

Android Application Pen-Testing

Configuration of Lab,
Android Studio,
Genymotion and
Santoku

ADB Commands

Mobile App Security
Pen-Testing Strategy

Android Application
Vulnerability
Exploitation

Insecure Login,
Hardcode Issue,
Insecure Data Storage
Issue, Input Validation

Access Control Issue,
Content Provider
Leakage, Client Side
Injection, Latest OWASP
top 10 vulnerability

ADB (Android Debug Bridge)

- Android Debug Bridge (adb) is a versatile **command-line tool** that lets you communicate with a device.
- The adb command facilitates a variety of device actions, such as **installing and debugging apps**, and it **provides access to a Unix shell** that you can use to run a variety of commands on a device.
- **A client**, which sends commands.
 - The client runs on your development machine.
 - You can invoke a client from a command-line terminal by issuing an adb command.
- **A daemon (adb),**
 - which runs commands on a device. The daemon runs as a background process on each device.

ADB Commands

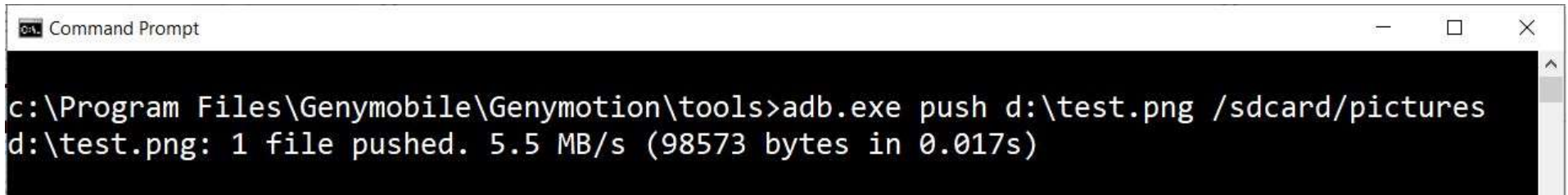
- `adb connect device_ip_address`
- `adb devices`
 - `offline` : The instance is not connected to adb or is not responding.
 - `device` : The instance is connected to the adb server.
 - `no device` : There is no device connected.
- `adb kill-server` - reset your adb host
- `adb start-server`

ADB Commands

- adb devices
 - List of devices attached
 - 192.168.67.101:5555 [serial no of the device]
 - emulator-5554 [serial no of the device]
- To Start the ADB Shell
 - adb -s 192.167.67.101:5555 shell
- if you wants to back to command prompt or santoku shell form the ADB the press ctrl + D or exit

ADB Commands

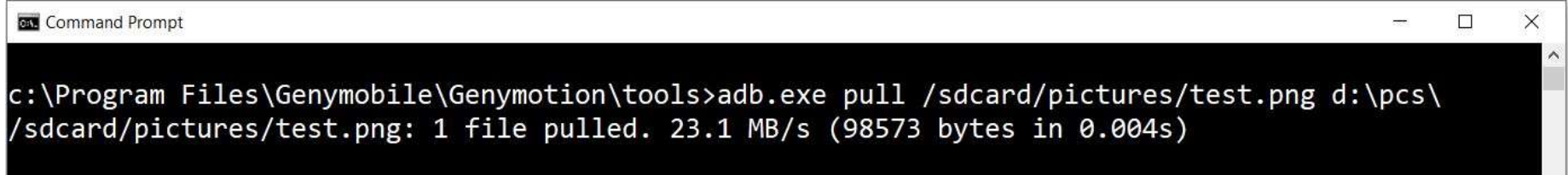
- `adb push <local> <remote>` - pushes the file <local> to <remote>
- Example
- `adb.exe push Pictures/index.png /sdcard/Pictures`

