Mobile Pentesting 101

Christian Villapando

About Me

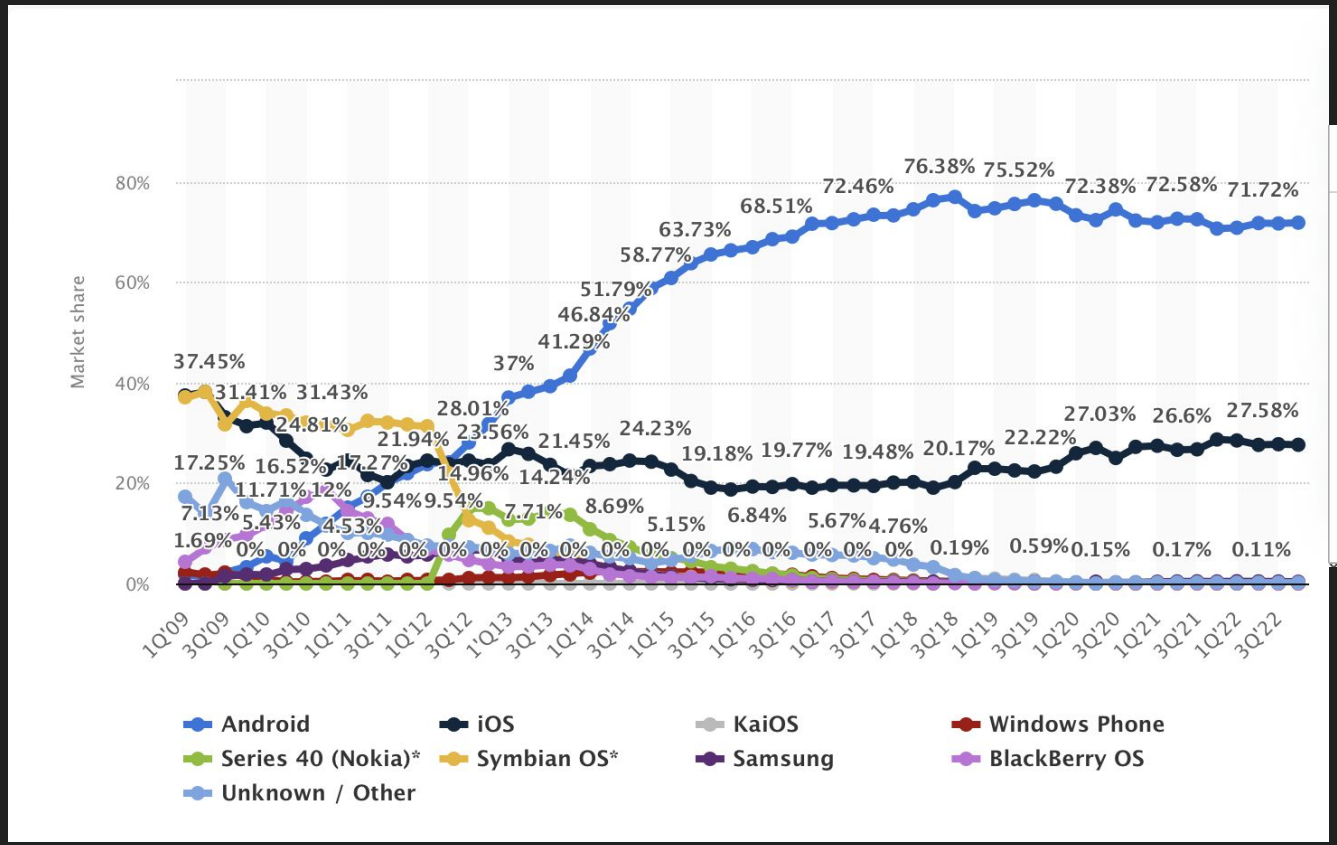
OUTLINE

- Mobile Security Overview
- iOS
- Android
- Static Analysis
- Dynamic Analysis
- OWASP MASVS
- Mobile Penetration Testing

MOBILE SECURITY

- Mobile devices are widely used today pose problems and opportunities for individuals and organizations
- The line between mobile devices and computers has become considerably blurred
  - Some smartphones are more powerful than some computers
  - Some computers (ultrabooks and Raspberry Pis) run on minimal hardware and use little power
- Mobile devices are productivity tools
- Individual
  - How many mobile devices do you have?
- Enterprises
  - Company-issued mobile devices
  - Personal devices used for work (BYOD)
  - Remote work

