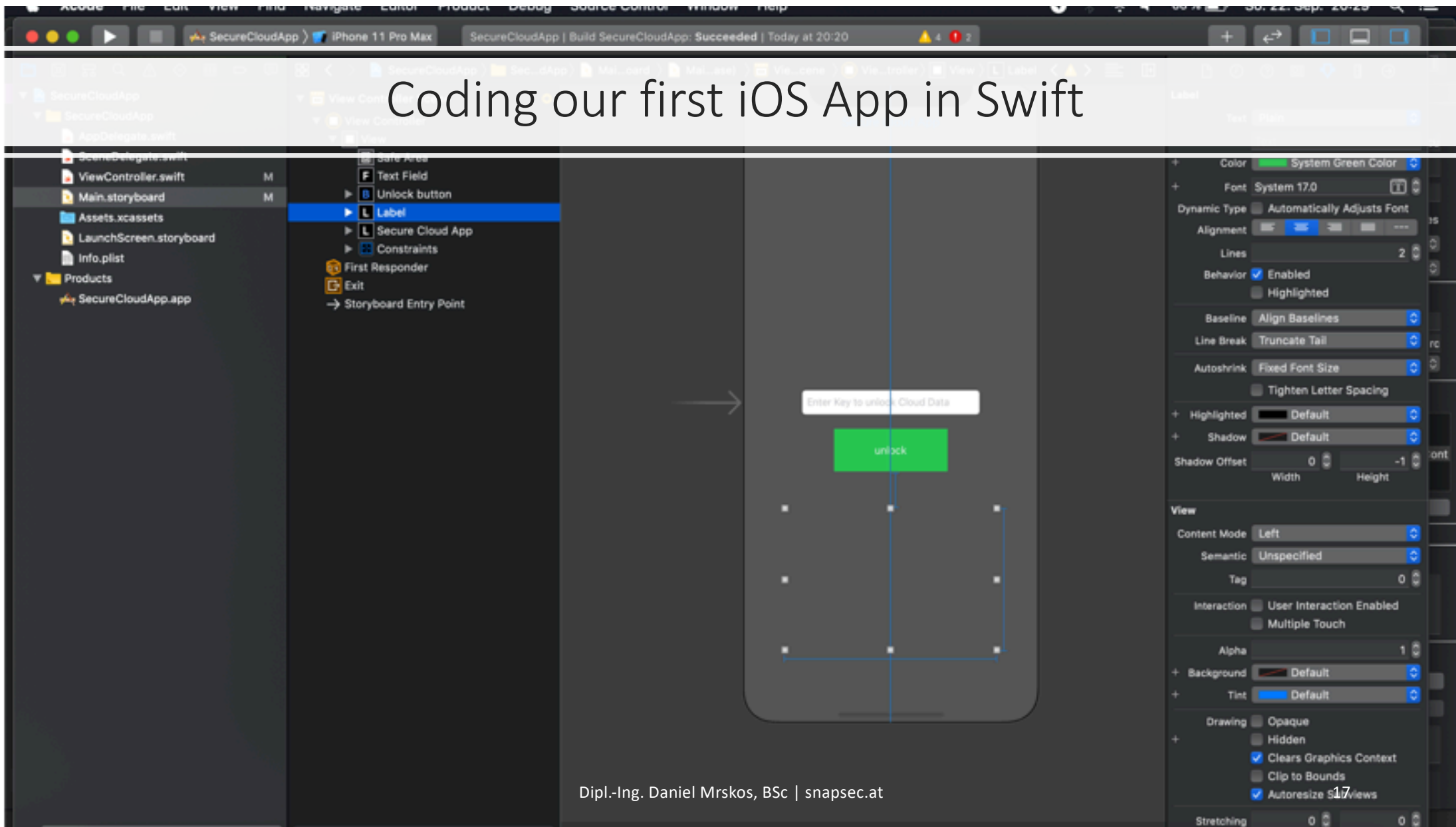
macOS

Xcode

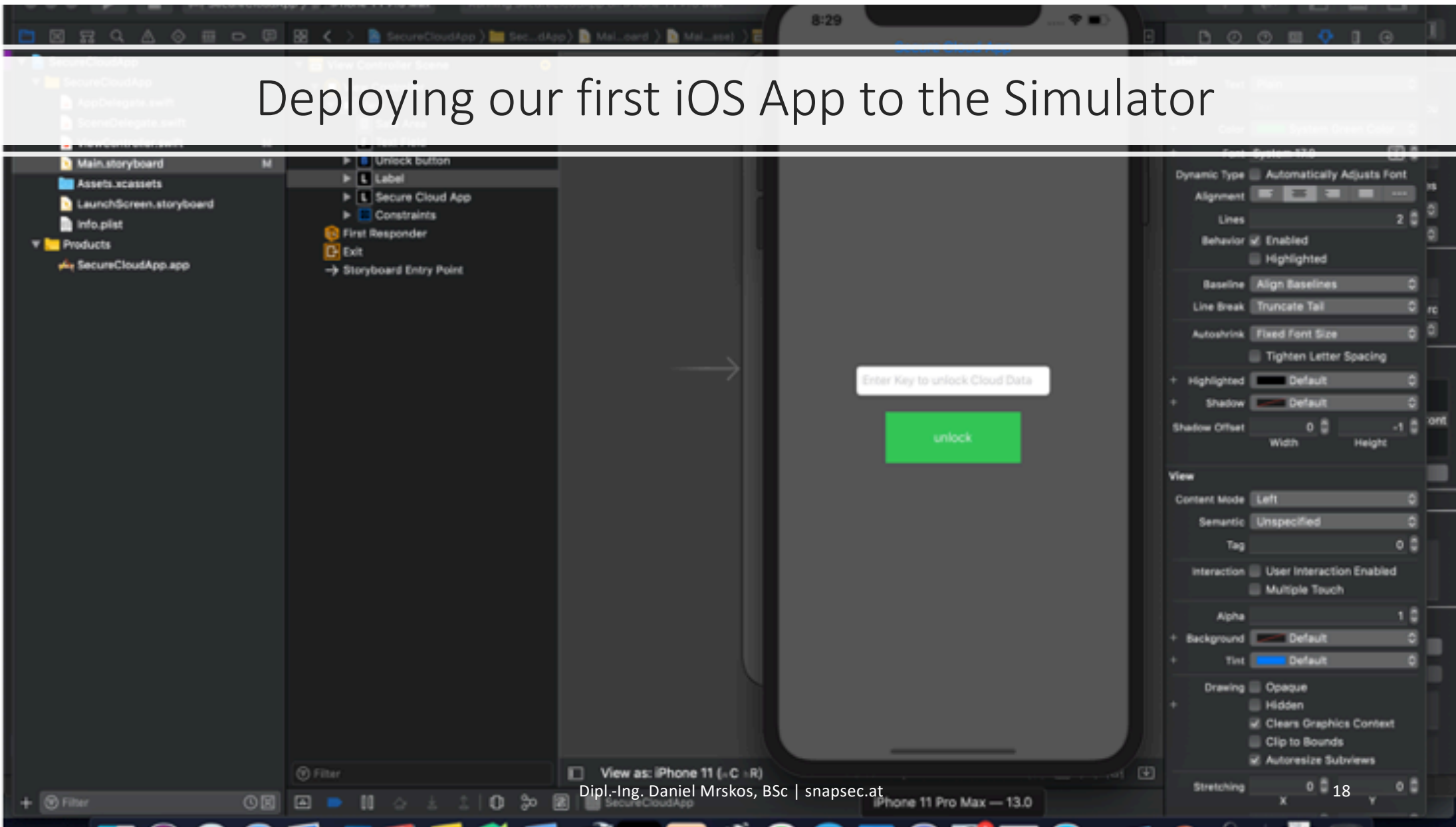
Some kind of an idea

A tiny little bit of software engineering
skills

