



November 29-30, 2018
Mechelen, Belgium

Fast forwarding Mobile Security with the MSTG

Jeroen Willemse – OWASP Benelux days

About me

Jeroen Willemse

@commjoenie

jeroen.willemsen@owasp.org

“Security architect”

“Full-stack developer”

“Mobile security”

@OWASP_MSTG



Agenda

- Introduction into the MASVS
- Introduction into the MSTG
- Some examples

The MSTG: mobile security?

QUESTION:

Can you do a CSRF or XSS attack on a native mobile app without a webview?

Answer:

XSS: No,

CSRF: No. Even with deeplinks it is not the same.



The MSTG: mobile security?

- So CSRF and XSS do not easily apply.

Natalie Silvanovich (@natashenka) posted a tweet on December 15, 2017, at 15:32. The tweet reads: "It's Android directory traversal day on the Project Zero Tracker bugs.chromium.org/p/project-zero/ ... bugs.chromium.org/p/project-zero/ ...". The tweet includes two links to the Project Zero tracker. The Twitter interface shows a "Folgen" (Follow) button and a dropdown menu.



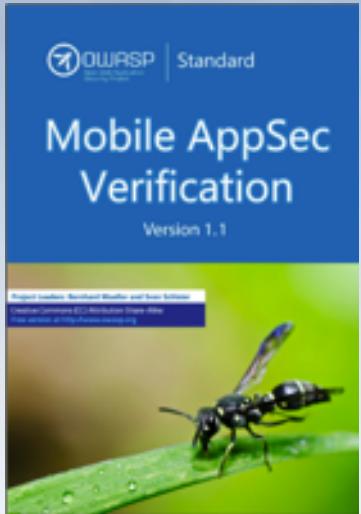
- But path-traversals do...

The MSTG: mobile security?

- So CSRF and XSS do not easily apply.
- But path-traversals do...
- And then there is... Data leakage
 - through logging,
 - through insecure storage,
 - Through IPC.
- What about weak authentication mechanisms?
- What about reverse engineering?



How do we fix this?



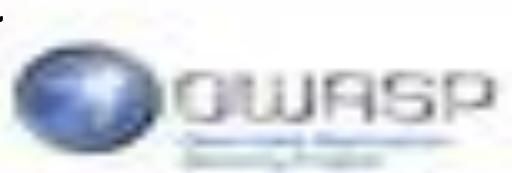
Mobile Application
Security
Verification Standard
<https://github.com/OWASP/owasp-masvs>



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



Mobile Appsec
Checklist

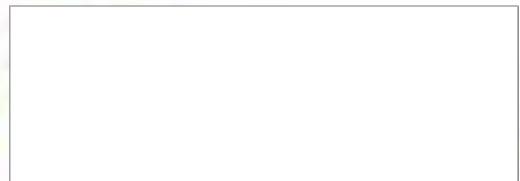
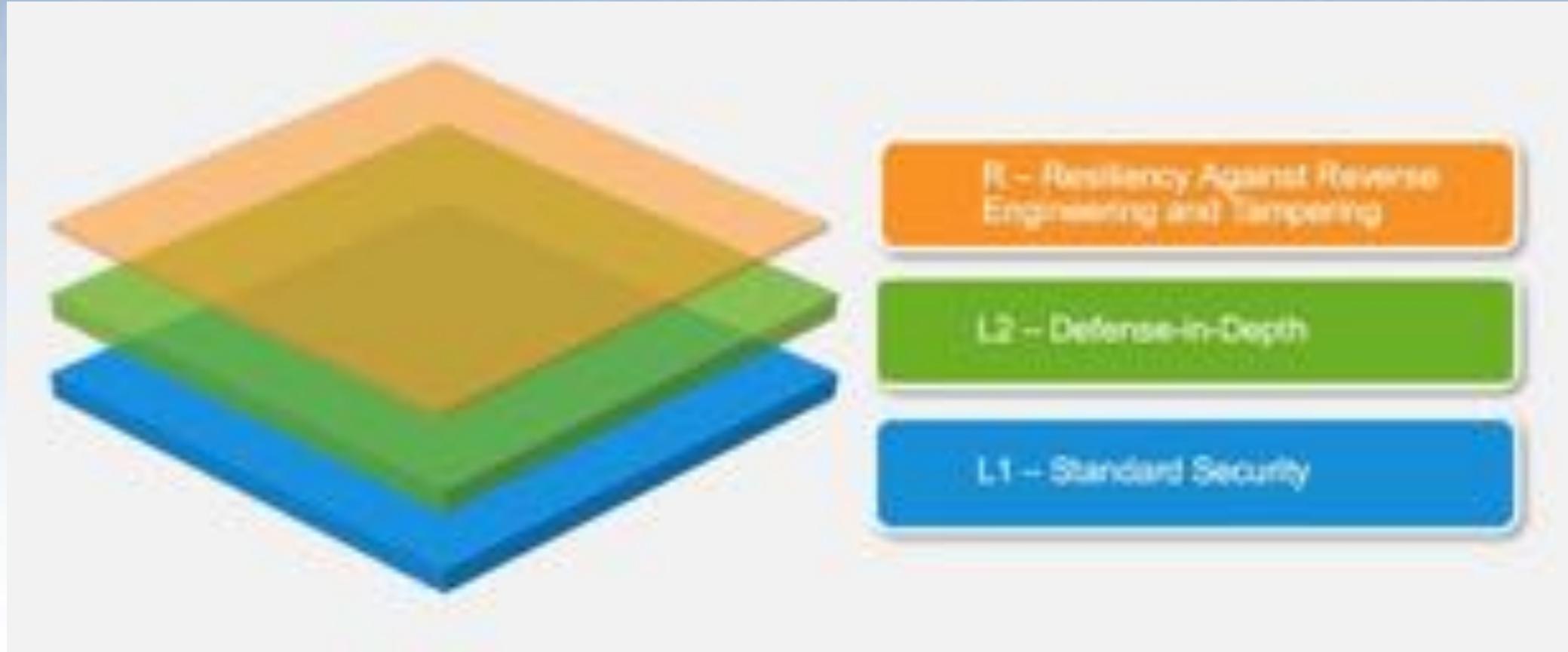


OWASP Mobile AppSec Verification Standard (MASVS)

- Started as a fork of the OWASP ASVS
- Formalizes best practices and other security requirements
- Mobile-specific, high-level, OS-agnostic
- Why?
 - Shift left: give security requirements a-priori



OWASP Mobile AppSec Verification Standard (MASVS)



OWASP Mobile AppSec Verification Standard (MASVS)

V2: Data Storage and Privacy Requirements

#	Description	L1	L2
2.1	System credential storage facilities are used appropriately to store sensitive data, such as user credentials or cryptographic keys.	✓	✓
2.2	No sensitive data is written to application logs.	✓	✓
2.3	No sensitive data is shared with third parties unless it is a necessary part of the architecture.	✓	✓
2.4	The keyboard cache is disabled on text inputs that process sensitive data.	✓	✓
2.5	The clipboard is deactivated so that selection may contain sensitive data.	✓	✓
2.6	No sensitive data is exposed via IPC mechanisms.	✓	✓
2.7	No sensitive data, such as passwords or pins, is exposed through the user interface.	✓	✓
2.8	No sensitive data is included in backups generated by the mobile operating system.		✓

How to use the MASVS?

