Mobile App Testing

Standards and Guidelines

Mobile App Vetting Process

Process for determining if an app meets the requirements of a specific organization.

Vetting the Security of Mobile Applications - NIST Special Publication 800-163r1

Vetting Process Overview

- App store vetting only ensures the app meets the store requirements
- Does not insure it meets organizational requirements
- A standard process allows sharing of results and insures thorough review
- Process should be based on organization's security policies
- May involve app vendor

App Intake Phase - Planning

- Planning
  - Develop app security requirements based on a **risk analysis,** the organization's security requirements and other procedures or systems that might mitigate risks
- Security Requirements
  - General requirements
    - Software characteristic or behavior that an app should exhibit in order to be considered secure
    - National Information Assurance Partnership (NIAP) - https://www.niap-ccevs.org/MMO/PP/-394-/pp_app_v1.2.htm
    - NIST Special Publication 800-53 - Security and Privacy Controls for Federal Information Systems and Organizations - https://csrc.nist.gov/publications/detail/sp/800-53/rev-4/final
    - OWASP Mobile Application Security Testing Guide
- Organizational requirements
  - Organization-specific risks and vulnerabilities

App Testing Phase

- Testing using one or more tools
- Vetting process is a function of the time and skills of the reviewers

App Approval/Rejection Phase

- Based on both general and organization specific criteria
- Include specifics about which controls or topics passed/failed
- **YOU NEED TO MAKE THIS JUDGEMENT FOR YOUR APP**

Reporting Phase

- Include artifacts to demonstrate conclusions
- Include this in your report for class

App Testing Principles

App Testing Objectives

- Enabling Authorized Functionality
- Preventing Unauthorized Functionality
- Limiting Permissions
- Protecting Sensitive Data
- Securing App Code Dependencies
- Testing App Updates

From original version of NIST Special Publication 800-163r1

Enabling Authorized Functionality

- The app must work as described; all buttons, menu items, and other interfaces must work. Error conditions must be handled gracefully, such as when a service or function is unavailable (e.g., disabled, unreachable, *etc.*).
- Test functions of UI and keyboard
- Test use of all appropriate sensors
- Test use of appropriate communications
- *This is not considered in testing for this class*

Preventing Unauthorized Functionality

- Unauthorized functionality, such as data exfiltration performed by malware, must not be supported.
- Other forms of malicious functionality include:
    - Injection of fake websites into the victim's browser in order to collect sensitive information, acting as a starting point for attacks on other devices, and generally disrupting or denying operation.
    - Use of banner ads that may be presented in a manner which causes the user to unintentionally select ads that may attempt to deceive the user. These types of behaviors support phishing attacks and may not be detected by mobile antivirus or software vulnerability scanners as they are served dynamically and not available for inspection prior to the installation of the app.
    - Use of the phone calls or SMS messages that is not clearly stated in the EULA or app description
    - Use of the unique phone identifier information in order to keep track of users instead of using a username/password scheme.

Limiting Permissions

- Apps should have only the minimum permissions necessary and should only grant other applications the necessary permissions
- Some apps have permissions that are not consistent with EULAs, app permissions, app descriptions, in-program notifications, or other expected behaviors and would not be considered to exhibit secure behavior (May just be poorly designed, not malicious)
    - Access to file input/output (I/O) and removable storage
    - Access to privileged commands
    - Access to APIs

Protecting Sensitive Data

- Apps that collect, store, and transmit sensitive data should protect the confidentiality and integrity of this data. This category includes preserving privacy, such as asking permission to use personal information and using it only for authorized purposes.
- Cryptography is often used to protect sensitive data, but keys may not be managed correctly.
- Privacy considerations need to be taken into account for apps that handle PII, including mobile-specific personal information like location data and pictures taken by onboard cameras, as well as the broadcast ID of the device.
- Another concern with protecting sensitive data is data leakage. For example, data is often leaked through unauthorized network connections.

Securing App Code Dependencies

- The app must use any dependencies, such as libraries, in a reasonable manner and not for malicious reasons
- Native Methods - typically calls to a library function that has already been loaded into memory.
- External Libraries and Classes:
  - Includes any third-party libraries and classes that are loaded by the app at run time.
  - Malicious apps can use library and class loading as a method to avoid detection
  - Introduce outside code to an app without the direct control of the developer
- Dynamic Behavior - not all operating behaviors are a result of user input. Executing apps may also receive inputs from data stored on the device.
- Inter-Application Communications - In Android platforms, inter-application communications are allowed but regulated by what Android calls "intents." An intent can be used to start an app component in a different app or the same app that is sending the intent.

Testing App Updates

- Test updates before they are deployed
- May have to use MAM to disable automatic updates

General Testing Types

Testing Types

- Black-box testing
- White-box testing
- Gray-box testing

Testing Approaches

- Testing in development vs testing in deployment
- Vulnerability Testing vs Penetration Testing
- Malicious Activity vs Security Vulnerabilities

Testing Approaches

- Source Code Versus Binary Code
  - Open source apps may provide source code, but most downloaded apps do not
  - Source code can be reviewed manually
  - Byte code or machine code may have to be decompiled or reverse engineered
- Static Versus Dynamic Analysis
  - Static analysis examines the app source code and binary code, and attempts to reason over all possible behaviors that might arise at runtime
  - Dynamic analysis operates by executing a program using a set of input use cases and analyzing the program's runtime behavior
    - Dynamic analysis is unlikely to provide 100 percent code coverage

Specific Mobile Security Testing Guidelines

Specific Testable Items

- OWASP Mobile AppSec Verification Standard(MASVS)
- For Government apps
  - NIAP Protection Profiles, Application Software, Mobility - https://www.niap-ccevs.org/Profile/PP.cfm

Testing Procedures

[OWASP Mobile Application Security Testing Guide (MASTG)]

MASVS vs MASTG

- Mobile Application Security Verification Standard (The Checklist)
  - Specific security standards that should be verified
  - Basis for your penetration testing report
- Mobile Application Security Testing Guide
  - What to test for and **how to test it**
  - Three sections
    - General testing guide
    - Android testing guide
    - IOS testing guide
  - **Test cases, and tool procedure suggestions**

Assignment 2: Select Test Apps

- It's recommended to select a primary app and a backup app
- Do not download apps to a host computer or your primary mobile device – they may contain a virus that can affect the host
  - Only download to VM or emulator
- Sources for potentially malicious apps
  - Lowest scores and Virus Total apps on Immuniweb - https://www.immuniweb.com/mobile/
  - Androzoo
    - Abreviated apps list in Canvas Learning Resources for this module
- Make sure your app runs in your emulator!

Assignment 2: Select Test Apps

- Downloading apps from Androzoo
  - From browser on **Android Studio Emulator:**
  - `https://androzoo.uni.lu/api/download?apikey=`**`${APIKEY}`**`&sha256=`**`${SHA256}`**
  - **`APIKEY = 97e238cdcee2ae34a73a81ee6d9c494e137ab6bf1a574623386a9a7256ca35f5`**
  - **`SHA256 = first column in spreadsheet`**

Summary

- App Testing Process
- App Testing Principles
- Specific Mobile Security Testing Guidelines