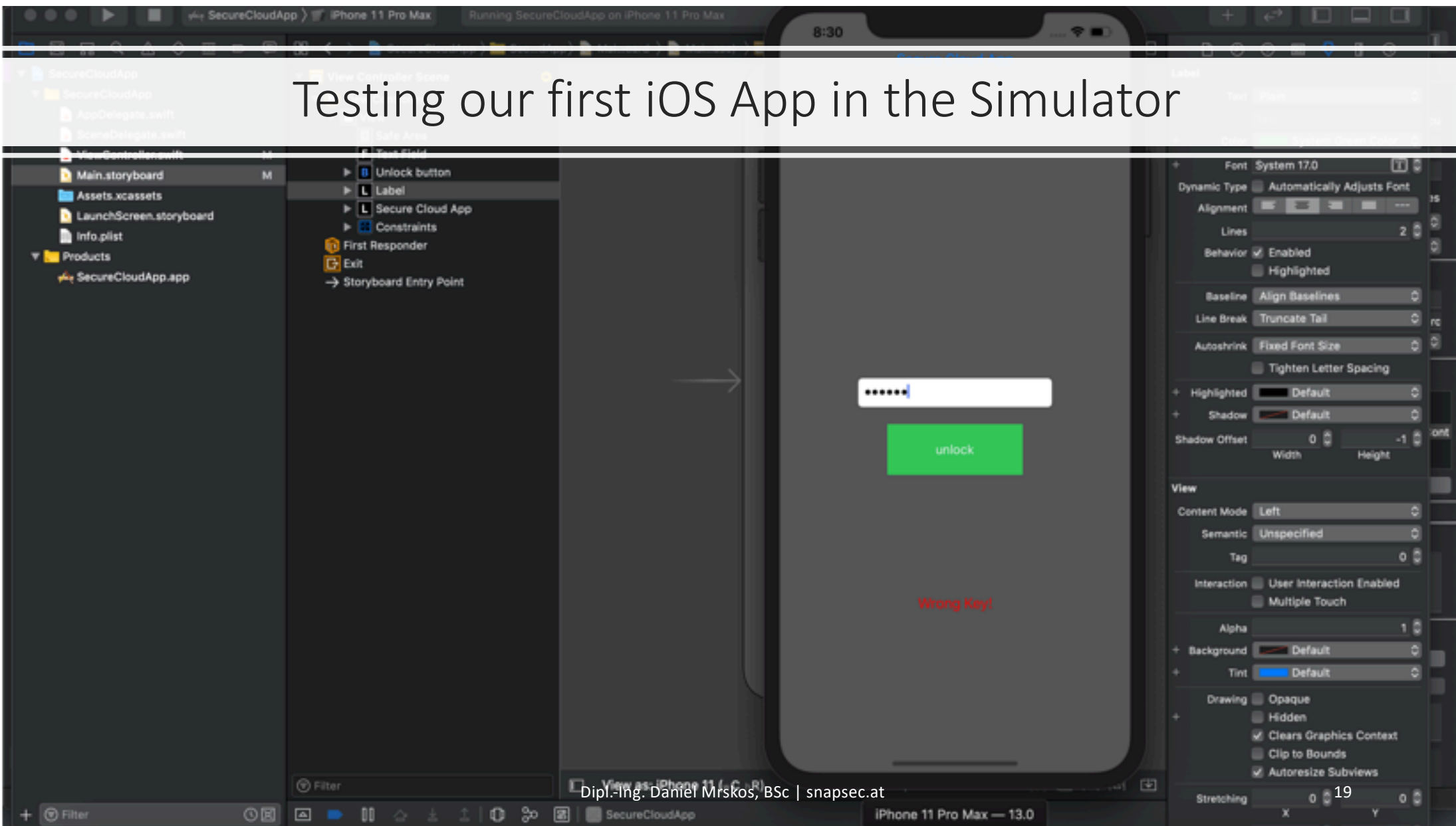
Coding our first iOS App in Swift



Deploying our first iOS App to the Simulator



Testing our first iOS App in the Simulator






