**CHAPTER 1**

**INTRODUCTION**

A few years ago, we introduced the first version of the Android operating system with the T-Mobile G1, and launched Android Market (now Google Play) the very same day. Android has grown up a lot. Since then-there are now more than 2 billion active Android devices round the world.

The operating system itself has gone through some major transformations, too. The G1 ran on Android 1.0-a version so early, we didn’t even name it after a dessert. The debut included features that you know and love today like pull-down notifications, sharing content across apps and multitasking between apps. But it didn’t have more advanced features like voice search, turn-by-turn navigation or NFC. Ten years later, we’ve come a long way! Our latest release of the operating system, Android 9 Pie, has all of those features and harnesses the power of artificial intelligence to make your phone smarter, simpler and more adaptive.

Every new Android version released comes with new features, improvements and security enhancements. The data discussed in this report shows that Android becomes more secure with every major version. The major security enhancements are documented on source.android.com in the Enhancements section.

In 2018, mobile apps were downloaded onto user devices over 205 billion times. Data by Marketing Land indicates that 57 percent of total digital media time is spent on smartphones and tablets. More often than not, our daily lives depend on apps for instant messaging, online banking, business functions, and mobile account management. According to Juniper Research, the number of people using mobile banking apps is approaching two billion-around 40 percent of the world's adult population.

Developers pay painstaking attention to software design in order to give us a smooth and convenient experience. People gladly install mobile apps and provide personal information, but rarely stop to think about the security implications.

Researchers have presented estimations that up to 90% of all applications have at least one vulnerability. While research about mobile applications in particular is scarce, high-profile data thefts like the Snapchat user database indicate serious security concerns even for apps that are used by millions.

There have been many types of research done by various firms on the topic of “Malicious codes/apps on Android OS”, like Trend Labs (2013), ZDNet (2012), Kaspersky (2014) and even newer ones like from GData (2017) and all of the statistics saw an increase in malicious code/apps as well as increase in number of Android users. To favour free and more enlarged environment, Android provides open sources for app development. This results in contribution from a large community for experts and many vulnerabilities have been successfully identified and patched.

As the awareness of privacy and its importance is spreading, the demand for security on smartphones has increased significantly. Even though every smartphone OS developer is trying to make their product more safer and also have established multiple security measures but still here are weak points that always pop up with time. These weak points allow malicious codes to either pass-through or misuse the security system against itself, causing information leakage (which may cause financial loss) or complete device takeover and loss of important data.

**1.1 What is android security?**

[Mobile security](https://www.webopedia.com/TERM/M/mobile_security.html) applications for Google's [Android](https://www.webopedia.com/TERM/A/Android_platform.html) platform help protect Android [smartphone](https://www.webopedia.com/TERM/S/smartphone.html) and [tablet](https://www.webopedia.com/TERM/T/tablet_PC.html) [mobile devices](https://www.webopedia.com/TERM/M/mobile_phone.html) from [malware](https://www.webopedia.com/TERM/M/malware.html) threats as well as unauthorized access following accidental loss or theft of the device.

Additional security features frequently offered by Android [mobile security apps](https://www.webopedia.com/TERM/M/mobile_security_apps.html) include securing data on the device, [VPN](https://www.webopedia.com/TERM/V/VPN.html) connectivity for protecting data in transit, scanning websites for potential [phishing](https://www.webopedia.com/TERM/P/phishing.html) schemes or other fraudulent activity, helping users locate their device if lost or stolen, and more.

Android mobile security applications are available from Google as well as well-known third-party security vendors such as Lookout, Avast, Kaspersky, Symantec and Qihu.

### 1.1.1 Android Security: System-Level Security Features

The Linux kernel provides Android with a set of security measures. It grants the operating system a user-based permissions model, process isolation, a secure mechanism for IPC, and the ability to remove any unnecessary or potentially insecure parts of the kernel. It further works to prevent multiple system users from accessing each other’s resources and exhausting them.

### 1.1.2 Android Application Security Features

This user-based protection allows Android to create an “Application Sandbox.” Each Android app is assigned a unique user ID, and each runs as a separate process. Therefore, each application is enforced at the process level through the Linux kernel, which does not allow applications to interact with one another, and gives them only limited access to the Android operating system.

This gives the user permission-based access control, and he/she is presented with a list of the activities the Android application will perform and what it will require to do them, before the app is even downloaded. The same goes for file system permissions – each application (or user) has its own files, and unless a developer explicitly exposes files to another Android application, files created by one application cannot be read or altered by another.

### 1.1.3 Android Application Security Scans

When building and testing the security of Android apps, developers should follow Android security best practices and keep the following in mind when performing security tests:

* Inbound SMS listeners (command and control)
* Unsafe file creation
* Improper database storage
* Unsafe use of shared preferences
* Storage of sensitive data on mass storage device
* Content provider SQL injection
* APN or proxy modification

### 1.1.4 Android Security: Geared Towards User-Friendly Security

All of Android’s more technical security features are designed to be simply presented to the user, meaning that they can be easily controlled through the interface. Straightforward methods of improving your Android device’s security can include: using a password or pin, setting your phone to lock after a period of inactivity, only enabling wireless connections that you use, and only installing Android apps you trust and have personally vetted.

Google also only allows tested and proven secure Android applications into its marketplace, meaning that the user has less of a chance of installing a malicious app. Furthermore, the Android security system prompts the user to allow the installation of an application, meaning that it is impossible to remotely install and run an application.

