CYBERTRUCK CHALLENGE
pw n3d
0x95 0x77 0xA7 0x86 0x07 0x52 0x85 0xA6 0x86 0xA7 0x77
0xE6 0x97 0xE7 0x52 0x25 0x66 0x97 0x97 0xA6 0x77 0x87

Android Security Workshop

Eduardo Novella

(NowSecure)

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

August 16-20, 2021 | Detroit (USA)

Outline

Main ideas

- Introduction Android

  Android OS internals
  Automotive Android OS

- Android Reverse Engineering

  Static & Dynamic analysis
  Tools used for RE-ing Android apps

- CyberTruck Challenge CrackMe Walkthrough

  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

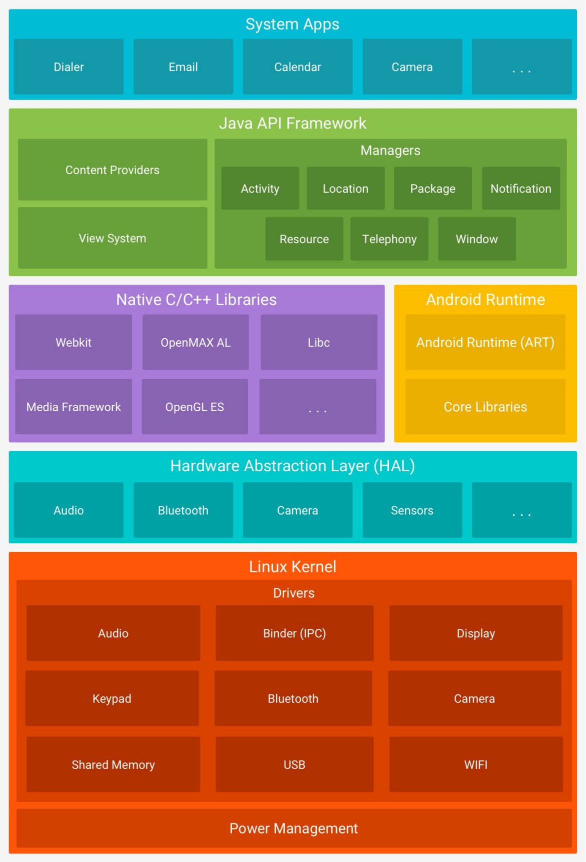
- **Previously** (Reverse Engineering)
  - Android **mobile** security: cloud-based payments (HCE wallets), DRM and TEE solutions
  - **Embedded** security : smartcards, smartmeter, PayTV, HCE, routers, any hardened IoT dev
  - Crypto: side-channel & fault injection attacks (hw). Whitebox cryptography (sw)

- Personal @ enovella.github.io
  - Based in Europe
  - Chess player, swimmer and nature lover

Android OS

Architecture

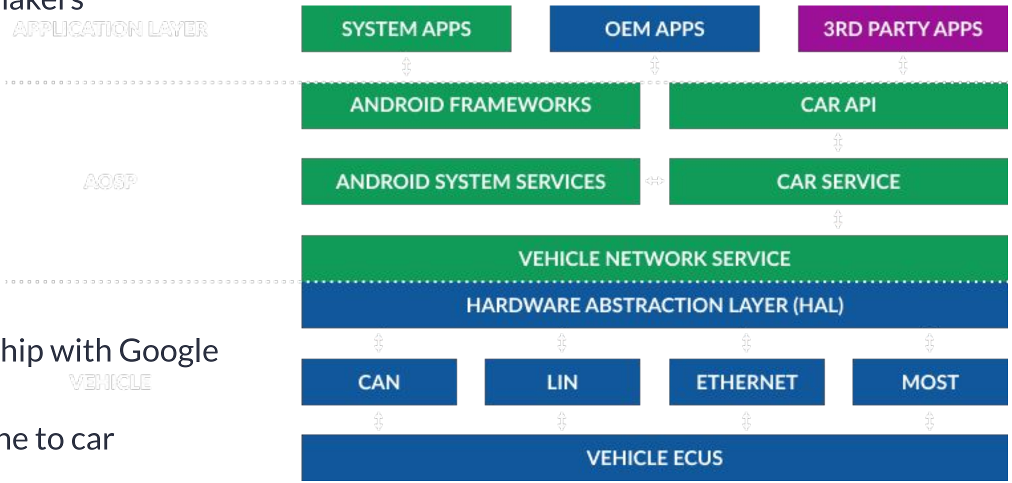
- Android OS developed by Google
  - Based on Linux (Open Source) with kernel- and user-space
  - Hardware manufacturers, app developers, AOSP contributors
  - Components:
    - Android Runtime (Dalvik VM vs ART)
    - Native core libraries (c/c++/rust), Linux Kernel, Java API
    - System apps, HAL, third-party applications
- Application Sandbox - least privilege
  - Each app operates in its own isolated environment (sandbox)
  - Unix-style permission model
  - Data directory `/data/data/package-name-app/`
  - App data sharing via IPC (content providers)
  - UID - User Identity. Greater than 10000 for normal apps



