KLIK OM TE STARTEN

Fast forwarding Mobile Security with the MSTG

Jeroen Willemsen – XSECCON Gamma

About me

Jeroen Willemsen
@commjoenie
jeroen.willemsen@owasp.org
“Security architect”
“Full-stack developer”
“Mobile security”

@OWASP_MSTG

Xebia Security

OWASP
Open Web Application Security Project

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:

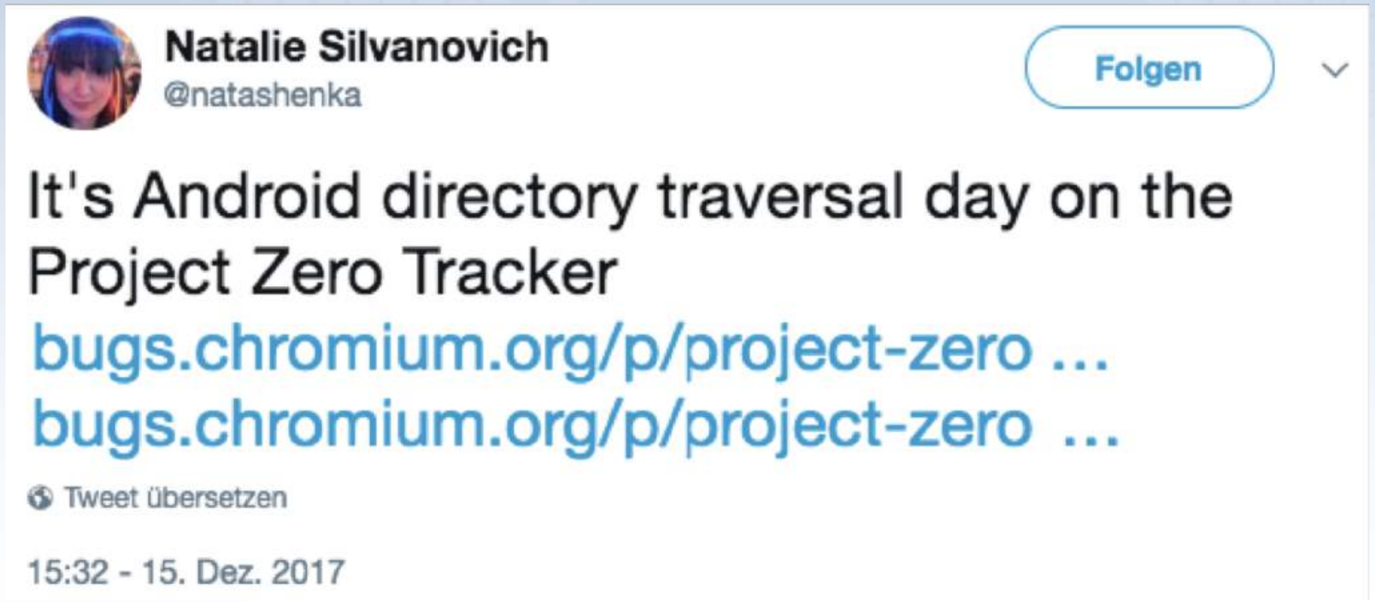
XSS: No,

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

OWASP
Open Web Application Security Project

The MSTG: mobile security?

- So CSRF and XSS do not easily apply.



Natalie Silvanovich
@natashenka

Folgen

It's Android directory traversal day on the Project Zero Tracker
bugs.chromium.org/p/project-zero …
bugs.chromium.org/p/project-zero …

Tweet übersetzen

15:32 - 15. Dez. 2017

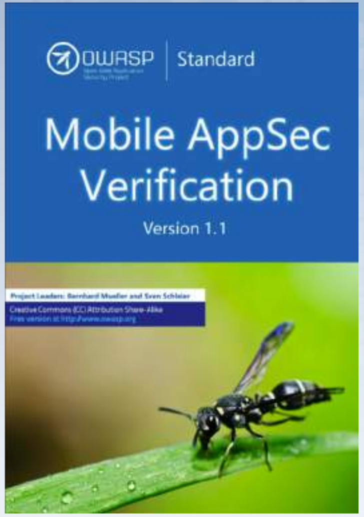
- But path-traversals do…

The MSTG: mobile security?

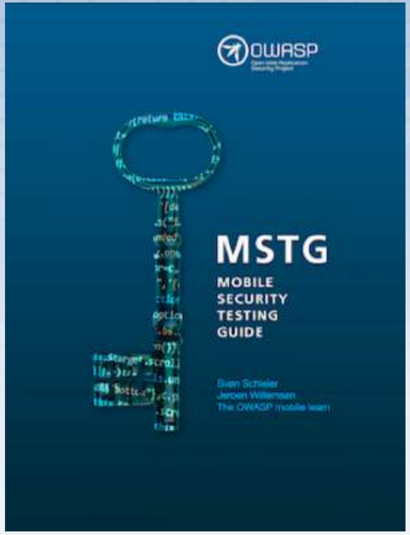
- So CSRF and XSS do not easily apply.
- But path-traversals do…
- And then there is… Data leakage
  - through logging,
  - through insecure storage,
  - Through IPC.
- What about weak authentication mechanisms?
- What about reverse engineering?

OWASP
Open Web Application Security Project

How do we fix this?



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



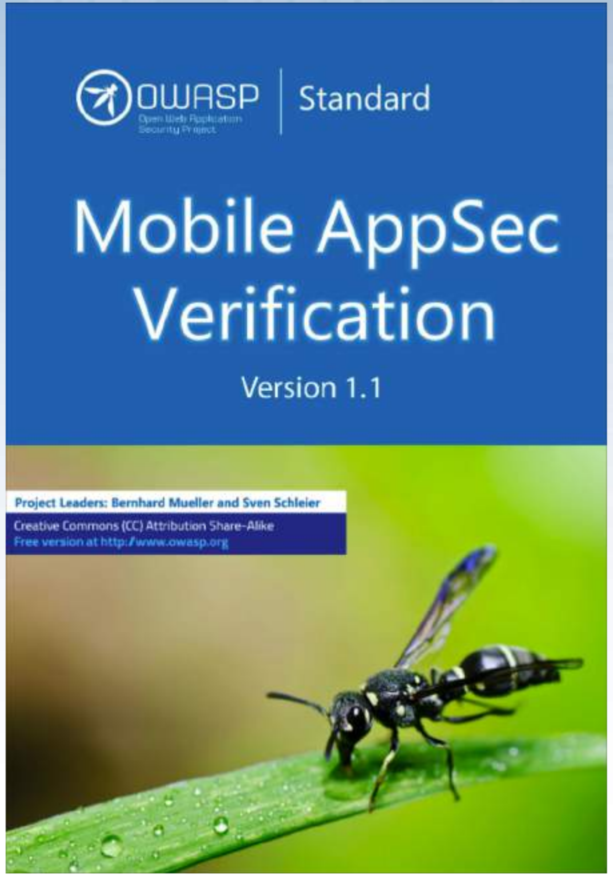
Mobile Security Testing Guide
https://github.com/OWASP/owasp-mstg

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 Open Web Application Security Project | Standard

Mobile AppSec Verification

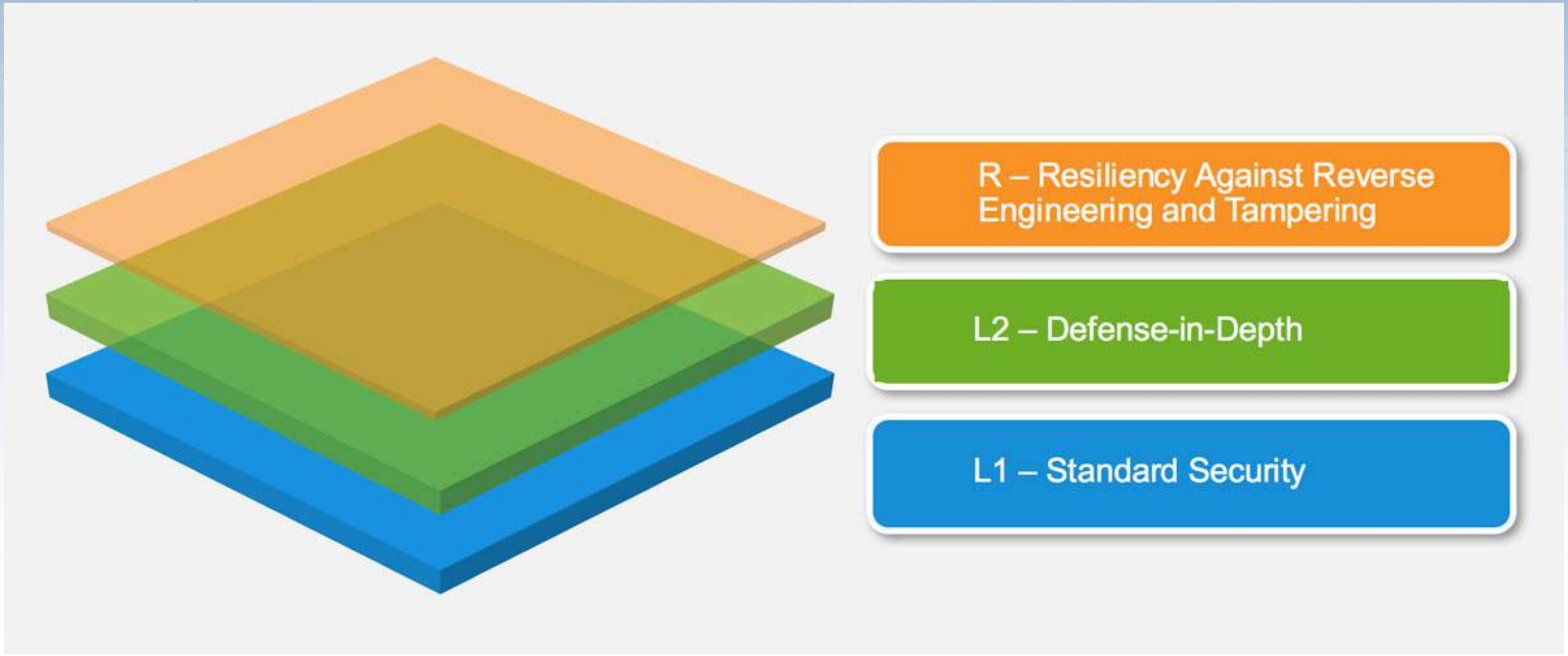
Version 1.1

Project Leaders: Bernhard Mueller and Sven Schleier

Creative Commons (CC) Attribution Share-Alike
Free version at http://www.owasp.org