### 1.2 What is an Application Vulnerability?

Application vulnerability is a system flaw or weakness in an application that could be exploited to compromise the security of the application. Once an attacker has found a flaw, or application vulnerability, and determined how to access it, the attacker has the potential to exploit the application vulnerability to facilitate a cyber-crime. These crimes target the confidentiality, integrity, or availability (known as the “CIA triad”) of resources possessed by an application, its creators, and its users. Attackers typically rely on specific tools or methods to perform application vulnerability discovery and compromise. According to Gartner Security, the application layer currently contains 90% of all vulnerabilities.

### 1.2.1 Common Application Vulnerability Exploits

While there are many different tools and techniques for exploiting application vulnerabilities, there is a handful that is much more common than others. These include:

* + Cross Site Scripting
  + SQL Injection
  + LDAP Injection
  + Cross Site Request Forgery
  + Insecure Cryptographic Storage

### 1.2.2 Application Vulnerability Management

It is common for software and application developers to use vulnerability scanning software to detect and remedy application vulnerabilities in code, but this method is not entirely secure and can be costly and difficult to use. Furthermore, scanning software quickly becomes out-dated and inaccurate, which only poses more issues for developers to address in trying to make their applications secure. The Figure 1.1 shows the vulnerabilities from 2011-2018

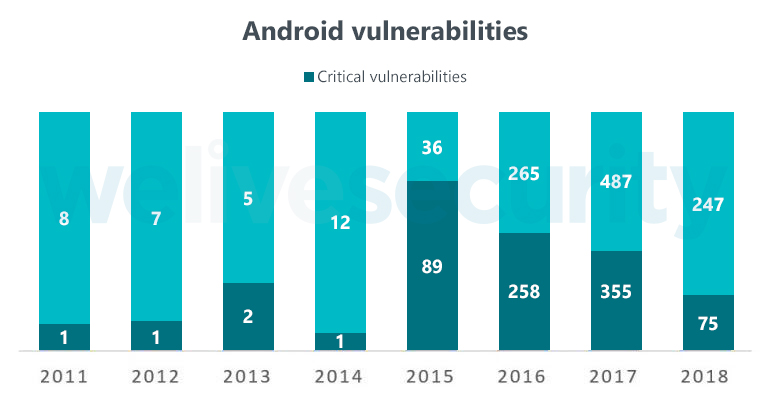


Figure 1.1: Android Vulnerabilities from 2011-2018

**1.2.3 Reducing Application Vulnerability Risk**

Veracode’s cloud-based service and systematic approach deliver a simpler and more scalable solution for reducing global application-layer risk across web, mobile and third-party applications. Recognized as a Gartner Magic Quadrant Leader since 2010, Veracode provides on-demand application vulnerability testing to detect and offer solutions for vulnerabilities and other security issues. Since Veracode offers a service instead of a scanning tool, companies are able to save costs by having their applications tested at the highest level of accuracy without the need for purchasing and updating software or hiring specialists to operate and maintain the software.

### 1.3 What Is Malicious Code?

Malicious code is the term used to describe any code in any part of a software system or script that is intended to cause undesired effects, security breaches or damage to a system. Malicious code is an application security threat that cannot be efficiently controlled by conventional antivirus software alone. Malicious code describes a broad category of system security terms that includes attack scripts, viruses, worms, Trojan horses, backdoors and malicious active content.

Malicious code can take the form of:

* + Java Applets
  + ActiveX Controls
  + Scripting languages
  + Browser plug-ins
  + Pushed content

Once inside your environment, malicious code can enter network drives and propagate. Malicious code can also cause network and mail server overload by sending email messages; stealing data and passwords; deleting document files, email files or passwords; and even reformatting hard drives.

### 1.3.1 Malicious Code Threatens Enterprise Security

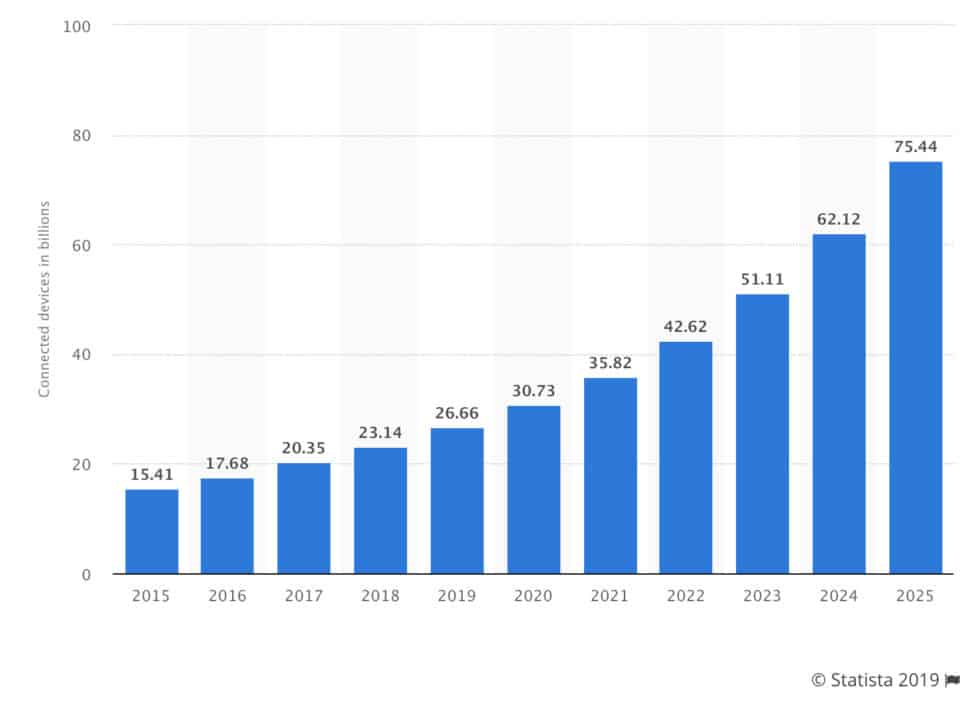
Malicious code can give a user remote access to a computer. This is known as an application backdoor. Backdoors may be created with malicious intent, to gain access to confidential company or customer information. But they can also be created by a programmer who wants quick access to an application for troubleshooting purposes. They can even be created inadvertently through programming errors. Regardless of their origin, all backdoors and malicious code can become a security threat if they are found and exploited by hackers or unauthorized users. As applications today tend to be built more and more often with reusable components from a variety of sources with varying levels of security, malicious code can pose a significant operational risk to the enterprise. That's why so many enterprises today are turning to Veracode to secure their applications. The Figure 1.2 shows the Android device attack estimations.