During early stages of development:

- Basis for (future) design decisions and enhancements
- Helps building internal baselines for Mobile Security and Coding Guidelines
- To determine security requirements early on. For example:

1.3	Security controls are never enforced only on the client side, but on the respective remote endpoints.	✓	✓
-----	---	---	---

While Implementing:

- Track the security requirements during development
- Redefine security requirements when business requirements are changing

During Penetration Test:

- Share the status of your security requirements with the tester



Current status MASVS

- Current release: 1.1 (English)
- Translations:
 - Released: Spanish, Russian
 - Ready: French, German, Japanese
 - In progress: Chinese (ZHTW)
 - Started: Persian



Current status MASVS

- Current release: 1.1
- Translations
- Lab-project status!



Current status MASVS

- Current release: 1.1
- Translations
- Lab-project status!
- NIST 800-163, revision 1

Draft NIST Special Publication 800-163
Revision 1

Vetting the Security of Mobile Applications

Michael Ogata
Josh Franklin
Jeffrey Voss
Vincent Srinivasan
Stephen Quinligan

COMPUTER SECURITY

Current status MASVS

Project Lead	Lead Author	Contributors and Reviewers
Sven Schleier & Jeroen Willemsen	Bernhard Mueller	Alexander Antukh, Mesheryakov Aleksey, Bachevsky Artem, Jeroen Beckers, Vladislav Chelnokov, Ben Cheney, Stephen Corbiaux, Manuel Delgado, Ratchenko Denis, Ryan Dewhurst, Tereshin Dmitry, Christian Dong, Oprya Egor, Ben Gardiner, Rocco Gränitz, Henry Hu, Sjoerd Langkemper, Vinícius Henrique Marangoni, Martin Marsicano, Roberto Martelloni, Gall Maxim, Rio Okada, Abhinav Sejpal, Stefaan Seys, Yogesh Shamrma, Prabhant Singh, Nikhil Soni, Anant Shrivastava, Francesco Stillavato, Romuald SZKUDLAREK, Abdessamad Temmar, Koki Takeyama, Chelnokov Vladislav

Future plans for the MASVS

- Ongoing: Integration with SKF
- Ongoing: Automate & simplify releases
- Ongoing conversations with the Cloud Security Alliance.
- Revisit Location & Connectivity requirements
- Re-evaluate the need for payload encryption
- Add more translations

Your turn!

- <https://github.com/OWASP/owasp-masvs>
- <https://mobile-security.gitbook.io/masvs/>



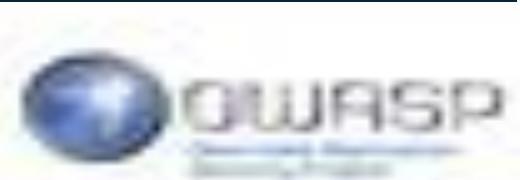
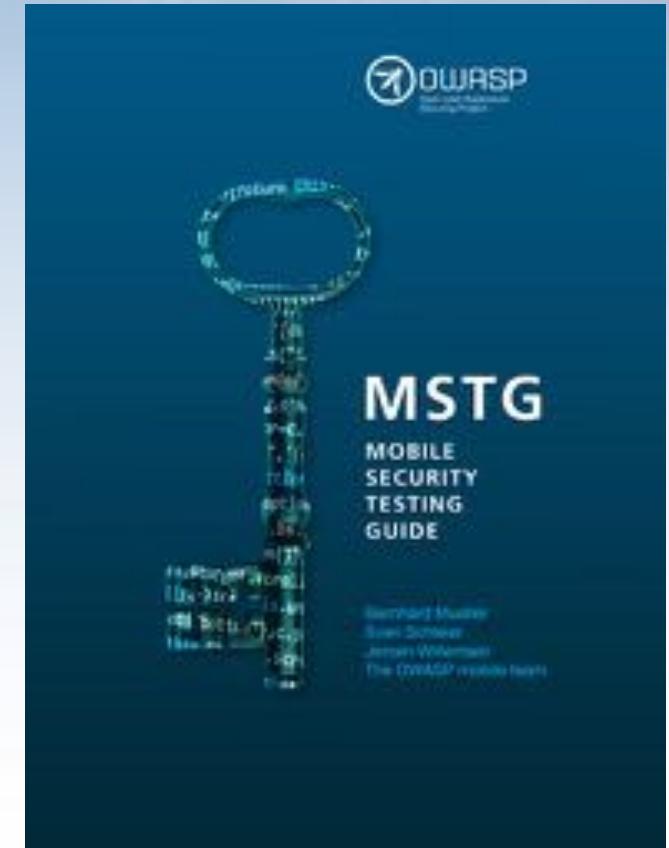
- ✓ Download it
- ✓ Read it
- ✓ Use it
- ✓ Give Feedback! Create an issue or a PR
- ✓ Tweet about it (@OWASP_MSTG)

Agenda

- Introduction into the MASVS
- **Introduction into the MSTG**
- Some examples

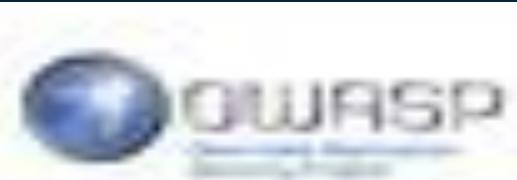
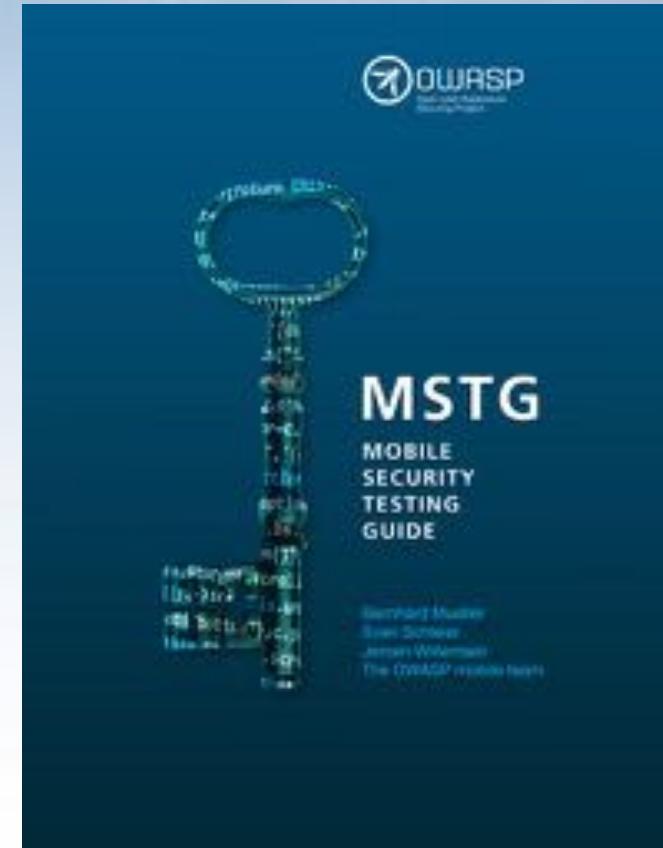
OWASP Mobile Security Testing Guide (MSTG)

- Manual for testing security maturity of iOS and Android (mostly) native apps.
- Maps on MASVS requirements.
- Why?
 - Educate developers and penetration testers.
 - Provide a baseline for automated checks



OWASP Mobile Security Testing Guide (MSTG)

- General testing guide
- Android Testing guide
- iOS Testing guide



OWASP Mobile Security Testing Guide (MSTG)

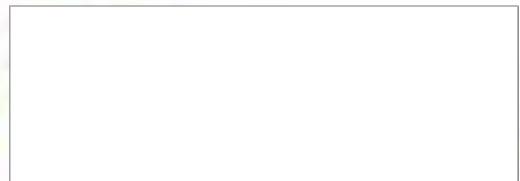
- General testing guide
- Android Testing guide
- iOS Testing guide
- Crackme's & Challenges



Kudos to Bernhard Mueller @bernhardm for his hard work!