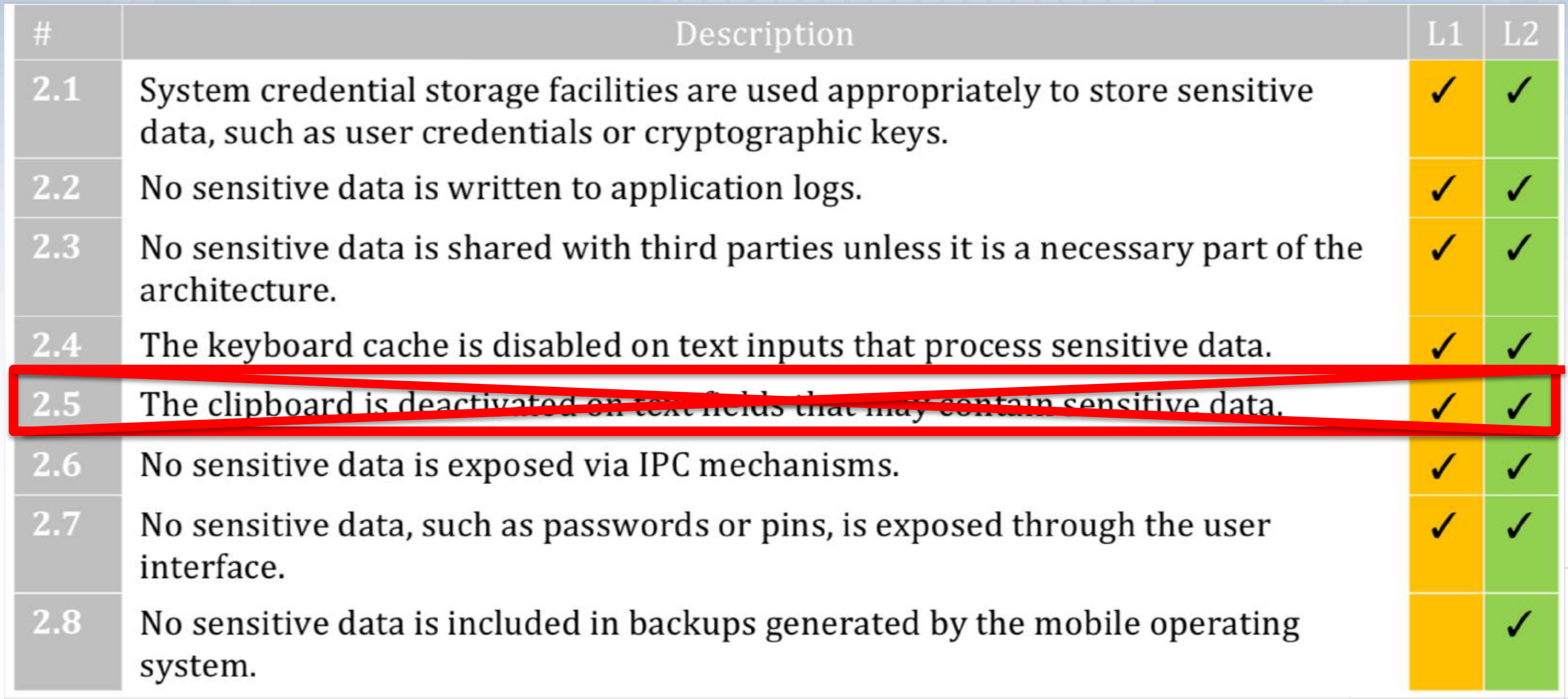
OWASP
Open Web Application Security Project

OWASP Mobile AppSec Verification Standard (MASVS)



OWASP Mobile AppSec Verification Standard (MASVS)

V2: Data Storage and Privacy Requirements

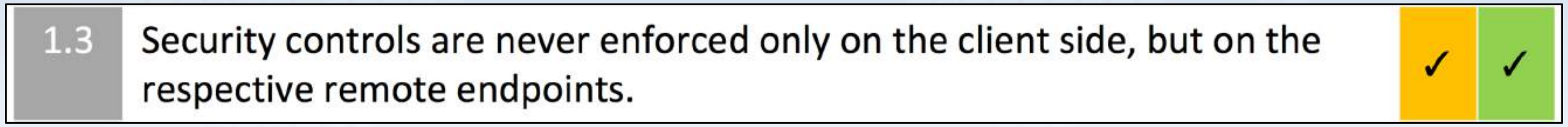


#	Description	L1	L2
2.1	System credential storage facilities are used appropriately to store sensitive data, such as user credentials or cryptographic keys.	✓	✓
2.2	No sensitive data is written to application logs.	✓	✓
2.3	No sensitive data is shared with third parties unless it is a necessary part of the architecture.	✓	✓
2.4	The keyboard cache is disabled on text inputs that process sensitive data.	✓	✓
~~2.5~~	~~The clipboard is deactivated on text fields that 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

OWASP
Open Web Application Security Project

Current status MASVS

- Current release: 1.1.3
- Translations: Spanish, Russian, French, German, Japanese, Chinese (ZHTW)
  - Started: Persian

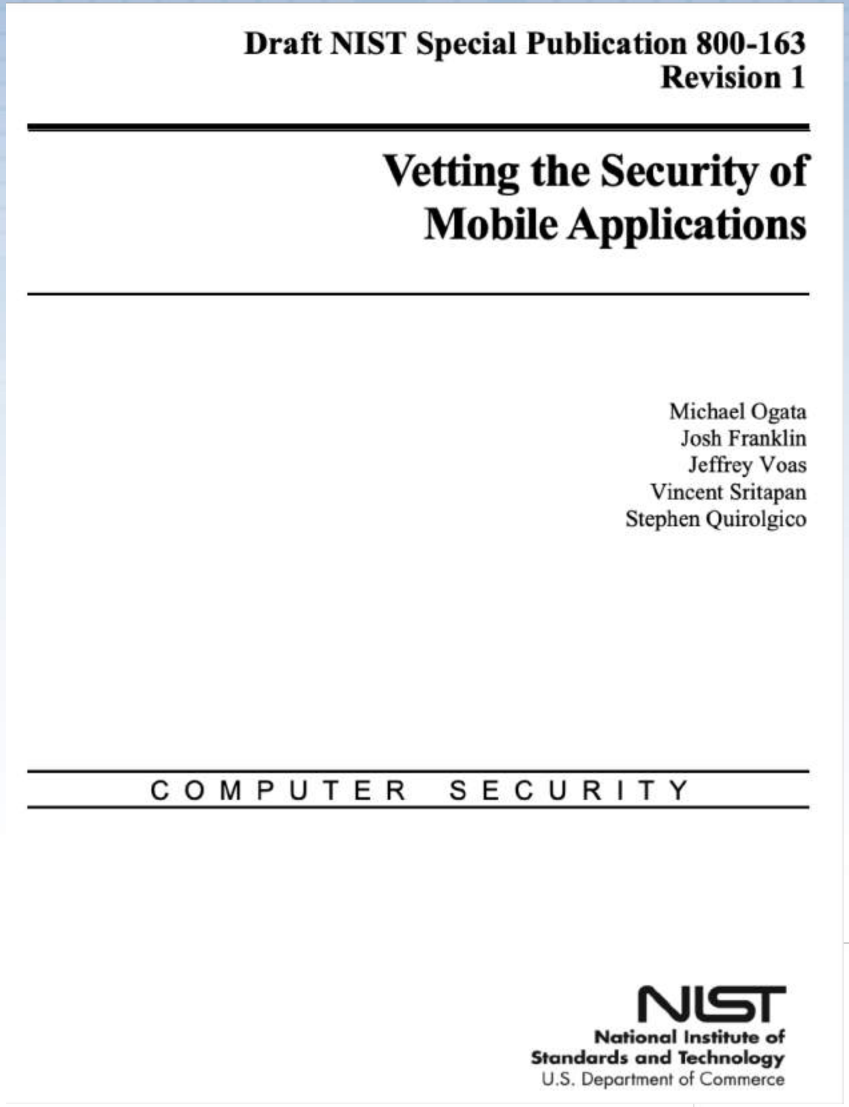
YOU KNOW WHAT GRINDS MY GEARS?
WHAT REALLY GRINDS MY GEARS
WE HAVE 8 LANGUAGES, BUT WOULD LOVE TO GET DUTCH IN AS WELL!!
imgflip.com

Current status MASVS

- Current release: 1.1.3
- Translations
- Lab-project status!