**1.3.2 How to Avoid Malicious Code**

One way to avoid malicious code in your applications is to add static analysis (also called [“white-box” testing](https://www.veracode.com/products/binary-static-analysis-sast)) to your software development lifecycle to review your code for the presence of malicious code. Veracode’s [static code analysis](https://www.veracode.com/products/static-analysis-sast/static-code-analysis) looks at applications in non-runtime environment. This method of security testing has distinct advantages in that it can evaluate both web and non-web applications and, through advanced modeling, can detect malicious code in the software’s inputs and outputs that cannot be seen through other testing methodologies.

### 1.3.3 Test for Malicious Code with Veracode

Veracode has the ability to detect applications for malicious code threats that include time bombs, hardcoded cryptographic constants and credentials, deliberate information and data leakage, rootkits and anti-debugging techniques. These targeted malicious code threats are hidden in software and mask their presence to evade detection by traditional security technologies. Veracode's detection capabilities provide comprehensive support to combat against backdoors and malicious code.

****Figure 1.2: Android device attacks and future estimations

## 1.4 Ransomware statistics 2019

**Ransomware infection rates are dropping but almost half of victims pay the ransom.**

Ransomware has been the core concern for cyber security professionals for years but in 2018 it finally started to decline in volume. However, it doesn’t serve us to get excited about progress just yet, as more and more companies are paying the ransom when they do get hit.

* + In 2018, enterprise **Ransomware increased by 12%**, accounting for 81% of all successful Ransomware infections ([2019 Internet Security Threat Report by Symantec](https://www.symantec.com/security-center/threat-report))
  + Overall, **Ransomware infection rates**“**declined approximately 60%** between March 2017 and December 2018, with intermittent increases across that period”

There are probably many causes for this overall decline, although Microsoft security researchers suspect that a primary factor is that both end users and organizations are becoming more aware of and dealing more intelligently with ransomware threats, including exerting greater caution and backing up important files so they can be restored if encrypted by ransomware Also, as described earlier, cybercriminals are opportunistic.

* 1. **Strategy**

The Android security team’s mission is to protect every one of the more than two billion Android users. They do this through massive investment and continuous improvement in our security and privacy technology and operations. The Figure 1.3 refers the strategy.

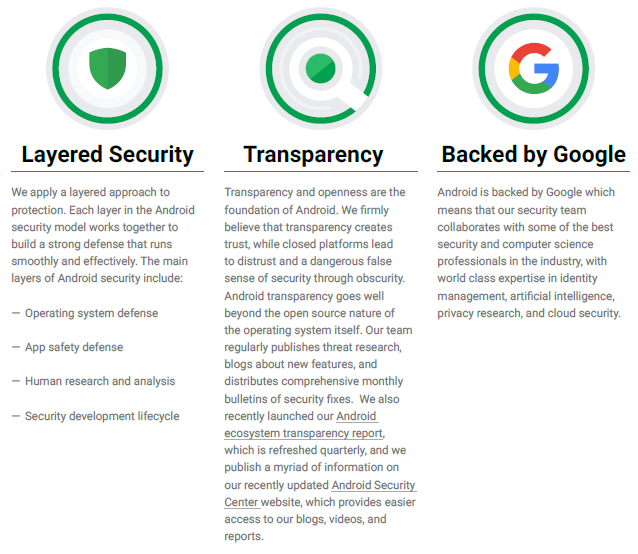
****

Figure 1.3: Strategy of Android security team

**CHAPTER 2**

**ANDROID ARCHITECTURE**

**Android architecture** is a software stack of components to support a mobile device needs. Android software stack contains a Linux Kernel, collection of c/c++ libraries which is exposed through an application framework services, runtime and application. The Figure 2.1 represents Android architecture.

