An Empirical Analysis of Contemporary Android Mobile Vulnerability Market

Keman Huang[1], Jia Zhang[2], Wei Tan[3], Zhiyong Feng[1]
[1]School of Computer Science and Technology, Tianjin University, Tianjin 300072, China
[2]Carnegie Mellon University Silicon Valley, Moffett Field, CA 94035, USA
[3]IBM Thomas J. Watson Research Center Yorktown Heights, NY 10598, USA
keman.huang@tju.edu.cn, jia.zhang@sv.cmu.edu, wtan@us.ibm.com, zyfeng@tju.edu.cn

Abstract— **The increasing popularity and rapid growth of the mobile ecosystem have been changing people's daily life. The vulnerabilities of mobiles resulting from the inherent vulnerable characteristic of software products, however, can be exploited, causing substantial economic loss or privacy leakage. This paper introduces an information metadata model, comprising a life cycle model and a heterogeneous network model, to investigate the evolving patterns of vulnerabilities in the current Android Mobile Vulnerability Market (AMVM). The test bed collects data from a variety of vulnerability datasets, comprising 19,711 vulnerable records. An empirical study is conducted over the test bed to trace the evolution process of vulnerabilities in the AMVM. The proposed network analysis method has opened a new way of studying mobile vulnerability market, in order to improve the security situation in the Android ecosystem.**

Keywords- *Android Mobile Vulnerability Market, Vulnerability life cycle, Heterogeneous Network, Evolving Pattern, Patching Behavior*

I. INTRODUCTION

The increasing popularity and rapid growth of the mobile ecosystem is changing the way that people communicate, work and socialize with each other [1]. Users nowadays are employing the mobile ecosystem to perform a variety of tasks, such as accessing social networks and running business operations. The revenue from the end users is significantly increasing, from barely existed to an expected value of $25.2 billion in 2017.

Despite its extensive adoption, the mobile ecosystem carries potential vulnerabilities that may be exploited by attackers to undermine systems and cause substantial economic loss or private information leakage [3]. As a matter of fact, the increasing potential benefit from the vulnerability exploitation in the mobile ecosystem has attracted significant attention from the black market [4]. Such vulnerabilities may not be avoidable since they are inherent from the vulnerable characteristic of software products [2] in the mobile ecosystem, ranging from the platform software supporting the operation system to third-party applications running on the platforms. Therefore, the security of the mobile ecosystem, especially tackling with vulnerabilities of the ecosystem, is becoming a crucial issue for the current business field [5].

Uncovering the patterns of the vulnerabilities is recognized as an effective approach to improve the security of the software ecosystem [6]. Many agencies have been built to collect and disclose detected software vulnerabilities, including the Computer Emergency Response Team (CERT) [1], Security Focus (BID) [2], the National Vulnerability Database (NVD) [3], and the Open Sourced Vulnerability Database (OSVDB)[4]. In addition, exploratory analysis of vulnerability life cycles can help vendors reduce potential vulnerabilities during their software development processes, suggest consumers to select safer services, and assist ecosystem managers in developing security policies to more effectively handle future attacks and threats, and identify the potential zero-day vulnerabilities [7]. Therefore, understanding the vulnerability market is important for both industry and academic fields [2-4, 7-10].

Current literature takes the software industry as a whole and only considers the vulnerabilities of software products developed for Windows (such as Internet Explorer, Firefox, and Adobe). One core reason is because the Microsoft Windows platform has been the main target of cyber attacks over the past decade. The vulnerabilities of mobile systems are oftentimes neglected by current research, causing an apparent literature gap of vulnerability research. The evolving patterns in the mobile ecosystem are quite different from those in the general software industry. Typical patterns include the emerging scale-free property, the vulnerable type transfer, and the severity evolution. Furthermore, current vulnerability study is based on single vulnerability data sets, which cannot depict an overall picture of the market.

