A 10 Year Time Span Analysis of Android Vulnerabilities

by

Adam Paul Hooper

A Dissertation Presented in Partial Fulfillment

of the Requirements for the Degree

Doctor of Science in Cybersecurity

CAPITOL TECHNOLOGY UNIVERSITY

May 2023

A 10 Year Time Span Analysis of Android Vulnerabilities

Approved:

Dr. Ian McAndrew, Chair

Dr. William H. Butler, Committee Member

Dr. Kellep Charles, External Examiner

Accepted and Signed:

Ian McAndrew 16th of May 2023
Ian McAndrew, Ph.D. Date

William Butler May 16, 2023
William H. Butler, D.Sc. Date

Kellep Charles May 16, 2023
Kellep Charles, D.Sc. Date

Ian McAndrew 16th of May 2023
Ian McAndrew, PhD, FRAeS Date
Dean of Doctoral Programs
Capitol Technology University

ABSTRACT

Vulnerabilities have been continually discovered and disclosed within the Android operating system since the initial public release. Such findings are recorded and tracked in Critical Vulnerabilities and Exposures and the National Vulnerability Database. As iterations in the operating system version are developed, vulnerabilities continue to evolve and occasionally reappear. Past flaws, which were mitigated in prior releases, have the potential to reappear as the code base evolves. Such instances may be exploited in the wild, with a likelihood of being re-exploited across numerous releases, including major revisions. Utilizing the metrics of past vulnerabilities and the affected versions, it is possible to demonstrate reoccurrence in a quantifiable method over a multi-year time span. Leveraging well established forecasting algorithms, it is possible to predict a likely continuation in the reoccurrence of vulnerabilities. Additionally, the descriptive statistical analysis method allowed for the use of a large index of data samples for future reoccurrence trend detection on various metrics of the Android OS and attack vectors of the vulnerabilities. Such metrics include the investigation of API and SDK and the correlation with the KEV, as highlighted by CISA utilizing the snowball approach of sampling in anticipation of parametric data with a fixed parameter set. Implications from this data strive to shed light on past mistakes and drive industry to consider lessons learned in order to reduce vulnerability reoccurrence and heighten overall operating system security.

Keywords: Android, version, vulnerability, mitigation, reoccurrence

DEDICATION AND ACKNOWLEDGEMENT

I dedicate this work to my family and am thankful for all of the support over the past few years to reach this milestone. First, my wife, Amy, who continually pushed me to press on even when life and work circumstances seemed insurmountable. Next, to my four beautiful daughters whose own accomplishments in education continue to be a regular motivator and make me proud of such a motivated and brilliant household. Finally, to extended family and friends, whose constant "Are you done yet?" questions nudged me towards completion.

I would also like to acknowledge my Chair, Dr. McAndrew, who help clear hurdles and press on towards submission. Additionally, Dr. Butler, as a professor and committee members whose encouragement and feedback helped me remain the course. Finally, to my peers which provided suggestions and motivation along the way.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

The adoption of operating systems (OS) was previously dominated by Microsoft on desktop platforms, and the market share since has been overcome by the presence of other OS distributions, partially due to the increased use of mobile devices (Shih, 2012). One OS available on mobile devices is Google's Android OS. As with each flavor of OS, Google's Android has demonstrated an increasing number of vulnerabilities with each release (Wu, 2015). The trending and analysis of vulnerabilities over an extended period of time across releases has not been performed on the Android OS, resulting in the industry lacking information on the potential improvements to security posture (Thomas, Beresford, & Rice, 2015). In regards to other OSs, mobile and desktop variants, taxonomies have been developed which analyze vulnerabilities on UNIX based releases; however, the data is constrained to a single distribution (Li & Li, 2006). This study seeks to analyze the vulnerabilities discovered and published in Common Vulnerabilities and Exposures (CVE) for the Android OS over a 10 year time span.

Chapter 1 of the dissertation will contain the background of the study, the problem statement, and the purpose of the dissertation study. Following the aforementioned sections will be the significance of the study, the nature of the study, the hypotheses, and research questions. Finally, the conceptual framework and definitions will be covered, ending with the assumptions, scope, limitations, delimitations, and the summary.

