2018

Security vulnerabilities in android applications

Crischell Montealegre
Edith Cowan University, mmonteal@our.ecu.edu.au

Charles Rubia Njuguna
Edith Cowan University, cnjuguna@our.ecu.edu.au

Muhammad Imran Malik
Edith Cowan University, muhammad.malik@ecu.edu.au

Peter Hannay

Ian Noel McAteer
Edith Cowan University, imcateer@westnet.com.au

DOI: 10.25958/5c5274d466691
Montealegre, C., Njuguna, C.R., Malik, M.I., Hannay, P., & McAteer, I.N. (2018). Security vulnerabilities in android applications. In *proceedings of the 16th Australian Information Security Management Conference* (pp. 14-28). Perth, Australia: Edith Cowan University.

SECURITY VULNERABILITIES IN ANDROID APPLICATIONS

Crischell Montealegre[1], Charles Rubia Njuguna[1], Muhammad Imran Malik[1], Peter Hannay[2], Ian Noel McAteer[1]
[1]School of Science, Edith Cowan University, [2]Asterisk Information Security
Perth, Australia
mmonteal@our.ecu.edu.au, cnjuguna@our.ecu.edu.au, muhammad.malik@ecu.edu.au,
peter.hannay@asteriskinfosec.com.au, i.mcateer@ecu.edu.au

Abstract

Privacy-related vulnerabilities and risks are often embedded into applications during their development, with this action being either performed out of malice or out of negligence. Moreover, the majority of the mobile applications initiate connections to websites, other apps, or services outside of its scope causing significant compromise to the oblivious user. Therefore, mobile data encryption or related data-protection controls should be taken into account during the application development phase. This paper evaluates some standard apps and their associated threats using publicly available tools and demonstrates how an ignorant user or an organisation can fall prey to such apps.

Keywords
Android, vulnerability scan, APK, AndroBugs, Ostorlab, Social Media

INTRODUCTION

In recent times, organisations have been deploying mobile applications to facilitate their business processes. Employees, customers, and vendors experience the exceptional exchange of services increasing productivity in the working environment through sharing of real-time information, free mobility, and better functionality.

Notwithstanding mobile apps benefits, however, usage of mobile apps can potentially lead to severe security hitches. Similar to obsolete enterprise applications, apps may contain vulnerabilities prone to attack. An attacker may exploit these vulnerabilities to gain unauthorised access to an organisation's information technology resources or a user's data (Quirolgico, Voas, Karygiannis, Michael, & Scarfone, 2015).

Literature Review

In the recent past, software delivery to an end user has taken a fundamental paradigm shift with easy-to-download, install, and use applications from mobile app markets. High-end user demand for Android apps has led to an increase in the production rate at which applications are developed and released in the market without overseeing authority. Although these contribute to an equal playing ground for both small organisations and prominent software development companies, the massive growth of new apps could equally compromise apps' security. Deploying new technology could have tragic consequences, causing a potential security threat to an organisation's IT resources, data, and users. ANZ Bank in rolling out their new ANZ app while retiring their GoMoney app is asking users to download and install the new app with their previous registered credentials. While new technologies may offer the promise of productivity gains and new capabilities, they may also present new risks. It is vital for an organisation's IT experts and users to be made fully aware of these risks and either develop plans to mitigate them or accept their consequences (Coyne, 2018).

Most large active enterprise data has reported having been sporadically leaked from mobile apps. For instance, Appthority Enterprise mobile security vendor scanned 1100 apps that use a communications Application Programming Interface (API) marketed by Twilio. Figure 1 illustrates vulnerabilities in apps as exposed in developers' hard-coded logs, which shows usernames and passwords credentials in their code (Appthority, 2017).

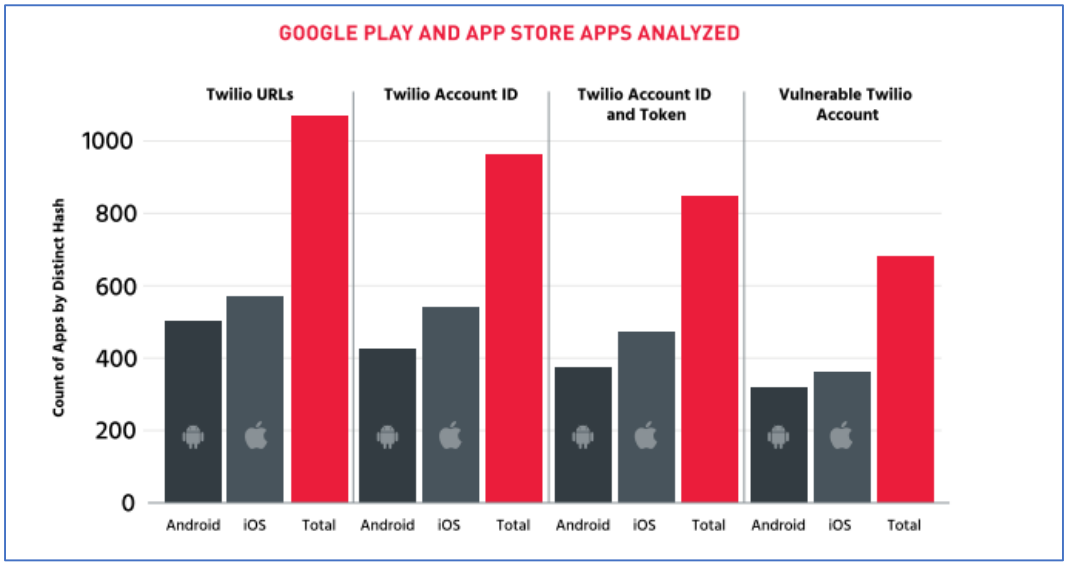

Figure 1 - Apps Exposing Hardcoded Credentials (Appthority, 2017)