A screenshot of a Windows Command Prompt window. The title bar reads "Command Prompt". The command prompt shows the following text:
c:\Program Files\Genymobile\Genymotion\tools>adb.exe push d:\test.png /sdcard/pictures
d:\test.png: 1 file pushed. 5.5 MB/s (98573 bytes in 0.017s)

```
Command Prompt
c:\Program Files\Genymobile\Genymotion\tools>adb.exe push d:\test.png /sdcard/pictures
d:\test.png: 1 file pushed. 5.5 MB/s (98573 bytes in 0.017s)
```

ADB Commands

- `adb pull <remote> [<local>]` - pulls the file <remote> to <local>.
- If <local> isn't specified, it will pull to the current folder.
- Example
 - `adb.exe pull /sdcard/Pictures/index.png`



```
Command Prompt
c:\Program Files\Genymobile\Genymotion\tools>adb.exe pull /sdcard/pictures/test.png d:\pcs\
/sdcard/pictures/test.png: 1 file pulled. 23.1 MB/s (98573 bytes in 0.004s)
```

ADB Commands

- In Android, the logcat command provides a way to view the system debug output.
- Logs from various applications and portions of the system are collected in a series of circular buffers which then can be viewed and filtered by this command:
- adb logcat
 - It allows you to view the device log in real-time.
- Example
 - adb.exe shell logcat -b radio -v time
 - <https://developer.android.com/studio/command-line/logcat.html>

ADB Commands

- `adb install <file>` - installs the given .apk file to your device
- <https://payatu.com/wp-content/uploads/2016/01/diva-beta.tar.gz>
- Example

Command Prompt

```
c:\Program Files\Genymobile\Genymotion\tools>adb.exe install C:\Users\Parag\Downloads\diva-beta\diva-beta.apk
Performing Push Install
C:\Users\Parag\Downloads\diva-beta\diva-beta.apk: 1 file pushed. 32.8 MB/s (1502294 bytes in 0.044s)
    pkg: /data/local/tmp/diva-beta.apk
Success
```


ADB Commands

- adb shell - launches a shell on the device
- root@vbox86p:/ \$ ls
 - acct
 - cache
 - config
 - ddata
 - default.prop , dev, efs, etc, factory etc...

ADB Commands

- Basic Linux Commands

- ls
- cat
- cd
- cp
- chmod
- rm
- mkdir
- mount
- exit

Santoku

- Santoku Linux flavor and its free and open source
- Name was suggested by Thomas Cannon of via Forensics (who happens to be the project leader of Santoku Linux)
- Santoku Linux is aimed at
 - Mobile Forensics,
 - Mobile Malware Analysis, and
 - Mobile Security Testing
- GNU/Linux distro designed to help you in every aspect of your
 - Mobile forensics,
 - Mobile malware analysis,
 - Reverse engineering and
 - Security testing

Santoku

- Tools available in the Santoku are:
 - Development Tools:
 - Penetration Testing:
 - Reverse Engineering:
 - Wireless Analyzers:
 - Device Forensics:
 - Mobile Infrastructure:

Santoku

- If you are into mobile security and mobile forensics then this distribution is definitely right for you.
- Mobile Forensics:
 - Firmware flashing tools for multiple manufacturers
 - Imaging tools, media cards, and RAM
 - Free versions of some commercial forensics tools
 - Useful scripts and utilities specifically designed for mobile forensics

Santoku

- Mobile Malware Analysis:
 - Mobile device emulators
 - Utilities to simulate network services for dynamic analysis
 - Decompilation and disassembly tools
 - Access to malware databases
- Mobile Security Testing
 - Decompilation and disassembly tools
 - Scripts to detect common issues in mobile applications
 - Scripts to automate decrypting binaries, deploying apps, enumerating app details, and more

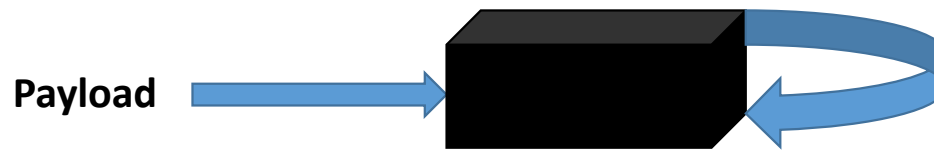
Santoku

- Configuration Steps of Santoku
 - <https://santoku-linux.com/>

Mobile Application Security Pen-Testing Strategy

- Security Pen-Testing Strategy

- Black Box –



- Irrespective of what is inside the application or without the knowledge of the source code, pen-tester provide the valid and invalid input and see the response.
 - Provide payload through input controls such as text box etc.,
 - Provide payload as a part of URL
 - Provide the payload as part of any string or variable by intercepting the request.
 - In-case developer don't have access of source, its good strategy.