This research, thus, aims to investigate the evolution patterns in the mobile vulnerability market for the ecosystem's security improvement. As Android continuously increases its popularity and market share in the mobile ecosystem [11], this paper focuses on the vulnerabilities which are relevant to the android ecosystem, named as the *Android Mobile Vulnerability Market (AMVM)*. It should be noted that our method can be apply to other mobile system market study. The research proposes an information metadata model, comprising a vulnerability life cycle model based on the vulnerability state transition and a heterogeneous network model based on the relationships among vulnerabilities, products and vendors. To systematically study the evolution process of the AMVM, the research collects the android vulnerability data since 2008 from multiple agencies, including the NVD, OSVDB, CERT, BID and vendor websites. Based on the collected datasets, network analysis is performed and suggestions for security improvement are discussed.

[1] http://www.kb.cert.org/vuls/
[2] http://www.securityfocus.com/bid
[3] https://nvd.nist.gov/
[4] http://osvdb.org/

978-1-4673-7284-8/15 $31.00 © 2015 IEEE
DOI 10.1109/MobServ.2015.34

The main contributions for this paper are two-folds:

- First, an information metadata model is proposed for mobile vulnerability market analysis, consisting of a life cycle model and a heterogeneous network model.
- Second, the systematical analysis over the network constructed, using our proposed information metadata model, from the aggregated vulnerability datasets for evolution of vulnerabilities for Android mobile vulnerability market has yielded suggestions for the security improvement in the mobile ecosystem.

The remainder of the paper is organized as follows. Section 2 presents the information metadata model oriented for the android mobile vulnerability market. Section 3 introduces the aggregated data set from different vulnerability agencies. Section 4 reports the vulnerability patterns analysis results. Section 5 discusses the related work and section 6 draws conclusions.

II. Information Metadata Model

In this section, we introduce an information metadata model tailored for mobile vulnerability market analysis. It comprises two components: a life cycle model and a heterogeneous network model.

A. *Life Cycle Model*

Figure 1 illustrates a life cycle of android mobile vulnerability. (1) A bug in the software products that is released and deployed in the mobile ecosystem will introduce a potential vulnerability in the market. (2) The vulnerability may be discovered by the vulnerability discovers, who can be the software vendors, the security researchers, or the underground discovers such as the hackers from the black market. (3) If the vulnerability is discovered by the software vendor, it can be fixed by the vendor before it becomes publicly known. (4) If the vulnerability is discovered by attackers, the exploit will be created and used to conduct attacks against the target. (5) The discovered vulnerability is publicly disclosed either by the vendor or on the public forums and mailing lists. (6) The disclosed vulnerability can attract attentions from the black market and results in exploitation activities. (7) On the other hand, the disclosed vulnerability will force the vendor to release the patching as soon as possible. (8) The identified exploitation of the vulnerability also provides knowledge for the patching. (9) The patching is released for the specific vulnerability while it may also bring in new bugs.

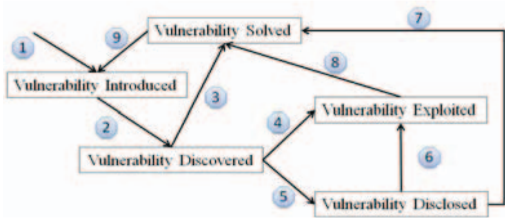


Figure 1. Life cycle model for the android mobile vulnerability.

The life cycle of vulnerability can be divided into five consecutive phases:

- ***Vulnerability Introduced (*** t_r ***)***: the time when the vulnerability is introduced into the ecosystem;
- ***Vulnerability Discovered (*** t_{dis} ***)***: the time when the vulnerability is discovered;
- ***Vulnerability Disclosed (*** t_d ***)***: the vulnerability is collected by the platform and disclosed to the public.
- ***Vulnerability Exploited (*** t_e ***)***: the exploit of the vulnerability is available.
- ***Vulnerability Solved (*** t_s ***)***: the patching for the vulnerability is released by the vendor.

Obviously, $t_r < t_{dis} < t_d, t_e, t_s$. Note that the sequence for the t_d, t_e, t_s is alterable. Hence, the following three durations may exist:

- ***Disclosure-Patching Duration*** ($t_{ds} = t_s - t_d$): the duration between the disclosure date and the solution date. It represents how quickly the vendors respond to the vulnerability and neutralize the threats. The larger the t_{ds} is, the longer the software product is in risks.
- ***Disclosure-Exploit Duration*** ($t_{de} = t_e - t_d$): the duration between the disclosure date and the exploit date. It represents how quickly the hackers exploit the vulnerability. The smaller the t_{de} is, the more attractive the product is for the hackers. Note that if $t_{de} \le 0$, the exploit is available before the vulnerability is publicly known. This is the well-known *zero-day vulnerability*.
- ***Patching-Exploit Duration*** ($t_{se} = t_e - t_s$): the duration between the solution date and the exploit date. It represents the length of competition between the vendors and the hackers.