Access from the hardcoded credentials gives the attacker the ability to run a mass surveillance operation at a scale typically only available to governments and carriers. The scale, measured by total app installs, is likely to be in the hundreds of millions of impacted mobile users whose calls and text messages may all be accessed.

The changing trend has contributed to mobile devices becoming indispensable components of our daily lives and integral parts of organisational culture. The current development of advanced processing, connectivity capabilities, and the ability for Android to host multi-purpose third-party applications, is a paradigm shift in organisation transformation that needs to be addressed. While mobile applications provide business agility, it also poses security challenges. Security threats for mobile apps have become common with more evidence of apps affected with malware in the Android market. Table 1 shows the list of malicious apps that were recently removed from the official Google Play Store. The mobile banking trojans concealed as device cleaners, boosters, or horoscopes have the capabilities to impersonate banking apps, intercept and send SMS, or install other applications (Stefanko, 2018).

Table 1 - Mobile Application Malware (Stefanko, 2018)

App Name	Package Name	Installs
Power Manager	com.puredevlab.powermanager	10+
Astro Plus	com.astro.plus	0+
Master Cleaner - CPU Booster	bnb.massclean.boost	5,000+
Master Clean - Power Booster	mc.boostpower.lf	100+
Super Boost Cleaner	cpu.cleanpti.clo	500+
Super Fast Cleaner	super.dupclean.com	500+
Daily Horoscope For All Zodiac Signs	ui.astrohoro.t2018	100 +
Daily Horoscope Free - Horoscope Compatibility	com.horochart.uk	500+
Phone Booster - Clean Master	ghl.phoneboost.com	1,000+
Speed Cleaner - CPU Cooler	speeeed.cool.fh	100+
Ultra Phone Booster	ult.boostphone.pb	1,000+
Free Daily Horoscope 2019	fr.dayy.horos	50+
Free Daily Horoscope Plus - Astrology Online	com.dailyhoroscope.free	1,000+
Phone Power Booster	pwr.boost.pro	1,000+
Ultra Cleaner - Power Boost	ua.cleanpower.boost	50+

App Name	Package Name	Installs
Master Cleaner - CPU Booster	bnm.massclean.boost	5,000+
Daily Horoscope - Astrological Forecast	gmd.horobest.ty	1,000+
Speed Cleaner – CPU Cooler	speeeed.cool.gh	0+
Horoscope 2018	com.horo2018i.up	1,000+
Meu Horóscopo	my.horoscop.br	1,000+
Master Clean - Power Booster	mc.boostpower.cf	50+
Boost Your Phone	boost.your.phone	1,000+
Phone Cleaner - Booster, Optimizer	phone.boost.glh	1,000+
Clean Master Pro Booster 2018	pro.cleanermaster.iz	10+
Clean Master - Booster Pro	bl.masterbooster.pro	5,000+
BoostFX. Android cleaner	fx.acleaner.e2018	50+
Daily Horoscope	day.horocom.ww	1,000+
Daily Horoscope	com.dayhoroscope.en	1,000+
Personal Horoscope	horo.glue.zodnow	1,000+

Background

The purpose of this study is to establish security issues in Android smartphone applications. Android smartphones store diverse data such as multimedia, sensor data, communication logs, and data created or consumed by applications, and so on. An Android user carries the device over multiple locations throughout the day and allows connections to various networks that are often not secure. As the same device may be used for both work and leisure purposes, installed Android applications often contain a combination of valuable personal and business data. Exposing potential security vulnerabilities in commonly used Android applications will help understand the risk associated with mobile apps within corporate premises.

Android smartphones extend the business perimeter, while existing security and privacy perimeter-oriented mechanisms are inadequate and easily compromised. In this context, the importance of Android applications interacting with corporate assets, make them economically attractive to attackers. This attraction happens because most people rarely consider Android-application threats when downloading from the Google Play Store. Furthermore, most vulnerability-assessment methods are not intended for individuals, but mainly for businesses. Thus, a targeted vulnerability assessment of Android applications is useful in assessing smartphone threats in a considerably more specific approach. We contribute towards this direction by identifying security threats on commonly used social media applications and compare different vulnerability-scanning frameworks tailored for Android applications.