MOBILE OPERATING SYSTEM MARKET SHARE



4Q'22	
Android	71.8%
iOS	27.6%
KaiOS	0.08%
Windows Phone	0.02%
Series 40 (Nokia)*	0.01%
Symbian OS*	0%
Samsung	0.34%
BlackBerry OS	0%
Unknown / Other	0.15%

MOBILE THREATS THAT ACTUALLY MATTER

- Loss of control and visibility with BYOD
- Always-on devices through multiple interfaces
- Device patching and extended vulnerability periods (Android)
- Device theft and loss
- Weak server-side controls
- Application flaws
  - Inherent to the app
  - Inherent to the platform
  - Inherent from the ad libraries

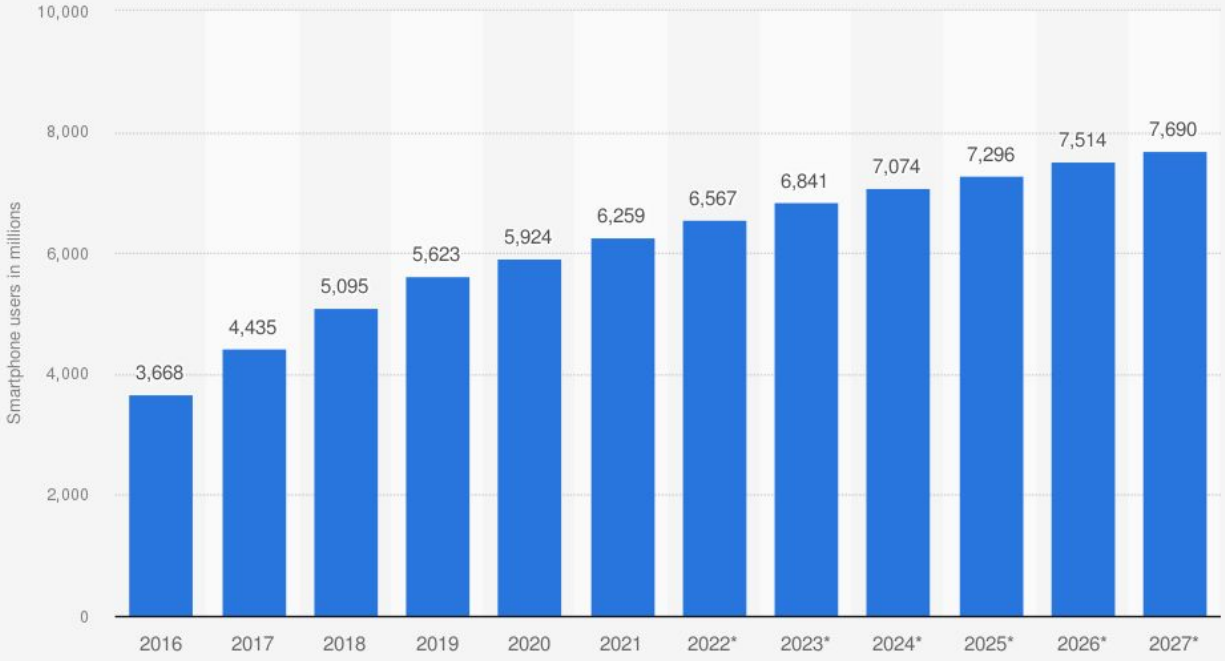
EVOLUTION OF MOBILE APPLICATIONS

- The first mobile phone applications were developed by handset manufacturers
  - Documentation was limited
  - Little information is known about operating internals
- This can perhaps be attributed to a fear from the vendors that opening the platforms to third-party development might have exposed trade secrets in what was not yet a fully developed technology
- The early applications were similar to many of the manufacturer-based apps found on today’s phone, such as contacts and calendars, and simple games such as Nokia’s popular Snake.

COMMON MOBILE APPLICATION FUNCTIONS

- Mobile applications have been created for practically every purpose imaginable
  - Online banking (Maya, Gcash, BDO, BPI, etc.)
  - Shopping (Lazada, Shopee, Amazon)
  - Social networking (Facebook, Twitter, Instagram)
  - Streaming (Netflix, HBO Go, Prime, Disney+)
  - Instant Messaging and Voice Chat (Messenger, Viber, WhatsApp)
  - E-mail (Gmail)
  - File sharing (Drive, Dropbox)
  - Games (ML, COC, etc.)

