ITMC411
Security in mobile computing

LECTURE 5
Mobile Vulnerability Scanners and Testing Tools

Common mobile application vulnerabilities

- Insecure data storage
- Memory leaks and corruption
- Supply chain vulnerabilities

Common mobile application vulnerabilities

Insecure Data Storage:

- **Sensitive data** (e.g., user credentials, financial info) improperly secured.
- **Risks**: Weak encryption, poorly protected database access, and exposed cookie storage.
- **Vulnerable to attacks**, especially on rooted devices or reverse-engineered apps.

Solutions:

- Use encryption and secure authentication.
- Conduct regular security audits.

Common mobile application vulnerabilities

Memory Leaks and Corruption:

- Common in apps using native languages like C, C++, Objective-C.
- Memory issues (e.g., leaks, buffer overflows) lead to app crashes or security exploits.

Risks:

- Can lead to denial-of-service (DoS) attacks.

Solutions:

- Apply best coding practices.
- Use Static Application Security Testing (SAST).

Common mobile application vulnerabilities

Supply Chain Vulnerabilities:

- Third-party components (libraries, frameworks) may contain bugs or malicious code.
- **Example**: ParkMobile breach — 21 million users' data compromised via a third-party vulnerability.

Solutions:

- Test third-party components thoroughly.
- Keep all components updated.
- Implement a "**shift-left**" security approach during development.

Types of mobile app security Tests

- Vulnerability scanning
- Penetration testing
- Risk assessment
- Security posture assessment

Types of mobile app security Tests

Vulnerability Scanning

- **Purpose**: Uses **automated** tools to find vulnerabilities in the app ecosystem.
- **Focus**: Looks for **known vulnerabilities**, particularly in **software dependencies** and **common code loopholes**.
- **Output**: Generates **reports** for developers/**QA** teams

Types of mobile app security Tests

Penetration Testing

- **Purpose**: **Simulates** attacks to identify weaknesses in the app.
- **Key Difference**: Involves ethical hackers, providing realistic, actionable threat data.
- **Output**: More detailed information on **exploit methods** and **loophole locations** compared to vulnerability scanning.

Types of mobile app security Tests

Risk Assessment

- **Purpose**: **Evaluates** the **risks** across **people**, **processes**, and **tools** in the app's ecosystem.

- **Steps**:
  - Catalog assets.
  - Identify potential threats.
  - Analyze how vulnerabilities can be exploited.

- **Output**: Provides insights into the severity and likelihood of risks, helping inform mitigation strategies.

Types of mobile app security Tests

Security Posture Assessment

- **Purpose**: **Prioritizes** risks from the **risk assessment** and **develops** strategies to **improve** the app's **security posture**.