B. *Heterogeneous Network*

Two relations are important among the software products, vulnerabilities and the vendors, the three core components in the AMVM: 1) the vendors release software products into the mobile ecosystem; 2) the potential vulnerabilities in the product are introduced into the mobile ecosystem. We formally define a heterogeneous network model for AMVM as follows, illustrated in Figure 2:

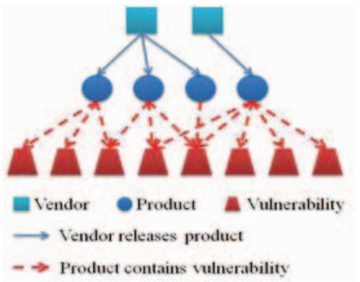


Figure 2. vu**L**nerability-**P**roduct- **V**endor (LPV) Heterogeneous Network.

Definition 1 ***(vuLnerability-Product-Vendor Network, LPV)*** $LPV = \{L, P, V, R_{lp}, R_{pv}\}$ where $L = \{l_i\}$ refers to the set of the disclosed vulnerabilities; $P = \{p_j\}$ refers to the vulnerable products; $V = \{v_k\}$ refers to the vendors; $R_{lp} = \{< l_i, p_j >\}$ represents the relations among the vulnerabilities and products; and $R_{pv} = \{< p_j, v_k >\}$ represents the relations among the products and the vendors.

1) Product:

Regarding the software products in the android mobile ecosystem, some are platforms supporting the Android operation system and we name them as *native products*; the others are developed by third party vendors to offer value-added services for consumers and we name them as *third-party products*. For the native products, our research follows the android security architecture[5] and classifies them into the following four types:

- *Kernel (NK)*: the software relevant to the Linux kernel for the android operation system.
- *Library (NL)*: the software provides basic functionalities for the system and the runtime environment.
- *Application Framework (NF)*: the software supports the direct communication with the application.
- *Application (NA)*: the software that the consumers consume to fulfill their requirements.

Similarly, for the third-party products, we can also classify them into the following three layers:

- *System (TS)*: the software products which are used to support the system running on the mobile devices.
- *Library (TL)*: the applications that developed by third-party vendors and offer common functionalities.
- *Application (TA)*: the third-party applications for specific functionalities.

Therefore, the research defines the products as:

$$p_j =< pn_j, pt_j, pv_j > \tag{1}$$

where pn_j refers to the name of the product, pv_j is the released version and $pt_j = \{NK, NL, NF, NA, TS, TL, TA\}$ refers to different types of the products.

2) Vulnerability:

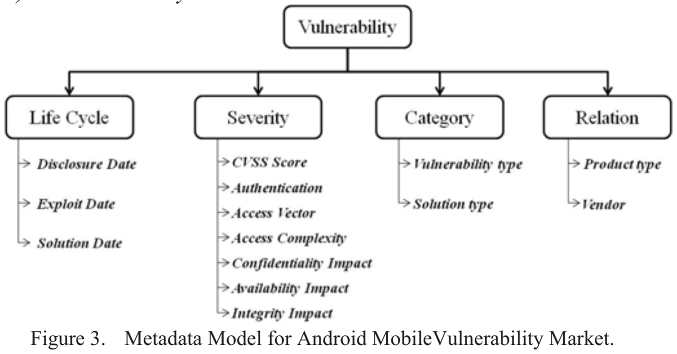


Figure 3. Metadata Model for Android MobileVulnerability Market.

To study the evolution pattern in AMVM, we consider how each vulnerability transfers between different lifecycle stages, how serious the vulnerability is, what type the vulnerability belongs to, and how vulnerability is fixed. In more detail, we build a metadata model for the Android Mobile Vulnerability Market as shown in Figure 3. This model builds a measurement gauge for our network analysis to be described in Section III and IV.