Current status MASVS

- Current release: 1.1.3
- 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 Voas
Vincent Sritapan
Stephen Quirolgico

COMPUTER SECURITY

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

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, Peter Chi, Lex Chien, 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, Riotaro Okada, Abhinav Sejpal, Stefaan Seys, Yogesh Shamrma, Prabhant Singh, Nikhil Soni, Anant Shrivastava, Francesco Stillavato, Romuald SZKUDLAREK, Abdessamad Temmar, Koki Takeyama, Chelnokov Vladislav, Leo Wang

Future plans for the MASVS

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

OWASP
Open Web Application Security Project

Your turn!

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

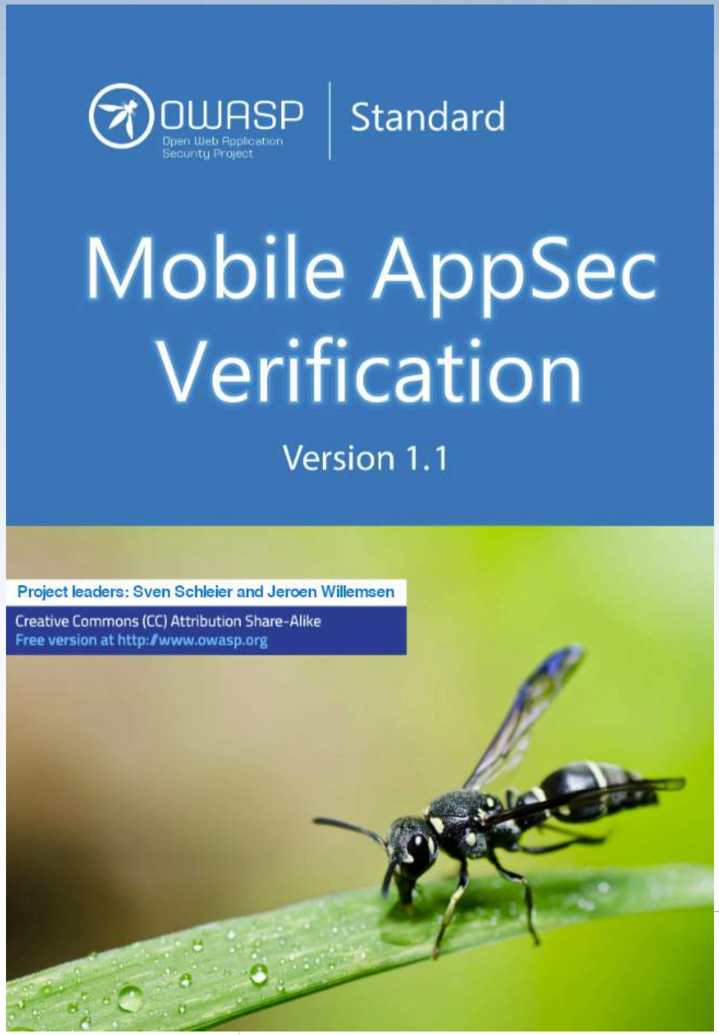
✓ Download it

✓ Read it

✓ Use it

✓ Give Feedback! Create an issue or a PR

✓ Tweet about it (@OWASP_MSTG)


OWASP Open Web Application Security Project | Standard

Mobile AppSec Verification

Version 1.1

Project leaders: Sven Schleier and Jeroen Willemsen

Creative Commons (CC) Attribution Share-Alike
Free version at http://www.owasp.org

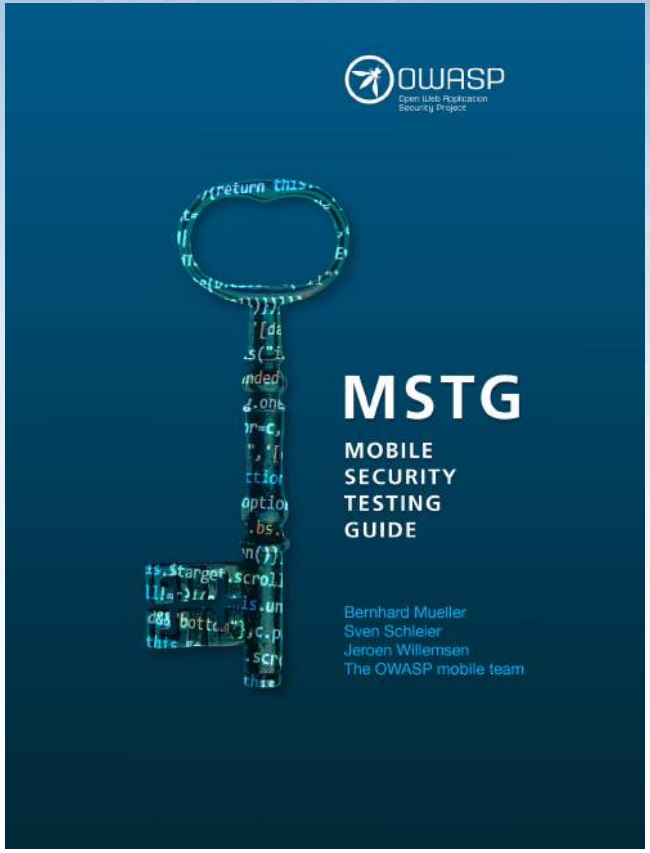
Agenda

- 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.

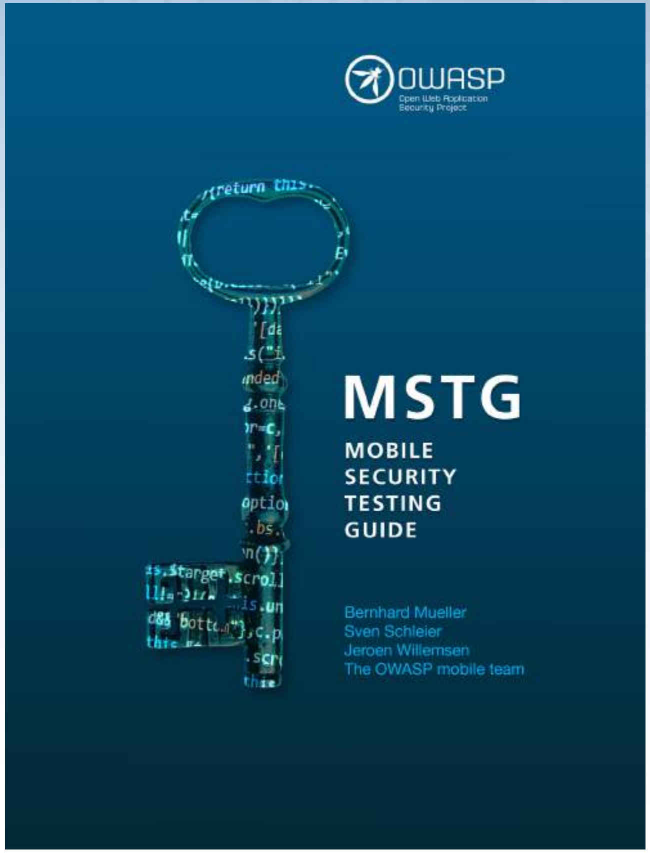
- Why?
  - Educate developers and penetration testers.
  - Provide a baseline for automated checks



OWASP Open Web Application Security Project

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

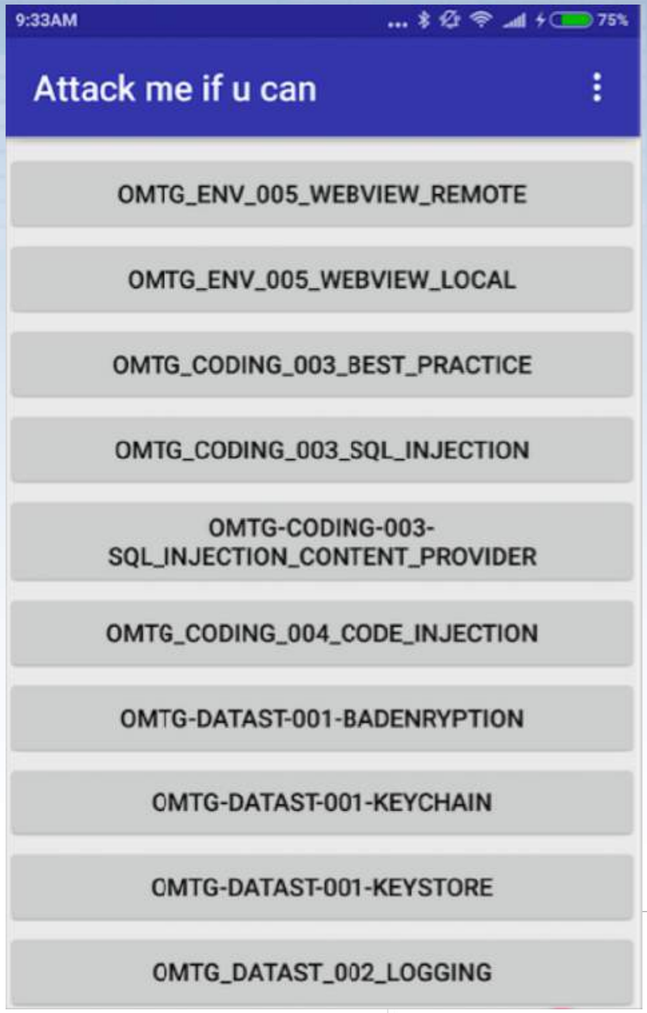
Kudos to Bernhard Mueller @bernhardm for his hard work!