Number of smartphone subscriptions worldwide from 2016 to 2021, with forecasts from 2022 to 2027 (in millions)



Source
Ericsson
© Statista 2023

Additional Information:
Worldwide; Ericsson; 2016 to 2021

COMMON MOBILE APPLICATION FUNCTIONS (cont.)

- In addition to the applications that are available in the various distribution markets, mobile applications have been widely adopted in the business world to support key business functions
- Many of these applications provide access to highly sensitive corporate data, including some of the following
  - Document storage applications allowing users to access sensitive business documents on demand
  - Travel and expenses applications allowing users to create, store, and upload expenses to internal systems
  - HR applications allowing users to access the payroll, time slips, holiday information, and other sensitive functionality

OPPORTUNITIES IN MOBILE SECURITY

- Lack of infosec professionals who are proficient in mobile security
  - Multiple technologies
  - VAPT, source code audit, secure deployment and management
  - This presents a gap in mobile security
- Mobile is here to stay

STOLEN DEVICE THREAT

- Mobile devices can be lost or stolen
  - Misplaced devices
- Devices that are stolen introduces threat to an organization
- Through policy, device management, and sufficient preparation the risk can be reduced
- What can an attacker do if the device can be unlocked?
  - Access local resources
  - Extract data
  - Synchronize the device to backup data to a computer
  - Jailbreak/root/unlock

Mobile Malware

Unlocking, Rooting, and Jailbreaking

- Every shipped mobile phone has restrictions from the manufacturer, vendor, or even the carrier
- These restrictions usually prevent users from installing software that does not qualify with their security requirement
- Hence, users will find a way to bypass this
- Jailbreaking refers to unlocking an iOS device
- Rooting refers to unlocking an Android device

Mobile Malware

- Mobile device malware is a growing threat for devices
  - New opportunities for exploiting users
  - New opportunities for attacker financial gain
- Mobile malware is a small fraction of the overall malware threat
  - Growing at an alarming rate
- Platform exposure varies significantly

User Credential Theft

- Mobile phones are increasingly relied upon for two-factor authentication via SMS
  - Banking applications and related financial services
- Zitmo (ZeuS-in-the-Mobile), a variant of the ZeuS trojan controls SMS and phone functionality, blocking select calls and intercepting messages.
- Integrated with the ZeuS trojan for effective banking control bypass
- It affected Symbian, Windows Mobile, BlackBerry and Android
- Allows the attacker to steal the mTAN (mobile transaction authorization number)

Android Malware vs iOS Malware

Android

- Android is used by most people, hence a lot of malware is developed for it
- It is common for an attacker to impersonate a known application and trick the user to install it

iOS

- Since the platform is very strict, malware is not that prominent compared to Android
- The strict application vetting and code signing by Apple greatly reduces the probability of a malicious application being installed on a user device

The Pegasus Spyware

- A spyware created by the NSO Group, an Israeli-based company, main clientele are governments
- Grants access to contacts, location, SMS, email, microphone, camera, etc.
- Multiple key government personnels, journalists and activists were targeted
- Uses multiple chained zero-day attacks to compromise its targets
- Pegasus was seen in the wild via:
  - Spear-phishing (text or email)
  - Zero-day attacks
- Can exploit Android or iOS
  - iOS version 15.1.1 (LATENTIMAGE)
  - Versions 15.5 and 15.6 (FINDMYPWN)
  - 16.0.3 (PWNYOURHOME)
- << update to include recent campaign >>

Triple Threat

NSO Group's Pegasus Spyware Returns in 2022 with a Trio of iOS 15 and iOS 16 Zero-Click Exploit Chains

By Bill Marczak, John Scott-Railton, Bahr Abdul Razzak, and Ron Deibert April 18, 2023

https://citizenlab.ca/2023/04/nso-groups-pegasus-spyware-returns-in-2022/

iOS

iOS Overview

- iOS is the mobile operating system of Apple
- Very popular
- Restrictive but works very well with the Apple ecosystem (“walled garden”)
- Apple has full control of both hardware and software
- EULA forbids reverse engineering
- iOS updates support old iPhones (iOS 17 supports iPhone XR)
- Apple can push updates faster

Jailbreaking iOS

- Most of the time, we need unrestricted access to the mobile operating system
- Jailbreaking voids the warranty of your product
  - You run also against the risk of “bricking” the device
- Be careful of the tools you use for jailbreaking!
- Please.. don’t jailbreak production devices