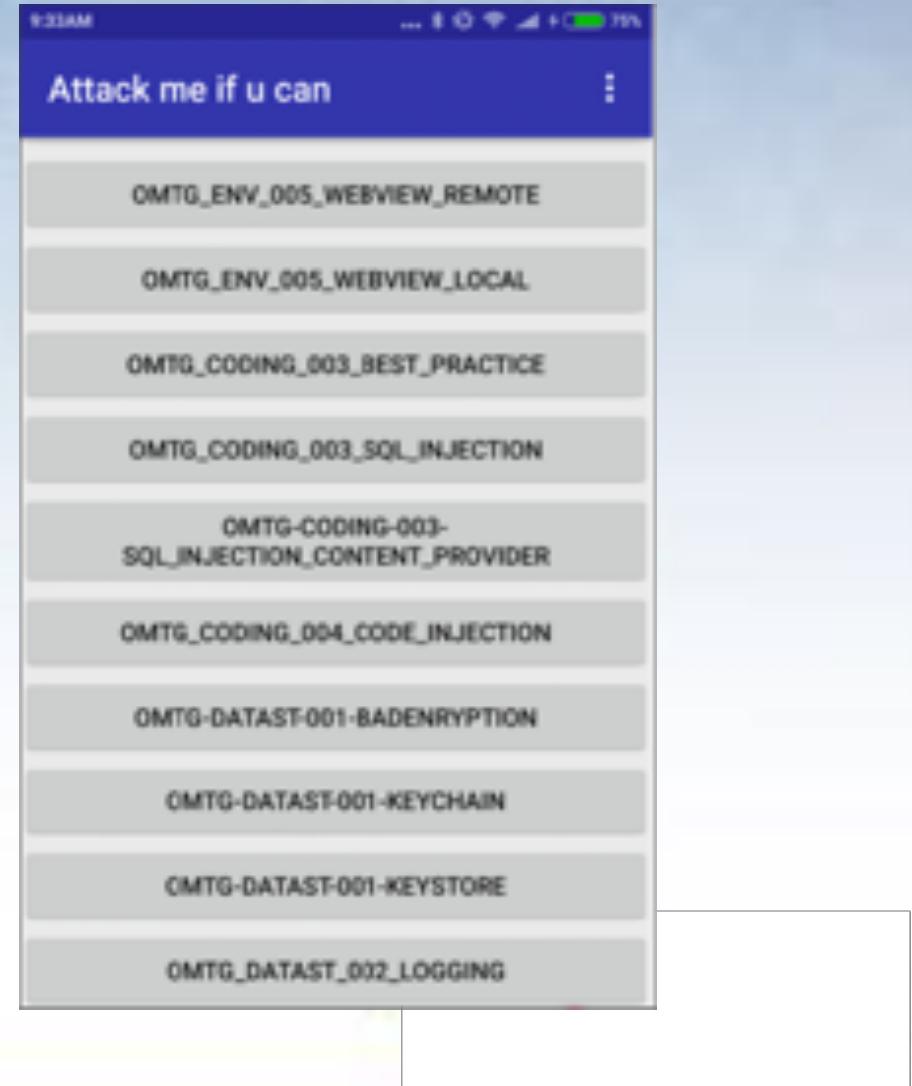
OWASP Mobile Security Testing Guide (MSTG)

- General testing guide
- Android Testing guide
- iOS Testing guide
- Crackme's & Challenges
- Mobile Appsec Checklist



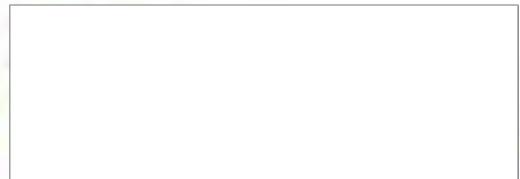
OWASP Mobile Security Testing Guide (MSTG)

- General testing guide
- Android Testing guide
- iOS Testing guide
- Crackme's & Challenges
- Mobile Appsec Checklist
- MSTG playground (External)



Current status MSTG

- We JUST released 1.1.0 TODAY!!!
- Lab-project & Mentioned in NIST 800-163, revision 1, 3K+ stars
- Automation: Simplified Crackme maintenance & document generation

