Android Security Workshop

Eduardo Novella
(NowSecure)

Connecting next generation talent with the heavy duty industry to keep vehicles secure

June 24-28, 2019 | Detroit (USA)

Outline

Main ideas

- **Introduction**

  Edu Novella and Android Workshop

- **Android RE**

  Tools used for RE-ing Android apps

- **Mobile CTF**

  Vulnerable keyless Android app to wirelessly unlock vehicles with your mobile "Mobile Keyless Remote System"

- **Takeaways**

  Things learned after this workshop

$ whoami

"I stay with problems longer"

- Mobile Security Research Engineer @ **NowSecure**
  - Focused on **Android** Reverse Engineering
- **Previously (**Reverse Engineering**)**
  - Android **mobile** security: cloud-based payments (HCE wallets), DRM and TEE solutions
  - **Embedded** security : smartcards, smartmeter, PayTV, HCE, routers, any hardened IoT dev
  - Crypto: side-channel & fault injection attacks (hw). Whitebox cryptography (sw)
- Personal @ enovella.github.io
  - Based in London (UK)
  - Chess player, swimmer and nature lover

CyberTruck Android Workshop

"Unlock your truck with your Android"

- **Mobile CTF simulating an app capable of unlocking vehicles via bluetooth**
  - Android CTF-like challenge (3 static + 3 dynamic flags = **6 flags** in total)
  - URL: http://192.168.1.34/challenges?category=android-by-eduardo-novella
  - Material: **smb://192.168.1.3/Documents/workshop** (student:student)
  - Run AVD: **/droidsec/android-studio/bin/studio.sh**
  - 1h workshop + extra time
  - Enable the TamperProof switch if you're brave :-)
- **Rules**
  - Don't share flags with other mates
  - Disturb the CTF platform availability
  - Up to you how to solve the challenges

Android RE

Tools

- Dalvik Bytecode
  - **JADX**
  - Bytecode Viewer
  - JEB
  - **Apktool**
  - Baksmali/smali
- Native
  - IDA Pro
  - **Radare2**
  - **Ghidra**
  - Binary Ninja
  - Hopper
- Dynamic Binary Instrumentation
  - **Frida**
  - Xposed
- Source code
  - Android Studio + AVD emulators

Android RE

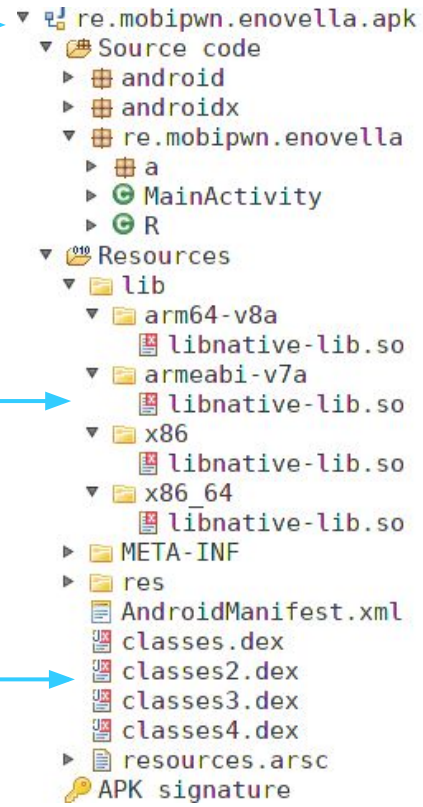
Most powerful OSS tools

- JADX
  - DEX decompiler
- Ghidra
  - Native decompiler
- Radare2
  - Unix-like reverse engineering framework
- Frida
  - Dynamic Binary Instrumentation
- R2frida
  - The ultimate static analysis on dynamic steroids

APK

Android Application Packaging (APK)

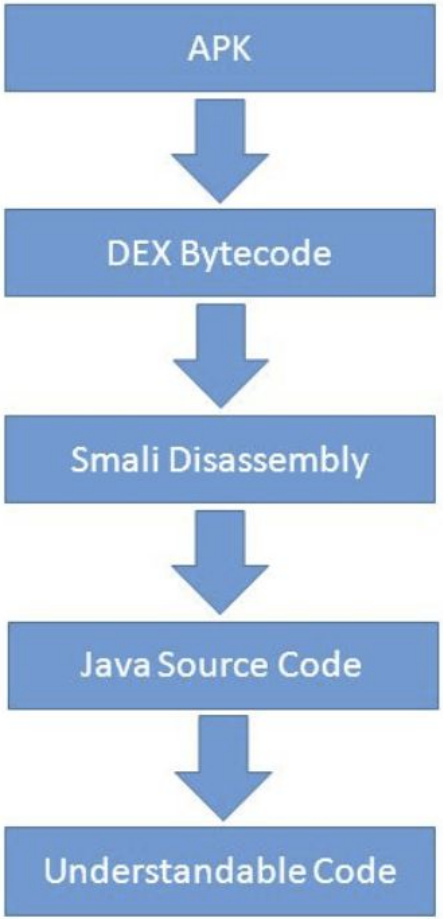
- **APK**
  - APK == ZIP
  - Manifest XML
  - Assets folder
  - Resources folder
- **Reverse Engineering**
  - APK
  - Native **code**
  - DEX byte**code**