Shipping to the Customer 😊

Because it seems really secure!



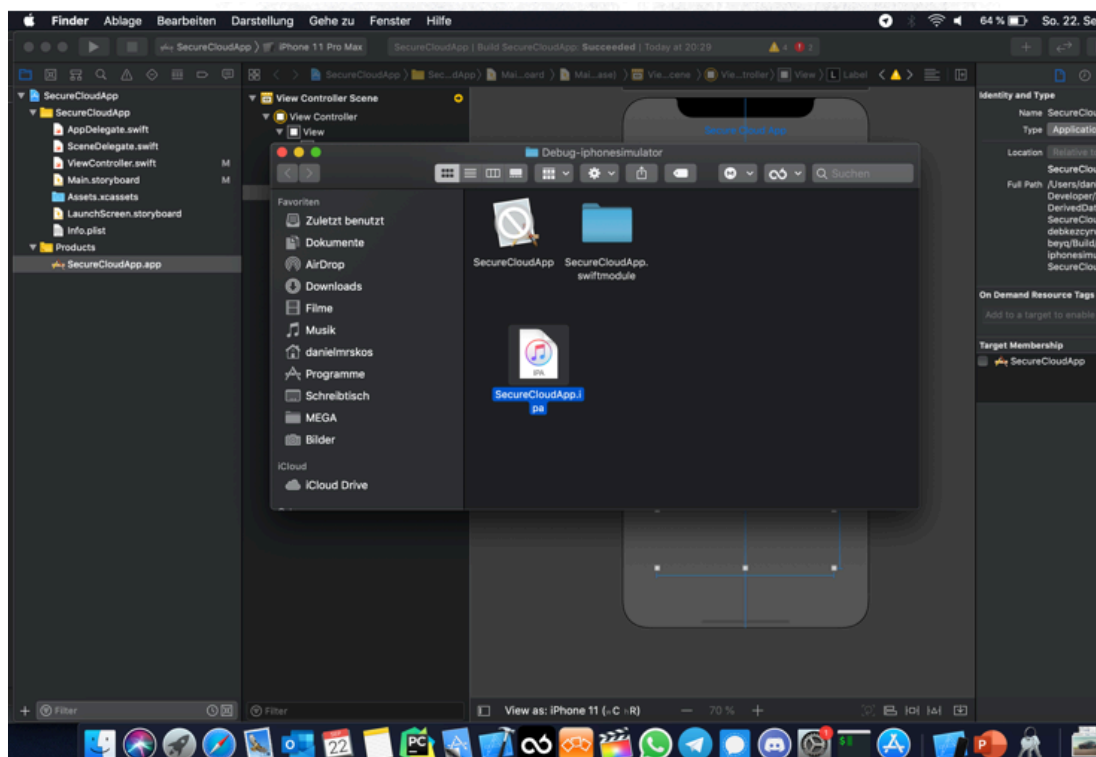
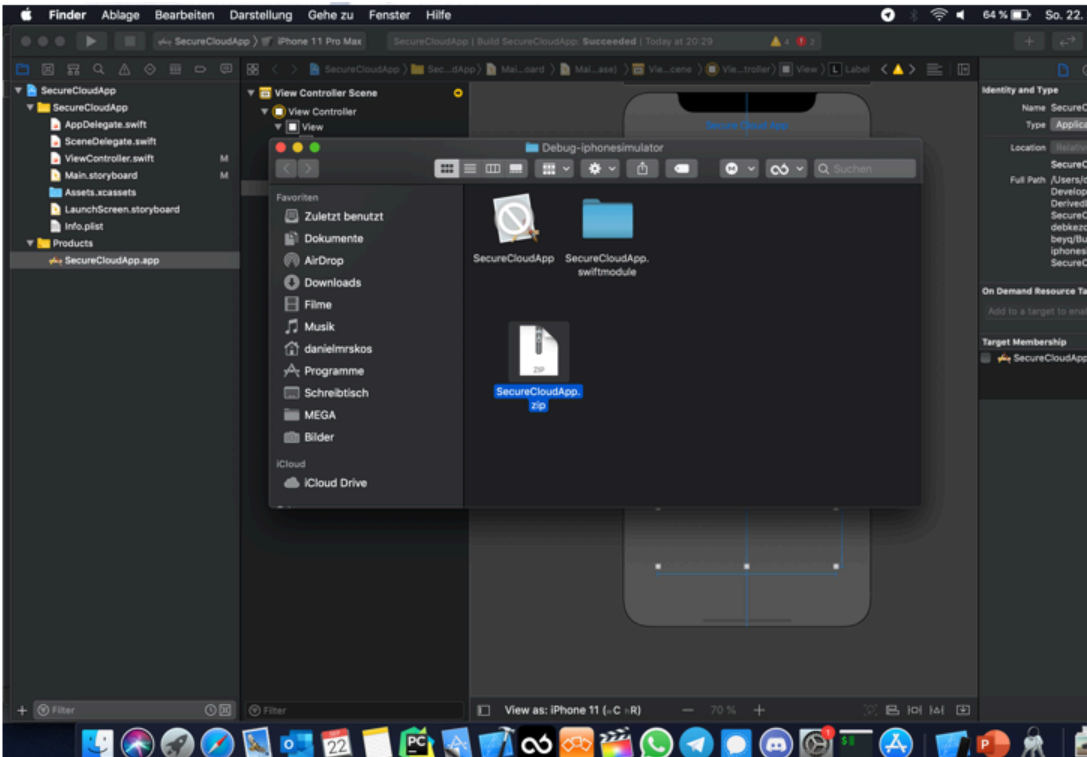
Hang on a minute!