METHODS

Static analysis

Static analysis is performed without executing the application on the provided or decompiled source code and accompanying APK files (Velu, 2016). This method indeed proves to be more thorough when using AndroBugs and Ostorlab frameworks, and cost-efficient with the ability to detect critical and non-critical threats to apps from the Google Play Store. Static analysis also unearths weakness that would not emerge in a dynamic test. Static analyses using frameworks are used for:

- Scanning of Android APK files
- Determination of possible vulnerabilities
- Reporting of any identified vulnerabilities

By demonstrating the effectiveness of different vulnerability-scanning frameworks and reporting the outcomes of mobile-app weaknesses and testing tools, organisations could use the same procedures to expose and determine the appropriateness of apps for deployment on an organisation's mobile devices. Ultimately, an organisation's security requirements could then be followed to determine the environment for deployment, usage, and ideal mobile technologies. For instance, hospital or airline environments could consider a more comprehensive apps-vetting process compared to a media or marketing company. This paper highlights some findings that are particularly important to consider before apps are approved for submission to the Play Store or Google Play markets.

Google Play Apps

Social networks are presently the most significant media spot in the world, and the most widely used channel for data, video and voice. Reporting and risk assessments generated by scanning frameworks demonstrate some exciting results. Table 2 below considers mobile applications downloaded from Google Play.

Table 2 - Android Apps Used in Testing

Apps	PackageName	Version	Size (MB)	Hash MD5 / SHA1
Facebook	com.facebook.katana	129.0.0.29.67	68	eb68e95931dbfe27bd096220919a6cb279b882db8cbfde5c04301f4e0c8c110e821e9a7f
IMO	com.imo.android.imoim	9.8.000000006751	7.3	955a28e1b3a67ac3b0fa6a3a7f073b06ca3b0291a478cdf3780d10eb533c7e96e66c021d
Instagram	com.instagram.android	10.26.0	27	48a284d59ff0f9affb7013962d5accfd80d103aaf6bae88c749f87321bf1e80c19b04a1a
LinkedIn	com.linkedin.android	4.1.61	27.4	47d198045715829fd73ac5134bf889c844ac8bce52e60352ab251702c5ab16ea81309374
Viber	com.viber.voip	6.9.5.9	30.8	c9376fe2f5fc6d928bedd95c1b87299af48d2a4fcf2b4ea944bc0077106e570b3ce5eb88
WhatsApp	com.whatsapp	2.17.231	35	b26bf9b3ddb2aab5af342ac46307604733eae8c80c1b8f157af3d91d104c608b45f95993
Twitter	com.twitter.android	7.0.0	34.2	e803a8f19ac3ddda5accbc6db87dd2e45b36e1a22003ffcec88e7bea05f2e644cc364193
WeChat	com.tencent.mm	6.5.8	40	cf237d05ab4782081ac70cbd2210ee3e32ff65d4ee3cbce62d1a1e924a98ddfef1da7e06

Planning

The study compared and analysed the outcomes of three different application-scanning frameworks to broadly gain an understanding of the various risks and vulnerabilities linked with the usage of leading social-media applications within business perimeters. However, on submission only the results of two frameworks have been analysed. The application-testing frameworks are:

- AndroBugs Terminal Framework
- OstorLab Cloud Framework

AndroBugs Terminal Framework

AndroBugs is an Android app-vulnerability scanner that provides an advanced security assessment for Android apps and uncovers potential critical issues relating to user applications. AndroBugs checks every component in the app to ascertain flaw-security and correct logic that would instead be exploited by hackers. Security problems can still perpetuate within the Android environment, because:

- Google may never employ app security that enforces or rates the security of an app.
- Google may never take down vulnerable apps from Google Play.
- Google already knows some security problems in certain Android applications.
- There are too many applications that use vulnerable APIs.

AndroBugs could help improve Android security, although companies need to give mobile security the same attention given to Web security by following Android-security researchers' exercise of responsible disclosure and acknowledge vulnerability reports they receive. Figure 2 below demonstrates how apps could be verified before being approved for installation by the end user (Quirolgico et al., 2015).