Automotive Android OS (AAOS)

Architecture

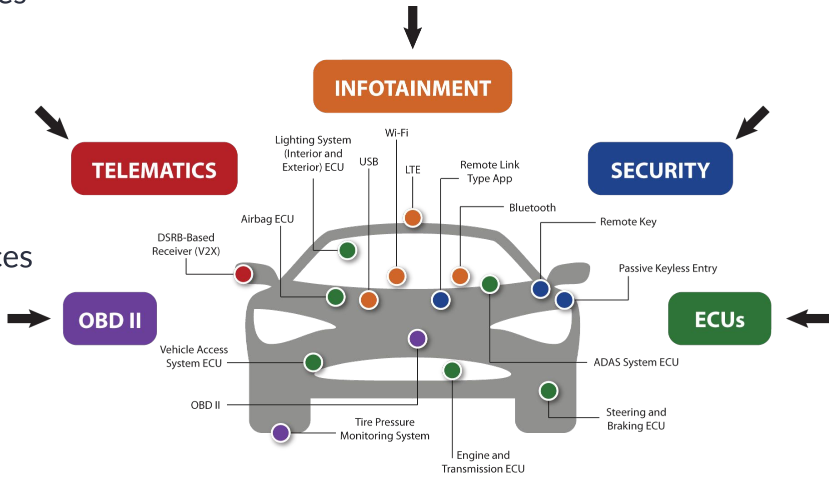
- Android Automotive OS (AAOS) - “Android for Cars”
  - Infotainment system built into cars by carmakers
  - Interface designed for car screens
  - Components
    - In-vehicle Infotainment (IVI)
    - Google Automotive Services (GAS)
    - Vehicle Map Service (VMS)
    - Exterior View System (EVS)
    - Heating, ventilation & AC (HVAC)
  - OEM receives access to GAS via a partnership with Google
- Android Auto
  - A framework to connect your Android phone to car



Android Threat Modelling

Attack Surface

- Physical access
  - USB port (ADB). Developer Options enabled
  - Hardware ports (UART, JTAG,...) for debugging purposes
  - Pull out proprietary apps from automotive vendors
  - Jailbreak from kiosk
  - ...
- Vendor's Applications
  - Identify critical assets within the app
    - IP, crypto, databases, Android shared preferences
    - Proprietary protocols and crypto
  - Network protocols (MITM), tracking, GPS spoofing
  - App security can be tampered with
  - Firmware updates for automotive components
  - ...
- Non-physical access
  - Wireless (WiFi, Bluetooth, NFC, LTE, Baseband)
  - Vulnerabilities on old Android OS
  - Web server accessible via browser
  - ...