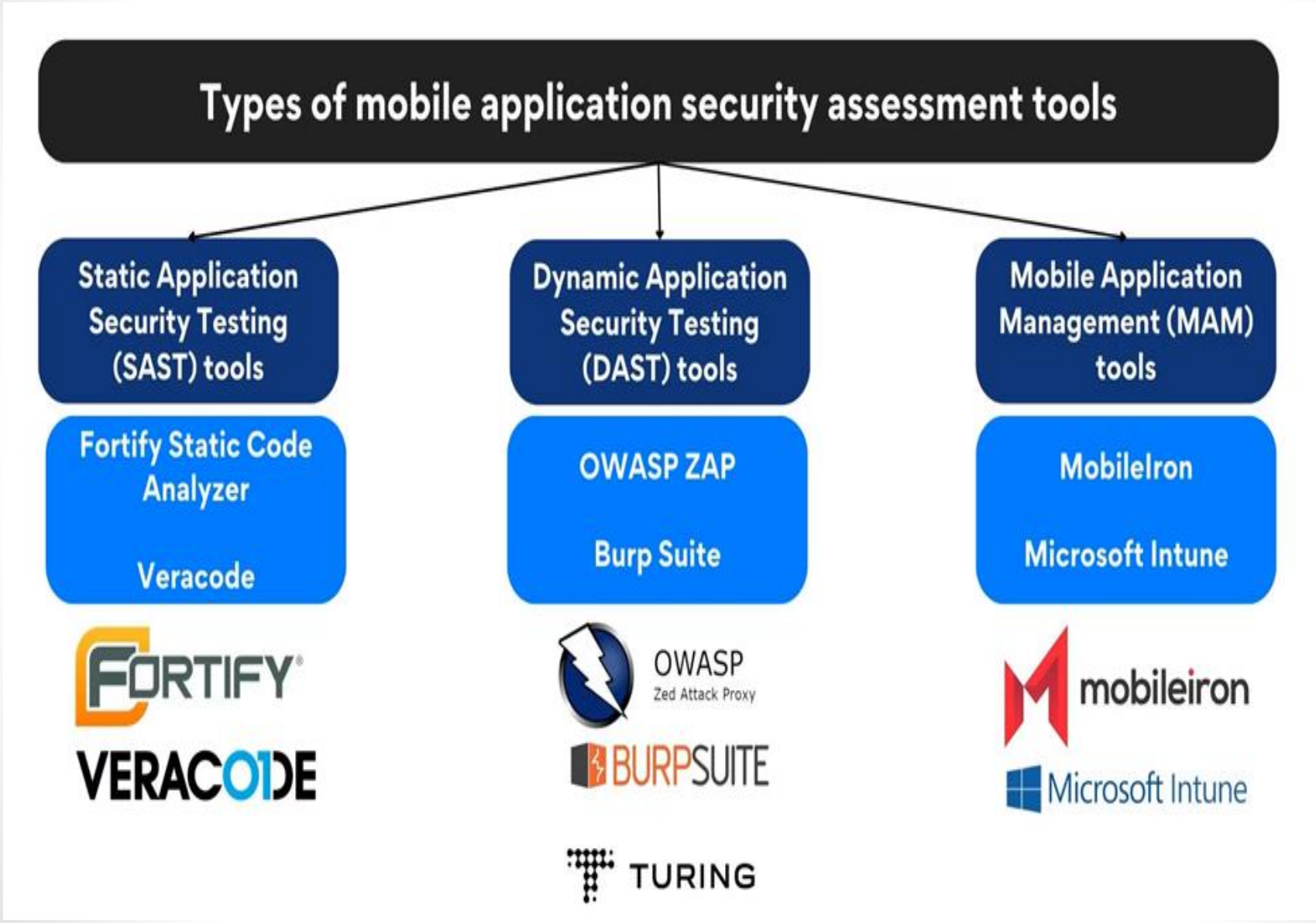
- **Strategies**: May include **stronger authentication**, **patching software**, **incident response** plans, and **continuous monitoring**.

- **Compliance**: Ensures alignment with **regulatory/industry** standards, protecting against **legal/financial** penalties.