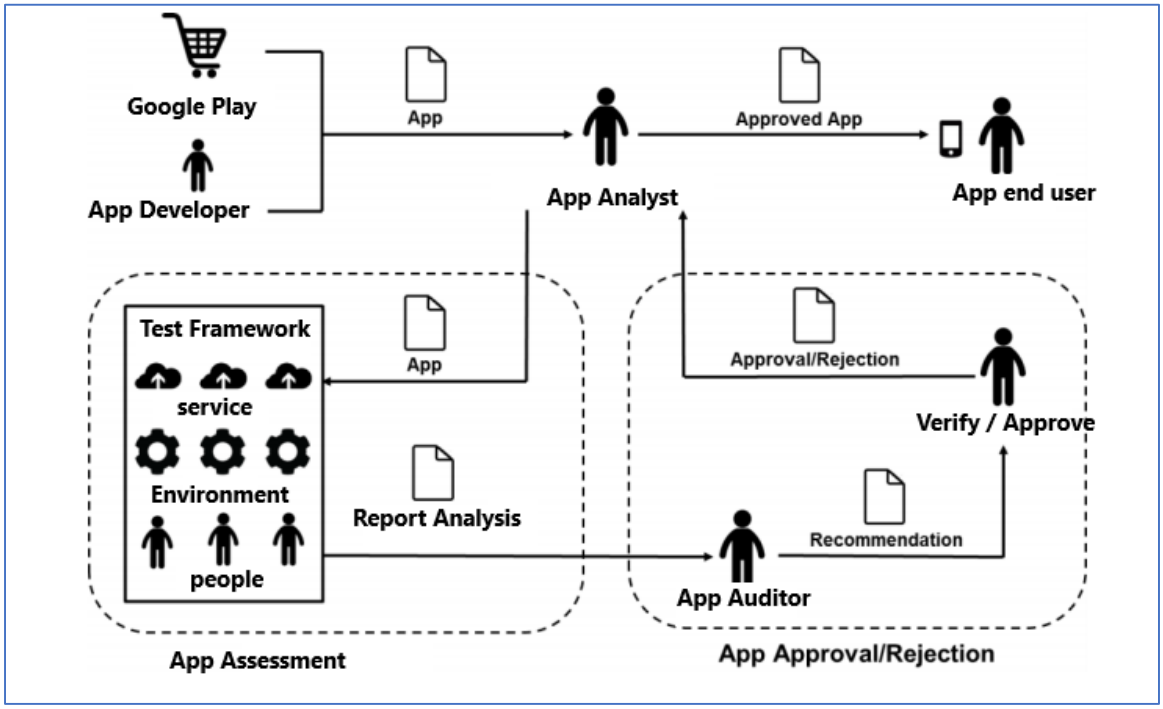

Figure 2 - Application Verification Process (Quirolgico et al., 2015)

Ostorlab Web-Based Framework

Ostorlab is a cloud-based application scanner framework for Android or iOS apps, and gives detailed information on the findings. An APK file is uploaded as an attachment to the framework which generates a security scan report in a few minutes. Figure 3 illustrates the Ostorlab scanning process in a web environment (Spreitzenbarth, 2013).

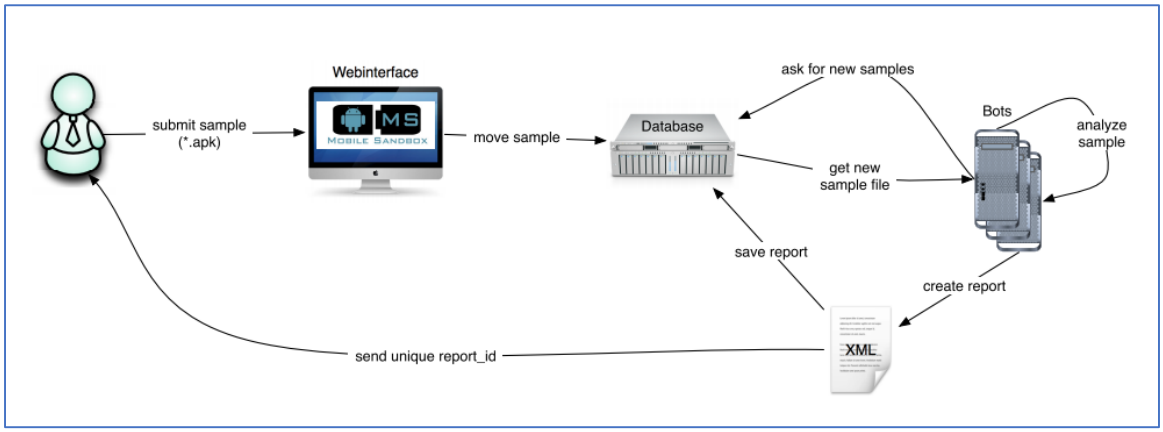

Figure 3 - Ostorlab Web-Based App Scanning Framework (Spreitzenbarth, 2013)

Scanning Process

The scanning process comprises of three main steps:

1. Generating APK files.
2. Scanning using AndroBugs.
3. Ostorlab framework.

Downloading APK files

All APK files were downloaded from Google Play using a web-services downloader. Following are the steps to download the APKs:

1. Open Evozi website (https://apps.evozi.com/apk-downloader/).
2. On a separate web browser, open Google Play (https://play.google.com).
3. Search and view the apps.
4. On the apps page, right-click on the “Install” button and select copy address (This step may be different depending on the web browser. We were using Opera).
5. Going back to Evozi, past the URL on the Google Play URL (see Figure 4).
6. Click Generate Download Link.
7. Click ‘Click here to download <*app package name*> now.
8. Verify MD5 hashes.