„There is something missing, is not it?“
„What is this security thing? It is overrated^^“

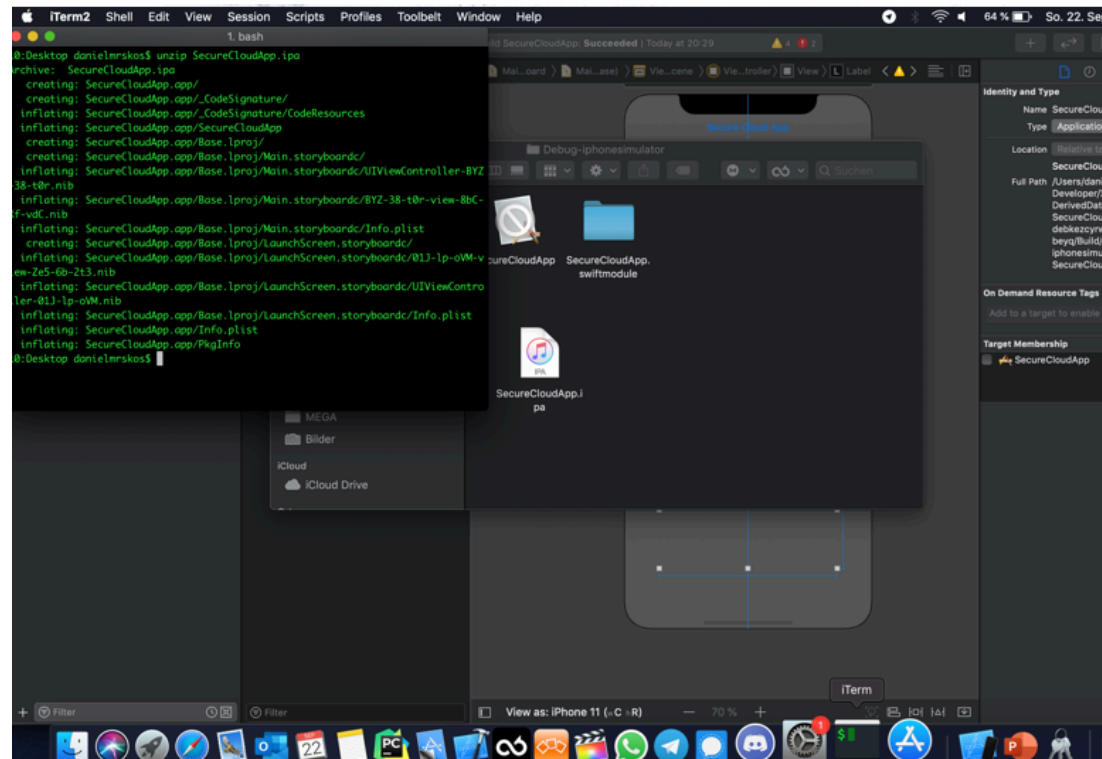
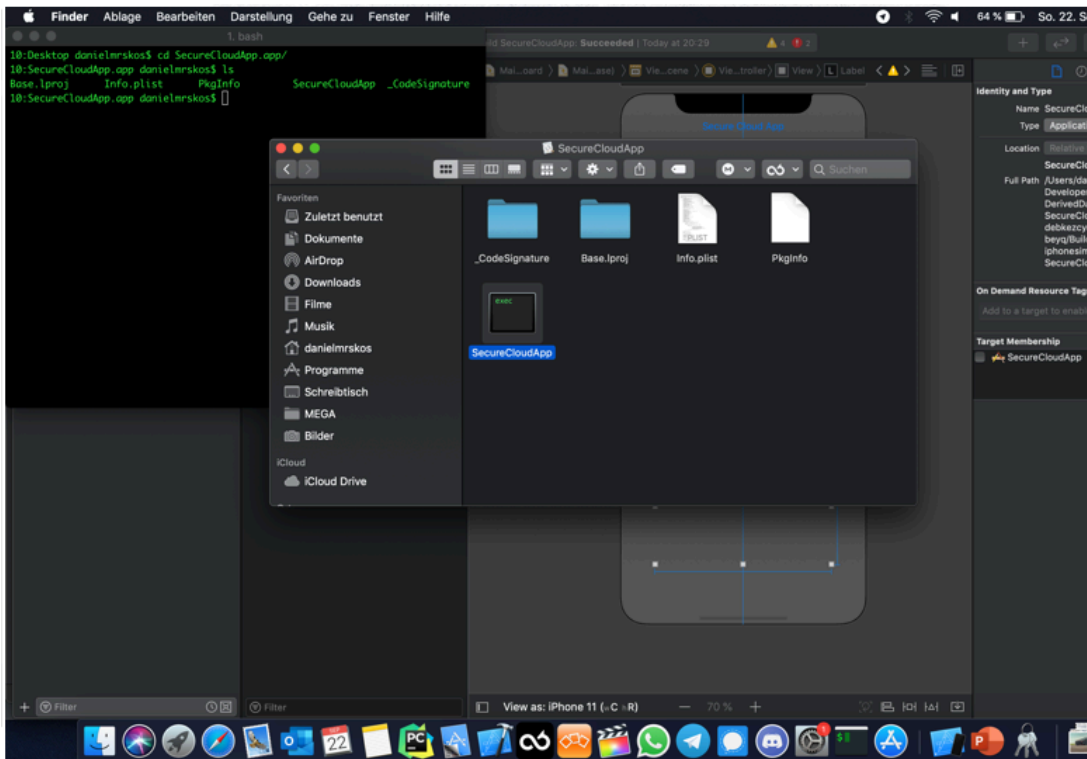