**2.1 Architecture**

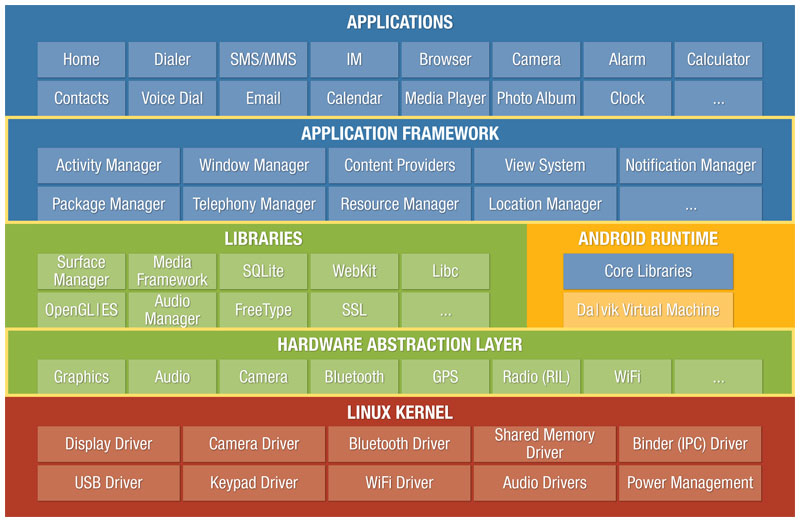


Figure 2.1: Android architecture

* Starting from the bottom we have **Linux Kernel**, Android is built up on the Linux Kernel. Linux is already being used extensively from so many years and its kernel had received so many security patches. Linux Kernel provides basic system functionality like process management, memory management, device management like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

**2.1.1 Linux Kernel Offers on Android**

As the base for a mobile computing environment, the Linux kernel provides Android with several key security features, including:-

* A user-based permissions model
* Process isolation
* Extensible mechanism for secure IPC
* The ability to remove unnecessary and potentially insecure parts of the kernel

As a multiuser operating system, a fundamental security objective of the Linux kernel is to isolate user resources from one another. The Linux security philosophy is to protect user resources from one another.

* **Hardware Abstraction Layer** just gives Applications direct access to the Hardware resources.
* Moving on to the third part comes the **Libraries, Android Runtime and Dalvik**. The libraries shown in the image are very necessary without which application will not run like Webkit library is used for browsing the web, SQLite library is used for maintaining SQL database and so on.
* **Dalvik Virtual Machine** which is specifically designed by Android Open Source Project to execute application written for Android. Each app running in the Android Device has its own Dalvik Virtual Machine
* **Android Runtime (ART)** is an alternative to Dalvik Virtual Machine which has been released with Android 4.4 as an experimental release, in Android Lollipop (5.0) it will completely replace Dalvik Virtual Machine. Major change in ART is because of Ahead-of-time (AOT) Compilation and Garbage Collection. In Ahead-of-time(AOT) Compilation ,android apps will be compiled when user installs them on their device whereas in the Dalvik used Just-in-time(JIT) compilation in which bytecode are compiled when user runs the app. Moving to the last one ,these are common.
* The **Application Framework** layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.
* Android apps are written in the Java programming language. The Android SDK tools compile your code-along with any data and resource files-into an APK: an Android package, which is an archive file with an .apk suffix. One APK file contains all the contents of an Android app and is the file that Android-powered devices use to install the app.The Figure 2.2 represents Application Framework process.

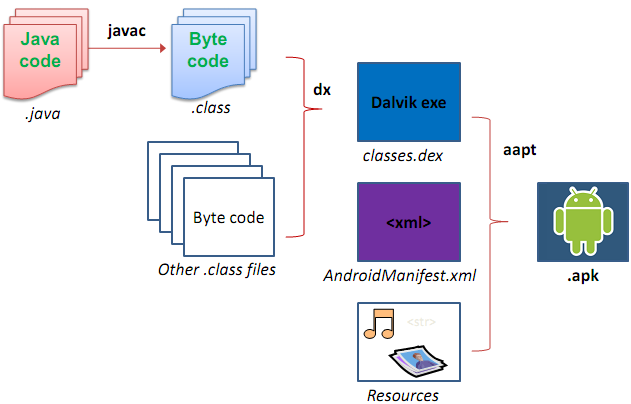
****

Figure 2.2: Application Framework

An **APK** file is an Archive that usually contains the following directories:

* AndroidManifest.xml: The AndroidManifest.xml file is the control file that tells the system what to do with all the top-level components (specifically activities, services, broadcast receivers, and content providers described below) in an application. This also specifies which permissions are required. This file may be in Android binary XML that can be converted into human-readable plaintext XML with tools such as [android-apktool](https://code.google.com/p/android-apktool/), or [Androguard](https://code.google.com/p/androguard/wiki/Usage#Androaxml) which we will cover in the upcoming post.
* META-INF directory:
  + MANIFEST.MF: the Manifest File
  + CERT.RSA: The certificate of the application
  + CERT.SF: The list of resources and SHA-1 digest of the corresponding lines in the MANIFEST.MF file.

.

* lib: the directory containing the compiled code that is specific to a software layer of a processor, the directory is split into more directories within it:
  + armeabi: compiled code for all ARM based processors only
  + armeabi-v7a: compiled code for all ARMv7 and above based processors only
  + x86: compiled code for X86
  + mips: compiled code for MIPS processors only
* res: the directory containing resources not compiled into resources.arsc
* assets: a directory containing applications assets, which can be retrieved by Asset Manager.
* classes.dex: The classes compiled in the dex file format understandable by the Dalvik virtual machine
* resources.arsc: a file containing precompiled resources, such as binary XML for example.