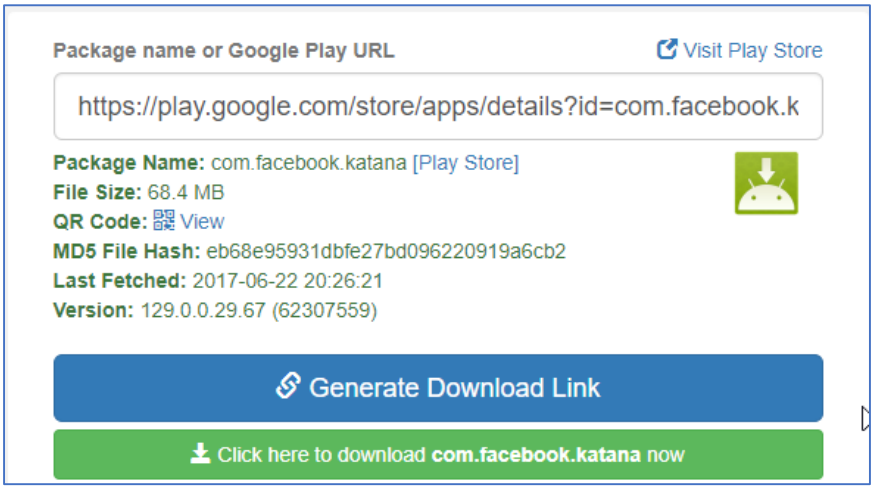

Figure 4 - Evozi Web Service: Download APK

Scanning using Ostorlab

1. Go to Ostorlab website (https://www.Ostorlab.co) and click the scan icon.
2. Fill in the fields, select your APK file and click scan (see Figure 5).
3. Ostorlab will email the link of the result.

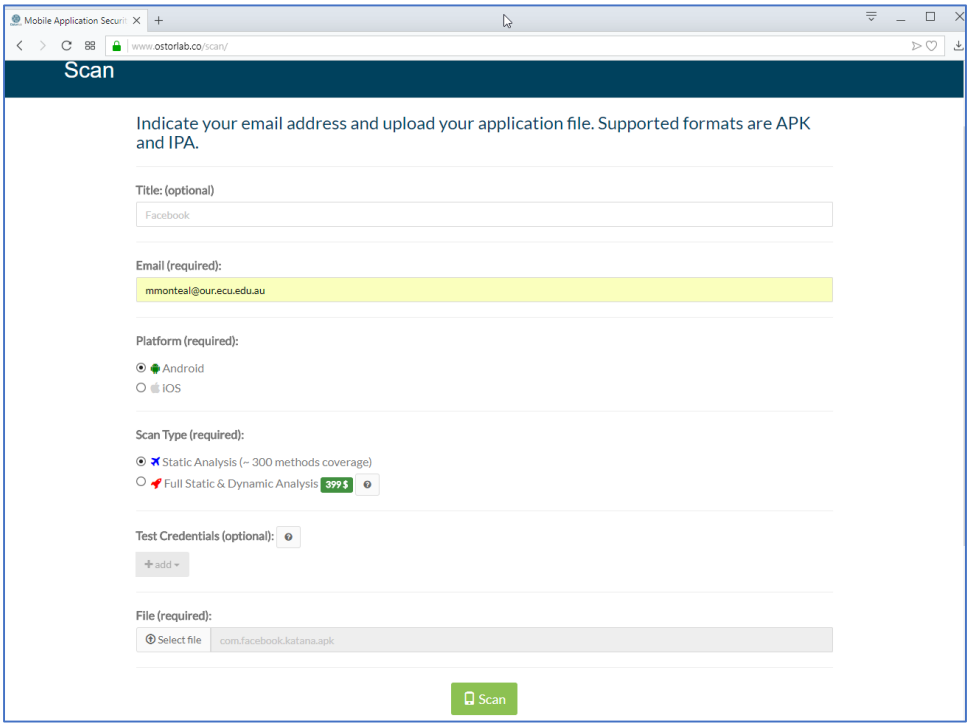

Figure 5 – Scanning on Ostorlab

Scanning using AndroBugs

1. Using Kali Linux, unzipped AndroBugs to the directory.
2. Open a terminal and execute the command:

   Python androbugs.py –f *apk file* -o *output result*

   Figure 6 demonstrates how to execute the command:

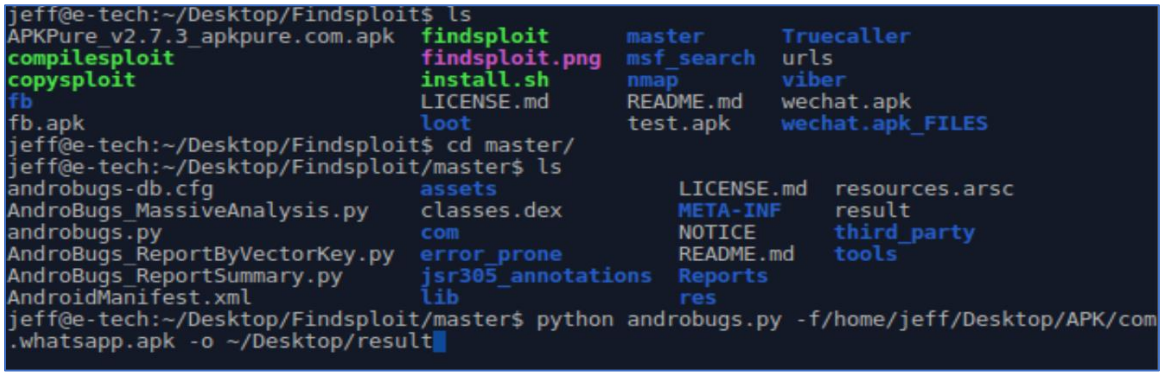

Figure 6 - Running Androbugs

RESULTS AND FINDINGS

Vulnerability Findings

Vulnerabilities from the frameworks used have been categorised in order of priority and potential impact to the end user. Table 3 shows the categorisation used for both tools:

Table 3 - Risk Categorization

AndroBugs	Ostorlab
Critical	High / Potentially
Warning	Important
Notice	Medium
Info	Note

The following vulnerabilities were discovered after running the eight mobile apps. Table 4 summarises the results from AndroBugs. Vulnerabilities with an Info categorisation, which indicates that there was no issue found on the specific static analysis, were not included in this table. Table 5 illustrates the results from Ostorlab.