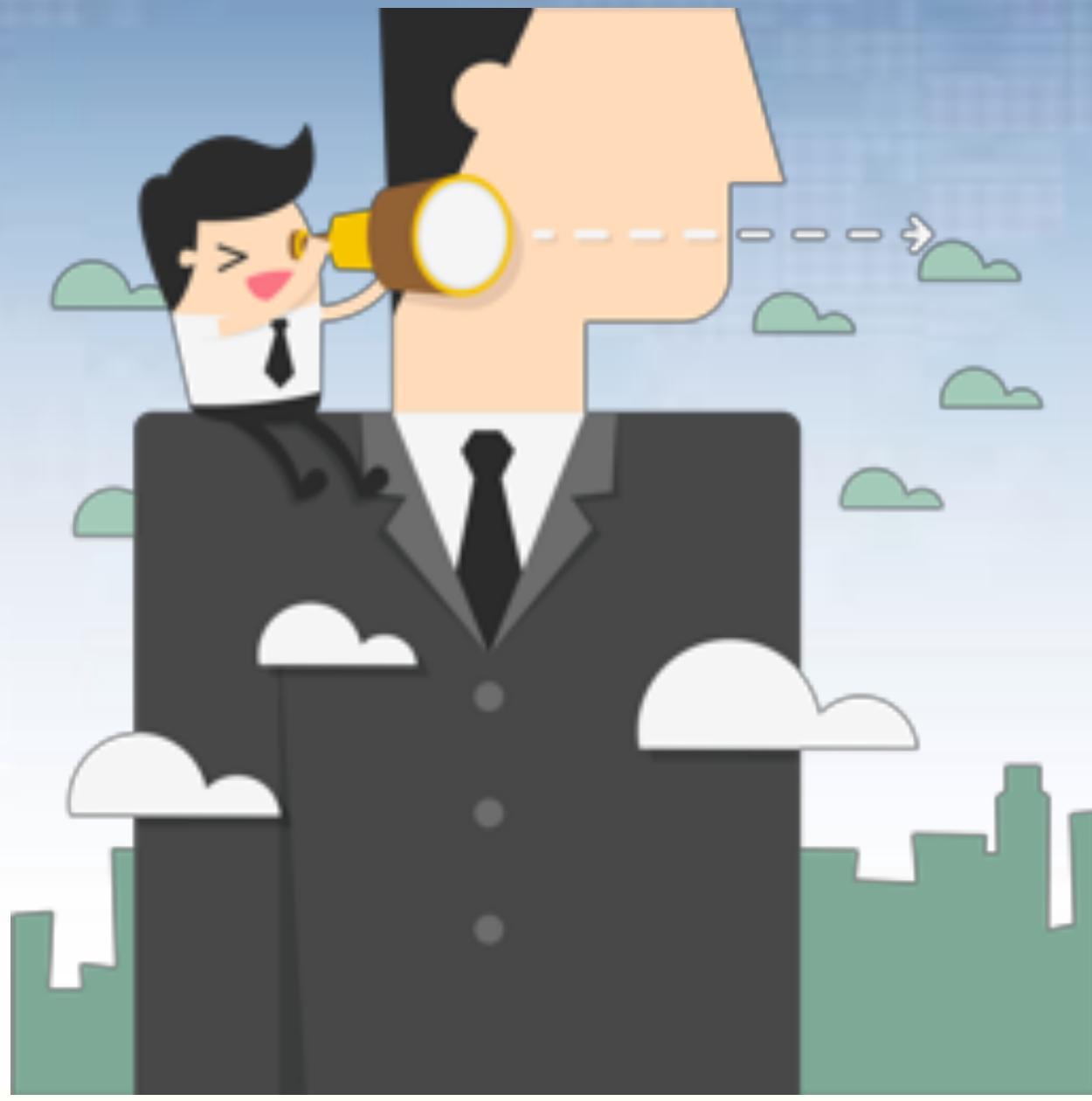


Current status MSTG

Authors	Co-Authors	Top Contributors	Reviewers	Editors
Bernhard Mueller Jeroen Willemsen (@jeroenwillemsen) Sven Schleier (@sushi2k)	Romuald Szkudlarek	Pawel Rzepa Francesco Stillavato Andreas Happe Alexander Anthuk Henry Hoggard Wen Bin Kong Abdessamad Temmar Bolot Kerimbaev Slawomir Kosowski	Sjoerd Langkemper Anant Shrivastava	Heaven Hodges Caitlin Andrews Nick Epson Anita Diamond Anna Szkudlarek

The full list of contributors is available on GitHub:

<https://github.com/OWASP/owasp-mstg/graphs/contributors>



Ongoing work for MSTG

- Adding code samples in Swift and Kotlin
- Adding Android 8/9 & iOS 12 updates (ongoing for 1.2)
- Translation to Japanese & Russian (ongoing)
- Getting hardcopies available

Future plans MSTG

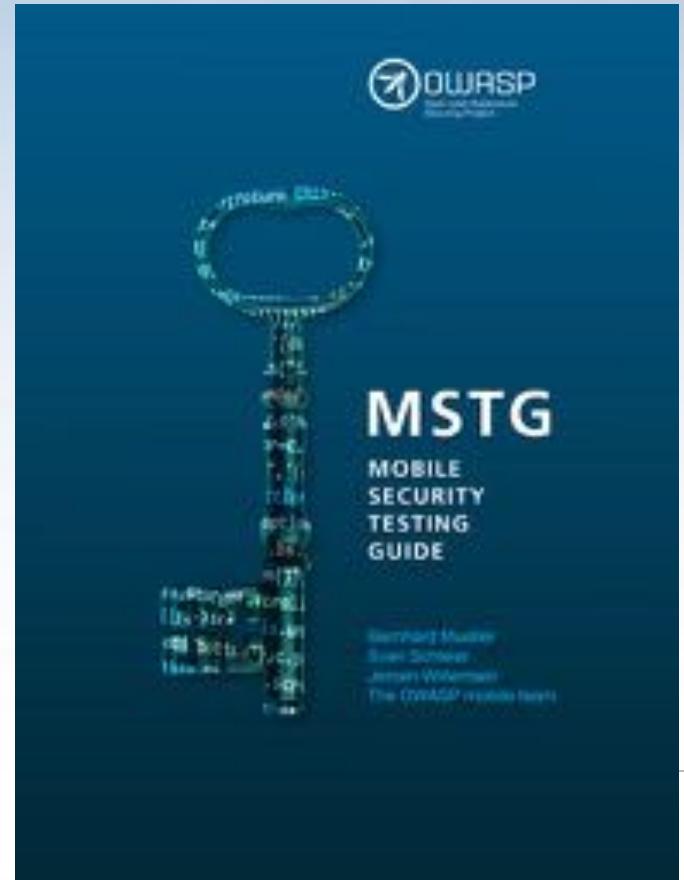
- Migrate crackmes and MSTG playground to one repository and develop more bad/good examples
- Restructure the MSTG to align with the MASVS
- Consider MDM write-ups (version 1.3)?
- Add more crackme exercises for iOS
- Seek collaboration with Apple / Google to speed up ?
- Collaborate with standardization bodies



Your turn!

- <https://github.com/OWASP/owasp-mstg>
<https://mobile-security.gitbook.io/mstg/>

- ✓ Download it
- ✓ Read it
- ✓ Use it
- ✓ Give Feedback (file an issue)
- ✓ **Fix issues: send in your Pull Requests!**
- ✓ Tweet about it (@OWASP_MSTG)



Agenda

- Introduction into the MASVS
- Introduction into the MSTG
- **Some examples**

Let's not repeat ourselves!

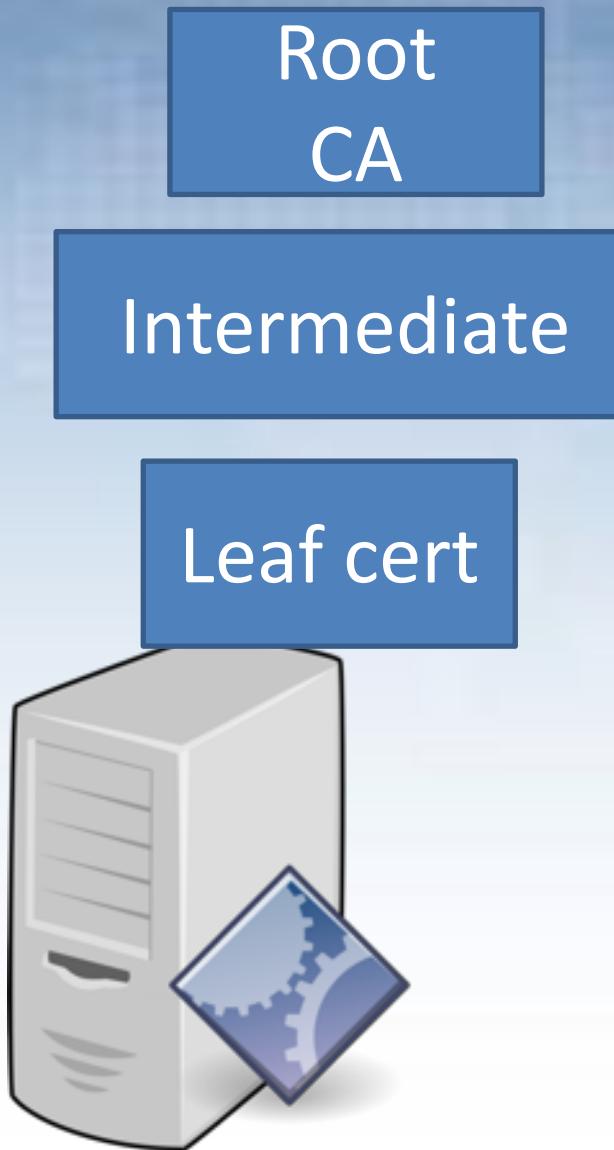
This happened yesterday:

Training 3 - Android security workshop by Jeroen Beckers & Stephanie Vanroelen

Let's give some love to iOS!