Reversing the iOS App

“Reversing will always win the fight”



Preparing the IPA



Unzip the IPA for Hopper

Loading the Binary in Hopper

The screenshot displays the Hopper Disassembler interface. The main window shows assembly code for a procedure, with comments in German explaining the code's purpose, such as "swift_bridgeObjectRelease" and "objc_release". The assembly code includes instructions like `mov rdi, qword [rbp+var_10]`, `call imp__stubs__swift_bridgeObjectRelease`, and `ret`. A red dashed line indicates the "BEGINNING OF PROCEDURE".

The left sidebar shows a "Tag Scope" with a list of addresses and names, including `0x100002e50 P _$s14SecureCloudApp14V...` and `0x1000037a0 P _main`. The right sidebar contains "File Information" (Path: /Users/danielmrskos/Desktop/Secu...), "Loader: Mach-O", "CPU: intel/x86_64", "CPU Syntax Variant: Intel", "Calling Convention: System V", "Address Information" (Type: Procedure, Prolog Heuristic: May be a procedure, Prolog Mode: Use Heuristic), "Navigation History", "Instruction Encoding" (55), "Graphic Views", "Format" (Argument --: Default, Signed, Negate, Leading Zeros, Relative to, Type, Field path), and "Comment".

At the bottom, a terminal window shows the command `>>> Python Command` and the output of a dataflow analysis of procedures in segment `__LINKEDIT`.

Labels Proc. Str ☆ ●
Q- wrong
Tag Scope
Wrong Key!

```

; ===== BEGINNING OF PROCEDURE =====
; Variables:
;   var_4: int32_t, -4
;   var_10: int64_t, -16
;   var_18: int64_t, -24
;   var_20: int64_t, -32
;   var_24: int32_t, -36

_main:
push    rbp
mov     rbp, rsp
sub     rsp, 0x30
xor     eax, eax
mov     ecx, eax ; argument #4 for method _$s14SecureCloudApp@C8DelegateCma
mov     dword [rbp+var_4], edi
mov     rdi, rcx ; argument #1 for method _$s14SecureCloudApp@C8DelegateCma
mov     qword [rbp+var_10], rsi
call   _$s14SecureCloudApp@C8DelegateCma ; _$s14SecureCloudApp@C8DelegateCma
mov     rdi, rax ; argument "aClass" for method imp___stubs__NSStringFromClass
mov     qword [rbp+var_18], rdx
call   imp___stubs__NSStringFromClass ; NSStringFromClass
mov     rdi, rax ; argument "instance" for method imp___stubs__objc_retainAutoreleasedReturnValue
call   imp___stubs__objc_retainAutoreleasedReturnValue ; objc_retainAutoreleasedReturnValue
xor     r8d, r8d
mov     edx, r8d
mov     rcx, qword [rbp+var_10]
mov     edi, dword [rbp+var_4]
mov     rsi, rcx
mov     rcx, rax
mov     qword [rbp+var_20], rax
call   imp___stubs__UIApplicationMain ; UIApplicationMain
mov     rcx, qword [rbp+var_20]
mov     rdi, rcx ; argument "instance" for method _objc_release
mov     dword [rbp+var_24], eax
call   qword [_objc_release_1000070e0] ; _objc_release, _objc_release_1000070e0, _objc_release
xor     eax, eax
add     esp, 0x30
pop     rbp

```

File Information

Path: /Users/danielmrskos/Desktop/Secu

Loader: Mach-O

CPU: intel/x86_64

Branch always stops procedures

CPU Syntax Variant: Intel

Calling Convention: System V

Address Information

Type: Procedure

Prolog Heuristic: Not a procedure pro

Prolog Mode: Use Heuristic

Navigation History

Clear Navigation Stack

Instruction Encoding

48 89 C7

Control Flow Graph

Searching for the string „wrong“

```

> dataflow analysis of procedures in segment External Symbols
> dataflow analysis of procedures in segment External Symbols
> Analysis pass 9/10: finalizing prototype search
> Analysis pass 10/10: searching contiguous code area
> Last pass done
Background analysis ended in 218ms

```

210 strings Python Command Dipl.-Ing. Daniel Mrskos, BSc | snapsec.at

Address 0x1000037c7, Segment __TEXT, _main + 39, Section __text, file offset 0x37c7 - Alt+Mouse, or trackpad gesture to zoom - Double-tap to focus on a block