Therefore, we defines the vulnerability as follow:

$$v_i =< vn_i, vt_i, vl_i, vf_i, vs_i > \tag{2}$$

where vn_i is the name of the vulnerability; vt_i is the vulnerability's type; vl_i is the lifecycle of the vulnerability; vf_i refers to the vulnerability's solutions and vs_i refers to the severity of the vulnerability.

As the vulnerability discovered date is often unavailable to the public and the product released date t_r is not straightly related to the vulnerability, the research only considers the disclosure date t_d, the exploit date t_e and the solved date t_s.

$$vl =< t_d, t_e, t_s > \tag{3}$$

To evaluate the vulnerability's risk severity, the study employs the widely used Common Vulnerability Scoring System (CVSS)[6], which is a real number between 0 and 10. The larger the score is, the higher the severity the vulnerability is. Specifically, it reflects the following six factors:

- ***Access Vector*** ***(*** $AV = \{Local, Adjacent\ Network, Network\}$ ***)***: indicates the way to exploit the vulnerability.
- ***Access Complexity*** ***(*** $AC = \{Low, Medium, High\}$ ***):*** indicates the complexity level to attack the software product through the vulnerability.
- ***Authentication*** ***(*** $AU = \{None, Single, Multiple\}$ ***)*** indicates the number of times the hacker must authenticate to a target in order to exploit the vulnerability.
- ***Confidentiality Impac***t ($CI = \{None, Partial, Complete\}$) indicates the impact on confidentiality of a successfully exploited vulnerability.
- ***Integrity Impact*** ***(*** $II = \{None, Partial, Complete\}$ ***)*** indicates the impact on integrity of a successful exploitation.
- ***Availability Impact*** ***(*** $AI = \{None, Partial, Complete\}$ ***)*** indicates the impact on the availability of a successful exploitation.

Hence, the severity of the vulnerability can be defined as:

$$vs =< cvss, AV, AC, AU, CI, II, AI > \tag{4}$$

III. Data Harvest & Overview

A. Data Harvest

The vulnerability disclosure can effectively promote the vendors to improve the software security. Many agencies such as NVD, OSVDB, CERT-VN, BID and SCP have been built to collect vulnerability information.

As this research focuses on the Android mobile vulnerability market, we first collect the relevant records

[5] https://source.android.com/devices/tech/security/index.html, Last accessed: 2015/4/22

[6] https://www.first.org/cvss/cvss-guide

from NVD using "android" as the query content. For each vulnerability, we parse the website to extract information including the CVE-ID, the description, the CVSS severity, the effected products, the vulnerability type, the effected products as well as the reference resources. Then we collect the data from OSVDB to get the CVE-ID, the time line information such as the disclosed date, the patching date as well as the exploit date. Furthermore, we collect the Android-relevant records from CERT-VN, and extract information such as the CVE-ID, vendor information, disclosed date, discovered date and patching date. We also get the vulnerable product's types from the SCP website[7]. Additionally, we obtain the patching information from the software vendors and the specific countermeasures such as the security focus (BID), the Google MISC etc. to enrich the data set. Finally, we aggregate all the fetched data by the CVE-ID to construct the data set for our study.

B. *Overview*

Based on the collected data, we constructed the LPV heterogeneous network using the information metadata model described in Section II. Table I reports the basic statistics and Figure 3 is the overview generated by Cytoscape [12].

TABLE I. BASIC STATISTICS OF COLLECTED DATASET

#CVE Records	1943
#Vulnerable Product Lines*	1658
#Vulnerable Products*	6449
#Vendors	1304
#Vulnerability Records	19711
#Components	1277
Average Vendor Degree	4.946
Average Product Vulnerability	3.056
Average Vulnerability Affected products	10.145

*Product lines refer to all the products with the same product name and vendor name but different product versions. Software products with different versions are considered as different products.

1) *Component Analysis:*

As shown in Figure 4, we found that most components (86.6%=1109/1277) are atomic modules consisting of one vendor, one product and one vulnerability. There are products lines in the third-party application level. Only four connected sub-components contain more than 100 nodes.

For the component-a, the vendors are "Apple," "Linux," "Google," "Adobe," "Android," "Microsoft," "Openssl," "Redhat" and so on. These vendors are core in the Android vulnerability market and their products are the native products. In addition, the vendor "Google" has the largest degree and betweeness in the component. This finding is reasonable because Google is the centrality in the ecosystem.