**2.2 App Components**

**App components** are the essential building blocks of an Android app. Each component is a different point through which the system can enter your app. Not all components are actual entry points for the user and some depend on each other, but each one exists as its own entity and plays a specific role-each one is a unique building block that helps define your app’s overall behavior. You can skip the content given below if you are already familiar with them. There are following four components of app:-

* **Content Provider** component supplies data from one application to others on request.
* **Activity** to be simple an activity represents a single screen with a user interface.
* A **Service** is a component that runs in the background to perform long-running operations or to perform work for remote processes.
* A **Broadcast** receiver is a component that responds to system-wide broadcast announcements.

**CHAPTER 3**

**ANDROID PERMISSION MODEL**

By default there are some Protected API’s in the Android Operating System which can only be accessed by Operating System. The Protected APIs include:-

* Camera functions
* Location data (GPS)
* Bluetooth functions
* SMS/MMS functions
* Network/data connections

If a particular application needs access to any of the API then it need to mention that permission in AndroidManifest.xml file. You might have observed that when installing a particular application from Google Play Store it ask for several permissions needed, if you don’t allow then app won’t install. If that user agrees to grant those permissions then Android operating system gives access to that Protected API.

# 3.1 Request Permissions in Android Application

Starting from **Android 6.0 (API 23)**, users are not asked for permissions at the time of installation rather developers need to request for the permissions at the run time. Only the permissions that are **defined in the manifest file** can be requested at run time.

**3.1.1 Types of Permissions:**

1. **Install-Time Permissions:** If the **Android 5.1.1 (API 22) or lower**, the permission is requested at the installation time at the **Google Play Store**. The Figure 3.1 represents Install-Time Permission.

If the user **accepts** the permissions, the app is installed. Else the app **installation is cancelled**.

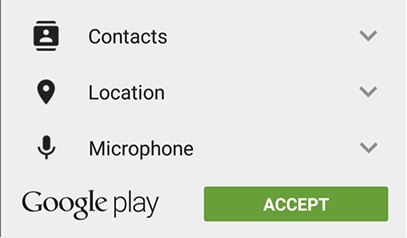


Figure 3.1: Install-Time Permissions Request

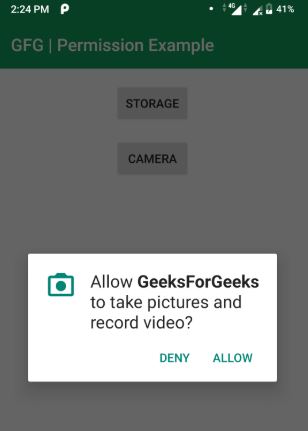
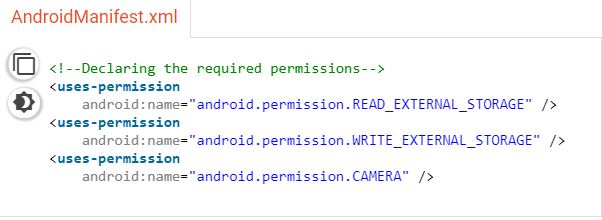
1. **Run-Time Permissions:** If the **Android 6 (API 23) or higher**, the permission is requested at the run time during the running of the app. The Figure 3.2 represents Run-Time Permission.

Figure 3.2: Run-Time Permissions Request

If the user **accepts** the permissions, then that feature of the app can be used. Else to use the feature, the app **requests the permission again**.

**Steps for Requesting permissions at run time:**

* **Declare the permission in**[**Android Manifest file**](https://www.geeksforgeeks.org/application-manifest-file-android/):In Android permissions are declared in **AndroidManifest.xml** file using the **uses-permission** tag. The Figure 3.3 represents declaration of AndroidManifest.xml file

**Figure 3.3: AndroidManifest.xml** file

* **Modify activity\_main.xml file to Add two buttons to request permission on button click:**  Permission will be checked and requested on button click. Open **activity\_main.xml** file and add two buttons in it. The Figure 3.4 represents activity\_main.xml file

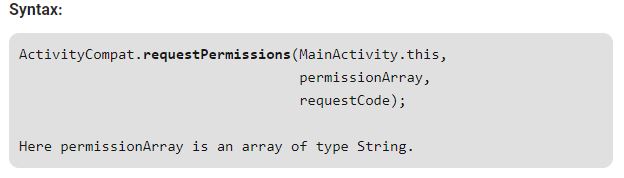
**Figure 3.4: activity\_main.xml file**

* **Check whether permission is already granted or not. If permission isn’t already granted, request user for the permission:**  In order to use any service or feature, the permissions are required. Hence we have to ensure that the permissions are given for that. If not, then the permissions are requested.

**Check for permissions:** Beginning with Android 6.0 (API level 23), the user has the right to revoke permissions from any app at any time, even if the app targets a lower API level. So to use the service, the app needs to check for permissions every time. The Figure 3.5 represents syntax to check permission.

**Figure 3.5: Syntax to checkSelfPermission**

**Request Permissions:** When **PERMISSION\_DENIED** is returned from the **checkSelfPermission()** method in the above syntax, we need to prompt the user for that permission. Android provides several methods that can be used to request permission, such as **requestPermissions()**. The Figure 3.6 represents the syntax to request permission.

**Figure 3.6: Syntax to requestPermissions**

* Override **onRequestPermissionsResult()** method:

**onRequestPermissionsResult()** is called when user grant or decline the permission. **RequestCode** is one of the parameters of this function which is used to check user action for corresponding request.