Examining data stored as CString

The screenshot shows the Hopper Disassembler v4 interface. The main window displays assembly code for a procedure named `aWrongKey:`. A search box in the top left contains the text `wrong`, and a tag scope on the left shows `Wrong Key!` selected. The assembly code includes several data definitions for strings, such as `aUnexpectedlyFo_100005830: // aUnexpectedlyFo` and `aWrongKey: "Wrong Key!", 0`. The code is disassembled into instructions, with the current instruction being `db "Wrong Key!", 0`. The right sidebar shows file information, address information (Type: ASCII), and navigation history. The bottom status bar indicates 210 strings and a Python command prompt.

```
00000000100005820 db 0
00000000100005827 db 0
00000000100005828 db 0
00000000100005829 db 0
0000000010000582a db 0
0000000010000582b db 0
0000000010000582c db 0
0000000010000582d db 0
0000000010000582e db 0
0000000010000582f db 0
00000000100005830 aUnexpectedlyFo_100005830: // aUnexpectedlyFo
db "Unexpectedly found nil while unwrapping an Optional value", 0 ; DATA XREF=_$s14SecureCloudApp14V
0000000010000586a db 0
0000000010000586b db 0
0000000010000586c db 0
0000000010000586d db 0
0000000010000586e db 0
0000000010000586f db 0
00000000100005870 aSup3rs3cur3pss:
db "_Sup3rS3cur3P@ssw0rd!", 0 ; DATA XREF=_$s14SecureCloudApp14V
00000000100005886 aWrongKey:
db "Wrong Key!", 0 ; DATA XREF=_$s14SecureCloudApp14V
00000000100005891 db 0
00000000100005892 db 0
00000000100005893 db 0
00000000100005894 db 0
00000000100005895 db 0
00000000100005896 db 0
00000000100005897 db 0
00000000100005898 db 0
00000000100005899 db 0
0000000010000589a db 0
0000000010000589b db 0
0000000010000589c db 0
0000000010000589d db 0
0000000010000589e db 0
0000000010000589f db 0
000000001000058a0 aSuperSecretClo:
db "Super Secret Cloud Data:\n!+1=10!", 0 ; DATA XREF=_$s14SecureCloudApp14V
000000001000058c1 db 0
000000001000058c2 db 0
000000001000058c3 db 0
000000001000058c4 db 0
000000001000058c5 db 0
```

> dataflow analysis of procedures in segment `__LINKED_IT`
> dataflow analysis of procedures in segment `External Symbols`
> Analysis pass 9/10: remaining prologs search
> Analysis pass 10/10: searching contiguous code area
> Last pass done
Background analysis ended in 218ms

210 strings
>>> Python Command

Finding the unlock key stored in plain text

The screenshot displays the Immunity Debugger interface with the following components:

- Assembly View:** Shows assembly instructions for a procedure. A red arrow points from the instruction `lea rdi, qword [aSup3rs3cur3Pssw@rd!]` to a jump instruction `jmp loc_100002487` in the code below.
- Right-Hand Panel:** Contains metadata and analysis options:
 - File Information: Path, Loader, CPU.
 - Branch always stops procedures:
 - CPU Syntax Variant: Intel
 - Calling Convention: System V
 - Address Information: Type: Procedure, Prolog Heuristic: Not a procedure pro, Prolog Mode: Use Heuristic
 - Navigation History: 0x1000037c7 (_main + 0x27), 0x100005870 (aSup3rs3cur3Pss)
 - Instruction Encoding: 48 8D 3D 5C 36 00 00
 - Control Flow Graph: A flowchart showing the execution path.
- Bottom Panel:** Shows analysis results:
 - 210 strings
 - Python Command
 - Address 0x10000220d, Segment __TEXT, _\$s14SecureCloudApp14ViewControllerC80doUnlockyySSF + 93, Section __text, file offset 0x220d - Alt+Mouse, or trackpad gesture to zoom - Double-tap to focus on a block

Trying out the found unlock key 😊

Leaving CFG node: No procedure at this address

wrong

Tag Scope

Wrong Key!

```
0000000100005825 db 0
0000000100005826 db 0
0000000100005827 db 0
0000000100005828 db 0
0000000100005829 db 0
000000010000582a db 0
000000010000582b db 0
000000010000582c db 0
000000010000582d db 0
000000010000582e db 0
000000010000582f db 0
0000000100005830 aUnexpectedlyFo_100005830: // aUnex
db "Unexpectedly found nil whi
000000010000586a db 0
000000010000586b db 0
000000010000586c db 0
000000010000586d db 0
000000010000586e db 0
000000010000586f db 0
0000000100005870 aSup3rs3cur3pss:
db "Sup3rs3cur3pssw0rd", 0
0000000100005886 aWrongKey:
db "Wrong Key!", 0
0000000100005891 db 0
0000000100005892 db 0
0000000100005893 db 0
0000000100005894 db 0
0000000100005895 db 0
0000000100005896 db 0
0000000100005897 db 0
0000000100005898 db 0
0000000100005899 db 0
000000010000589a db 0
000000010000589b db 0
000000010000589c db 0
000000010000589d db 0
000000010000589e db 0
000000010000589f db 0
00000001000058a0 aSuperSecretClo:
db "Super Secret Cloud Data:\n
00000001000058c1 db 0
00000001000058c2 db 0
00000001000058c3 db 0
> dataflow analysis of procedures in segment __LINKEDIT
> dataflow analysis of procedures in segment External Symbols
> Analysis pass 9/10: remaining prologs search
> Analysis pass 10/10: searching contiguous code area
> Last pass done
Background analysis ended in 218ms
>>> Python Command
```

.....

unlock

Super Secret Cloud Data:
1+1=10!