Table 4 - AndroBugs Result

Title	Category	Facebook	IMO	Instagram	LinkedIn	Twitter	Viber	WhatsApp	WeChat
<SSL_Security> SSL Connection Checking	Critical	true	true	true	true	true	true	true	true
<WebView><Remote Code Execution><#CVE-2013-4710#> WebView RCE Vulnerability Checking:	Critical	true	true	true	true	true	true		true
<Implicit_Intent> Implicit Service Checking	Critical	true	true		true		true	true	true
AndroidManifest ContentProvider Exported Checking	Critical	true		true					true
<SSL_Security> SSL Certificate Verification Checking	Critical	true		true				true	
<SSL_Security> SSL Implementation Checking (Verifying Host Name in Custom Classes)	Critical	true		true					
App Sandbox Permission Checking	Critical		true		true		true	true	true
<Hacker> Base64 String Encryption	Critical		true						true
<#CVE-2013-6272#> AndroidManifest Exported Lost Prefix Checking	Critical	true							
<Command> Runtime Command Checking	Critical			true			true	true	true
<KeyStore><Hacker> KeyStore Protection Checking	Critical			true					
AndroidManifest Dangerous ProtectionLevel of Permission Checking	Critical					true			
<#BID 64208, CVE-2013-6271#> Fragment Vulnerability Checking	Critical							true	
External Storage Accessing	Warning		true	true	true	true	true	true	true
AndroidManifest Exported Components Checking	Warning	true	true	true	true	true	true	true	true
<Sensitive_Information> Getting ANDROID_ID	Warning	true	true	true	true	true	true	true	true
<WebView> WebView Local File Access Attacks Checking	Warning	true	true	true	true	true	true		true
Dynamic Code Loading	Warning		true	true	true			true	true
<WebView> WebView Potential XSS Attacks Checking	Warning	true	true	true	true	true	true		
Codes for Sending SMS	Warning		true						

Title	Category	Facebook	IMO	Instagram	LinkedIn	Twitter	Viber	WhatsApp	WeChat
<Sensitive_Information> Getting IMEI and Device ID	Warning	true		true			true		true
AndroidManifest "intent-filter" Settings Checking	Warning				true				
<SSL_Security> SSL Certificate Verification Checking	Warning						true		
AndroidManifest Normal ProtectionLevel of Permission Checking	Warning								true
<Database><#CVE-2011-3901#> Android SQLite Databases Vulnerability Checking	Notice		true	true	true	true	true	true	true
<Signature><Hacker> Getting Signature Code Checking	Notice		true	true	true	true	true	true	true
Native Library Loading Checking	Notice		true	true		true	true	true	true
<Hacker> APK Installing Source Checking	Notice		true	true		true	true	true	
File Unsafe Delete Checking	Notice	true	true	true	true	true	true	true	true
AndroidManifest Exported Components Checking 2	Notice	true	true	true	true	true	true	true	true
<Command> Executing "root" or System Privilege Checking	Notice	true	true	true		true		true	true
<Debug><Hacker> Codes for Checking Android Debug Mode	Notice	true		true			true		
<KeyStore><Hacker> Possible KeyStore File Location	Notice					true			
AndroidManifest Adb Backup Checking	Notice						true		
<KeyStore><Hacker> KeyStore File Location	Notice						true		
<KeyStore><Hacker> KeyStore Protection Information	Notice						true		
<Database> Android SQLite Databases Encryption (SQLite Encryption Extension (SEE))	Notice		true				true		

Table 5 - Ostorlab Results

Title	Category	Facebook	IMO	Instagram	LinkedIn	Twitter	Viber	WhatsApp	WeChat
Virustotal malware analysis (MD5 based search)	High		true						
Services declared without permissions	Potentially	true	true	true	true		true	true	true
Backup mode enabled	Potentially					true	true		
Insecure Filesystem Access	Potentially								true
Intent Spoofing	Potentially							true	
Exported activites, services and broadcast receivers list	Important	true	true	true	true	true	true	true	true
Decompiled source code	Important		true	true	true	true	true	true	
ELF binaries do not enforce secure binary properties	Medium	true				true			true
Application code not obfuscated	Medium					true			
Hardcoded SQL queries list	Note	true	true	true	true		true		
Hardcoded urls list	Note	true	true	true	true		true		
APK files list	Note	true	true	true	true	true	true	true	true
APK attack surface	Note	true	true	true	true	true	true	true	true
Virustotal malware analysis (MD5 based search)	Note	true		true	true	true	true	true	true
Hardcoded strings list	Note	true	true	true	true	true	true	true	true
Android Manifest	Note	true	true	true	true	true	true	true	true
List of JNI methods	Note	true	true	true	true	true	true	true	true
Implementation of a FileObserver	Note		true	true					
Obfuscated methods	Note		true	true	true	true	true	true	
Call to XML parsing API	Note		true		true	true	true	true	
Call to native methods	Note		true	true	true	true	true	true	
Application checks rooted device	Note		true	true	true	true	true		
Implementation of a WebViewClient	Note		true	true	true	true	true		
Call to Inter-Process-Communication (IPC) API	Note		true	true	true	true	true	true	