Background of Study

The research is aimed to discover patterns and trends of vulnerabilities known to exist in the Android OS. Unmitigated vulnerabilities in recent instances of the Android

OS deployed to consumer devices have not been studied to determine potential real-world security impacts. The study would leverage a time series analysis over a 10 year period to analyze the rate of vulnerability change. The study would attempt to achieve improvements in security posture across all Android OS architecture by analyzing data across respective versions and releases. The lack of analysis of data trends over an extended time period represents a gap in knowledge from a scientific perspective which would have an impact on the industries understanding. Knowledge gained from the research may allow for a more focused improvement on the Android security landscape.

A study has not been performed on the current iterations of the Android OS above version 5.x of vulnerabilities and malware detected running in the wild, leaving the industry without a scholarly knowledge of known threats. Previous studies in the area of vulnerabilities and malware detected in the Android OS has been limited to versions 5.x and prior. No known studies have been published which cover vulnerabilities from the initial version with available CVEs dating from version 1.5 in 2009. Current implementations of Android are considered to be versions 6.x-11.x, which are presently distributed by manufacturers on existing devices. The OS developer, Google, and vendors which utilize the OS in their distributed devices do not have relevant scientific data to further their development of future releases or mitigate vulnerabilities in current releases.

One question the study seeks to answer is to what extent is the rate of vulnerabilities changing. The importance of the question of the extent of vulnerability changes may be evident by the increased detection of vulnerabilities over time since the inception of the OS. An additional question to answer is if each platform or the overall posture has improved over time. Changes in security posture over time may demonstrate

trending that can be used to target common vulnerabilities and mitigate the threats before the OS version is released to the public. Finally, is to determine if any vendors are performing better from a security perspective than others. Rating the performance of one vendor over another will allow the industry as a whole to determine better practices for implementing security standards during the development and sustainment life cycles.

The pool of data from which the study will be based on will be pulled from published CVE data. The social relevance of the published data may be evident by the potential availability of unpublished or zero-day vulnerabilities. Although not a consideration of this study, the examination of vulnerabilities not available via the CVE data demonstrates a knowledge gap observed by the general public. The overall goal of the study is to accomplish the creation of a defined matrix which demonstrates vulnerabilities across a fixed period among all Android versions. Made publicly available, the data generated by the study may generate future theoretical interest if a detectable trend or pattern is observed in the outcome of the matrixed data's results.

Ratazzi (2016) demonstrated the flaws in his work by studying only Android 5.x and below. It is well known that version 6 was released in 2015 and version 7 was released in 2016, with successive releases on a regular cycle. The more seminal work of Mulliner, Oberheide, Robertson, and Kirda (2013) focused on detecting vulnerabilities in versions 2.x and 4.x. Sato, Chiba, and Goto (2013) studied malware in manifest files dating back to 2012, to which the latest Android release was between 2.x and 4.x as well. The most comprehensive study, performed by Sufatrio, Tong-Wei, and Thing (2015), investigates vulnerabilities and malware detection techniques studied over the years, but also fails to consider current OSs, most likely due to the lack of availability of newer

releases at the time of their publication. It is worth noting the lack of published CVEs on the initial public release version 1.1 from 2009. The importance of the study is evident by the lack of relevant research on previous and current version releases of the Android OS. Additionally, it is to be noted that all versions mentioned in the previous studies are no longer supported, with version 6.x being the earliest supported release as of 2018.

Thomas et al. (2015) provide a measurement technique applied to older versions of the Android OS, which can be applied to newer research on the current OS known as a free, update, mean (FUM) score. Utilizing this metric would enable a better understanding of historical and current vulnerabilities on the threat landscape of the Android OS. Wu (2015) highlights various methods of detection using proven methodologies for application and kernel level vulnerabilities in older iterations of Android. Applying similar detection methodologies to more recent releases will provide better insight into the overall security posture of the OS over time. It is believed that the seminal work of the previous researchers can create a sound basis for providing a cohesive study of the current climate of vulnerabilities and malware running on modern releases of the Android OS. Developing newer measurement techniques or adapting previous methods of gathering metrics will further the industry's ability to gauge vulnerability impact.

Problem Statement

Not all vulnerabilities in the Android OS are addressed by vendors or service providers on the devices used by consumers after the initial deployment, as made apparent by the detection of zero-day vulnerabilities post release and the need for patching schedules by vendors during the OS' supported lifecycle, as outlined by Wu

(2015). The general problem statement is: upon release, the Android OS demonstrates more vulnerabilities with each version (Wu, 2015). The specific problem is: the trending of vulnerabilities with each historical version of Android OS over a 10 year span is not available to inform improvements in industry security posture (Thomas et al., 2015).