Information

Path: /Users/danielmrskos/Desktop/Secu

Header: Mach-O

CPU: intel/x86_64

Branch always stops procedures

U Syntax Variant: Intel

Calling Convention: System V

Address Information

Type: ASCII

Prolog Heuristic: Not a procedure pro

Prolog Mode: Use Heuristic

Navigation History

00037c7 (_main + 0x27)

0005870 (aSup3rs3cur3pss)

000220d (_\$s14SecureCloudApp14ViewCont

Clear Navigation Stack

Public Views

Comment

Strings and Tags

Area: Set Clear

Procedure: Set Clear

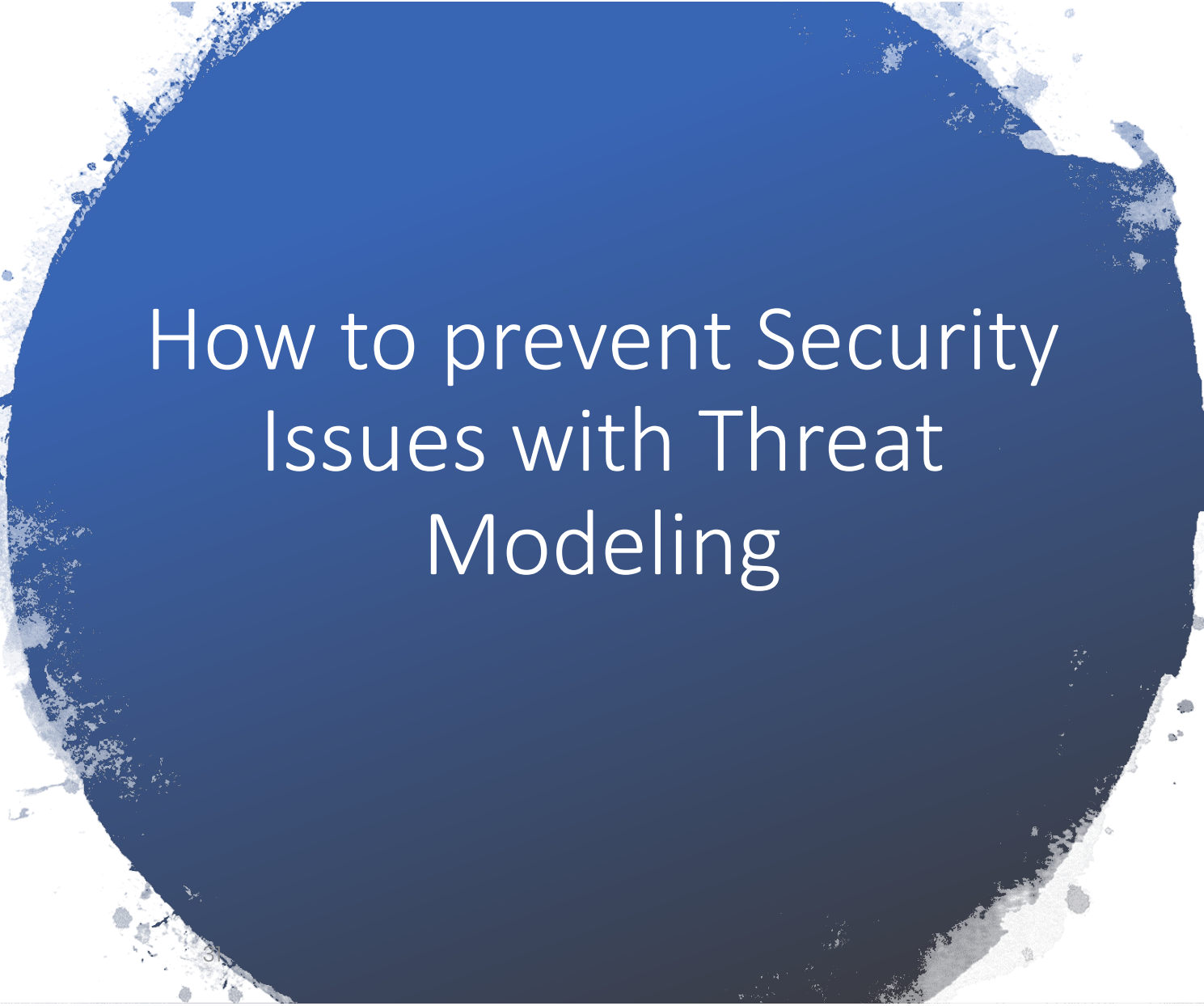
Address:

Block:

Let's take a look at the Code to identify the Issue

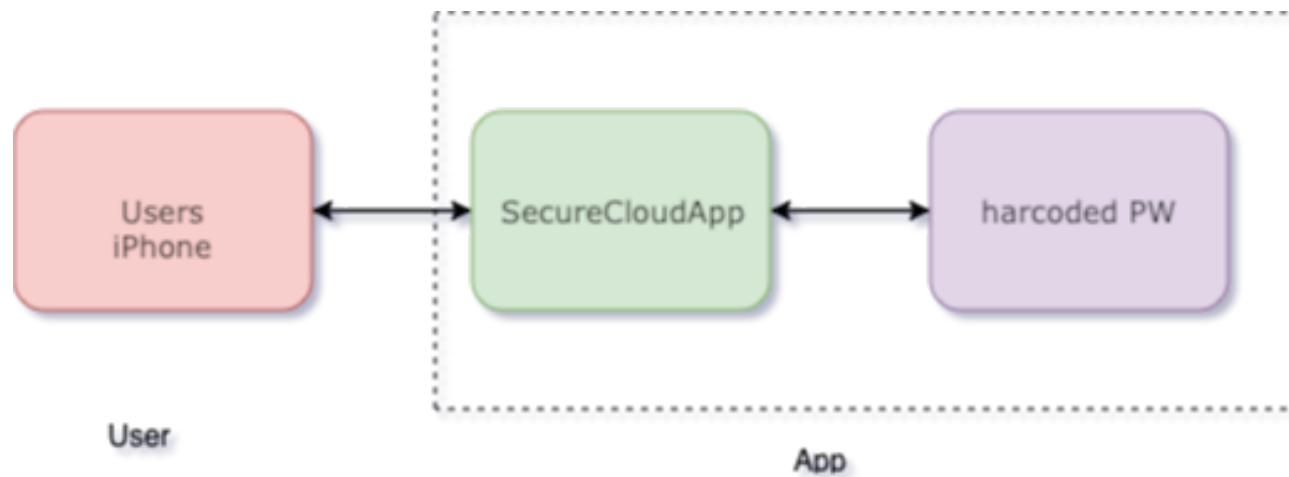
```
9 import UIKit
10
11 class ViewController: UIViewController {
12     @IBOutlet var _label: UILabel!
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         // Do any additional setup after loading the view.
16     }
17
18     @IBAction func UnlockButton(_ sender: Any) {
19         let unlock = textField.text
20
21         if(unlock == ""){
22             return
23         }
24
25         DoUnlock(unlock!)
26     }
27
28     func DoUnlock(_ unlock:String){
29         if(unlock == "_Sup3rS3cur3P@ssw0rd"){
30             _label.text = "Super Secret Cloud Data:\n!+1=10!"
31             _label.textColor = UIColor.green
32         }else{
33             _label.text = "Wrong Key!"
34             _label.textColor = UIColor.red
35         }
36     }
37 }
38
39
40
41
42
```

M2-Insecure Data Storage found!



How to prevent Security Issues with Threat Modeling

Drawing a Data Flow Diagram of the Application



Using STRIDE per Interaction to Identify Threats in our Scenario

Interaction	S	T	R	I	D	E
User sends key		X	X		X	X
Checking the plaintext key				X		

Design Issue found: Key is stored in plaintext!

Using Attack Libraries to identify Threats as well

[Global AppSec Amsterdam, September 23-27]

Mobile Top 10 2016-M2-Insecure Data Storage

[2016 Table of Contents](#)
 [← M1-Improper Platform Usage](#)
 [2016 Mobile Top 10 List](#)
 [M3-Insecure Communication →](#)

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	Application / Business Specific
Threats agents include the following: an adversary that has attained a lost/stolen mobile device; malware or another repackaged app acting on the adversary's behalf that executes on the mobile device.	In the event that an adversary physically attains the mobile device, the adversary hooks up the mobile device to a computer with freely available software. These tools allow the adversary to see all third party application directories that often contain stored personally identifiable information (PII) or other sensitive information assets. An adversary may construct malware or modify a legitimate app to steal such information assets.	Insecure data storage vulnerabilities occur when development teams assume that users or malware will not have access to a mobile device's filesystem and subsequent sensitive information in data-stores on the device. Filesystems are easily accessible. Organizations should expect a malicious user or malware to inspect sensitive data stores. Usage of poor encryption libraries is to be avoided. Rooting or jailbreaking a mobile device circumvents any encryption protections. When data is not protected properly, specialized tools are all that is needed to view application data.	This can result in data loss, in the best case for one user, and in the worst case for many users. It may also result in the following technical impacts: extraction of the app's sensitive information via mobile malware, modified apps or forensic tools. The nature of the business impact is highly dependent upon the nature of the information stolen. Insecure data may result in the following business impacts:	Insecure data storage vulnerabilities typically lead to the following business risks for the organization that owns the risk app:	
				<ul style="list-style-type: none"> Identity theft; Privacy violation; Fraud; Reputation damage; External policy violation (PCI); or Material loss. 	<ul style="list-style-type: none"> Identity Theft Fraud Reputation Damage External Policy Violation (PCI); or Material Loss.

Am I Vulnerable To 'Insecure Data Storage'?

This category insecure data storage and unintended data leakage. Data stored insecurely includes, but is not limited to, the following:

- SQL databases;
- Log files;
- XML data stores or manifest files;
- Binary data stores;
- Cookie stores;

How Do I Prevent 'Insecure Data Storage'?

It is important to threat model your mobile app, OS, platforms and frameworks to understand the information assets the app processes and how the APIs handle those assets. It is crucial to see how they handle the following types of features:

- URL caching (both request and response);
- Keyboard press caching;
- Copy/Paste buffer caching;
- Application backgrounding;

Threat Modeling combined with iOS App Development



The logo for snapSEC is centered on the left side of the slide. It features the word "snapSEC" in a bold, dark blue sans-serif font. A vertical green bar is positioned to the right of the "SEC" part of the logo. The logo is set against a white circular background that has a blue, splattered border.

snapSEC |

Thanks for your attention!

- #GotAnyQuestions?
hello@snapsec.at
- #InterestedInMobilePentesting?
[OWASP MSTG](#)
- #ThanksToFHSTP
fhstp.ac.at