CompTIA Mobile App Security+ Certification Exam (Android Edition) Live exam ADR-001 Beta Exam AD1-001

INTRODUCTION

This exam will certify that the successful candidate has the knowledge and skills required to securely create a native Android mobile application, while also ensuring secure network communications and backend Web services.

Test Purpose: The CompTIA Mobile App Security+ Certification Exam (Android Edition) is suited for those individuals with at least 24 months of application development experience, as well as solid familiarity with Java, the Android SDK, and principles of secure application development.

The successful candidate should have the knowledge and skills to:

- Describe fundamental principles of application security
- Describe the security model of Android devices
- Describe common threats to mobile application security
- Develop moderately complex applications using the Android SDK
- Describe Web services security model and vulnerabilities
- Properly implement SSL/TLS for Web communications
- Utilize the security features of the Android operating system and APIs
- Properly implement secure coding techniques
- Avoid insecure retention of data in memory
- Describe common implementations of cryptography such as PKI
- Leverage encryption for storage and/or communications
- Understand access control and file permissions
- Harden an application against attack to levels appropriate for the risk model of the application

Prerequisite knowledge: JAVA programming, Android SDK, SQL coding, mobile and application security essentials, and implementing encryption.

Domain	% of Examination
1.0 Mobile application security, SDLC, and threat models	18%
2.0 Android SDK, APIs, and security features	20%
3.0 Web service and network security	23%
4.0 Data security and implementing encryption	23%
5.0 Application hardening and reverse engineering	5%
6.0 Secure Java coding	11%
Total	**100%**

CompTIA Authorized Materials Use Policy

CompTIA Certifications, LLC is not affiliated with and does not authorize, endorse or condone utilizing any content provided by unauthorized third-party training sites, aka 'brain dumps'. Individuals who utilize such materials in preparation for any CompTIA examination will have their certifications revoked and be suspended from future testing in accordance with the CompTIA Candidate Agreement. In an effort to more clearly communicate CompTIA's exam policies on use of unauthorized study materials, CompTIA directs all certification candidates to the CompTIA Certification Exam Policies webpage:
http://certification.comptia.org/Training/testingcenters/policies.aspx

Please review all CompTIA policies before beginning the study process for any CompTIA exam. Candidates will be required to
abide by the CompTIA Candidate Agreement
(http://certification.comptia.org/Training/testingcenters/policies/agreement.aspx) at the time of exam delivery.

If a candidate has a question as to whether study materials are considered unauthorized (aka brain dumps), he/she should perform a search using CertGuard's
engine, found here:
http://www.certguard.com/search.asp

Or verify against this list:
http://certification.comptia.org/Training/testingcenters/policies/unauthorized.aspx

**Note: The lists of examples provided in bulleted format below each objective are not exhaustive lists. Other examples of technologies, processes or tasks pertaining to each objective may also be included on the exam although not listed or covered in this objectives document.

CompTIA is constantly reviewing the content of our exams and updating test questions to be sure our exams are current and the security of the questions is protected. When necessary, we will publish updated exams based on existing exam objectives. Please know that all related exam preparation materials will still be valid.

1.0 Mobile Application Security, SDLC, and Threat Models

1.1 Identify reasons for significance of secure mobile development.

- U.S. Regulatory requirements: PCI, HIPAA, FFIEC, FISMA
- International requirements: E.U. privacy
- Business requirements
- Consumer expectations (including privacy)
- Security risks which are unique or higher for mobile
    - Lost/stolen device (physical access)
    - Untrusted Wi-Fi networking (DNS attack, MITM)
    - Users running modified OS (rooting)
    - Telephony-related attacks (SMiShing, MitMo, toll fraud)

1.2 Compare relative severity of security issues.

- Unprotected Web interfaces
- Vulnerability to SQL injection
- Storage of passwords, sensitive data without encryption
- Transmission without encryption (TLS/SSL)

1.3 Explain a secure development process throughout application development.

- Security testing/review on release (and during development)
- Business requirements
- Specifications
- Threat model/architectural risk analysis
- Code review
    - Automated
    - Manual
- Perform security testing
    - Fuzzing
    - Security functionality
    - Dynamic validation
    - Risk-based testing
    - Penetration testing
- Types of documentation
    - Regulatory or corporate policy security or privacy requirements.
- Schedule on-going security tests post-OS upgrades