Known iOS Jailbreaks

- checkm8
- checkra1n
- Unc0ver
- Fugu15
- palera1n

iOS Applications (1)

- All third-party applications are run under the "mobile" user
- Each application are "sandboxed" from each other
- Applications can be written in C, Objective-C or Swift
- Apps are distributed in IPA format
  - Technically .ZIP files
- <<INSERT IPA directory structure>>

iOS Applications (2)

- Custom URL handlers
  - Custom URL schemes provide a way to reference resources inside your app. Users tapping a custom URL in an email, for example, launch your app in a specified context. Other apps can also trigger your app to launch with specific context data; for example, a photo library app might display a specified image.
  - Offers additional attack vectors for your application
- Universal Links
  - When users tap or click a universal link, the system redirects the link directly to your app without routing through Safari or your website.
  - In addition, because universal links are standard HTTP or HTTPS links, one URL works for both your website and your app. If the user has not installed your app, the system opens the URL in Safari, allowing your website to handle it.
  - When users install your app, the system checks a file stored on your web server to verify that your website allows your app to open URLs on its behalf.
  - Only you can store this file on your server, securing the association of your website and your app.

iOS Application Code Signing

- iOS requires that all executable code is signed using an Apple-issued certificate
- Developers join the Apple Developer Program and a certificate is issued
- Developers create their applications via the Xcode IDE and sign with the certificate and submit it for publication to the App Store
- Apple sign the application with their private key

iOS Security Features

- System security
  - Hardware and software tightly knit together
- Encryption and data protection
- Network security
  - App Transport Security (ATS) requires that all HTTP connections made use HTTPS.
  - ATS blocks connections that fail to meet minimum security specifications.
- Application security
  - Code Signing
- Apple Pay
- Internet Services
- Device Control
  - Secure wipe
- Privacy Controls
  - “Ask app not to track”

iOS Testing Environment

- You will need a an operating system that you can use to interact and login with the mobile device
  - This can be any virtual machine - Kali Linux is a good start
- You will need a mobile device where you can freely install applications that you are testing
  - Preferably, an old jailbroken iPhone
  - iOS Simulator
  - Apple Security Research Program
  - Correlium
- Points of consideration when selecting devices: price, speed, ease of rooting, restoration

Android

Android Overview

- Android was developed by the company Open Handset Alliance, a project sponsored by Google (acquired eventually)
- Android is the largest installed base and widely adopted my several manufacturers
- Device manufacturers manipulate the OS according to their needs
- Though there is a common base OS, the disparity in hardware and software contributes to the difficulty in securing everything “Android OS”

Rooting Android

- Rooting an Android device varies highly depending on the device hardware and vendor
- Several approaches in rooting an Android device
  - Unlocking the bootloader (preferred)
  - Exploits (such as CVE-2019-2215, CVE-2022-0847)

Android Applications

- Applications are developed using Java or Kotlin
- APK (Android PacKage) is .zip compressed file
- Applications bytecode compiled into DEX format, then interpreted by an Android virtual machine (VM)
- The VM for Android 5.0 is called AOT (Ahead-of-Time) compilation
  - For lower versions, the VM is called Dalvik
- Each application is assigned a unique UID and GID at the time of installation
- Android developers sign their own applications
  - Definitely not as strict as iOS
- Signing DOES NOT validate the identity of the developer

Android Security Fix Process

1. First, a flaw is disclosed to the Open Handset Alliance (OHA)
    a. This flaw can be present in code outside the OHA, requiring reaching out to owners
    b. If a fix is made, the OHA is notified
2. The fix is shared with multiple vendors
    a. Each vendor will test this thoroughly, considering dependencies with their own applications
3. After the fix and update is vetted, it is shared to the mobile operator (MO)
    a. Repeats the testing process (!!)
4. If testing and integration is OK, it can now be shared to Android users

THIS TAKES A LOT OF TIME!!

Android Testing Environment

- Physical device
- Emulator
  - Android Studio
  - Corellium
- Host systems
- Virtual machines
- Considerations of physical vs emulator still applies!