The population selected for the study will be limited to current releases at the time of research. For example, the previous versions of Android are 6.x-8.x; however, versions 9.x-11.x have been released by the time the study is to begin. A complete sampling of all known vulnerabilities will be utilized for the research study as it allows the most flexibility in collecting data not based on geological or user metrics but can be later restricted to available OS versions and correlated to the previously mentioned variables.

The rationale is most appropriate, as it allows flexibility to examine which releases are available and collect statistics for all versions analyzed. The study will not be limited by demographics or geographic locations and can easily be expanded on as new versions are released. Upon completion, this study will provide quantitative evidence that can be used to determine if the problem of vulnerable OS software from Google's Android OS is becoming more secure or increasingly less secure.

Purpose of Dissertation Study

The purpose of this quantatative dissertation study is to discover trends of vulnerability patterns from device vendors utilizing a time series analysis of Android OS vulnerabilities over a 10 year period on the most popular hardware platforms. Hardware platforms are expected to affect the outcome but due to similarity between vendors, this becomes less of a factor to consider, as seen in later findings. Additionally, it is sought to

determine if rate of vulnerabilities is changing to achieve an improvement in security posture for the architecture.

The research method to be selected for the time span analysis study of OS vulnerabilities will be quantitative. Qualitative research is stated to "not directly test for cause and effect" (Salkind, 2003); therefore will not be selected as the research method. The study used a grounded theory method; as stated by Creswell "as the theory is grounded in data" (Creswell, 2005) and "existing theories do not address your problem" (Creswell, 2005).

Grounded theory will help to answer the research questions of vulnerability rate change and improvements to platforms, posture, and vendor progress due to the fact grounded theory is a systematic process (Creswell, 2005). Trending of CVE data will be laid out in an orderly manner, to be determined as the study progresses. Additionally, leveraging the data available from CVE databases will also require the "self-correction nature" (Creswell, 2005) of grounded theory to ensure the analysis truly represents the potential trends over the 10 year time period.

The primary study upon which the initial concept of a long duration trend analysis comes from the paper "Security Metrics for the Android Ecosystem" by Thomas et al. (2015). Metrics on the vulnerabilities in versions 2.x and 4.x of the Android OS were presented on a short-term scale and locked to the particular point in time. The key concept of an expanded study across multiple OSs and over a longer duration originates from Thomas et al.'s (2015) statement of "the importance of a longitudinal study: this type of analysis requires years of data" It can be surmised the researchers believed a study would benefit from the increased duration and expanded scope.

The initial steps the research will take will be a grounded theory method to govern metrics that will help ascertain the available vulnerabilities list from which to base the study. The stated initial step will be required as all of the future analysis will focus on vulnerabilities of the available versions. Next, a structure will be developed to collect the current and historical CVE data in an analyzable format. The format will be foundational to the study to ensure the collected data can be rationalized and presented in a methodological approach. Finally, the data will be inputted into a system to formally analyze and determine any potential trending presented in a well-defined matrix.

The known biases of the time analysis of OS vulnerabilities can be correlated with the versions of the OS as well as the date the vulnerabilities are disclosed. As stated by Cone and Foster (2006), the techniques used to operationalize the variables and ensure the validity of data to be collected will be imperative and will be leveraged with the research design to reduce bias. The research design composed of a quantitative grounded theory method, may address the bias by analyzing all vulnerabilities across all of the categories and including vulnerabilities from inception within the study's time frame.

The population of the study will be all vulnerabilities across the Android OS collected from the CVE database. As stated by Salkind (2003), a sampling is to be selected to accurately represent the population as a whole, which will be defined by the independent variables. The independent variables of the study will consist of the versions and the time period, ending with the most relevant data, which is currently believed to be a 10 year span. Creswell (2005) states the independent variables can influence the dependent variable (Creswell, 2005), which will be explored in the study as the rate of change. The dependent variables will consist primarily of the Common Vulnerability

Scoring System (CVSS) score and the algorithm used to rank and score the rate of vulnerability change.

The proposed data collection methodology will be a constant comparison. Creswell (2005) defines the data collection methodology of constant comparison as a recursive procedure from specific to broad allowing for a more concise categorization of the data to be collected and analyzed (Creswell, 2005). The National Vulnerability Database (NVD) will be leveraged, and the collected data will also include values from the database source of CVSS metrics.