**3.2 Application Signing**

* Android requires that all apps be digitally signed with a certificate before they can be installed. Android uses this certificate to identify the author of an app
* To run application on the device, it should be signed. When application is installed on to an device then package manager verifies that whether the application has been properly signed with the certificate in the apk file or not.
* Application can be self-signed or can be signed through CA.
* Application signing ensures that one application can’t access any other application except through well-defined IPC and also that it is passed unmodified to the device.

**3.3** **Application Verification**

* Android 4.2 and later support application verification. Users can choose to enable “Verify Apps” and have applications evaluated by an application verifier prior to installation.
* App verification can alert the user if they try to install an app that might be harmful; if an application is especially bad, it can block installation.

**3.4** **Android Sandbox**

Once installed on a device, each Android app lives in its own security sandbox: - The Android operating system is a multi-user Linux system in which each app is a different user.

* By default, the system assigns each app a unique Linux user ID (the ID is used only by the system and is unknown to the app). The system sets permissions for all the files in an app so that only the user ID assigned to that app can access them.
* Each process has its own virtual machine (VM), so an app’s code runs in isolation from other apps.
* By default, every app runs in its own Linux process. Android starts the process when any of the app’s components need to be executed, then shuts down the process when it’s no longer needed or when the system must recover memory for other apps.

In this way, the Android system implements the principle of least privilege, which is each app by default, has access only to the components that it requires to do its work and no more. This creates a very secure environment in which an app cannot access parts of the system for which it is not having permission. As every Android app runs in its own sandbox environment and cannot affect other apps by default but two apps can have same Linux User ID and can also share the same Dalvik VM if they are signed with the same Certificates.

**CHAPTER 4**

**HOW TO PROTECT YOUR SMARTPHONE FROM HACKERS AND INTRUDERS**

Having your smartphone hacked kind of feels like someone robbed your house or maybe even worse. It’s a massive invasion of privacy, a violation of your personal space, and it may take a while to figure out what is missing. Your smartphone doesn’t just hold your valuables; it signals to intruders which of your valuables are the most important to you. If something is on the phone you always have with you, then by definition, it rates.

Smartphones are small devices that can easily get lost or stolen, and they are constantly online sending and receiving signals, so they are always a target for criminals. To keep your phone and its contents away from prying eyes and sticky fingers, you need to develop a strategy for protecting your valuable information. Here are some tips on how to protect your smartphone from hackers and intruders.

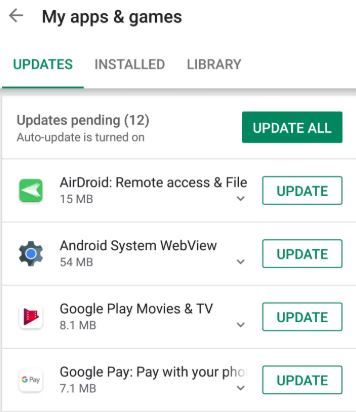
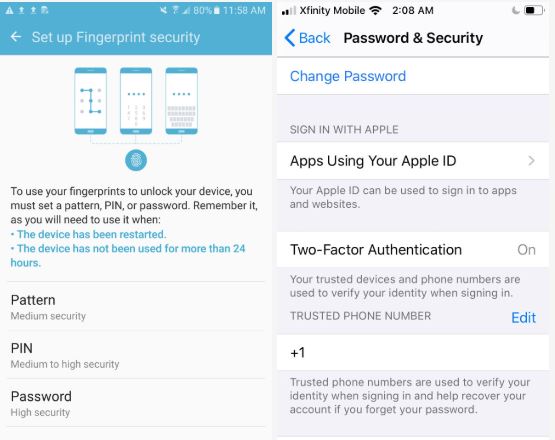
**4.1 Update your OS apps**

Figure 4.1: OS apps updating

Software companies are constantly updating software, and not only for cosmetic reasons. Many software updates and bug fixes contain security improvements that help guard your smartphone against data breaches and intrusions, and close off vulnerabilities, making it harder for hackers to break through. When an update is announced for your smartphone’s operating system or for any of the apps you use, install it without delay. The Figure 4.1 represents OS apps updation.

**4.2 Avoid public Wi-Fi**

By now, everyone should know about the dangers of using open Wi-Fi for anything, because free public Wi-Fi in shopping centers, cafes, airports, or any other public venue, is open season for all kinds of online mischief. Try to use only your private cell connection whenever possible and switch off Wi-Fi on your mobile phone altogether whenever you are in a public place. If that is not possible, consider using a VPN app, a utility that tunnels network communications through an encrypted connection. But choose carefully - not all VPNs are equal in quality. Also consider disabling Bluetooth while you’re out and about unless you are wearing a smart watch that requires it.

**4.3 Lock your Smartphone**

**Figure 4.2: Lock interface**

Always engage a four or six-digit passcode to enter your device. Passcodes may not be super convenient, but peace of mind dictates that if your smartphone falls out of your pocket while you’re trying out new sofas at Ikea, the first person who picks it up should not be able to get your life story from your email, contacts, photos, and banking information. Consider setting an even longer passcode with both numbers and letters. Not a big passcode fan? No worries. Fingerprint scanning and Face ID are easy, fast alternatives to punching in numbers. While you’re at it, make sure apps with personal information are also locked behind passwords. The Figure 4.2 represents lock interface.