Mobile Application Security Testing

Mobile App Security Testing

- It is a catchall phrase referring to the evaluation of mobile app security via static and dynamic analysis
- Terms such as "mobile app penetration testing" and "mobile app security review" are used somewhat inconsistently in the security industry, but these terms refer to roughly the same thing
- A mobile app security test is usually part of a larger security assessment or penetration test that encompasses the client-server architecture and server-side APIs used by the mobile app

Mobile App Security Testing

- Mobile app security testing is usually used in two contexts
  1. Testing a nearly finished or production-ready version of the app, identify security issues, and write a report
  2. Implementation of requirements and the automation of security tests from the beginning of the software development life cycle
- The same basic requirements and test cases apply to both contexts, but the high-level method and the level of client interaction differ

Principles of Testing

- **Black-box testing**
  - Conducted without the tester's having any information about the app being tested
  - The main purpose of this test is allowing the tester to behave like a real attacker in the sense of exploring possible uses for publicly available and discoverable information
- **White-box testing**
  - The tester has full knowledge of the app (all information is provided)
  - The information may encompass source code, documentation, and diagrams
- **Gray-box testing**
  - In between the two aforementioned testing types: some information is provided to the tester (usually credentials only), and other information is intended to be discovered
  - This type of testing is an interesting compromise in the number of test cases, the cost, the speed, and the scope of testing
  - Gray-box testing is the most common kind of testing in the security industry

Vulnerability Analysis: Static and Dynamic

- Vulnerability analysis is usually the process of looking for vulnerabilities in an app
  - Static
  - Dynamic
- Although this may be done manually, automated scanners are usually used to identify the main vulnerabilities
- **Static Application Security Testing (SAST)**
  - Involves examining an application's components without executing them, by analyzing the source code either manually or automatically
- **Dynamic Application Security Testing (DAST)**
  - Involves examining the app during runtime, can be manual or automatic
  - Usually doesn't provide the information that static analysis provides, but can detect interesting elements (assets, features, entry points, etc.) from a user's point of view

Static Analysis

- During static analysis, the mobile app's source code is reviewed to ensure appropriate implementation of security controls
- In most cases, a hybrid automatic/manual approach is used
  - Automatic scans catch the low-hanging fruit
  - Manual approach explores the code base with specific usage contexts in mind

Static Analysis: Manual Code Review

- Manually analyzing the mobile application's source code for security vulnerabilities.
- Methods range from a basic keyword search via the 'grep' command to line-by-line examination
  - IDEs (Integrated Development Environments) often provide basic code review functions and can be extended with various tools.
- A common approach is identifying key security vulnerability indicators by searching for certain APIs and keywords
  - Such as database-related method calls like "executeStatement" or "executeQuery" as code containing these strings is a good starting point for manual analysis
- Great for identifying vulnerabilities in the business logic, standards violations, and design flaws
  - Especially when the code is technically secure but logically flawed
- Slow, tedious, time-consuming process

Static Analysis: Automatic Source Code Analysis

- Automated analysis tools can be used to speed up the review process of Static (SAST)
- Checks the source code for compliance with a predefined set of rules or industry best practices, then typically display a list of findings or warnings and flags for all detected violations
    - Tools can run against the compiled app only, some must be fed the original source code, and some run as live-analysis plugins in the Integrated Development Environment (IDE)
    - Produces a lot of false positives if they are not configured for the target environment

Dynamic Analysis

- The focus of DAST is the testing and evaluation of apps via their **real-time execution**
- The main objective is finding security vulnerabilities or weak spots in a program while it is running
- Conducted both at the mobile platform layer and against the backend services and APIs, where the mobile app's request and response patterns can be analyzed
- Usually used to check for security mechanisms that provide sufficient protection against the most prevalent types of attack, such as
  - Disclosure of data in transit
  - Authentication and authorization issues
  - Server configuration errors
- MobSF is a good start
- Frida is KING

Avoiding False Positives in Automated Scanning Tools

- Automated testing tools' lack of sensitivity to app context is a challenge as these may produce false positives
- In any case, consider exploit scenarios when you perform the assessment; don't blindly trust your scanning tool's output

Avoiding False Positives in Clipboard

- When typing data into input fields, the clipboard can be used to copy in data
- The clipboard is accessible system-wide and is therefore shared by apps, therefore this sharing can be misused by malicious apps to get sensitive data that has been stored in the clipboard
- Before iOS 9, a malicious app might monitor the pasteboard in the background while periodically retrieving the `[UIPasteboard generalPasteboard].string`.
  - As of iOS 9, pasteboard content is accessible to apps in the foreground only, which reduces the attack surface of password sniffing from the clipboard dramatically

Avoiding False Positives in Clipboard (2)