1.4 Summarize general application security best practices.

- Sanitizing input, input validation
- Contextually appropriate output escaping
- Good design considerations:
    - Logic in applications
    - Storage of variables
    - Database design

- Debugging
- Error handling
- Secure storage
- Secure communications
- Authentication and authorization
- Session management
- Ensuring application and data integrity
- Security by design vs. obscurity
- Sandbox

1.5 Identify the major architectural risks to weaknesses in an application.

- Build an architecture diagram of an application (including back-end services), along with descriptions of each component
- Establish a deep and comprehensive understanding of the application and its components
- Break the architecture into specific security zones for individual consideration
- For each zone, articulate and enumerate each of the following:
    - Who has access to the zone?
    - What would motivate someone to attack the system?
    - What would an attacker target, specifically, in each zone? (e.g., data, functions)
    - How could each target be attacked? (Reference Microsoft's STRIDE process.)
    - What would be the impact to the business of a successful attack?
    - What remediation could be implemented to reduce the likelihood of a successful attack?
    - Recommend and justify remediation based on their corresponding likelihood and magnitude of impact vs. their costs to the business

2.0 Android SDK, APIs and Security Features

2.1 Summarize the Android security architecture.

- System and kernel level security
- Application sandbox
- Application signing
    - Purpose
    - Key management
- Permissions
    - File system
    - Application-defined
    - URI permissions

2.2 Explain the Android permission model.

- Protected APIs

- Requesting permissions
- Defining permissions
- Use of signatures
- Protection levels
- Summarize the Device Administration API
    - Purpose and appropriate use
- Letting the user control access to sensitive data
    - Start the contacts activity to let the user select a contact for use by the application rather than require permission to access all contacts
    - Start the camera application to let the user take a picture for use in the application without requiring camera permissions

2.3 Describe secure inter-process communication.

- Public and private components
- Protecting access to
    - Services
    - Broadcast receivers
    - Activities
    - Content providers
    - Databases
- Securely accessing third-party components with IPC
- Types of attacks
    - Confused deputy
    - Intent sniffing
    - Intent hijacking
    - Data disclosure

2.4 Securely implement common features.

- Web view
- KeyChain

3.0 Web Service and Network Security

3.1 Summarize the risks in performing Web and network communications.

- Clear text transmission of data
- Man-in-the-middle attacks
- APN modification
- Insufficient validation of certificates/certificate chain
- SSL compromise
- DNS hijacking

3.2 Implement an SSL session with validation.

- Encryption/confidentiality of data
    - Basics of public-key crypto as used in SSL
    - Understanding of threats that SSL encryption protects against

- Authentication of the server (basic)
    - How server certificates are verified (default CA chain checking)
    - Understanding of threats that SSL server authentication protects against
- Authentication of the server (advanced)
    - Custom Hostname Verifiers
    - Customize trust (configuring application to only trust certain certs)
    - Use self-signed certificates for server authentication
- Authentication of the client
    - Explain basics of mutual-authentication SSL
    - Understanding of threats that SSL client authentication protects against
    - Deploy client certificate into application's keystore
    - Deploy client certificate into system keystore
    - Configure application to present client certificates for client authentication

3.3 Distinguish sound security protections for authentication.

- Explain pros/cons and implement device/application authentication techniques
    - Mutual-authentication SSL for client device authentication
    - Web service API keys for client application authentication
- Explain pros/cons and implement user authentication techniques
    - Storing/accessing user credentials using AccountManager
    - Basic authentication
    - Digest authentication
    - Token-based authentication using OAuth

3.4 Explain common threats and protections for Web services.

- Explain input validation
    - Need for input validation (lack of client/communicating-party authentication)
    - Postel's Law
    - Positive (whitelist) vs negative (blacklist) validation
    - Pros/cons of whitelist validation
    - Pros/cons of blacklist validation
- Explain cross site scripting (XSS) attacks
    - Stored XSS
    - Reflected XSS
    - XSS prevention strategies
- Explain cross site request forgery (CSRF attacks)
    - CSRF attack general principles
    - Login CSRF
    - CSRF prevention strategies
- Explain command/SQL injection attacks