SSL pinning

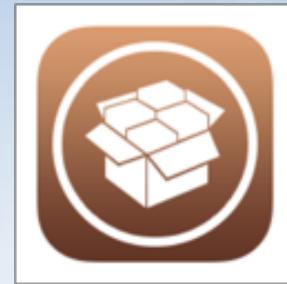


Version
Certificate Serial Number
Certificate Algorithm
Identifier for Certificate Issuer's Signature
Issuer
Validity Period
Subject
Subject Public-Key Information
Algorithm Identifier
Public-key Value
Issuer Unique Identifier
Subject Unique Identifier
Extensions
Certification Authority's Digital Signature

SSL pinning – SSL killswitch V2

Two easy ways to break most pinners:

1. Jailbreak → use Cydia & SSL Killswitch V2



2. Do dynamic instrumentation on a non-jailbroken device

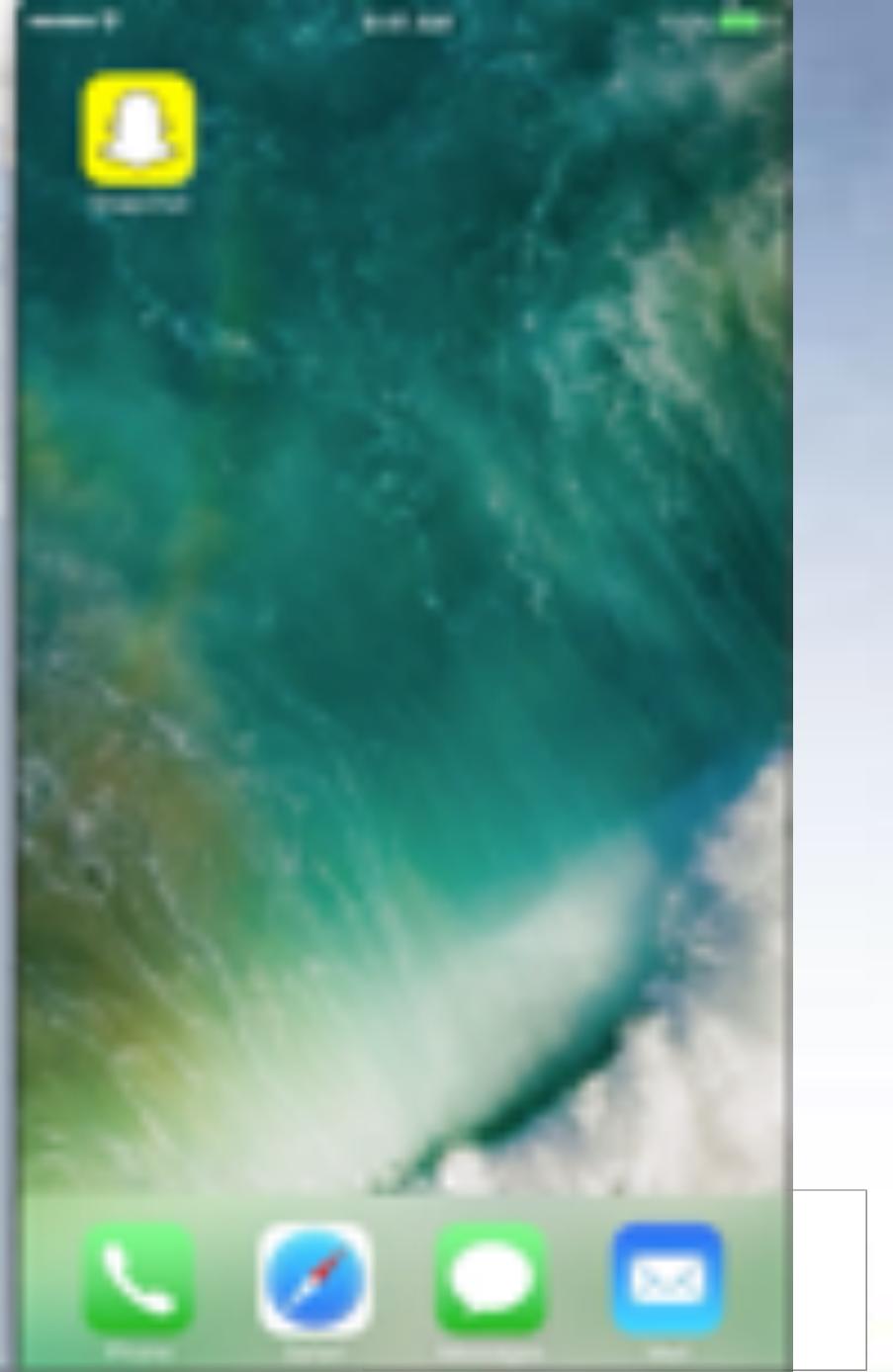


See <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x04f-Testing-Network-Communication.md>

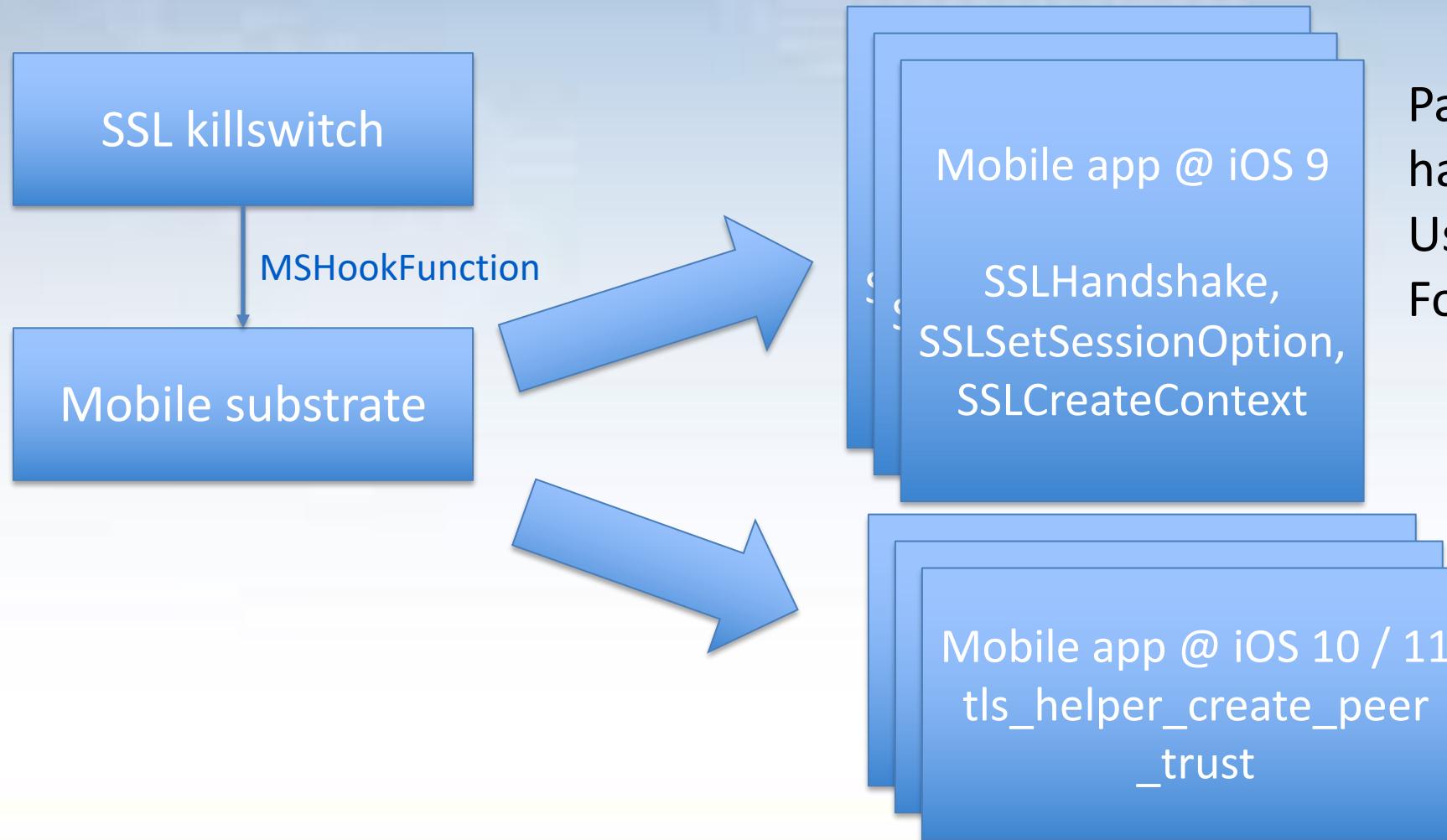
and <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06g-Testing-Network-Communication.md>