Static and Dynamic Analysis

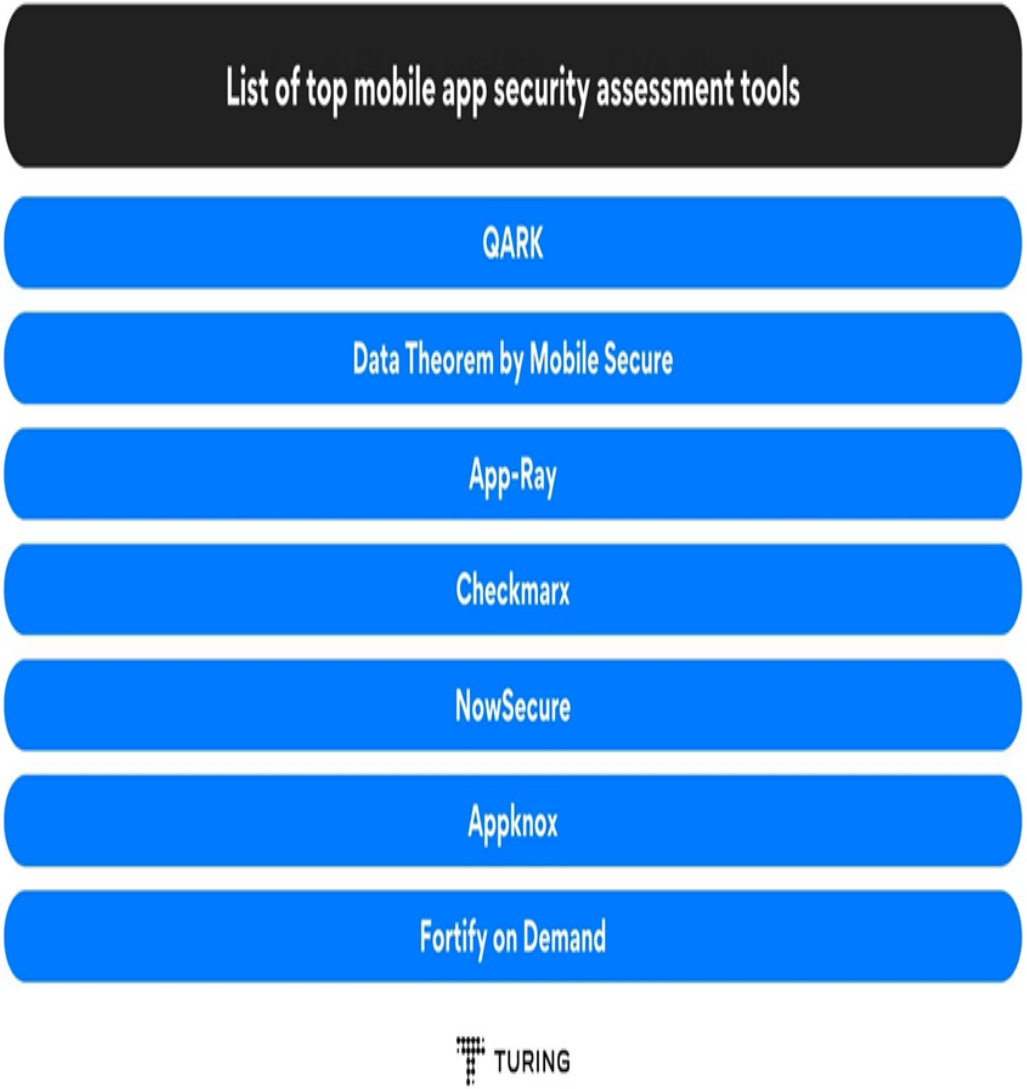
- **Static application security testing (SAST)**
  - **Tests** the **application code** for vulnerabilities **before running** it in an **app**.
  - Tools such as **Klocwork** and **Checkmarx** are useful for achieving **SAST**.
- **Dynamic application security testing (DAST)**
  - focuses on a **running app**.
  - **DAST tools** scan apps to check for any **loopholes** that may lead to security risks.
  - An example of a **DAST** tool for mobile is **HCL AppScan**.

Static and Dynamic Analysis



Top mobile app security assessment Tools

1. QARK
2. Data Theorem
3. App-Ray
4. Checkmarx
5. NowSecure
6. Appknox
7. Fortify on Demand
8. HCL AppScan
9. AppSweep
10. Veracode
11. Synopsys
12. Ostorlab



QARK

Purpose: Open-source tool for Android app security.

Key Features:

- **Static** code analysis, permission mapping, manifest analysis.
- Combines **static** and **dynamic** analysis.

Pros:

- Free and open-source.
- Generates detailed reports.
- Integrates with **CI systems**.

Cons:

- Android-only.
- Requires technical expertise.

Data Theorem by Mobile Secure

Purpose: Comprehensive tool for **Android** and **iOS** security.

Key Features:

- **Static** and **dynamic** analysis, vulnerability assessment, compliance testing.
- **Real-time** behavior monitoring.

Pros:

- Supports both **iOS** and **Android**.
- Continuous monitoring.

Cons:

- High pricing.
- Additional configuration for complex architectures.

App-Ray

- **Purpose**: Security testing for **iOS**, **Android**, and **Windows**.
- **Key Features:**
  - **Static** and **dynamic** analysis for **vulnerabilities** and **data leaks**.
- **Pros**:
  - Supports multiple platforms.
  - User-friendly with continuous monitoring.
- **Cons**:
  - Limited community support.
  - Requires an internet connection for analysis.

