Mobile hacking

Motivations

- **Surveillance**: • Audio • Camera • Call logs • Location • SMS messages
- **Financial**: • sending high rate SMS • stealing transaction authentication numbers (TANs) • extortion via ransomware • fake antivirus • making expensive calls
- **Data theft**: • Account details • Contacts • Call logs • phone number • stealing IMEI
- **Botnet activity**: • launching DDoS attacks • click fraud • sending SMS
- **Impersonation**: • SMS redirection, sending e-mail messages, posting to social media

Attack vectors

- Malware
- Data exfiltration
- Data tampering
- Data loss

Vulnerabilities and risks

- Malicious apps in stores
- Mobile malware
- App sandboxing vulnerabilities
- Weak device and app encryption
- OS and app update issues
- Jailbreaking and rooting
- Mobile application vulnerabilities
- Privacy issues (Geolocation)
- Weak data security
- Excessive permissions
- Weak communication security
- Physical attacks
- Insufficient code obfuscation
- Insufficient transport layer security
- Insufficient session expiration

Security issues from App Stores

- Used to distribute malware/malicious apps
- App: Software that runs on mobile devices
- App Store: Distribution platform
  - Can be from owners of the OS e.g. Apple, Google play, Microsoft.
  - Or third parties e.g. Amazon Appstore, GetJar, and APKMirror.
- Can be distributed through
  - legitimate app stores

        - because of insufficient or no vetting of apps
    - unofficial app stores
        - user is social engineered to download and execute

Sandboxing

- App sandboxing
    - Limits resources available to an app
    - Isolates one from another
    - ❗ A vulnerable one can still be exploited

Mobile spam

- Also known as SMS spam, text spam, or m-spam
- Advertisements or malicious links
- E.g. you've won a prize, click here to claim it.

SMiShing

- SMS phishing
- Acquire personal information through SMS with malicious links
- Effective as
    - Easy through e.g. using prepaid SMS card using fake identity
    - Usually not checked by antiviruses
    - Users are not familiar
- E.g. "PayPal - your account has been locked"

Pairing

- Pairing with malicious devices may enable e.g. BlueSnarfing and BlueBugging

Hacking mobile platforms

Mobile platform attack vectors

- Enabled by extensive usage and implementation of bring your own device (BYOD) policies
- Device
  - Phishing
- Network
- Data-center/cloud

OWASP Top 10 Mobile Threats

1. **Improper Platform Usage**
   - Misuse of a platform feature or failure to use a platform security controls
   - Caused by insecure coding towards an exposed API
   - E.g. • Android intents • platform permissions • cloud risks • TouchID misuse • Keychain
   - Allows feeding malicious inputs or unexpected sequences of events to the vulnerable endpoint
2. **Insecure Data Storage**
   - Caused by assumption that users/malware will not have access to a mobile device's filesystem.
   - Filesystem are easily accessible through computer connection and specialized tools.
   - 💡 Use strong encryption
   - 💡 Careful when logging, caching, storing data, sending analytics, setting cookies, etc.
3. **Insecure Communication**
   - Allows exploiting vulnerabilities to intercept sensitive data in transit
   - Communications include TCP/IP, WiFi, Bluetooth/Bluetooth-LE, NFC, audio, infrared, GSM, 3G, SMS...
   - Attacks include e.g. compromising local network in coffee shop, carrier/network devices or installing malware
   - Caused by e.g. • poor handshaking (with weak encryption) • not using SSL/TLS • lack of certificate • weak negotiation • cleartext communication of sensitive assets.
4. **Insecure Authentication**
   - Captures notions of authenticating the end user or bad session management
   - Usually attacked through automated tools
   - Caused by poor or missing authentication schemes
   - Typically done via mobile malware within the device or botnets owned by the attacker.
   - 💡 Use access tokens, never store keys locally, store data encrypted, do not use TouchID or 4-digit pins (even hashes can be cracked with rainbow tables easily), always authenticate on back-end
5. **Insufficient Cryptography**
   - Cryptography was attempted, but it wasn't done correctly
   - Can be caused by weak algorithms, poor key management processes

- Creation and Use of Custom Encryption Protocols
- 💡 Do not trust code encryption from underlying OS 🤗 e.g. in iOS
  - Apps are encrypted and signed by trustworthy sources
  - iOS app loader will decrypt the app in memory execute after signature validation
  - Attacker can use jailbroken device and take snapshot of application memory once it's decrypted