HTTP Request History							
Timestamp	Method	URL	Request	Response	Headers	Body	Actions
2023-03-10 10:00:00	GET	/index.html	GET /index.html	200 OK	Content-Type: text/html; charset=UTF-8	[REDACTED]	View Details
2023-03-10 10:00:10	POST	/submit-form	POST /submit-form	200 OK	Content-Type: application/json	[REDACTED]	View Details
2023-03-10 10:00:20	GET	/api/data	GET /api/data	200 OK	Content-Type: application/json	[REDACTED]	View Details



SSL pinning – SSL killswitch V2



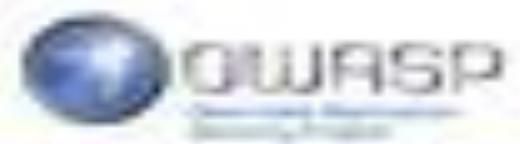
Patch underlying SSL handshake implementation
Used by NSURLConnection
For all apps...

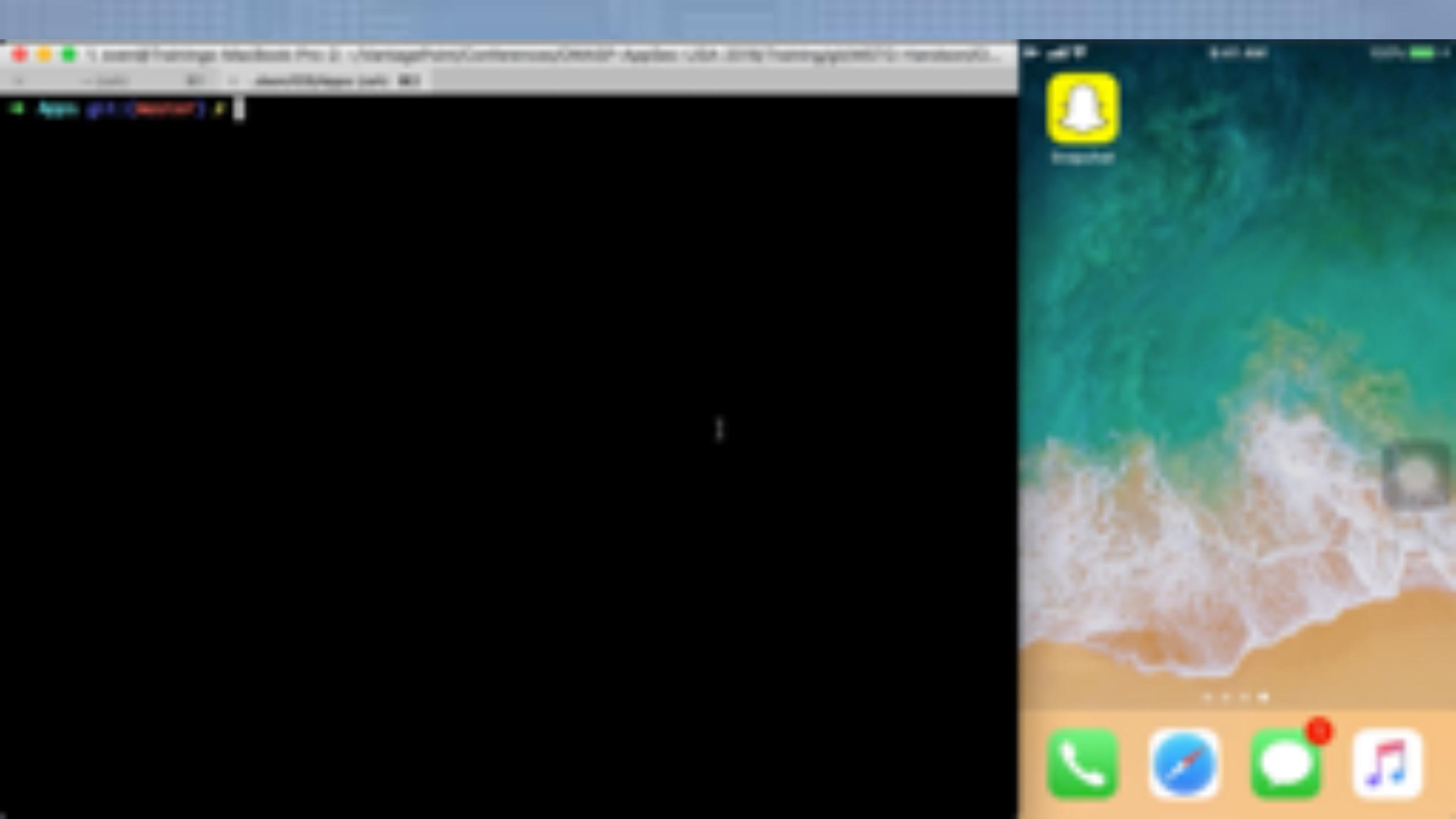
What if you don't want to jailbreak?

- Jailbroken devices require maintenance
- Jailbreaks are getting harder to find
- What about jailbreak protection of the app?
- Let's patch the app itself!



FЯIDA





SSL pinning – Objection



Patch underlying SSL
handshake implementation
Used by NSURLConnection
For one app.