- Basics of command/SQL injection
- Input validation as a defense strategy
- Parameterized queries/prepared statements

3.5 Describe proper implementation of session security.
- Highly random token
- Expire on timeout or exit
- Store in memory not in data
- Avoid static user token
- Android ID, Device ID and other device identifiers

4.0 Data security and Implementing Encryption

4.1 Explain how encryption and hashing works.
- Symmetric-keys and public-key cryptography
- One-way functions (e.g., hashes)
- Why is the salting of passwords needed
- Key generation, why 10,000 rounds is better than 2000
- Security by design vs obscurity

4.2 Summarize methods for securing stored data.
- Certificates
- Permissions and access rights
- Database security
  - Strong SQL database passwords
  - Sanitizing inputs (stopping SQL injection)
- Encryption

4.3 Distinguish proper implementation of encryption in an Android application.
- Encrypting the application (when it is enabled again)
- Why using GPG/OpenPGP might be better than your own version
- Using some known value like the MEID to obscure data, better than nothing?
- Obfuscation the code to stop someone reversing the algorithm that being using
- How to store passwords and keys so they cannot be extracted from the application
- How to encrypt or hash data stored in SQL databases
- Checking if the device encryption is enabled

4.4 Implement data security using the Android permissions model.
- Create a custom permission for your application
- Secure a service with the correct permissions
- Grant temporary permission to open a downloaded file

5.0 Application Hardening and Reverse Engineering

5.1 Explain reverse engineering.

- Explain what reverse engineering is
  - Explain "good" reasons for reverse engineering applications
  - Explain "bad" reasons for reverse engineering applications
  - Nature of the Android and Java platforms and why reverse engineering is easy/easier
- Explain basic reverse engineering techniques and approaches
  - Process/stages
  - APK components (classes, dex, certificates, manifest, layouts, jars, native libraries, resource/assets, etc.)
  - Static analysis
    - Strings/resources analysis
    - Disassembly
    - Decompilation
    - Pros and cons
  - Dynamic analysis
    - Sandboxes
    - Observing network communications
    - Android emulator
    - Live debugging
    - Pros and cons
  - Understand forward engineering (code => javac => dx => classes.dx) to understand reverse engineering
  - Reverse engineering tools
    - apktool, dex2jar, jd-gui

5.2 Explain reverse engineering countermeasures.

- ProGuard
  - Explain what ProGuard is
  - Explain why ProGuard can make reverse engineering more difficult
  - Deploy a default installation of ProGuard into an Android project
  - Explain why native methods can make ProGuard application more difficult
  - Explain why reflection can make ProGuard application more difficult
  - Explain the various options that can be configured in a ProGuard configuration
  - Provide an example where the default ProGuard configuration must be altered
  - Explain how native libraries impact reverse engineering
- Information leakage
  - Remove/reduce logcat logging info

    - Remove/reduce debug code
    - Remove/reduce stacktrace leaks
    - Catch exceptions
    - Things used to debug applications will help reverse the application if left in code
- Make man-in-the-middle (MITM) harder for network protocol reverse engineering
    - Use SSL for communications to server
    - Use certificate pinning: hardcode the cert for the server
- Detect package modifications
    - Check integrity of APK
    - MD5/SHA checksum of application public key? Check if individual fields match?
- Password Storage
    - Explain why reverse engineering can make recovery of static passwords easy
    - Explain what information, contained in an application, can be recovered via reverse engineering
    - Explain the trade-offs between storage, derivation, and user-supply of secret information

6.0 Secure Java Coding

6.1 Explain Java language structure and object oriented development.

- Classes, objects, methods, fields
- Exception handling, try/catch
- Packages

6.2 Demonstrate proper handling of sensitive information.

- Purge sensitive information from exceptions and from memory after usage
- Avoid logging highly sensitive information

6.3 Explain general secure Java coding best practices.

- Correct naming
- Limit the extensibility of classes and methods
- Define wrappers around native methods
- Get proper reference to external storage like SD-card
- Isolate unrelated code