**4.4 Keep your mobile phone number private**

Just like you wouldn’t give out your old landline phone number to anyone who asked for it, don’t automatically offer your mobile number to any app that prompts you. The more places that have your number, the more vulnerable you are to SMS intrusions and scams, and even invasion of your protected 2FA accounts. Consider adding a second line to your mobile phone. Google Voice is an excellent way to shield your phone number from online miscreants, as are apps like Sideline, Line2, and Hushed, which facilitate adding a second line to your mobile phone.

**4.5 Travel Light**

Don’t store personal information, documents, or files on your phone, and limit the number of geotagged photos in your Camera Roll. Make a habit of keeping your phone relatively pristine by offloading images and documents to your computer, and eliminating confidential emails from financial, employer, and health-related accounts.

**4.6 Use two-factor authentication**

Here’s another obnoxious security measure that most people can’t stand. Two-factor authentication (2FA) is hated because it requires an extra step, and it’s really a pain if you forget to have your phone or watch nearby. But like passwords, it serves a purpose by providing an extra layer of protection in case someone gets ahold of your password. The Figure 4.3 represents Two factor authentication

Figure 4.3: Two factor authentication

**4.7 Use strong passwords**

Everyone hates passwords. But when it comes to assigning them, don’t take half measures. Use only strong passwords that are not easily cracked by hackers. They should contain 16-20 characters with a mix of letters and numbers, upper and lowercase letters, and symbols. Brute force password crackers will still dismantle many strong passwords, but making it easy for hackers by using your birthday, your pet’s name, or the same password for everything is a truly terrible idea.

There are plenty of secure password generators online, so you don’t have to think them up for yourself. Change your passwords every six months to a year, or as soon as you hear about a data breach of any program you use.

**CHAPTER 5**

**A CASE STUDY IN OPEN SOURCE SOFTWARE SECURITY AND PRIVACY: ANDROID ADWARE**

Security and privacy are complicated things. This is why open development is a key factor and a precondition for creating secure solutions. Security and privacy is getting more important every day. Also due to the development of machine learning applications many data driven solutions are poisoned with privacy related data.

When development happens in the open, you can directly verify if a vendor is actively pursuing security and privacy and watch how it treats issues. The ability to study the process followed, the source code developed makes that anyone can perform an independent audit. Not only on code, but also on process used!

So beside code, open development means that an open processes is followed. A process where we can see and check whether mandatory baselines and principles are used.

To increase and improve security and protect our privacy open source solutions are more and more seen as a very good solution. Within more and more companies worldwide we notice a trends towards adopting open source solutions for security and privacy protection. Governments worldwide cannot depend and trust on closed source software for their security infrastructure anymore.

**5.1 Review of Mobile Security Literature**

So far mobile malware has largely targeted the Android operating system as opposed to other systems, for example, iOS and Blackberry OS. Some of the recent threats have been elevated in the media. Skeptics may attribute this partly to computer security software companies wanting to create a new market on mobile systems for their products

Researchers continue to discover new vulnerabilities on Android devices. For example, some of the apps pre-installed by hardware manufacturers may be exploited because their permissions can be leaked to other apps. Another serious example was a botnet involving numerous apps in the Android Market. As a result of those apps, a great number of users have downloaded a malware, which was at the time called Android

According to Google’s Vice President of Engineering for

According to Google’s Vice President of Engineering for Android, a new service called Bouncer will scan the Android Market (Google Play) for known malware and also simulate newly added apps to catch the misbehaving ones.

According to Google, Android included certain core security features from the start, including sandboxing (putting virtual walls between applications and other software on the device) and using a permission system, which shows the user during installation what types of access each new app demands.

In addition, behavioural biometric security will play a larger role with Android’s newest version 4 as it comes with a facial gesture recognition app for security. Another example of behavioural biometric security also available for mobile devices is a product called BehavioWeb, which understands and constantly verifies each unique user’s keystroke and user Interface patterns.

Although some of this discussion is technical and empirical, it is also inevitably influenced by the authors’ own background and even stereotypes. Similar to Linux, the Android brand has a socially benevolent image as a virtue of having open-source code (though the hardware manufacture and much of the software development is for profit). This lends sympathy in society in general and among computer enthusiasts toward the cute little robot (the Android logo). As it is costless and easy to access, many organizations and consumers have become prone to blindly trusting open-source software. On the other hand, one study by a computer security company (Vera Code) found that open source application teams are able to fix flaws much faster than both internally developed and commercial application teams

**5.2 Case Study by Emre Erturk**

There are two ways to analyse malware in general whether desktop or mobile: dynamic analysis and static analysis. Dynamic analysis involves automated tools to execute the malware in a controlled system environment and check for malicious patterns. Therefore a large sample of malware can studied quickly. On the other hand, dynamic analysis is not yet common for mobile devices because of the difficulty of replicating the numerous mobile hardware, mobile operating systems, and their many different versions. Mobile malware also have different ways of spreading as opposed to desktop malware, which further makes dynamic analysis more difficult.

The payload of Plankton adware consists of executable Java byte code that is initiated in the background. After collecting information about the infected device, Plankton contacts a specific web site and is capable of downloading further payload. It also allows commands to be given remotely from the contacted site. Earlier reports classified Plankton as a Trojan for potentially stealing data files but more recent security vendor reports suggest that it may rather be part of a very aggressive ad network.

The family of android malware that includes Spy-E and SNDAPPS displays unsolicited advertisements, generates notifications, gathers user and phone information, and communicates with an ad web site. If the variant NickiSpy.A is considered to be a part of this family, then the payload also includes draining money by sending SMS from the infected device to a premium number.