Significance of the Study

The study is unique in nature due to the scope of the research. Previous studies in the domain are limited to an older subset of Android releases. The study will be performed against vulnerabilities published for all releases to date. Additionally, previous studies were performed for only current versions at the time of research. This study strives to examine the historical non-supported versions as well as the current vendor supported releases.

Potential beneficiaries of the study are far-reaching. Beginning with the OS developer, new insights may be gained on trends with reoccurring vulnerabilities. Future versions of the OS would have improved security posture if the developer used this insight to avoid recreating historical mistakes. Vendors utilizing the OS on their devices would benefit by using the new knowledge to improve the mitigation of vulnerabilities based on the trending analysis. Finally, the end user would benefit from an overall more secure device.

Theoretical significance of the study potentially demonstrates the ability of vendors and service providers to mitigate known vulnerabilities addressed by Google in the Android OS in current patch release cycles. Additionally, new detection mechanisms for vulnerabilities on Android devices that can be run ad-hoc on the device or remotely with low overhead that could be leveraged by future publicly available security suites.

The practical significance of the study potentially increases user's awareness of mobile device security by demonstrating the vulnerabilities detected on their devices and the potential for loss of sensitive data on platforms assumed to be secure. Additionally, the study provides an accurate sample of vulnerabilities found in recent instances of the Android OS between versions 6.x and 11.x on devices being utilized in the wild.

Nature of Study

The research method will be quantitative for the study. A quantitative study will allow for the accurate measurement of vulnerabilities that have been addressed in patches from Google or by mitigations from vendors but not resolved on the end consumer device. The research design that will best facilitate is a case study. As stated by Creswell, "the study of a case within a real-life contemporary context or setting" can provide insight into actual vulnerabilities in the wild. A case study will allow for trend analysis of vulnerabilities contextually. As stated by Yin (1994), a case study will allow for a systematic study using multiple sources of evidence. Additionally, the research can potentially extend or strengthen knowledge in the Android security focus area. Finally, previous research studies in similar areas can be leveraged for techniques and methods to determine the best approach for analyzing the data collected.

The research method of a purely quantitative study is appropriate due to the population being examined. The raw CVE details, CVSS scores, and NVD database are highly numeric in value, and precise measurements can be taken of the published vulnerabilities. Qualitative would leave more room for speculation; however, the study strives to be analytical. The grounded theory approach will enable the study to proceed in a systematic way.

The design will focus on the collection and organization of the raw vulnerability data. Additionally, the analysis of the data will help achieve the goal of the study by demonstrating potential trends and patterns stemming from historical findings and correlation with current and future vulnerability discoveries. Finally, a deeper analysis using a scoring metric such as the CVSS will allow additional insights into the severity of the vulnerabilities.

Hypotheses/Research Questions

The main research question to be answered by the study is to what extent is the rate of vulnerabilities changing. Three sub-questions are sought to be answered in support of the main research question. First of the sub-questions, is to determine if each platform improved, which will help to create a system to gauge the progress of each platform. Next, has the posture improved over time will determine either the positive or negative rate of change over the given period. Finally, determining if any vendors are doing better than others through a ranking system will provide the ability to overlay the results from each platform's improvements and security postures. The expected result will be a charted and graphed set of data to display the main question of vulnerability rate change.

Next, the question will be examined of how many vulnerabilities have gone unmitigated by vendor implementations of the Android OS on mobile devices running versions 6.x through 11.x after patches have been made publicly available by Google. Additionally, to what extent is the real-world impact in terms of exploitability and loss of elements of the confidentiality, integrity, and availability (CIA) triad from vulnerabilities which have gone unmitigated by vendor implementations of the Android OS in current releases, ranging from versions 6.x through 11.x?

Utilizing the research methodology of quantitative analysis is to be followed by grounded theory. The grounded theory process is explained as a series of events or interactions which occur over a period of time (Creswell, 2005). The correlation of CVE across multiple Android OS versions over a predefined time period fits the definition by Creswell (2005) for a basis to perform the study.

According to Salkind (2003), historical research, also known as historiography, seeks to answer the nature of past events. The study to be performed may not fit the research method of histography, as the vulnerabilities, although exposed at a time in the past, are always present in a configuration without proper mitigations. It is surmised vulnerabilities remain existent in past, present, and future releases of OSs which may be due to the complexity of a vulnerability, negligence of developers, or other factors.