Mobile Application Security Pen-Testing Strategy

- Security Pen-Testing Strategy
 - White Box –
 - Pen-Tester will perform walk through of the application to understand the source code of the application
 - Pen-Tester should have access to the source code.
 - Pen-Tester should have through knowledge front end and back end (programming language) of the application.
 - Pen-Tester should know various reverse engineering techniques.
 - This also good strategy in the case of Mobile Application as more than 50% code is resides in the mobile phone it self.
 - This strategy will fail in the case when binaries are locked

Mobile Application Security Pen-Testing Strategy

Common Architecture

Entry	Notes
AndroidManifest.xml	The binary xml file that provides information that a device needs in order to run the app.
classes.dex	The application code compiled in the dex format.
resources.arsc	Binary XML file containing precompiled application resources.
res/	Contains other folders with resources for your application: GUI layouts, icons, menus, and so forth.
assets/	Contains other media that you might want to use in your app, such as videos, sounds, and large images that are not used directly in GUI layouts.
META-INF/	It contains the MANIFEST.MF file, which stores meta data about the contents of the JAR. The signature of the APK is also stored in this folder.
lib/	This folder contains compiled the code – i.e. native code libraries or Contains precompiled third-party libraries (JAR archives) that you want to use in your app.

Android Application Vulnerability Exploitation

Practical Example in DIVA

- 1. Insecure Logging**
- 2. Hard Coding Issues**
- 3. Insecure Data Storage Issues**
- 4. Input Validation Issues**
 - **Client Side Injection (SQL Injection)**
- 5. Access Control Issues**
 - **Content Provider Leakage**
 - **Path Traversal Vulnerability**
 - As the name suggests, a path traversal vulnerability in an application allows an attacker to read other system files using the vulnerable application's providers.

OWASP TOP-10 Vulnerabilities for Mobiles

- Open Web Application Security Project (OWASP) is one of the standards when it comes to security and finding vulnerabilities.
- It also releases a top 10 list that includes the most common and important vulnerabilities in various platforms.
- The OWASP top 10 guide for mobile could be found at <https://owasp.org/www-project-mobile-top-10/>
- If we have a look at the OWASP mobile project, here are the 10 security issues it covers for mobile applications:

OWSAP TOP-10 Vulnerabilities for Mobiles

- If we have a look at the OWASP mobile project, here are the 10 security issues it covers for mobile applications:
 1. Weak Server Side Controls
 2. Insecure Data Storage
 3. Insufficient Transport Layer Protection
 4. Unintended Data Leakage
 5. Poor Authorization and Authentication
 6. Broken Cryptography
 7. Client Side Injection
 8. Security Decisions Via Untrusted Inputs
 9. Improper Session Handling
 10. Lack of Binary Protections

OWSAP TOP-10 Vulnerabilities for Mobiles

- **Weak Server Side Controls**

- In the first OWASP vulnerability, Weak Server Side Controls, as the name suggests, is not sending the data from the mobile application to the server side in a secure way, or exposing some sensitive APIs while sending data.
- For instance, consider an Android application login credentials to the server for authentication, without validating the inputs.
- An attacker could modify the credentials in such a way so as to get access to sensitive or unauthorized areas of the server.
- This vulnerability could be considered as a vulnerability in both mobile applications as well as web applications.