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)


9:33AM
75%
Attack me if u can
OMTG_ENV_005_WEBVIEW_REMOTE
OMTG_ENV_005_WEBVIEW_LOCAL
OMTG_CODING_003_BEST_PRACTICE
OMTG_CODING_003_SQL_INJECTION
OMTG-CODING-003-SQL_INJECTION_CONTENT_PROVIDER
OMTG_CODING_004_CODE_INJECTION
OMTG-DATAST-001-BADENRYPTION
OMTG-DATAST-001-KEYCHAIN
OMTG-DATAST-001-KEYSTORE
OMTG_DATAST_002_LOGGING

Current status MSTG

- Version 1.1.0
- 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

Open Web Application Security Project

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)

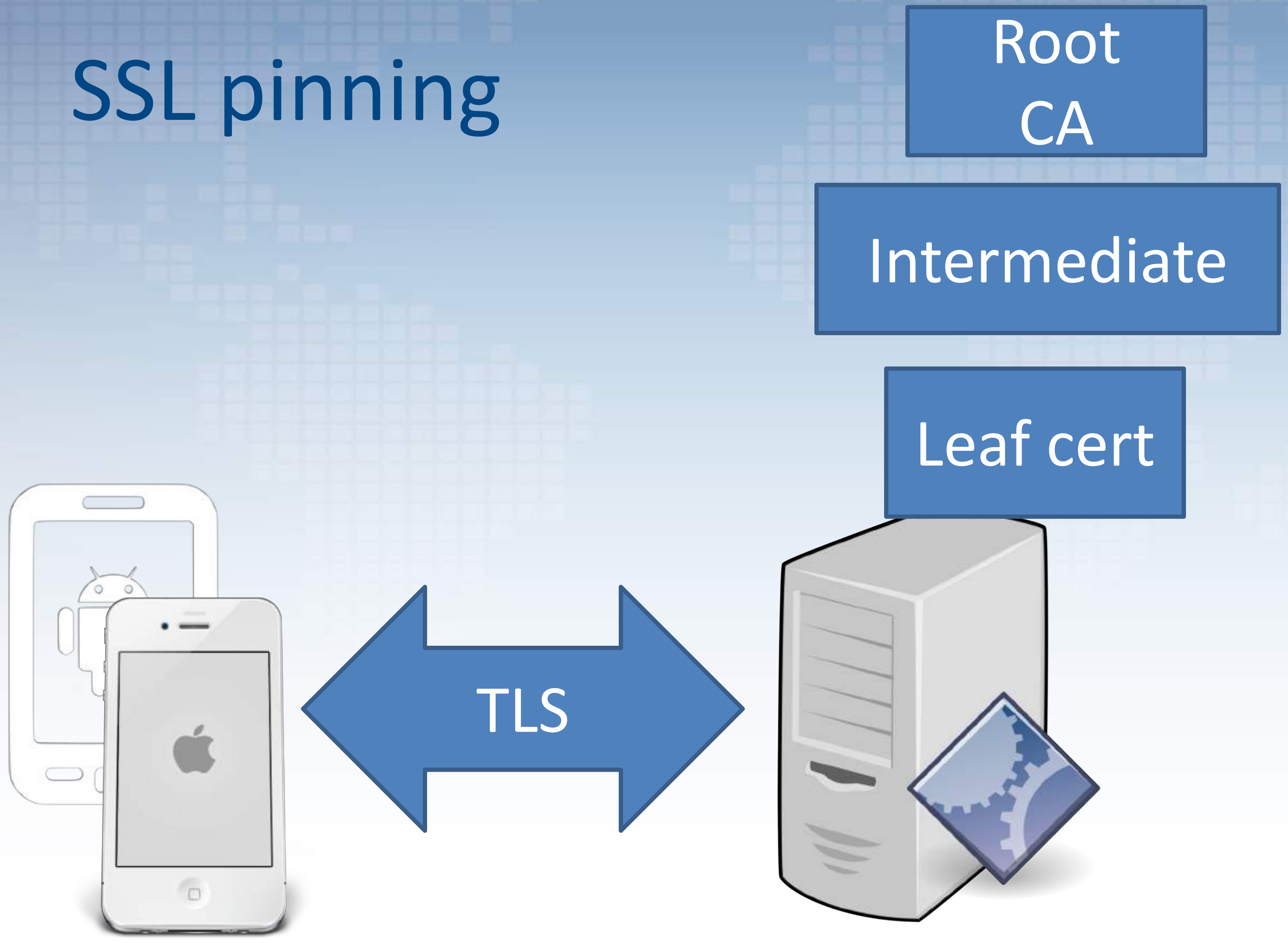
✓ **Fix issues: send in your Pull Requests!**

✓ Tweet about it (@OWASP_MSTG)

Agenda

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

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

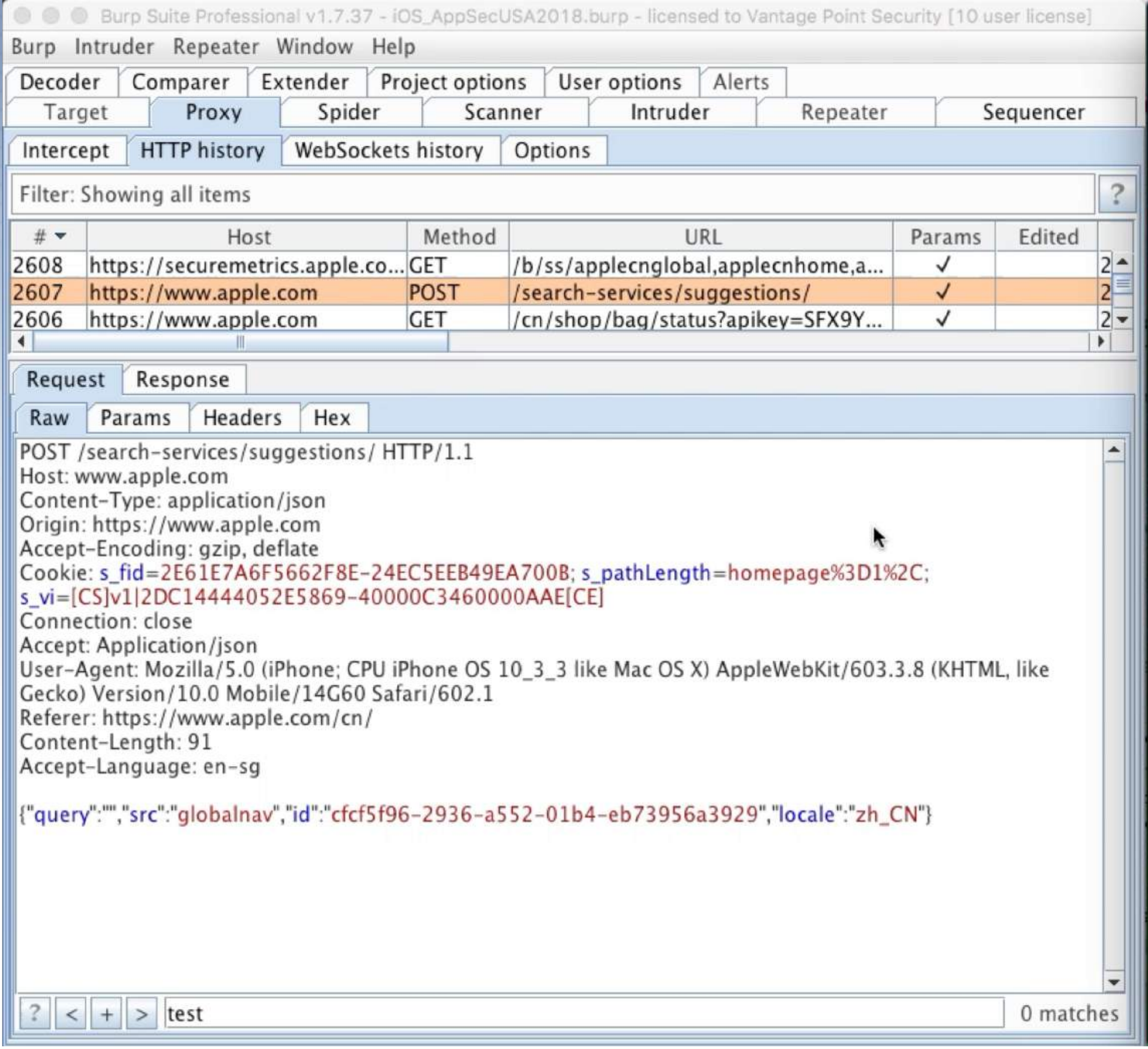
SSL pinning – SSL killswitch V2

Two easy ways to break most pinners:

1. Jailbreak → use Cydia & SSL Killswitch V2

2. Do dynamic instrumentation on a non-jailbroken device

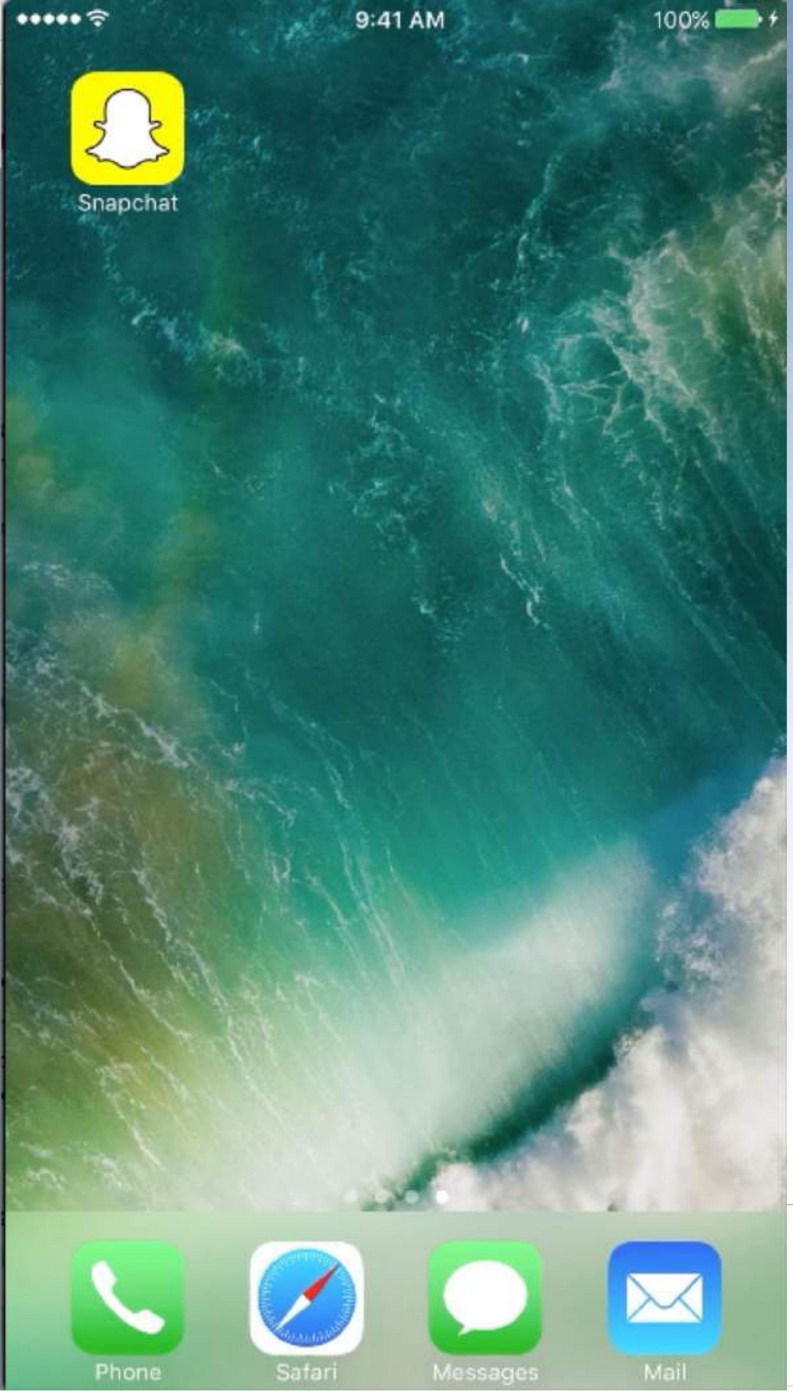
See https://github.com/OWASP/owasp-mstg/blob/master/Document/0x04f-Testing-Network-Communication.md
and https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06g-Testing-Network-Communication.md


Burp Suite Professional v1.7.37 - iOS_AppSecUSA2018.burp - licensed to Vantage Point Security [10 user license]
Burp Intruder Repeater Window Help
Decoder Comparer Extender Project options User options Alerts
Target Proxy Spider Scanner Intruder Repeater Sequencer
Intercept HTTP history WebSockets history Options
Filter: Showing all items
Request Response
Raw Params Headers Hex
test
0 matches

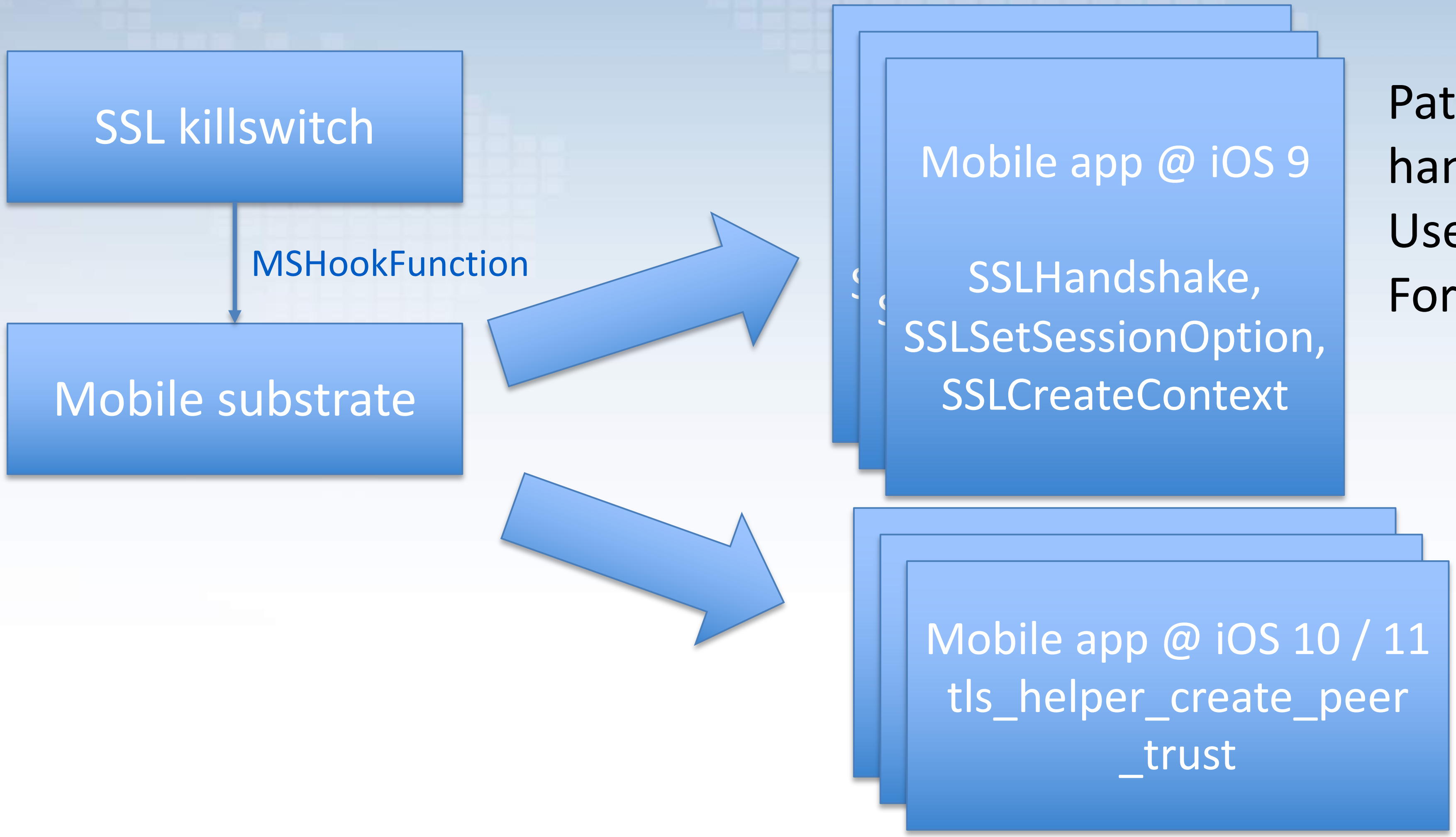
#	Host	Method	URL	Params	Edited
2608	https://securemetrics.apple.co...	GET	/b/ss/applecnglobal,applecnhome,a...	✓	
2607	https://www.apple.com	POST	/search-services/suggestions/	✓	
2606	https://www.apple.com	GET	/cn/shop/bag/status?apikey=SFX9Y...	✓	

```
POST /search-services/suggestions/ HTTP/1.1
Host: www.apple.com
Content-Type: application/json
Origin: https://www.apple.com
Accept-Encoding: gzip, deflate
Cookie: s_fid=2E61E7A6F5662F8E-24EC5EEB49EA700B; s_pathLength=homepage%3D1%2C;
s_vi=[CS]v1|2DC14444052E5869-40000C3460000AAE[CE]
Connection: close
Accept: Application/json
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 10_3_3 like Mac OS X) AppleWebKit/603.3.8 (KHTML, like
Gecko) Version/10.0 Mobile/14G60 Safari/602.1
Referer: https://www.apple.com/cn/
Content-Length: 91
Accept-Language: en-sg

{"query":"","src":"globalnav","id":"cfcf5f96-2936-a552-01b4-eb73956a3929","locale":"zh_CN"}
```