1. Frida server in Gadget waits
2. Objection connects to server with explore REPL
3. Objection calls script that patches underlying SSL handshake implementation

TouchID the wrong way: using LAContext

There are 2 ways to use TouchID:

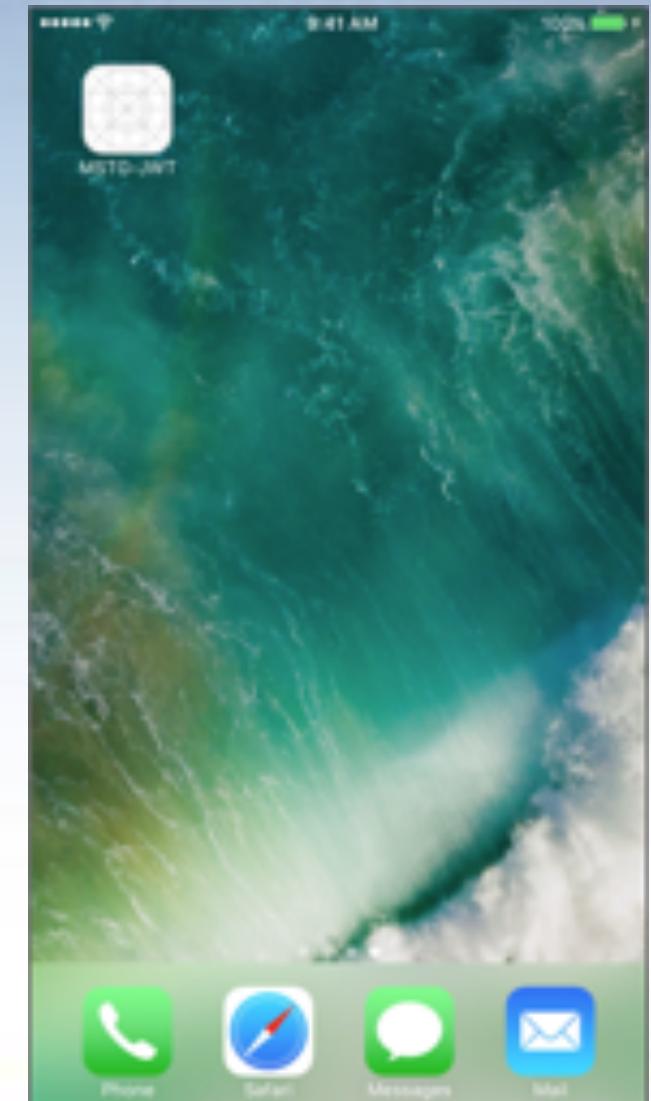
1. Protect an entry in the keychain and unlock it via TouchID

2. Use the LocalAuthenticationContext :

```
LocalAuthenticationContext.evaluatePolicy(.deviceOwnerAuthenticationWithBiometrics, localizedReason: reasonString) {  
    success, evaluateError in  
}
```

```
If success {  
    successmethods()  
} else {  
    ....  
}
```

What if we call the
successmethods() directly?



Bypassing Touch-ID

- With **needle**



- With

- Both cases: use Frida to hook onto
`evaluatePolicy:localizedReason:reply`
 - Ensures that when *evaluatePolicy* is called that the reply its success is set to true (E.g.: call success methods)

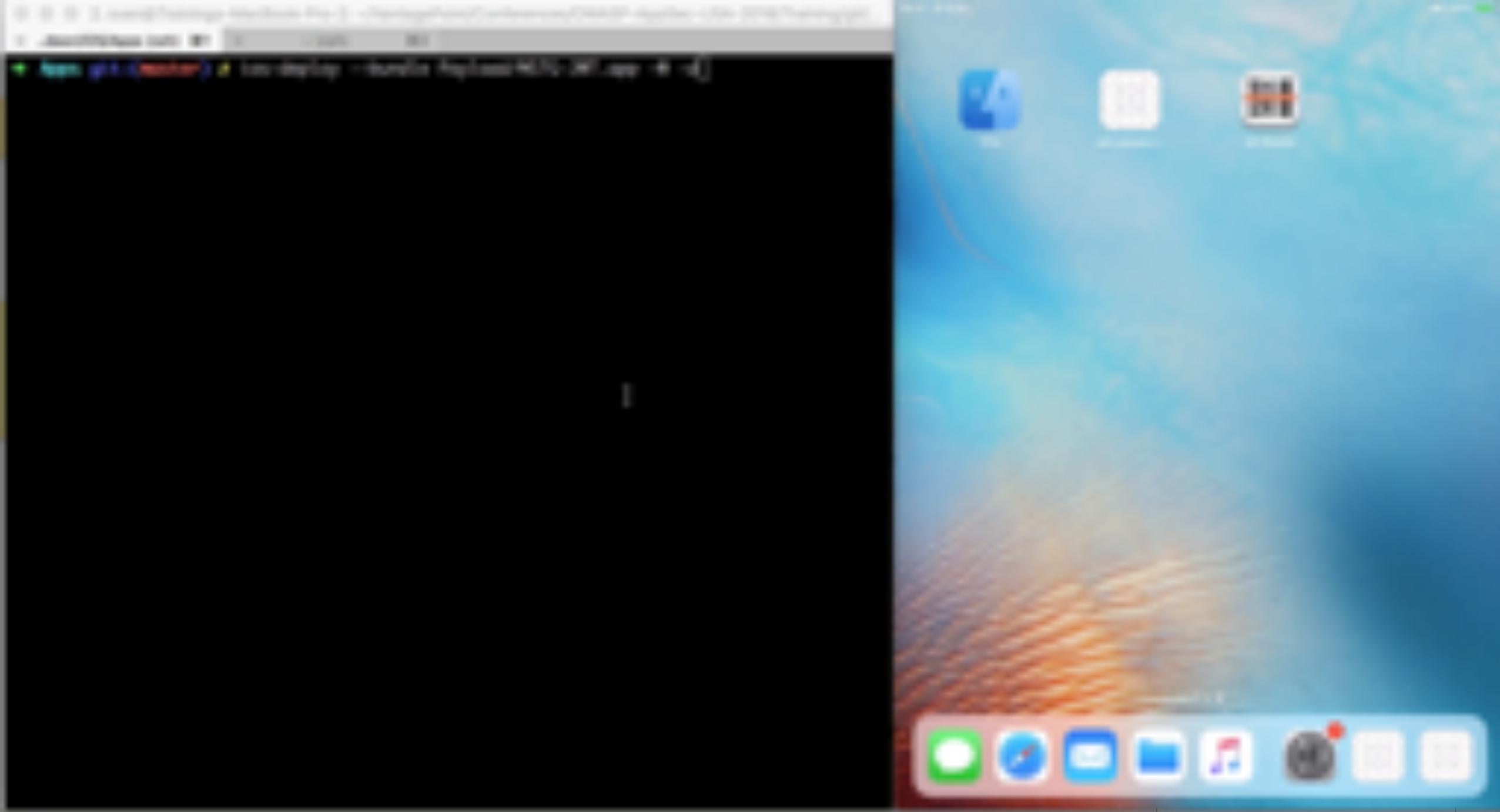
See <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06f-Testing-Local-Authentication.md>



```
 ④ macbook-Pro:~ t.pearce$ ./needle
```