Android Reverse Engineering

Tools

- Java/Kotlin → 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 Reverse Engineering

Most powerful OSS tools

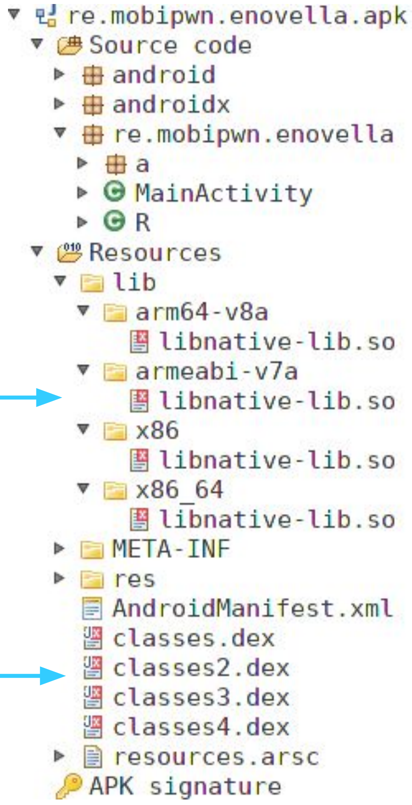
- JADX
  - DEX decompiler
- Ghidra
  - Native decompiler
- Radare2
  - Unix-like reverse engineering framework
- Frida
  - Dynamic Binary Instrumentation
- R2frida
  - The ultimate static analysis on dynamic steroids

APK

Android Application Packaging (APK)

- **APK**
  - APK == ZIP
  - Manifest XML
  - Assets folder
  - Resources folder

- **Reverse Engineering**
  - APK
  - Native **code**
  - DEX byte**code**

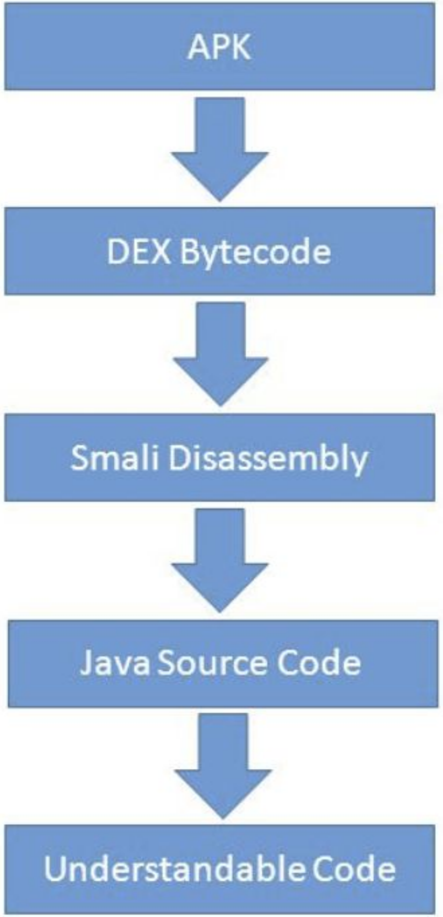


```
▼ re.mobipwn.enovella.apk
  ▼ Source code
    ▶ android
    ▶ androidx
    ▼ re.mobipwn.enovella
      ▶ a
      ▶ MainActivity
      ▶ R
  ▼ Resources
    ▼ lib
      ▼ arm64-v8a
          libnative-lib.so
      ▼ armeabi-v7a
          libnative-lib.so
      ▼ x86
          libnative-lib.so
      ▼ x86_64
          libnative-lib.so
    ▶ META-INF
    ▶ res
      AndroidManifest.xml
      classes.dex
      classes2.dex
      classes3.dex
      classes4.dex
    ▶ resources.arsc
  APK signature
```

Android RE

Static Analysis

- Static Analysis
  - Understand app logic
  - Find security bugs
  - Reveal critical assets
  - Locate places to perform dynamic analysis
- Steps
  - Decompile binary code → Pseudo code (readable)
  - Navigate codebase & search for
    - strings, crypto keys, passwords, network traffic, ..
    - obfuscation
      - Rename variables, functions (if stripped)
  - Tamper with the app integrity
    - Include your modifications
      - enable logging
      - disable checks
      - GPS locations