SSL pinning – SSL killswitch V2



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

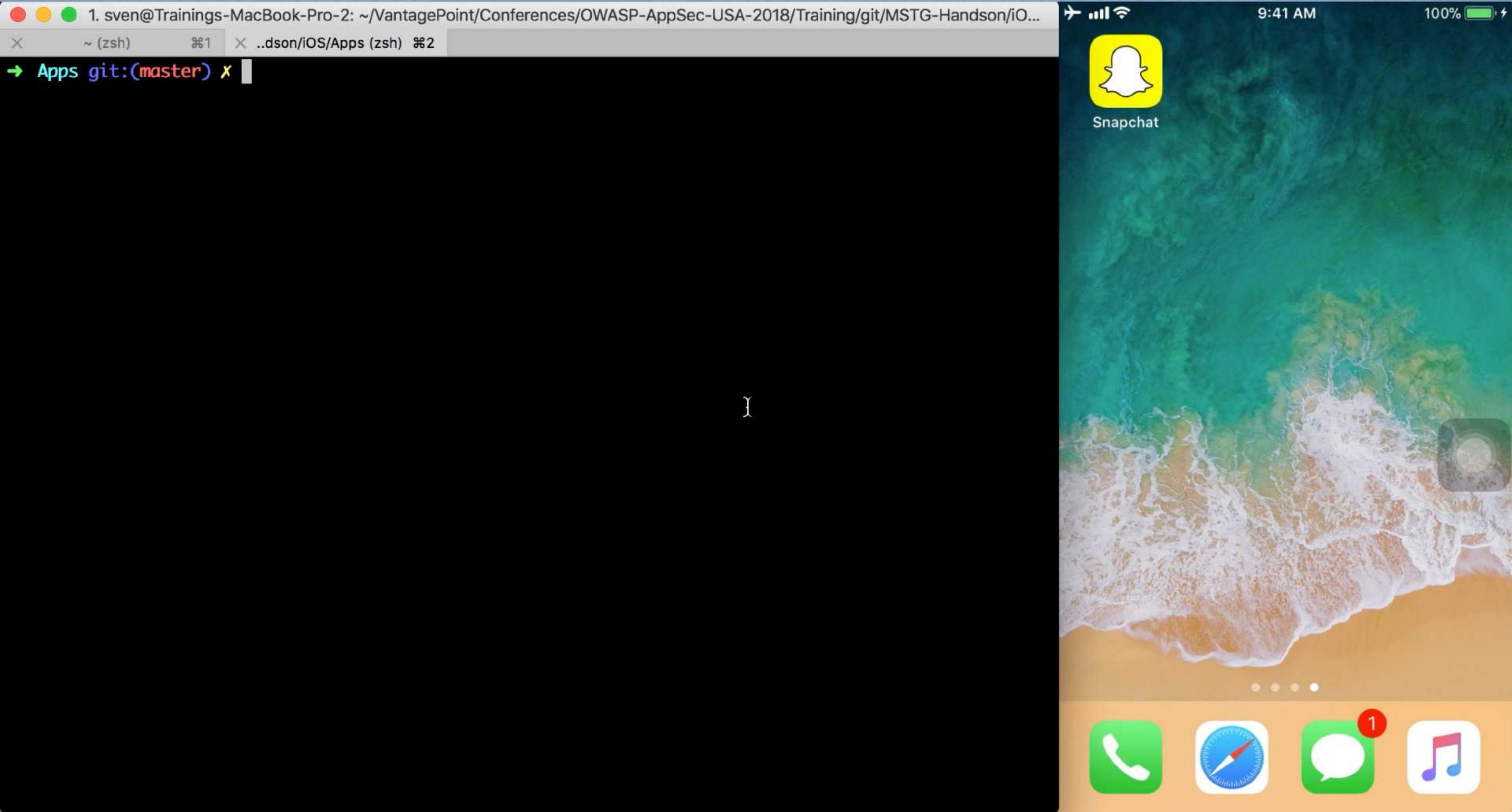
What if you don’t want to jailbreak?

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

OBJECTION
RUNTIME
MOBILE
EXPLORATION
GIT.IO/OBJECTION

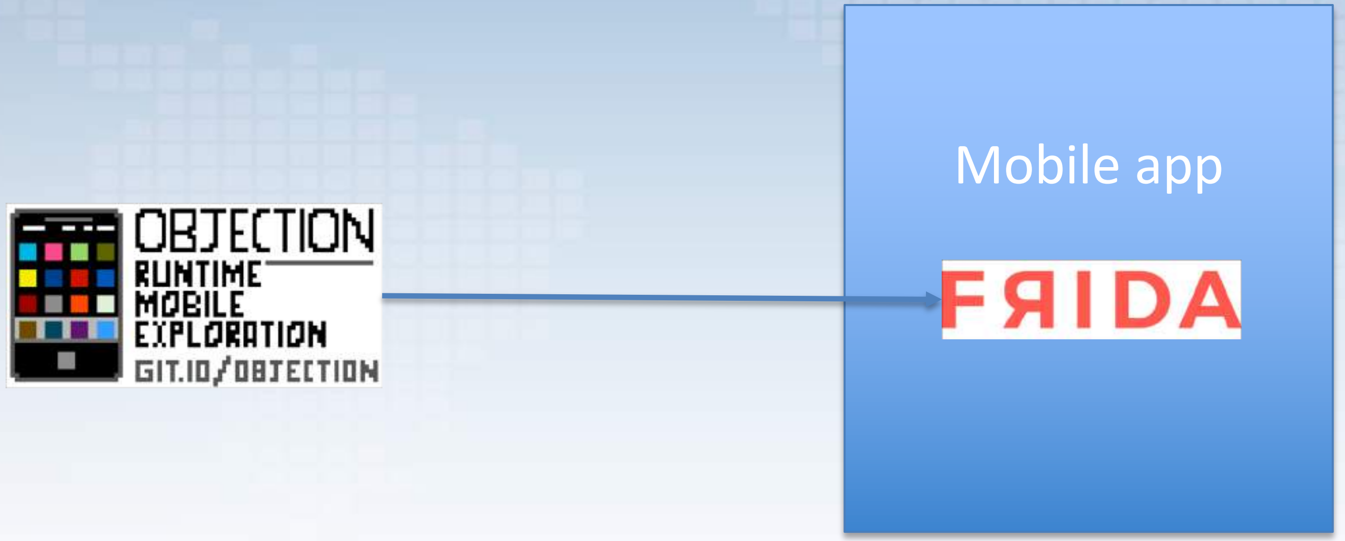
FRIDA

OWASP
Open Web Application
Security Project



```
1. sven@Trainings-MacBook-Pro-2: ~/VantagePoint/Conferences/OWASP-AppSec-USA-2018/Training/git/MSTG-Handson/iO...
~ (zsh)  ⌘1    ..dson/iOS/Apps (zsh)  ⌘2
➜ Apps git:(master) ✗
```

SSL pinning – Objection

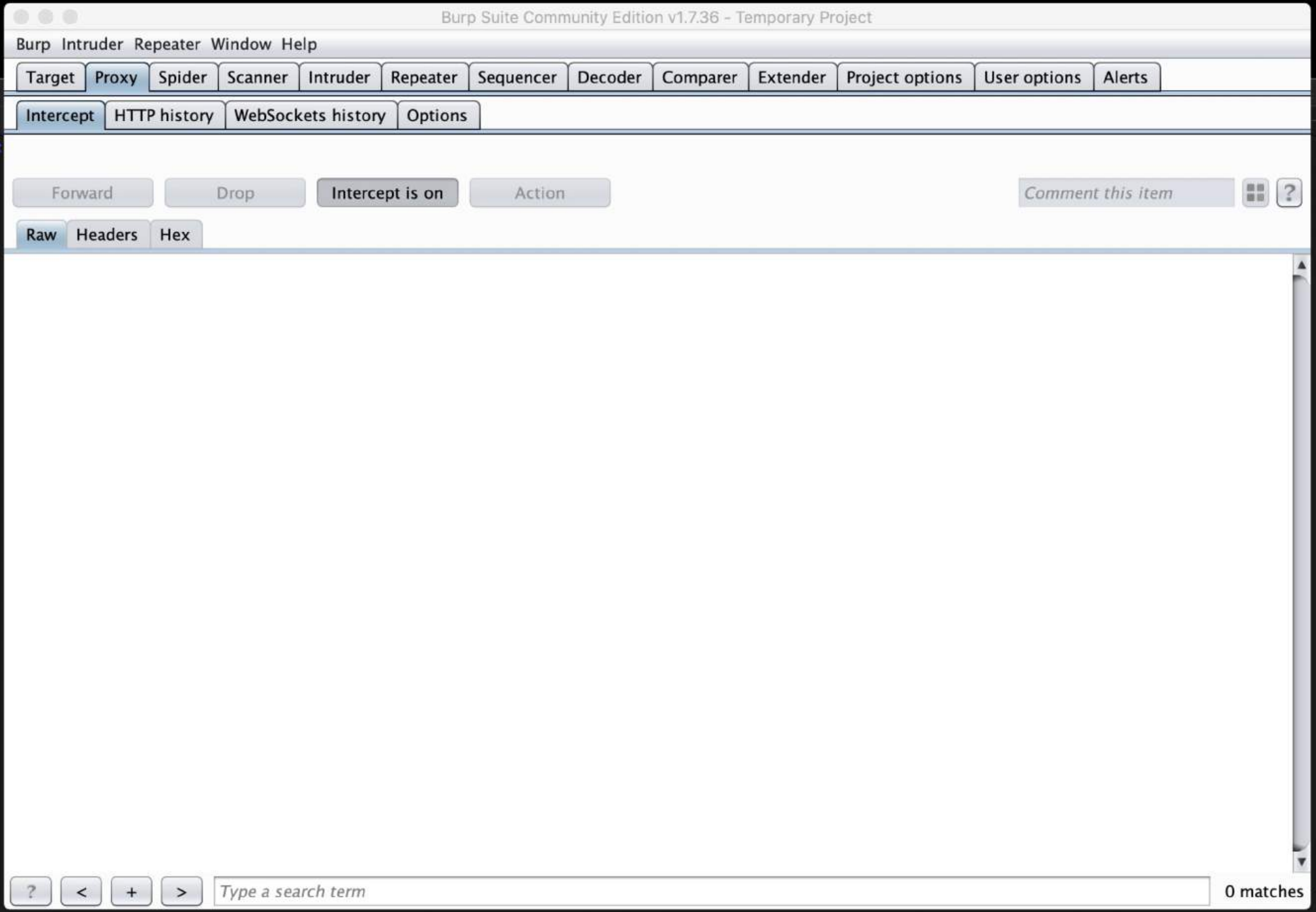
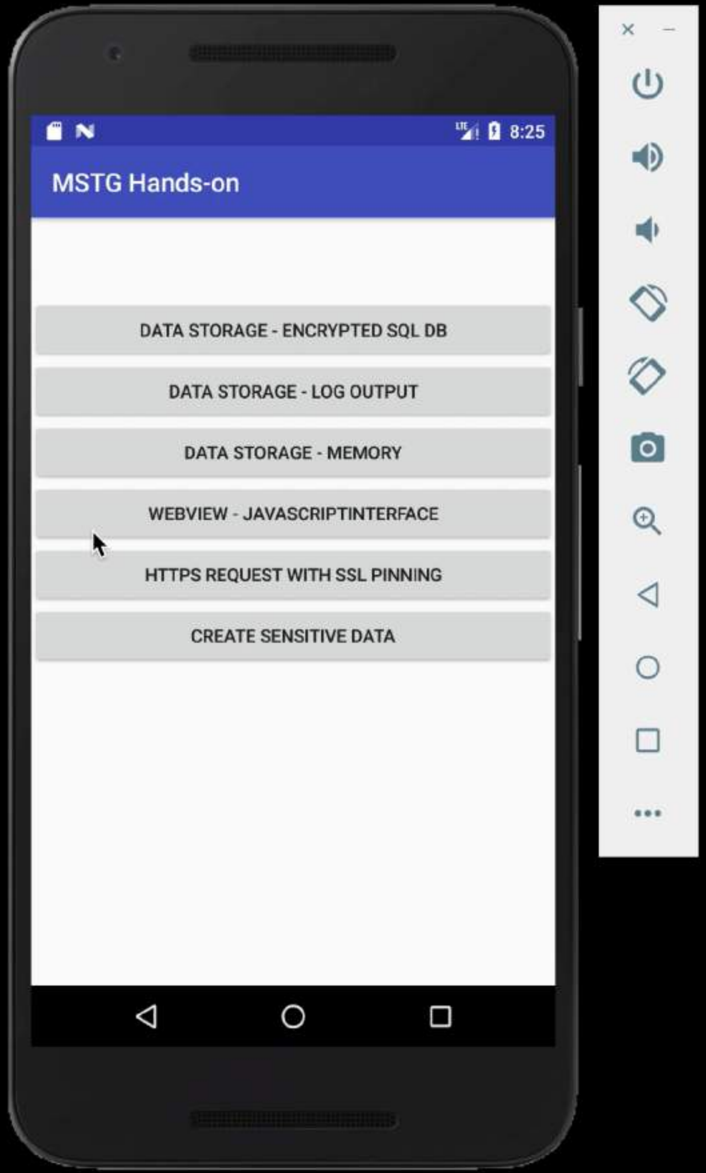