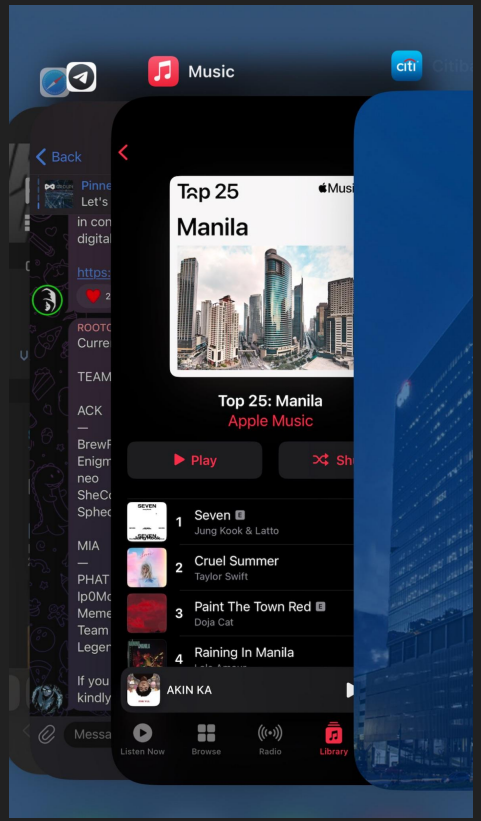
- Disabling pasting in passwords input fields is generally not advised, as:
  - Preventing pasting into input fields of an app does not prevent that a user from copying sensitive information
    - Since the information has already been copied before the user notices that it's not possible to paste it in, a malicious app has already sniffed the clipboard
  - Users might even choose weaker passwords that they can remember and they cannot use password managers anymore, which would contradict the original intention of making the app more secure
- When using an app you should still be aware that other apps are reading the clipboard continuously, as the Facebook app did
- Still, copy-pasting passwords is a security risk you should be aware of, but also cannot be solved by an app

OWASP MASVS

OWASP MASVS

- An active project by OWASP and currently the industry standard for mobile app security
- It can be used by mobile software architects and developers seeking to develop secure mobile applications, as well as security testers to ensure completeness and consistency of test results.
- MASVS Control Groups represent the most critical areas of the mobile attack surface
    - MASVS-STORAGE, MASVS-CRYPTO, MASVS-AUTH, MASVS-NETWORK, MASVS-PLATFORM, MASVS-CODE, MASVS-RESILIENCE
- Tests are associated with security levels: Level 1, Level 2, Level R
- Many controls, application context matters

Sample Test: Data Storage - Level 2



Mobile Pentesting

Mobile Penetration Testing (Pentesting)

- The classic approach involves all-around security testing of the app's final or near-final build, e.g., the build that's available at the end of the development process
- For testing at the end of the development process, the Mobile App Security Verification Standard (MASVS) and the associated checklist is a great baseline
- A typical security test is structured as follows:
  - Preparation
  - Intelligence Gathering
  - Mapping the Application
  - Exploitation
  - Reporting

Mobile Pentesting: What Do We Actually Test?

- In a typical network penetration tests, we aim to exploit hosts, servers, etc.
  - These machines are “owned” by the organization
- In the case of a mobile penetration test, the mobile device is typically a personal device - not “owned” by the organization
- In a mobile penetration test, we focus on testing the mobile application and its environment
  - Any device that the mobile application talks to should be considered to be included in the scope
- In most cases, the scope of a test includes the mobile application itself, how it stores data, and the requests it sends to the back-end server
  - *Web app pentest-ish*

Intercepting Mobile Traffic

- Similar to a web application penetration test, it is essential to intercept TLS traffic to and from the mobile application
- Intercepting TLS traffic allows us to view, edit, and run test cases on the data sent
- Intercepting TLS traffic in Android is different with iOS
- If you do not understand how controls preventing interception are implemented, you *will* have a difficult time
- Useful tools: PortSwigger Burp Suite

SSL Pinning

- Defense mechanism against malicious certifications
- You “pin” a certificate or public key for each domain
- Prevents “in-the-middle” attacks and scenarios
  - Including HTTPS interception
- Tools to bypass
  - iOS: SSLKillSwitch2, Objection, Custom Frida Scripts.
  - Android: Custom Frida Scripts, Objection
- Off-the-shelf tools will not work with custom SSL pinning implementations
  - Consider custom TCP/IP implementations, non-HTTP protocols