Android RE

Static Analysis

- Static Analysis
  - Understand the logic → who, how, where, why, what is going on
  - Decompile binary code → Pseudo code (readable)
  - Search for → strings, crypto keys, passwords, network traffic, ..
  - Manual RE → Rename variables, functions if stripped



Android RE

Dynamic Analysis

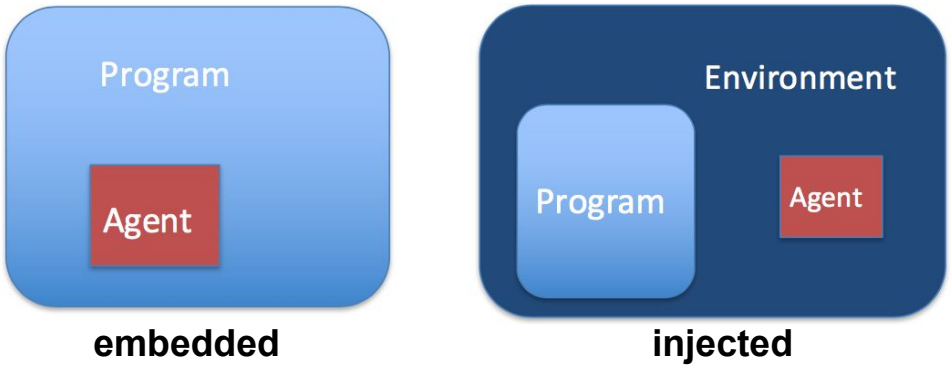
- Dynamic Binary Instrumentation (DBI)

"A method of analyzing the behavior of a binary application at runtime through the injection of instrumentation code"

  - Access process memory (stack,heap,code,...)
  - Hook, trace, intercept functions
  - Change return values, variables, globals, function args,...
  - Overwrite function implementations while app is running
  - Call arbitrary functions from imported classes
  - Find object instances on the heap
  - Bypass client-side security checks