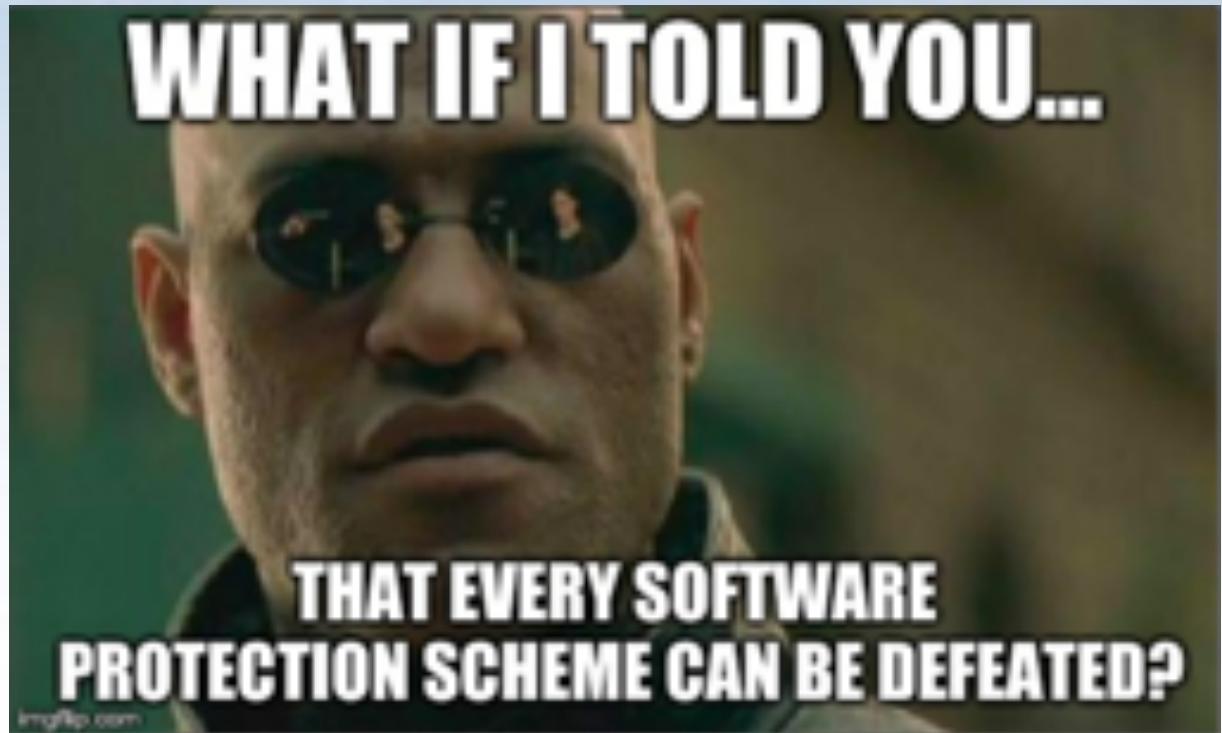
```
needle v1.0.5
IP: 192.168.0.198

Listening
Stopped Listening
Client Disconnected
[1027.0.0.1] DPCODE: init_apps
[1027.0.0.1] DPCODE: init_version
[1027.0.0.1] DPCODE: init_version
New connection from: 1027.0.0.1
Listening
Stopped Listening
Listening
Client Disconnected
Stopped Listening
[1027.0.0.1] DPCODE: init_apps
[1027.0.0.1] DPCODE: init_version
[1027.0.0.1] DPCODE: init_version
New connection from: 1027.0.0.1
Listening
Stopped Listening
Client Disconnected
Client Disconnected
A client is already connected, rejecting new
connection request from: 1027.0.0.1
[1027.0.0.1] DPCODE: init_version
New connection from: 1027.0.0.1
```



There is much more!

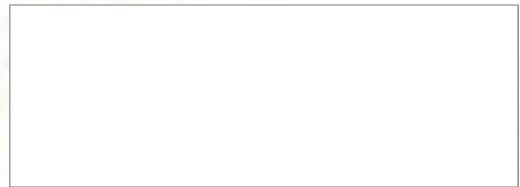
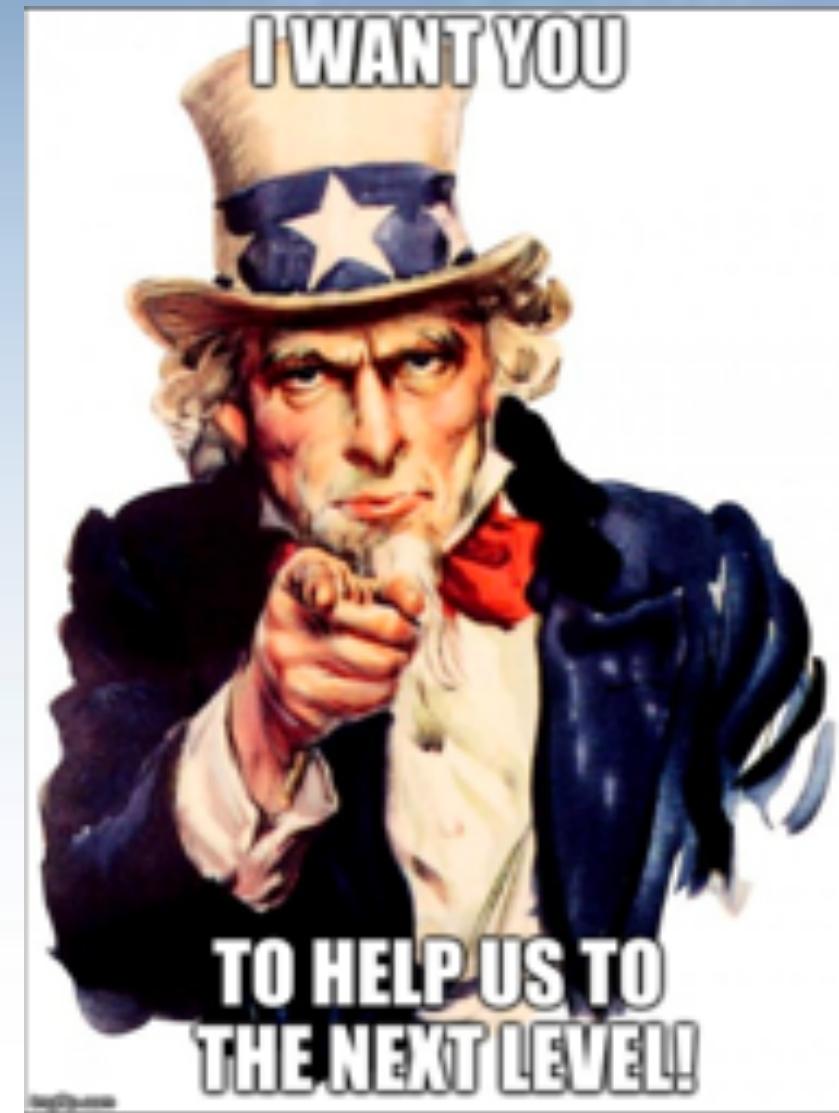
- Reverse Engineering
 - ✓ Root / Jailbreak Detection
 - ✓ Anti-Debugging
 - ✓ Detecting Reverse Engineering Tools
 - ✓ Emulator Detection / Anti-Emulation
 - ✓ File and Memory Integrity Checks
 - ✓ Device Binding
 - ✓ Obfuscation



There is much more!

- Reverse Engineering
- Analysis & best practices for
 - Storage
 - Cryptography
 - Local Authentication
 - Network Communication
 - Code quality & build settings







QUESTIONS?

@OWASP_MSTG

jeroen.willemsen@owasp.org



THANK YOU!

@OWASP_MSTG

jeroen.willemsen@owasp.org