Title	Category	Facebook	IMO	Instagram	LinkedIn	Twitter	Viber	WhatsApp	WeChat
Call to External Storage API	Note		true	true	true	true	true	true	
Call to dangerous WebView settings API	Note		true	true	true	true	true	true	
Call to Socket API	Note		true	true	true	true	true	true	
Call to logging API	Note		true	true	true	true	true	true	
Application components list	Note		true	true	true	true	true	true	
Call to SSL/TLS API	Note			true	true	true	true	true	
Application certificate information	Note		true		true	true	true	true	
Call to Reflection API	Note		true	true	true	true	true	true	
Call to Crypto API	Note		true	true	true	true	true	true	
Call to Random API	Note		true	true	true	true	true	true	
Call to SQLite query API	Note		true	true	true	true	true	true	
Call to dynamic code loading API	Note		true	true	true	true	true	true	
Call to command execution API	Note			true		true	true	true	
MoPub Framework detected	Note					true			
URL Scheme listURL Scheme list	Note		true					true	true

Vulnerability and Risk Assessments

Based on the findings generated by the above tools, eighty different social media applications that are commonly used in Android smartphones were analysed. The result indicates various risk levels that exist in Android mobiles and therefore putting the user in high risk of leaking both individual and organisational data when interfacing with installed apps from Google Play. Consequently, this merits targeted specific app-vulnerability assessment.

Threats on vulnerable apps are:

- Personal user data-leakage over a network, e.g. email, IMEI, GPS, MAC.
- Unencrypted communication over the network.
- Having world-readable/writable files.
- Poor authorization and authentication.
- Information-stealing malware.
- Known vulnerabilities.

Sensitive data leakage

Sensitive data leakage can happen when an app is improperly storing user information. Mobile apps should avoid unnecessary storage of data on a device. According to OWASP "Insecure data storage, occurs when development teams assume that users will not have access to the phones file system and store sensitive pieces of information in data-stores on the phone. Devices file systems are often easily accessible, and the user should expect a malicious entity to be inspecting the data stores. Rooting or jailbreaking a device usually circumvents any encryption protections, and in some cases, where data is not protected properly, all that is needed to view application data is to hook the phone up to a computer and use some specialised tools " (OWASP, 2014a).

Three apps may be susceptible to the Android SQLite Database Vulnerability (CVE-2011-3901). More so, it is suggested that all of the apps are using an unsecured way to delete files. By using file.delete(), any attacker, especially on rooted devices, may recover everything you delete. Also, almost all are using or have API calls to external data storage. It is imperative to ensure that sensitive information is handled well. It is also worth noting that the apps are reading the ANDROID_ID, IMEI and deviceID information.

Unencrypted communication

The most important feature of the client-server architecture is information exchange. When data is transmitted, it may be exchanged through the carrier network or the Internet. While developing an application, if care is not taken while sharing data between the client and server, there is a chance that the data may be compromised in transit. The best way to protect data in transit is to encrypt it. Encryption prevents sniffed data from read, particularly in the case of usernames, passwords, and credit card information. According to OWASP "Unfortunately, mobile applications frequently do not protect network traffic. They may use SSL/TLS during authentication, but not elsewhere, exposing data and session IDs to interception. Also, the existence of transport security does not mean it is implemented to its full potential. Detecting basic flaws is easy. Just observe the phone's network traffic. More subtle flaws require inspecting the design of the application and the application's configuration" (OWASP, 2014b).

All apps include URLs that are not using SSL. Referencing the AndroBugs result, WeChat has the most number of URLs (44), followed by Instagram (16), Twitter (11), IMO (6), Viber (5), LinkedIn and WhatsApp (4), and Facebook (1). Also, Facebook, Instagram and WhatsApp do not check the validation of the SSL Certificate which allows self-signed, expired or mismatched Common Name (CN) certificates for SSL.

Information disclosure read and writes

Disclosure of relevant data stored in apps such as passwords and credit card details, which should remain hardcoded, is a requirement for any developer to prioritize because most applications developed for mobile devices can reveal the code when reversed engineered. A hacker could access this sensitive information that to further facilitate access to company resources compromising their reputation.

Five apps were found to have permitted "MODE_WORLD_READABLE" or "MODE_WORLD_WRITEABLE" on some of the services, in particular in using Advertising ID API.

Poor Authorization and Authentication

Authentication and authorisation refer to user privileges granted for using an application. In an application with functionalities beyond publicly usable features, permission may be required for accessing free functions. Authentication refers to who you are in an application. For instance, email: xyz@a.b.c | Website: www.abc.edu.