One of the newcomers among Android malware is TigerBot. This malware allows remote access by the attacker and can be controlled via SMS messages. It will listen for specific messages, which can steal contacts lists and screenshots, change network settings, and control running processes. During the static analysis, the apps containing this malware have been shown to obtain (if installed) an unusual variety of intrusive permissions. TigerBot is beyond adware, actually a Trojan going so far as to disguise itself with a Google icon.

**CHAPTER 6**

**Advantages of Android OS over Other OS:-**

* **It's largely supported by Google:**

Think without them being in the picture Android might have only gained mild success in terms of its push on the market. Especially the first iteration, it essentially looked like a BlackBerry copycat.

* **It's open and free:**

The beauty of Android is that you can customize to your heart's content, provided that the manufacturers are willing to hand over the source code.

Plus due to the very fact of it being open, it makes it free and thus the only cost is primarily the device.

* **It's pretty much runs on everything mobile, and otherwise:**

When looking at Android we see that right now it runs on mobile phones, tablets, wearable and even the likes of set-top boxes. This gives it the edge in the market in which you can be sure to write in one language that should cover most devices available. If anything IoT (Internet of Things) will be running Android to boot.

* **It's the current market leader:**

With around 1 BILLION devices activated monthly, it gives developers and users the advantage of most probably having their devices supported.

* **Expandable Storage Option:**

Whenever you want to increase the storage capacity of any Apple device such as iPhone or iPad, you need to go for a costly upgrade and it has a lot of hassle. But in the case of Android devices, you simply insert a micro SD memory card into your device card slot and expand the memory as per your requirement.

* **Customized or Modified Rom:**

If you are bored of using your stock Android and want to have some new and exciting features you can easily install any custom ROM and enhance your user experience.

* **Open Ecosystem:**

Android operating system gives you a lot of choices when it comes to installing your favourite apps.

**CONCLUSION**

Android devices, being an important part of our daily routine, contain a large amount of private information about its user in it. Currently Android relies on Sandbox to protect the data and the code of one application from another. In addition, the concept of Content Provider and URIs offers a safer way of sharing the database among client applications. However, the attacks can still attempt any illegitimate way and can use reverse engineering to steal all private information. The open source nature of Android OS is both making their devices most supported and most vulnerable at the same time. ProGuard is one effective way of preventing reverse engineering of application but still hold no guarantee to prevent information extraction. Putting all together, we conclude that Android devices are not safe and the attackers can easily access private information and require no particularly complicated way to get that.

.

**BIBLIOGRAPHY**

[1] Gartner, ""Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016"," *July 2017.*

*Available:* [*http://www.gartner.com/newsroom/id/3609817*](http://www.gartner.com/newsroom/id/3609817).

[2] Yury Namestnikov, Denis Maslennikov, "Kaspersky Security Bulletin 2012,"

*July 2017. Available:* [*https://securelist.com/kaspersky-security-bulletin-2012-theoverall-statistics-for-2012/36703/.*](https://securelist.com/kaspersky-security-bulletin-2012-theoverall-statistics-for-2012/36703/)

[3] G. Abendan, "Mobile Malware," July 2017.

*Available:* [*https://blog.trendmicro.com/trendlabs-security-intelligence/mobile-malware-high-risk-apps-hit-1m-mark/*](https://blog.trendmicro.com/trendlabs-security-intelligence/mobile-malware-high-risk-apps-hit-1m-mark/)*.*

[4] A. Ustenko, "Android Application Structure," *July 2017.*

*Available:* [*https://www.slideshare.net/AlexeyUstenko/android-application-structure*](https://www.slideshare.net/AlexeyUstenko/android-application-structure)

[5] Android, "Security," *July 2017.*

*Available:* [*https://source.android.com/security/#platform-security-architecture*](https://source.android.com/security/#platform-security-architecture)*.*

[6] D. Palmer, ""Android alert"," July 2017.

Available:*[http://www.zdnet.com/article/android-alert-this-cutesymalware-](android%20security.docx)*

*[has-infected-millions-of-devices/](android%20security.docx).*

[7] Chintan, "The security architecture of Android," July 2017.

Available:*[http://www.androidauthority.com/thesecurity-architecture-of-android-11063/.](http://www.androidauthority.com/thesecurity-architecture-of-android-11063/)*

[8] Android, "App Manifest," July 2017.

Available:*[https://developer.android.com/guide/topics/manifest/manifestintro.](android%20security.docx)*

*[html.](android%20security.docx)*

[9] Android, "Sharing your code and resources," July 2017.

Available:*<https://developer.android.com/studio/build/shrink-code.html.>*

[10] GuardSquare, "ProGuard," July 2017.

Available: [*https://www.guardsquare.com/en/proguard.*](https://www.guardsquare.com/en/proguard.)

[11] Q. Kennemer, "New Android Malware," July 2017.

Available:*[http://phandroid.com/2016/04/25/androidmalware-ransomware-website/.](http://phandroid.com/2016/04/25/androidmalware-ransomware-website/)*

[12] J.Smith, "Google Play Protect rolling out to Android devices for better security," July 2017. Available:[*http://www.zdnet.com/article/google-play-protect-nowrolling-out-to-android-devices/.*](http://www.zdnet.com/article/google-play-protect-nowrolling-out-to-android-devices/.)