Descriptive research is defined by Salkind (2003) as having the purpose "to describe the current state of affairs at the time of the study" As descriptive research attempts to understand current events (Salkind, 2003) and the study will include past and present data, the descriptive methodology will not be a selected approach. The scope of time the study is, in essence, a disqualifier for selecting descriptive research.

The correlational research methodology was selected as it measures the strength of a relationship between two variables, known as the correlational coefficient (Salkind, 2003). Salkind (2003) explains correlational research may result in the understanding of the commonality or predictability of an outcome with specified pieces of information (Salkind, 2003). The correlational research method will allow the study proposed the desired result to analyze the trending of vulnerabilities and the predictability of future vulnerabilities.

Creswell (2005) states the case study methodology focuses on individuality rather than grouping (Creswell, 2005). The proposed study seeks to analyze many vulnerabilities and a selected group of OSs. Due to the nature of the study's scope, the case study methodology will sufficiently handle the data subset to be analyzed, such as limited vulnerabilities and OS versions.

In summary, grounded theory with a correlational and case study methodology is elected to be the research method of choice. As stated by Creswell (2005), grounded theory allows for a broad theory to be generated when no current theories answer the problem to be studied (Creswell, 2005). The open nature of studying CVEs over a time period for previously unknown trends will seek to generate a new broad theory, which may result in future work.

Conceptual Framework

The Android OS developed by Google is the most prolific OS across mobile devices. Other vendors support their own proprietary OSs, but Android's open-source nature allows the OS to be widely adopted by various hardware manufacturers. As seen with the previously most popular desktop OS, Microsoft Windows is known to have a

plethora of publicly available vulnerabilities and compiled exploits. This same issue plagues Android due to its wide availability, open-source nature, connectivity options, and ease of application deployability.

The first issue to be explored is the availability of the Android OS. The open architecture allows vendors to compile the OS for their platform of choice. Under Google's own brand of hardware, created by third-party vendors specifically to Google's specifications, the Nexus brand of devices receives regular updates addressing known vulnerabilities under the tech goliath's patch management cycle (Thomas et al., 2015). Unfortunately, these updates do not get deployed to all vendor devices.

The work of Ratazzi (2016) states that mobile devices released by other vendors, such as Amazon, LG, or HTC, are restricted by multiple issues, limiting their ability to receive the hotfixes. First, regressions testing must be performed by the vendor to verify the compatibility with their release of the Android OS version. Additionally, vendor customizations may create new vulnerabilities not applicable to other versions of the same revision (Wu, 2015). Second, other providers are limited by the mechanisms to deliver updates to the devices. WiFi may be an option, but updates are often limited or delayed by a service provider, such as cellular carriers (Ratazzi, 2016). Third, vendors do not have the desire to keep ageing devices updated with the latest releases, as the next wave of devices is always right around the corner. The consumer's desire to continually upgrade to the latest technology leaves the vendors with no option but to move forward on only the most recent platforms, lest they take a hit to their profit on the bottom line.

As stated by Thomas et al. (2015), Google's line of Nexus devices have the reputation of being the most updated and supported. Their reputation is primarily due to

the vendor's directly supporting their software on their selected hardware and adopting a long-term supported device list. Vendors such as BlackBerry and LG have chosen their own of band patch cycle, often beating Google at patching monthly vulnerabilities. It is stated by BlackBerry Limited (2016) that BlackBerry maintains a group of security experts to patch and release updates to their Android based handhelds. The limited number of devices that keep current with security updates leaves organizations with the requirement of selecting only homogeneous Android based mobile devices to be used in their network environment, limiting the availability of a bring your own device (BYOD) policy.

Instead of selecting Android based devices that are regularly maintained by the hardware vendor, some organizations elect to deploy a mobile device manager (MDM) solution as part of their BYOD solution. A framework developed by Zahadat (2016) inserts patch management into policy to be adopted by an organization. These MDM platforms allow patch management to be supported by the organization, therefore, enabling a timely patch management strategy. Additionally, recent research has led to the development of custom patch management solutions.

Previously, root level access was required to deploy patches to devices. As seen in the Linux architecture that Android is based on, root allows administrative privileges to the underlying OS. Vendors maintain this level of administrative access to their devices; however, custom solutions may not have the required privileged-level access. Not all devices have the option of obtaining root access, and the devices that do usually result in an un-warrantied status after root is achieved. One solution developed by the research of Mulliner et al. (2013), PatchDroid, allows for patching of Android devices without the

requirement for root access. The technology utilized allows for the injection of code to mitigate known vulnerabilities without the support of the OS or hardware vendor. Direct patching of the underlying OS is not required, as hotfixes are available at the application level.