Checkmarx

- **Purpose**: **Code-level** security testing tool.
- **Key Features:**
  - Comprehensive **SAST** (**Static** Application Security Testing) with **manual** and automated options.
- **Pros**:
  - Seamless integration with development workflows.
  - Multi-language support.
- **Cons**:
  - Expensive.
  - Requires setup time.

NowSecure

- **Purpose**: Security testing for **iOS** and **Android**.
- **Key Features:**
  - **Dynamic** analysis, **real-time** monitoring, network and storage vulnerability detection.
- **Pros**:
  - Actionable reports with clear steps.
  - Advanced mobile forensics.
- **Cons**:
  - Limited language support.
  - Higher cost for large app portfolios.

Appknox

- **Purpose**: **Cloud-based** security tool for **Android** and **iOS**.
- **Key Features:**
  - Automated testing with focus on vulnerabilities and improper authentication.
- **Pros**:
  - Easy-to-use interface.
  - Integration with CI/CD tools.
- **Cons**:
  - Limited to **cryptographic** vulnerabilities.
  - Higher pricing for advanced features.

Fortify on Demand

- **Purpose**: **Cloud-based** security testing by Micro Focus.
- **Key Features:**
  - Combines **static** and **dynamic** analysis, focusing on **code** and **network** vulnerabilities.
- **Pros**:
  - Seamless integration with dev environments.
  - Detailed reports.
- **Cons**:
  - High pricing.
  - Complex configuration for large apps.

AppSweep

- **Purpose**: **Cloud-based** tool for **Android** and **iOS**.
- **Key Features:**
  - Automated testing with focus on **data leakage** and **insecure communication**.
- **Pros**:
  - Easy-to-use with CI/CD integration.
- **Cons**:
  - Limited iOS support.
  - Higher pricing for advanced features.

HCL AppScan

- **Purpose**: **Enterprise-grade** tool for **Android** and **iOS**.
- **Key Features:**
  - Comprehensive vulnerability scanning with detailed reports.
- **Pros**:
  - Strong integration with CI/CD.
  - Advanced automation.
- **Cons**:
  - Complex setup.
  - High cost for small enterprises.

Veracode

- **Purpose**: **Enterprise-grade** tool for **Android** and **iOS**.
- **Key Features:**
  - Combines **static** and **dynamic** analysis with **network communication** security.
- **Pros**:
  - Detailed, actionable insights.
  - Strong dev environment integration.
- **Cons**:
  - Expensive for small businesses.
  - Requires setup for complex architectures.

Synopsys

- **Purpose**: Security testing tool for **Android**, **iOS**, and Windows.
- **Key Features:**
  - Combines **static**, **dynamic**, and **interactive** analysis.
- **Pros**:
  - Comprehensive testing capabilities.
  - Supports multiple platforms.
- **Cons**:
  - Expensive.
  - Requires complex setup.

Ostorlab

- **Purpose**: Security testing tool for **Android**, **iOS**
- **Key Features:**
  - Provides **static**, **dynamic** analysis.
- **Pros**:
  - Comprehensive Security Analysis.
  - User-Friendly Interface.
- **Cons**:
  - Limited Features in Free Version.
  - Performance Issues.

Top mobile app security assessment Tools

8 Mobile App Security Testing Tools

Tool	Overview	Key Features
Checkmarx	Cloud AppSec Platform.	SAST, SCA, API Security.
Appknox	Automated Mobile Security.	Automated & Manual Testing.
Data Theorem	Full-stack App Security.	Mobile, API, Web Security.
NowSecure	Automated Mobile Testing.	Continuous Testing, Pen Testing.
App-Ray	Mobile App Protection.	Security Testing, Fuzzing.
Veracode	App Security Solutions.	eLearning, SAST, DAST.
Ostorlab	Automated Security Testing.	Mobile, Web App Testing.
Q-MAST by Quokka	Cloud Mobile Security.	Analysis Methods, Automated Scanning.

Smali

What is Smali?