Patch underlying SSL handshake implementation
Used by NSURLConnection
For ***<u>one</u>*** app.

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

SSL Pinning in Android

Let’s do similar runtime patching in Android…


Burp Suite Community Edition v1.7.36 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Forward
Drop
Intercept is on
Action
Comment this item
Raw Headers Hex
Type a search term
0 matches

8:25
MSTG Hands-on
DATA STORAGE - ENCRYPTED SQL DB
DATA STORAGE - LOG OUTPUT
DATA STORAGE - MEMORY
WEBVIEW - JAVASCRIPTINTERFACE
HTTPS REQUEST WITH SSL PINNING
CREATE SENSITIVE DATA

TouchID the wrong way: using LAContext

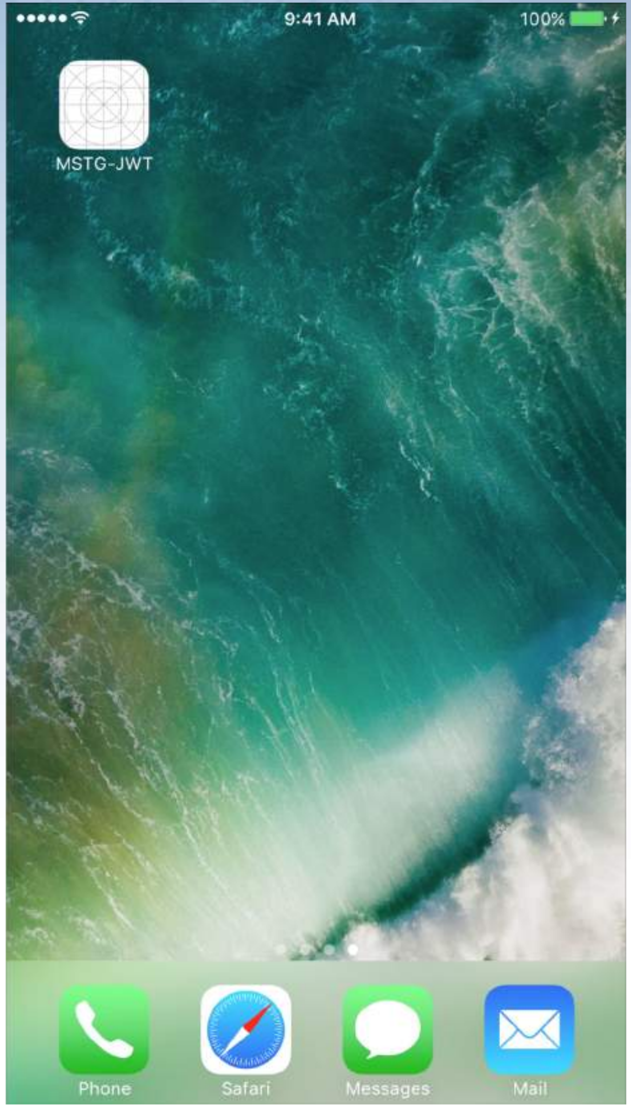
There are 2 ways to use TouchID:

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?

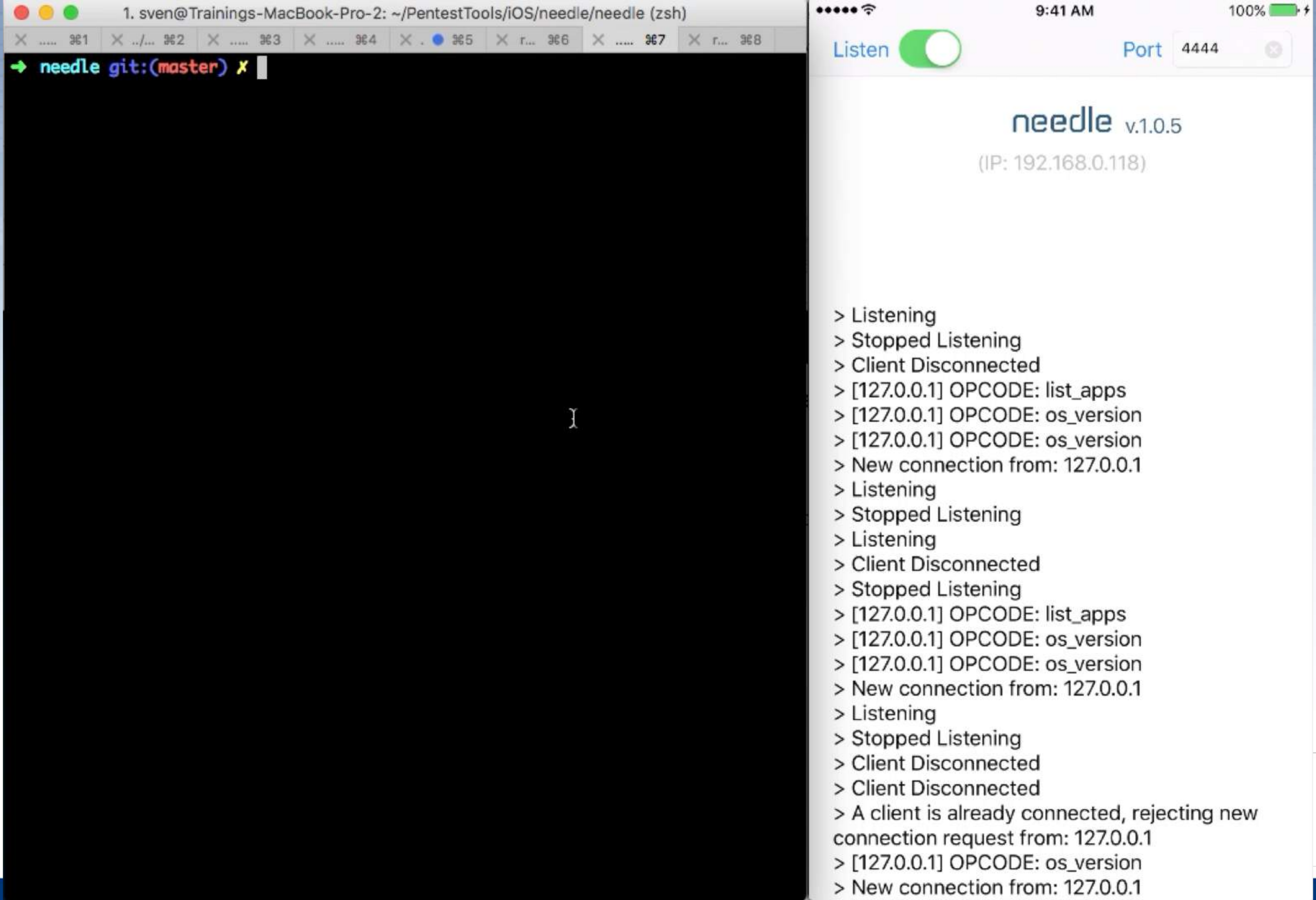


Bypassing Touch-ID

- With needle
- With OBJECTION RUNTIME MOBILE EXPLORATION GIT.IO/OBJECTION
- Both cases: use Frida to hook onto *`evaluatePolicy:localizedReason:reply`*
  – Ensures that when *evaluatePolicy* is calls 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