The next aspect of security-related issues with the Android OS is with applications. As cited by Zhou, Wang, Zhou, and Jiang (2012), the Google Play Marketplace had surpassed 200,000 apps by 2011. Out of a sampling of apps, the custom solution, DroidRanger, detected approximately 2% of apps contained some form of malware infection. It is also cited that alternative app stores, such as one provided by Amazon, had a much higher rate of infection by malicious software, potentially due to less stringent controls (Zhou et al., 2012). Furthering the issue, once these malicious packages are reported or detected at the marketplace level, no restrictions are preventing other open marketplaces from distributing the infected applications. Losses from this malicious software epidemic can result in various security issues, from breaches in data privacy to remote control of the device.

Unfortunately, the implementation of encryption at the app level has the capability of detection prevention. As stated by Suarez-Tangil, Tapiador, and Peris-Lopez (2014), utilizing steganography, apps can hide their malicious components under a hardened layer of encryption. Code obfuscated by encryption can prevent custom tools from detecting the malware running on a device resident in memory or prior to being uploaded to an app store. Additionally, it is stated that apps may not carry a payload before they are downloaded from an app store but, in fact, have the ability to contact a remote server and hook into a remote malicious payload that executes upon or after

launch (Suarez-Tangil et al., 2014). Other methods can be implemented to detect these types of hooking attacks, such as analyzing the runtime.

Methods to identify and prevent malicious apps have been previously limited to blacklisting, as stated by Sato et al. (2013). Their research led to the development of a new concept of detection that took the form of DroidMat. DroidMat is capable of analyzing API calls through manifest files with a sufficient level of success (Sato et al., 2013). New methods of detection need to be continually pursued in an effort to stay ahead of the increasingly advanced malware.

Another conventional detection technique is utilizing anomaly-based detection. Banuri et al. (2012) proposes an open architecture application which uses a framework to augment anomaly detection. Similar to other models, this utility analyzes the application's runtime to define the potential risk of a running app. Additionally, the utility reviews the access permission requested and utilized by the apps being analyzed to reduce the number of false positives (Banuri et al., 2012).

It is suggested by Jeon, Micinski, Vaughan, Fogel, Reddy, Foster, and Millstein (2012), that enabling fine-grained permissions within the Android OS may prevent some forms of malicious attack. It is observed that many apps do not elect to maintain the standard procedure of least privilege. The malicious apps can take advantage of the unaware user by requesting access to unneeded resources. The solution developed is known as RefineDroid, which can create the new level of granularity without requiring root access or recompile of code (Jeon et al., 2012). Enabling fine-grained permissions within the Android OS may be capable of preventing or blocking the aforementioned hooking techniques of malicious apps. Apart from these custom approaches, vendors

such as Samsung and BlackBerry have taken steps to further secure the Android operating system with their own security-based customizations.

According to BlackBerry Limited (2016) and Samsung Research America (2016), features enabled by the two vendors, Samsung and BlackBerry, take additional steps to secure the underlying OS. Each has marketed their solutions under the brands Knox and DTEK, respectively. The standard approach utilized by both vendors leverages certificate-based authentication and verification. The architecture allows hardcoded certificates that are device unique to be deployed during the manufacturing process. These certificates enable app verification, protected bootloaders, and trusted platforms to be restricted to the devices (BlackBerry Limited, 2016; Samsung Research America, 2016).

The third and last issue to be discussed is connectivity options available to mobile devices. As stated by Wibowo and Ali (2016), the accessibility of Android allows users to access many productivity features including online services, file sharing, and peripheral connectivity. The three most common forms of wireless communication include WiFi, cellular, and Bluetooth connectivity options. The productivity options are obvious, but WiFi and cellular connectivity allow vendors to push and provide downloadable security patches and hotfixes as well as updates to applications. The downfall to achieving this accessibility is potential attack vectors introduced by their inherent interoperability. The Cabir malware is stated to have been one of the earliest worms to propagate between mobile devices and exploit a vulnerability in Bluetooth technology (Wibowo & Ali, 2016). Vulnerabilities in text and multimedia-based messaging is, achievable through cellular communications, is also cited as having known