OWSAP TOP-10 Vulnerabilities for Mobiles

- **Insecure Data Storage:**
- This simply means storing the application-related information in a way on the device accessible by the user.
- Many Android applications store secret user-related information, or app information, in shared preferences, SQLite (in plain form) or in external storage.
- Developers should always keep in mind that even if the application is storing sensitive information in the data folders (/data/data/package-name), it will be accessible by a malicious application/attacker as soon as the phone is rooted.

OWSAP TOP-10 Vulnerabilities for Mobiles

- **Insufficient Transport Layer Protection:**
- Many Android developers rely on insecure mode of sending data over the network such as in the form of HTTP or not properly implementing SSL.
- This makes the app vulnerable to all the different types of attacks happening on the network, such as traffic interception, manipulation of parameters while sending data from the application to the server, and modifying responses in order to gain access to locked areas of the application.

OWSAP TOP-10 Vulnerabilities for Mobiles

- **Unintended Data Leakage:**
- This vulnerability occurs in applications when the application stores data at a location which in itself is vulnerable.
- These might include the clipboard, URL Caches, Browser Cookies, HTML5 data storage, analytics data, and so on.
- An example would be a user logging in to their banking application who has copied their password to the clipboard.
- Now, even a malicious application could access that data in the user's clipboard.

OWSAP TOP-10 Vulnerabilities for Mobiles

- **Poor Authorization and Authentication:**
- Android applications, or in general mobile applications, are mostly vulnerable if they try to authenticate or authorize a user based on client-side checks without proper security measures.
- It should be noted that most client-side protections could be bypassed by an attacker once the phone is rooted.
- Therefore, it is recommended that application developers use server-side authentication and authorization with proper checks, and once that is successfully done, use a random-generated token in order to authenticate the user on the mobile device.

OWSAP TOP-10 Vulnerabilities for Mobiles

- **Broken Cryptography:**

- This simply means use of nonsecure cryptographic functions in order to encrypt the data components.
- This might include some of the known vulnerable ones, such as MD5, SHA1, RC2, or even a custom developed one without proper security measures.

- **Client Side Injection:**

- This is possible in Android applications mostly due to the use of SQLite for data storage.
- ' or '1' == '1

OWSAP TOP-10 Vulnerabilities for Mobiles

- **Security Decisions Via Untrusted Inputs:**
- In mobile applications, developers should always sanitize and verify user-supplied inputs or other related inputs, and shouldn't use them as they are in the application.
- Untrusted inputs could often lead to other security risks in the application such as Client Side Injection.
- **Improper Session Handling:**
- While performing session handling for a mobile application, the developer needs to take care of a lot of factors, such as proper expiration of the authentication cookies, secure token creation, cookie generation and rotation, and failure to invalidate sessions at the backend.
- A proper secure sync has to be maintained between the web application and the Android application.

OWSAP TOP-10 Vulnerabilities for Mobiles

- **Lack of Binary Protections:**
- This means not being able to properly prevent the application from being reversed or decompiled.
- Tools such as Apktool and dex2jar could be used to reverse an Android application, which exposes the application to various kinds of security risks if proper developing practices have not been followed.
- To prevent analysis of applications by reversing from attackers, developers could use tools such as ProGuard and DashO

Gapps Project

- GApps (short for Google Apps) packages are essential in the Android aftermarket development community.
- They are specially crafted optional add-ons for custom ROMs that can be used to get Google apps such as Google Play Services and the Play Store on your device.
- **Why GApps?**
- Google requires every Android device maker to follow the Compatibility Definition Document (CDD) to pass the Compatibility Test Suite (CTS) so they can be allowed to pre-load their devices with Google apps and services.
- Custom developers however can't easily bundle these Google apps and services with their builds.
- As these apps are not using the Apache or GPLv2 license, bundling them within the ROM presents legal challenges.

Gapps Project

- This is exactly where the GApps packages come in.
- The GApps maintainers rely on build scripts that allow for the automated creation of new updated packages at regular intervals.
- ROM developers, on the other hand, usually build the custom ROMs in such a way that an end-user can flash a third-party GApps distribution on top of their builds and seamlessly gain the ability to use the Google Play Store or any other applications that require Google Play Services.