6. **Insecure Authorization**
   - Allowed by poor or missing authorization schemes
   - Usually attacked through automated tools
   - Typically done via mobile malware within the device or botnets owned by the attacker.
   - Attacks include
     - binary attacks when application is in offline mode
     - using low-privilige session to gain more access by manipulating GET/POST request
   - Weaknesses include
     - Insecure Direct Object Reference (IDOR) i.e. endpoints without authorization checks.
     - Hidden endpoints as developers assume they'll not be seen by anyone
     - User role or permission transmissions from app to back-end
   - 💡 Always verify claims/roles in back-end independently from client.
7. **Client Code Quality**
   - All of the code-level implementation problem in the mobile client
   - Caused by passing untrusted inputs to method calls
   - Covers security decisions via untrusted inputs
   - May lead to
     - memory leaks and buffer overflows through app on the mobile device
     - foreign code execution or denial of service on remote server endpoints
8. **Code Tampering**
   - Usually done through changing forms of the apps to collect data
   - Attacker can directly modify the code, change the contents of memory dynamically, change or replace the system APIs that the application uses, or modify the application's data and resources
   - Malicious apps are typically distributed in third party stores
   - Phishing attacks can be used to trick users to install the apps.
   - E.g. • binary patching • local resource modification • method hooking • dynamic memory modification
   - 💡 Countermeasures
     - Integrity checks during run-time.
     - Do not run if the device is rooted/jailbroken.
9. **Reverse Engineering**
   - Analysis of the final core binary to determine the source code, libraries, ...
10. **Extraneous Functionality**
    - Attacker analyzes the apps for test-code left behind, hidden functionality, switches etc.

    - Attacked to learn about back-end systems or running unauthorized high-privileged actions

Basic Threats

- Malware / rootkit
- Data Loss
- Data Tampering
- Data Exfiltration

Vulnerabilities And Risks on Mobile Platforms

- Malicious third-party application / in the store
- Application vulnerability
- Data security
- Excessive permissions
- Weak encryptions
- Operating System update issue
- Application update issue
- Jailbreaking / rooting
- Physical attack

OS Sandboxing Issue

- Sandbox is a security mechanism for separating running programs, usually in an effort to mitigate system failures or software
vulnerabilities from spreading
- Sandbox limits the app's access to files, preferences, network resources, ...
- Advanced malware designed to bypass it, by fragment code or put sleep timer in the script to bypass the inspection process

Android

Device Administration API

- Provides device administration features at the system level
- This API allows to create security-aware apps that are useful in the enterprise settings, where require rich control over employee devices

Rooting

- A process of allowing user to attain privileged control
- Needed for modify settings, get full control over the kernel or install custom ROMs
- E.g. SuperOneClick allows to root Android phones.

iOS

Jailbreaking

- Rooting the iOS
- Escalating the privileges on iOS to remove or bypass the factory default restrictions

Jailbreaking exploits

- **Userland exploit**
  - 📝 Allow user-level access without escalating iBoot-level access
- **iBoot exploit**
  - 📝 Allow user-level and boot-level access
- **BootROM exploit**
  - 📝 Allow user-level and boot-level access

Jailbreaking Techniques

Untethered jailbreak

- Does not require to reboot with a connection to your computer
- Exploit bypass the iBoot sequence

Tethered jailbreak

- Need a connection to your computer to reboot, without it, the boot stuck with an Apple logo
- Offers complete jailbreak features

Semi-Untethered jailbreak

- Allows to boot into the iOS device, but with limited functionality
- The jailbreak functions will be disabled until the launch of a jailbreak app

Semi-Tethered jailbreak

- Allows you to boot with limited functionality
- To get the full functionality, a reboot with a tethered jailbreak required
- Tethered jailbreak + a package to allow reboot with limited functionality
- ❗ A reboot no longer retains the patched kernel
  - But the installed jailbreaking tool can be used if admin privileges are required.

Windows Phone

- Windows Phone 8 using the Windows NT Kernel
- Windows Phone 8 include app sandboxing, remote device management, native code support (C++)

BlackBerry OS

- Support for Java Micro Edition MIDP 1.0 and MIDP 2.0
- OS update with BlackBerry over the air software loading service (OTASL)

Mobile Device Management (MDM)

- Deployment, maintenance and monitoring of mobile devices

MDM Functions

- Enforce device to be locked after certain failed login
- Enforce strong password policy for all BYOD.
- MDM can detect attempt of hacking BYOD device and then limit the network access of the affected device
- Enforce confidentiality by using encryption as per organization's policy
- Administration and implementation of Data Loss Prevention (DLP)

MDM Deployment Methods

On-site MDM Deployment

- Install MDM application on local servers
- Management is done by local staff
- Provide full control over the MDM

Cloud-based MDM Deployment

- MDM application is installed and maintained by a third party
- Less administration needed
- The deployment and maintenance is the responsibility of the service provider

Bring Your Own Device (BYOD)

BYOD is a trend of employees using their personal devices for work. It could be a laptop, a phone, etc...