- **Smali** : low-level assembly-like language designed for the **Dalvik Virtual Machine (VM)**
- It serves as an **intermediate language** between **Java source code** and the **executable code** on Android devices.
- **Smali** is mainly used in **reverse engineering**, particularly for **analyzing** or **modifying** Android applications.

Smali

Common Uses of Smali

1. **Reverse Engineering**: Modify **Android APK** files after decompiling to change their behavior.

2. **Malware Analysis:** Used by researchers to understand the behavior of **malware** on **Android**.

3. **Debugging**: Applied when the original source code is unavailable, helping to **debug apps**.

4. **Removing Ads/License Restrictions***:* Modify apps to remove unwanted features or protections by altering **DEX** files.

Smali

Key Tools for Working with Smali

- **Baksmali**: Decompiles **DEX** files into **Smali** code.
- **Smali Tool**: Recompiles **Smali** code back into **DEX** format.
- **JEB & APKTool:** Common tools for decompiling/recompiling APK files and working with **Smali** code.

Smali

How to Work with Smali

- Use **APKTool** to extract **APK** resources, including **DEX** files, which can be **converted** to **Smali** code using **Baksmali**.

- After **modifying** the **Smali code**, use **Smali Tool** to **recompile** the **DEX file**, and **APKTool** to **repackage** it back into **APK** format.

Tools for Working with Smali

Working with **Smali** often involves various tools that facilitate

- decompiling
- editing
- reassembling
- analyzing **APK** files and .**dex bytecode** for Android applications.

Tools for Working with Smali

ApkTool

Purpose:

- Decompiles and reassembles **APK** files,
- **converting** .dex files into **Smali code** and allowing for **modification** of both **code** and **resources**.

Usage:

- **Decompile**: *apktool d app.apk* (creates **Smali** files and resources).
- **Recompile**: *apktool b app_folder* (rebuilds the **APK** after edits).

Tools for Working with Smali

JEB (Java Executable Bytecode)

Purpose:

- Professional-grade Android **decompiler**
- converting **.dex** files into **Smali** and Java code
- performing interactive code analysis.

Features:

- Interactive **GUI** with **decompiled Java, Smali**
- Advanced support for **obfuscation** and **native code** analysis.
- **Python scripting** support for automating tasks.

Tools for Working with Smali

JADX (Java Decompiler for Android)

Purpose:

- A **decompiler** that converts .**dex** files into **readable Java code**, with some support for viewing Smali.

Usage:

- **Open APK**: Load an **APK** file in **jadx-gui** to explore the code.
- **Export Smali**: View **Smali cod**e for **methods/classes** when needed.

Tools for Working with Smali

Baksmali and Smali

Purpose:

- Tools specifically for disassembling (**baksmali**) and assembling (**smali**) **.dex** files.

Usage:

- **Disassemble**: *baksmali disassemble app.dex* (produces .smali files).
- **Assemble**: *smali assemble smali_folder -o classes.dex* (compiles .smali files into a .dex file).

Tools for Working with Smali

Android Studio (with JD-GUI or JADX plugin)

Purpose:

- While Android Studio isn't a Smali editor, it can be configured with **plugins** to support **Java decompilation** and some **Smali viewing**.

Tools for Working with Smali

JD-GUI

Purpose:

- A standalone **Java decompiler** that can be used with **dex2jar** to inspect **Java code** for analysis.

Usage:

- **Open JAR file**: Load the **.jar** file created with **dex2jar** to view **decompiled Java code**.

Tools for Working with Smali

dex2jar

Purpose:

- Converts **.dex** files into **.jar** files, which can then be **decompiled** into **Java** using a **decompiler** like **JD-GUI** or **JADX**.

Usage:

- **Convert**: *d2j-dex2jar app.dex* (generates a .jar file from the .dex file).

Workflow Example with Smali Editing

1. **Decompile APK**: Use *apktool d app.apk* to get **.smali** files.
2. **Edit Smali Code**: Open **.smali** files in a text editor (like **VSCode**) and make changes.
3. **Recompile APK**: Use *apktool b app_folder* to rebuild the modified APK.
4. **Sign and Install APK**: Use **jarsigner** or **apksigner** to sign the **APK** and then install it on an Android device for testing.