Authorisation points to what you are authorised to do in an application. When the authorisation and authentication schema fails to protect the application, the privileged functions in the application are compromised, rendering it vulnerable to attacks. Authorisation and authentication should be dealt with accurately while developing an application to ensure that unauthorised users are not granted access to sensitive information. This can be achieved by ensuring secure session-handling and login functions.

Information Stealing Malware

Mobile smartphones, in particular, Android devices, provide several ways for applications to be downloaded and installed, for example from official Google Play Store, from other third-party marketplaces, or from APK-downloading sites. Currently, most distribution of malware for smartphones have utilised third-party app stores. Mainly, malware distributed through the third-party target to steal data from the mobile device (Kaspersky, 2016). ZitMo is one of the most rampant pieces of mischievous code for a smartphone. It started in 2010 for Symbian OS, and designed to forward SMS messages. ZitMo for Android targeted the SMS-based banking two-factor authentication (Maslennikov, 2011) (Alliance, 2012).

Malware apps target information on Facebook accounts. The malicious apps try to collect data from Facebook. If no account can be collected, it launches the spoofed Facebook login UI to steal user credentials. Once malware connects to a real Facebook page, it takes full control of user profile (Zhang & Aimoto, 2018).

When downloading APKs, the Snapshot app was mistakably analysed instead of the Snapchat app. The lapse was later discovered after receiving a different hash check. The error was due to a Google search result, which indexed items being searched on the address.

Known Vulnerabilities

Table 6 shows the known vulnerabilities identified:

Table 6 - Known Vulnerabilities

Vulnerability Reference	Apps	Description	Affected Platform
CVE-2013-6271	WhatsApp	Allow an attacker to remove the device locks and bypass restrictions ("CVE-2013-6271," 2013)	Before Android 4.4 (API 19)
CVE-2011-3901	Viber / WhatsApp / IMO	SQLite Journal Information Disclosure vulnerability ("CVE-2011-3901," 2011)	Before Android 4.0
CVE-2013-4710	Facebook / IMO / Viber / LinkedIn / Twitter / Instagram / WeChat	This method can be used to allow JavaScript to control the host application ("CVE-2013-4710," 2013)	Prior to Android 4.2

CONCLUSION

Organisations face frequent threats to data security and privacy, and prioritising these in the noise of continuously developing security concerns is difficult. The main focus of this research was to demonstrate vulnerabilities that exist in commonly used social network apps and analyse the threats with the highest potential impact on the business environment. The results highlight security issues to be considered by organisations and application users.

A vulnerable app that has access to corporate data is a potential channel for such threats, and is rarely monitored when interfacing with the restricted commercial environment. Google Play, with a high volume of apps, largely

stores unmitigated mobile applications. App-data leakage, un-encrypted communication, and unauthorised access vulnerabilities demonstrate the need for organisations to understand and protect against a broader set of app risks to sensitive data. Both AndroBugs and Ostorlab frameworks show how simple errors by a developer can deluge substantial amounts of data across hundreds of apps, opening the possibility for the mass data exposure and user surveillance of an organisation.

REFERENCES

Alliance, C. S. (2012). Security Guidance for Critical Areas of Mobile Computing (CLOUD SECURITY ALLIANCE Security Guidance for Critical Areas of Mobile Computing, V1.0), Mobile Working Group

Appthority. (2017). How a Mobile Developer Error is Exposing Millions of Conversations.

Coyne, A. (2018). ANZ retires Grow, goMoney apps. Retrieved from https://www.itnews.com.au/news/anz-retires-grow-gomoney-apps-485437

CVE-2011-3901. (2011). Available from National Vulnerability Database Common Vulnerabilities and Exposures. Retrieved from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-3901

CVE-2013-4710. (2013). Available from National Vulnerability Database Common Vulnerabilities and Exposures. Retrieved from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-4710

CVE-2013-6271. (2013). Available from National Vulnerability Database Common Vulnerabilities and Exposures. Retrieved from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-6271

Kaspersky. (2016). Mobile Malware Evolution 2016. Retrieved from https://securelist.com/files/2017/02/Mobile_report_2016.pdf

Maslennikov, D. (2011). ZeuS-in-the-Mobile for Android. Retrieved from https://securelist.com/zeus-in-the-mobile-for-android-10/29258/

OWASP. (2014a). Mobile Top 10 2014-M2. Retrieved from https://www.owasp.org/index.php/Mobile_Top_10_2014-M2

OWASP. (2014b). Mobile Top 10 2014-M3. Retrieved from https://www.owasp.org/index.php/Mobile_Top_10_2014-M3

Quirolgico, S., Voas, J., Karygiannis, T., Michael, C., & Scarfone, K. (2015). Vetting the Security of Mobile Applications. doi:10.6028/nist.Sp.800-163

Spreitzenbarth, M. (2013). Forensic Analysis of Android and its malicious Applications.

Velu, V. K. (2016). Mobile Application Penetration Testing.

Zhang, M., & Aimoto, S. (2018). Android Malware Harvests Facebook Account Details. Retrieved from https://www.symantec.com/blogs/threat-intelligence/android-malware-harvests-facebook-details