Android Reverse Engineering

Dynamic Analysis

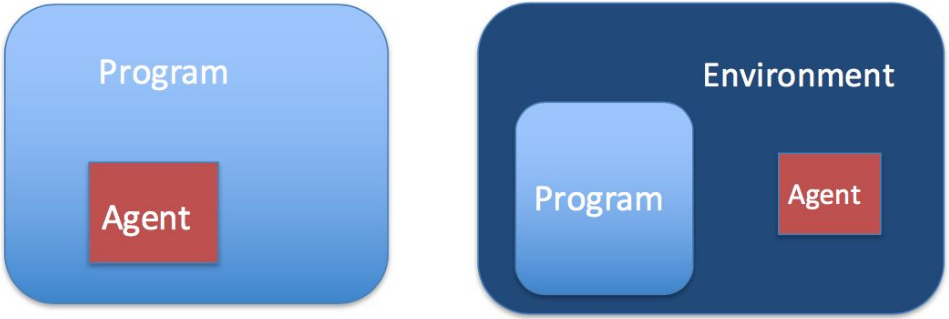
- Dynamic Binary Instrumentation (DBI)

"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

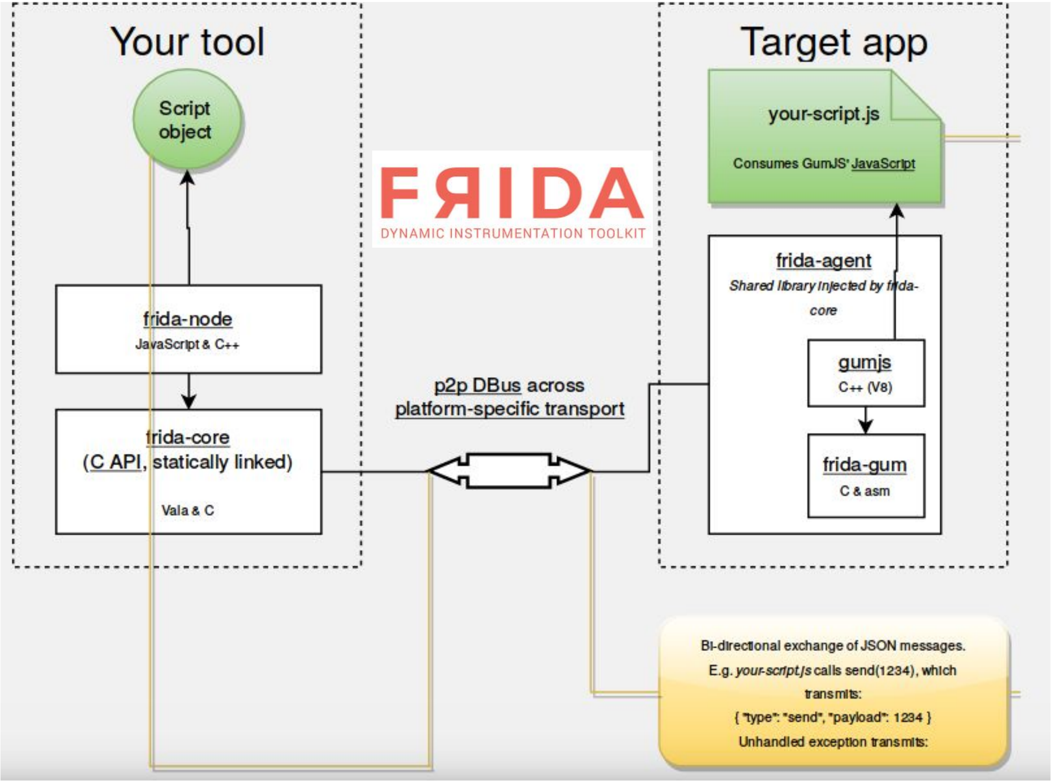
DYNAMIC INSTRUMENTATION TOOLKIT



embedded **injected**

Android Reverse Engineering

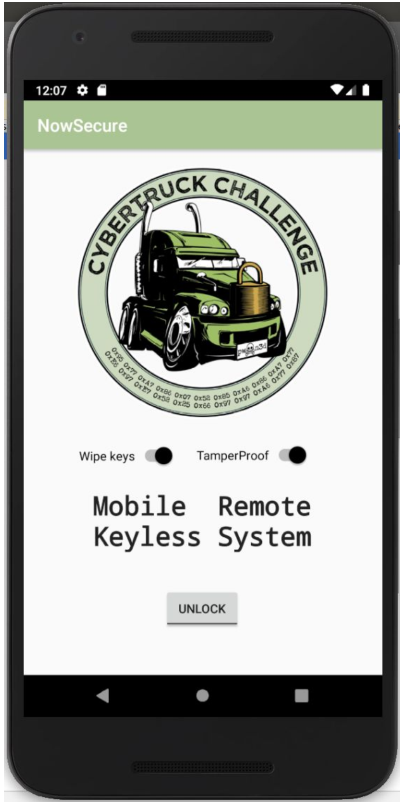
Process Injection via Frida


Your tool
Script object
frida-node
JavaScript & C++
frida-core
(C API, statically linked)
Vala & C
FRIDA
DYNAMIC INSTRUMENTATION TOOLKIT
p2p DBus across platform-specific transport
Target app
your-script.js
Consumes GumJS' JavaScript
frida-agent
Shared library injected by frida-core
gumjs
C++ (V8)
frida-gum
C & asm

Bi-directional exchange of JSON messages.
E.g. *your-script.js* calls send(1234), which transmits:
{ "type": "send", "payload": 1234 }