BYOD Policies

BYOD policies should include:

- Device: which devices and operating systems are supported
- Password: require all devices to be password protected
- Access: determine which data can be accessed from employee's device
- Application: which applications allowed, which should be banned

Mobile security guidelines

- Avoid auto-upload of files
- Perform security assessment of applications
- Turn off Bluetooth
- Allow only necessary GPS-enabled applications
- Do not connect to open network
- Install applications from trusted sources
- Use strong password
- Use Mobile Device Management (MDM) softwares
- Update operating system often
- Do not allow rooting / jailbreaking
- Encrypt phone storage

- Periodic backup
- Configure mobile device policies

Mobile attacks

Mobile threats

- Takes advantage of the lack of security control in smartphones
- Can also be caused by a malicious app
- Older OS versions have known vulnerabilities
  - Patched in newer versions but users may not update them.
  - Vendors may not update phones after a while, maybe before warranty period.
- Vendors having own modified version of Android increases security risks
- Data transmission threats through Bluetooth, WiFi, 3G/4G/5G or wired connection to a computer.

Attack vectors

Attacks on the device

- **Malicious code signing**
  - Obtaining a code-signing key from the code signing service to create a malicious application
- **JAD File Exploit**
  - Attacker crafts a `.jad` file with spoofed information
  - Java Application Description (`.jad`) contains attributes of Java application

Browser-based attacks

- **Framing**
  - Integrating another page through `iframe` element of HTML
  - Enables clickjacking to steal information
- **Man-in-the-Mobile**
  - Also known as • **MitMo** • **Man-in-the-Phone**
  - Malware to spy on e.g. SMS OTPs (one-time passwords) or voice calls and relay them back to the hackers
- **Buffer Overflow**
  - Caused by not truncating input data when it's longer than the reserved space and leads to overwriting other data in memory.
  - 📝 Both iOS and Android are vulnerable as they use C/C++ in their kernels
- **Data caching**
  - Inspected to gain sensitive information stored in them
- Clickjacking

Phishing

- Redirecting uses to legitimate looking malicious sites through e.g. pop-ups, emails
- Mobile users are more vulnerable due to smaller size of the browsers, URLs, warnings etc.
- See also Phishing | Social Engineering

Phone/SMS-based attacks

- **Baseband Attacks**
  - Exploits GSM/3GPP processor (exchanges radio signals with cell towers)
- See Mobile-based social engineering | Social engineering types including e.g. • SMS phishing (SMSiShing) • Fake security apps • Repackaging legitimate apps • Malicious apps

Application-based attacks

- Sensitive data storage
- No encryption / weak encryption
- Improper SSL validation
- Configuration manipulation
  - E.g. through external configuration files
- Dynamic runtime injection
  - E.g. stealing data in memory
- Unintended permissions
- Escalated privileges
- Access to device and User info
- Third-party code
- Intent hijacking
- Zip directory traversal
- Side channel attack
- UI overlay/pin stealing
- Intent hijacking
- Clipboard data
- URL schemes
- GPS spoofing
- Weak / no local authentication
- Integrity / tampering / repackaging
- Unprotected app signing key
- App transport security

System attacks

- Malware
    - Attacks the underlying system
- No passcode / weak passcode
- iOS jailbreaking
- Android rooting
- OS data caching
- Accessible passwords and data
- Carrier-loaded software
    - Through e.g. bloatware
- No encryption / weak encryption
- User-initiated Code
- Zero-day exploits
- Device lockout
- Kernel driver vulnerabilities
- Confused deputy attack
- TEE/secure enclave processor
- Side-channel leakage
- Multimedia/file format parsers
- Kernel driver vulnerabilities
- Resource DoS
- GPS spoofing

Network attacks

- Wi-Fi
    - E.g. no-encryption or weak encryption
- Rogue Access Point
- Packet sniffing
- Man-in-the-Middle (MITM)
    - **SSLStrip**: websites are downgrades to use HTTP
    - **Fake SSL certificates** issued by attacker
- Session hijacking
- DNS poisoning
- BGP hijacking
- HTTP proxies
- **Sidejacking**
    - Listening to traffic to steal exchanged cookies to extract session IDs

Data center/cloud attacks

Web server attacks

- Platform vulnerabilities in OS and server software
- Server misconfiguration
- **Cross-site scripting (XSS)**
- Cross-site forgery (XSRF)
- Weak input validation
- Brute force attacks
- Cross Origin Resource Sharing
- Side channel attack
- Hypervisor attack
- VPN

Database attacks

- SQL injection
- Privilege escalation
- Data dumping
- OS command execution