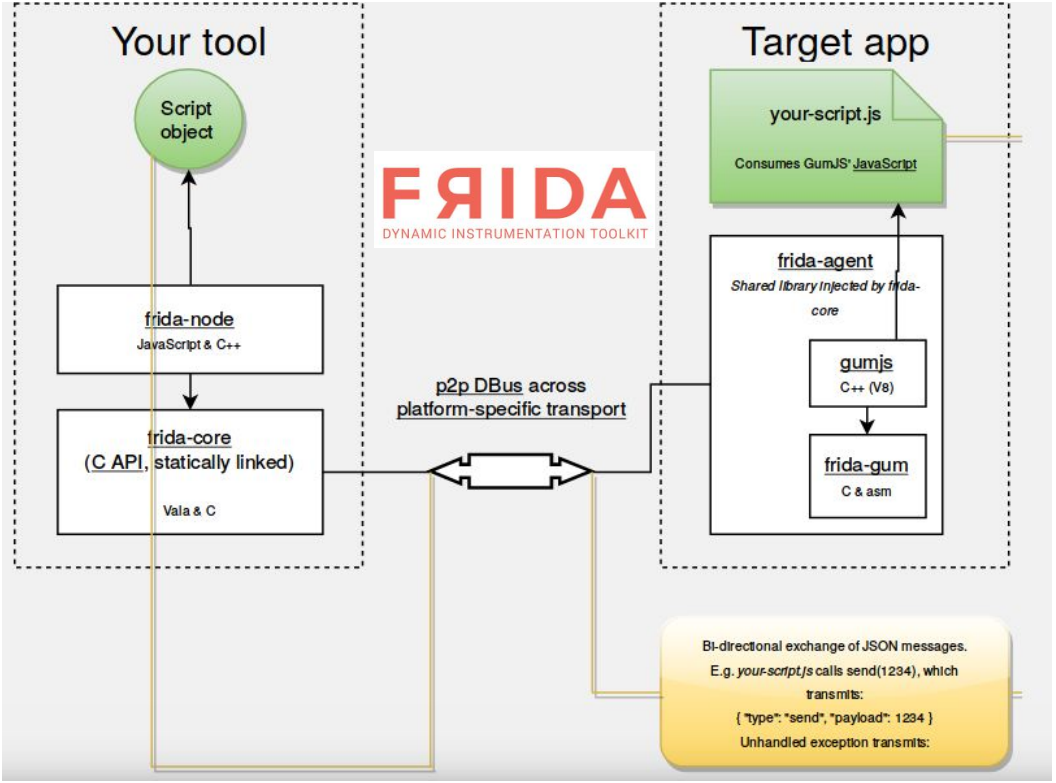
FRIDA

DYNAMIC INSTRUMENTATION TOOLKIT



Android RE

Injection via Frida

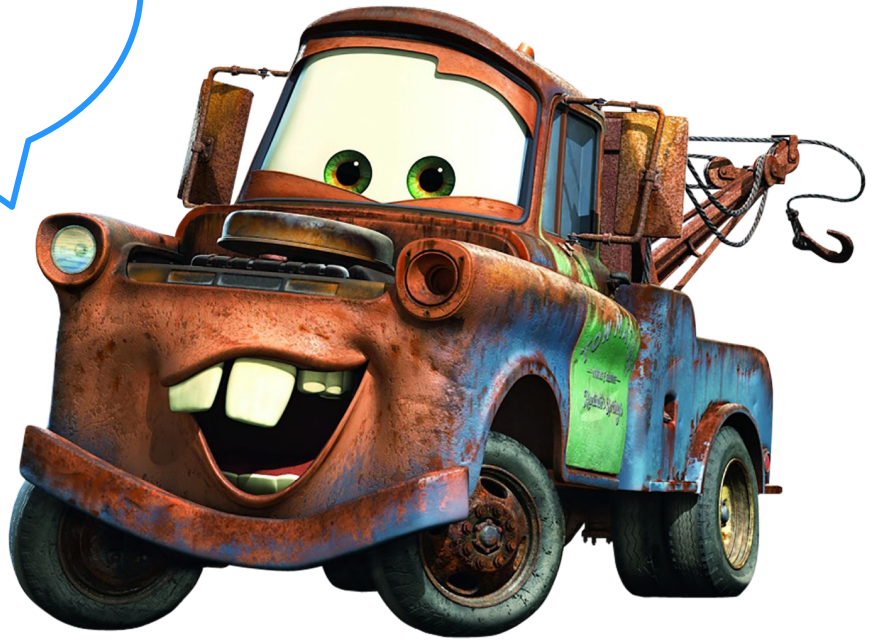


Android Crackme

Can you unlock this uncrackable car keyless system?

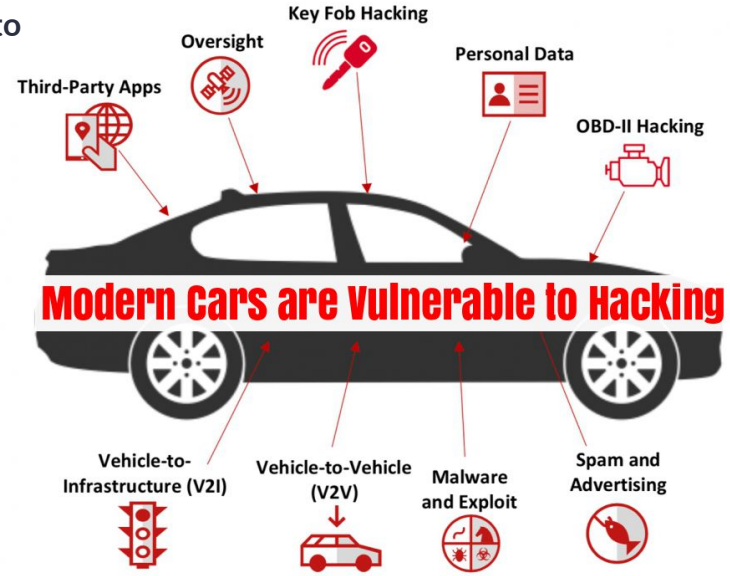
Mobile CTF

Let’s play!



Takeaways

- Secure vehicles can be hard → Security by obscurity is not the solution
- Focus on the **design** and ensure **strong** key hierarchy → Client-side apps will be eventually compromised
- Follow security **guidelines** → OWASP MSTG
- **Minimum** privilege principle → Reduce the attack surface
- Do not **hardcode** secrets within your code → Use **encryption** at rest
- Use **hardware-backed** keystore to keep secrets instead of SW-based implementations
- Protect IP → **Code hardening** (obfuscation, anti-tampering, anti-rooting, anti-debugging, ...)
- Ensure proper **randomness** source → Use strong & secure **crypto**
- Implement multi-factor authentication (MFA)
- Enforce certificate pinning to slow down MITM attacks
- Bug **bounty** your application before you got hacked
- Employ hardened OS features → **TrustZone** (TEE)
- Google security → **SafetyNet**



Links

Where to search

- Radare2 && Frida (NowSecure)
- The Mobile Security Testing Guide (MSTG)
- Awesome Mobile CTFs
- Awesome Frida && Frida CodeShare
- MOBISEC lectures
- Android App Reverse Engineering 101
- RedNaga Security
- Gio's blog
- A bunch of mobile security blog posts on the Internet

THANK YOU!
Q&A

Eduardo Novella

Mobile Security Research Engineer

enovella@nowsecure.com

@NowSecureMobile

@enovella_

Special thanks to

@RomainKraft @fs0c131y @Hexploitable

for providing feedback on the crackme