Unhandled exception transmits:

Android CyberTruck Crackme

Can you unlock this uncrackable car keyless system?



Mobile Remote Keyless System

https://github.com/nowsecure/cybertruckchallenge19

Android CyberTruck Challenge Crackme

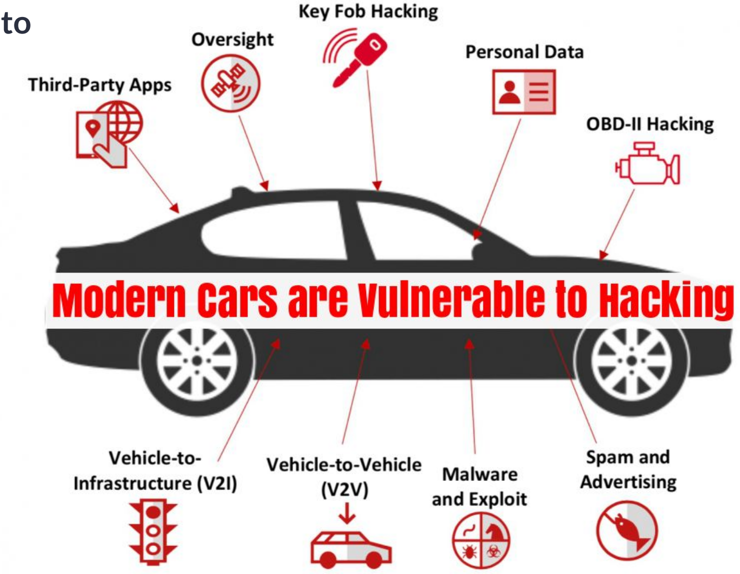
"Unlock your truck with your Android"

- **Mobile CrackMe simulating an app capable of unlocking vehicles via bluetooth**
    - Material: https://github.com/nowsecure/cybertruckchallenge19
    - Android CTF-like challenge (3 static + 3 dynamic flags = **6 flags** in total)
    - Run it in Android emulator
    - 1h workshop + extra time
    - Enable the TamperProof switch if you're brave :-)
- **Rules**
    - Don't share flags with other mates
    - Up to you how to solve the challenges

Mobile CTF

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