OWASP Open Web Application Security Project

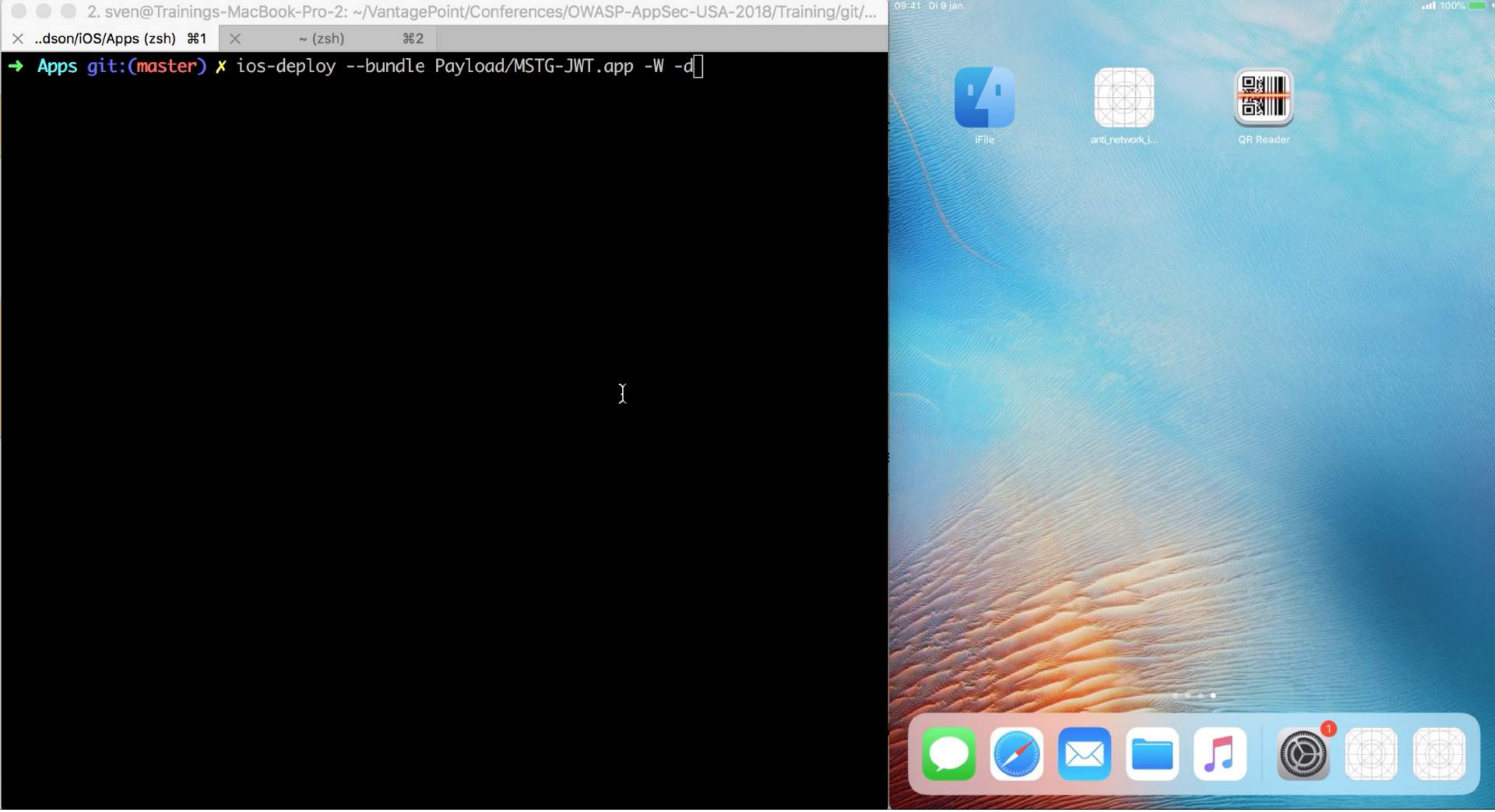


```
➜ needle git:(master) ✗
```

Port 4444

(IP: 192.168.0.118)

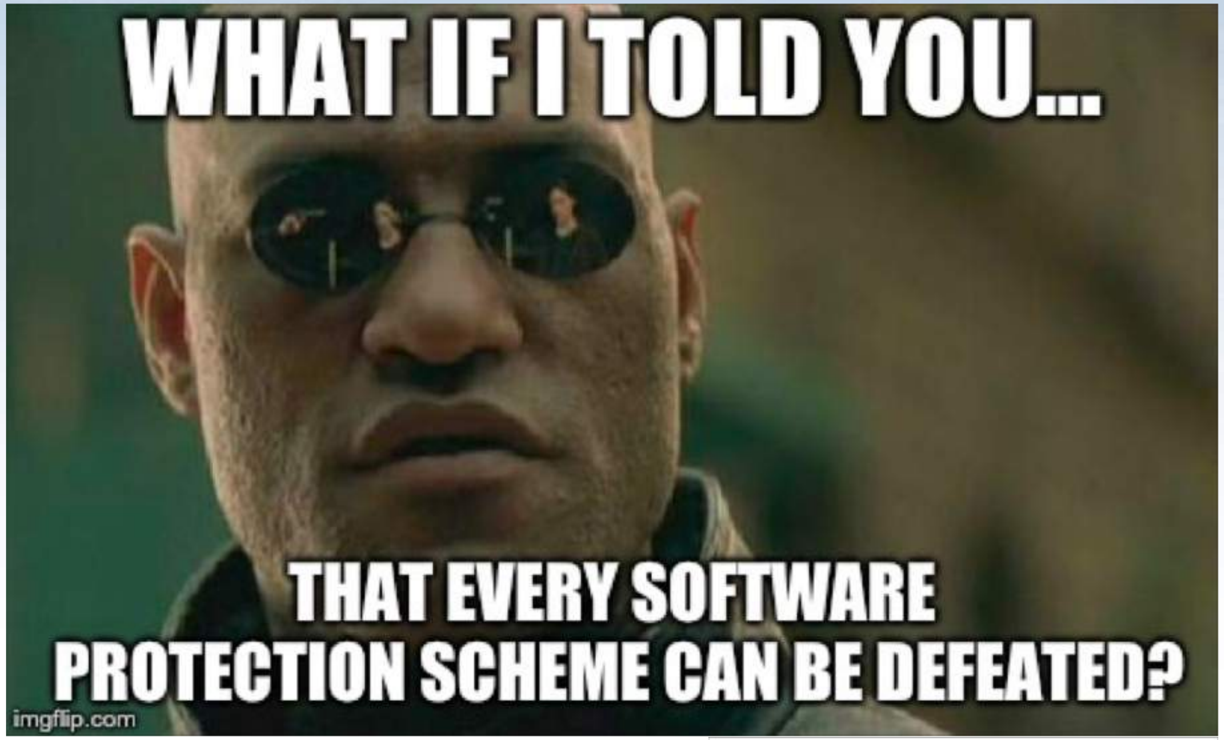
```
> Listening
> Stopped Listening
> Client Disconnected
> [127.0.0.1] OPCODE: list_apps
> [127.0.0.1] OPCODE: os_version
> [127.0.0.1] OPCODE: os_version
> New connection from: 127.0.0.1
> Listening
> Stopped Listening
> Listening
> Client Disconnected
> Stopped Listening
> [127.0.0.1] OPCODE: list_apps
> [127.0.0.1] OPCODE: os_version
> [127.0.0.1] OPCODE: os_version
> New connection from: 127.0.0.1
> Listening
> Stopped Listening
> Client Disconnected
> Client Disconnected
> A client is already connected, rejecting new
connection request from: 127.0.0.1
> [127.0.0.1] OPCODE: os_version
> New connection from: 127.0.0.1
```


2. sven@Trainings-MacBook-Pro-2: ~/VantagePoint/Conferences/OWASP-AppSec-USA-2018/Training/git/...
..dson/iOS/Apps (zsh) ⌘1
~ (zsh) ⌘2
09:41 Di 9 jan.
100%
iFile
anti_network_i...
QR Reader

```
➜ Apps git:(master) ✗ ios-deploy --bundle Payload/MSTG-JWT.app -W -d
```

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

I WANT YOU
TO HELP US TO
THE NEXT LEVEL!
imgflip.com

QUESTIONS?

@OWASP_MSTG

jeroen.willemsen@owasp.org

OWASP Open Web Application Security Project

THANK YOU!

@OWASP_MSTG

jeroen.willemsen@owasp.org

Open Web Application Security Project

Addition: Android and objection

OBJECTION DEMO ON ANDROID?