For the component-b, it contains only one vendor "Mozilla" and three product lines "Thunderbird," "Firefox" and "Seamonkey." They are all popular third-party applications in the Android ecosystem.

For the component-c, it contains three different vendors "*Qualcomm,*" "*Codeaurora*" and "*Motorola.*" "*Qualcomm*" and "*Motorola*" are well-known leaders in the mobile technologies. The vendor "*Codeaurora*" is hosted by the Linux Foundation. It serves the mobile wireless industry and supports upstream projects such as the Linux Kernel and the Android system. These products belong to the native kernel or the third-party system.

For the component-d, it contains only one vendor "*Kernel*" and one vulnerability "*CVE-2009-1185*" relevant to all the different versions of the product line "*UDEV.*" They are all native-kernel products.

The vulnerabilities in different components are obviously related to the different layers of the Android ecosystem.

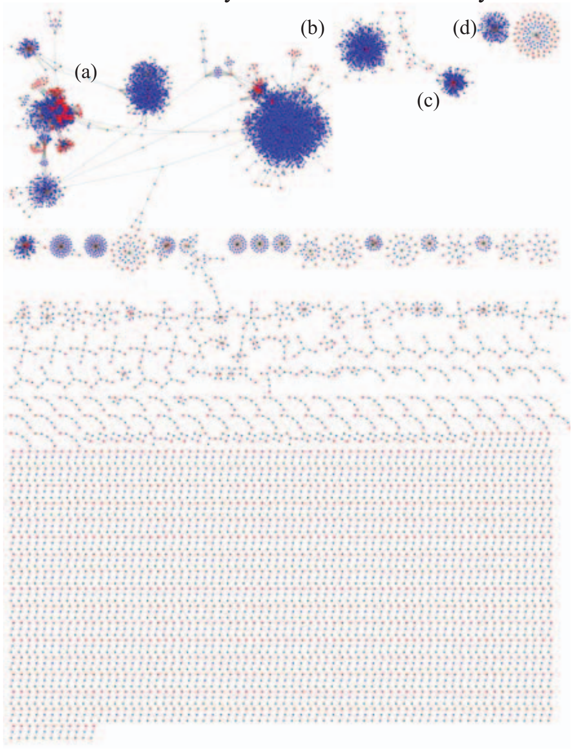


Figure 4. Overview of LPV Network. Red ellipse represents the vulnerability; blue diamond is the product; black triangle is the vendor.

2) *Modular Analysis*

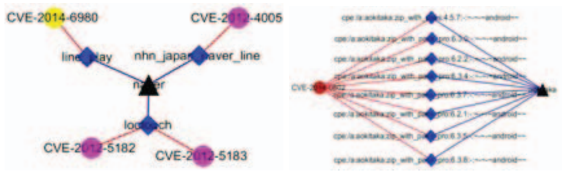
Based on Figure 4, we can observe two relation modular between vulnerability, product and vendor:

- ***Star Modular***: Each product released by a vendor owns one or more unique vulnerabilities.
- ***Spindle Modular***: All the products released by the same vendor share the same vulnerabilities.

For the star modular, the products released by the same vendor have different vulnerabilities. Each vulnerability requires a specific patching. However, for the spindle modular, the products share the same vulnerability. It can be seen from Figure 4 that most of the vendor-product-vulnerability relations for the main sub-components belong

[7] http://android.scap.org.cn/index.html

to the spindle modular. One reason is that these products are developed based on the same foundational framework/ platform. Therefore, if the product is released by the same vendor based on the same vulnerable framework/platform, it has a large possibility to inherit the same vulnerability. Therefore, we can employ this feature to predict the potential vulnerability for the new released products.



(a) Star Modular **(b) Spindle Modular**

Figure 5. Modular in the Vulnerability-Product-Vendor Network.

3) *Degree Distribution Analysis:*

We define the following three degree distributions for the generated network:

- Vendor degree distribution: it refers to how many vendors have the given number of vulnerable products.
- Product degree distribution: it refers to how many products are with the given number of vulnerabilities.
- Vulnerability degree distribution: it refers to how many vulnerabilities are within the given number of products.

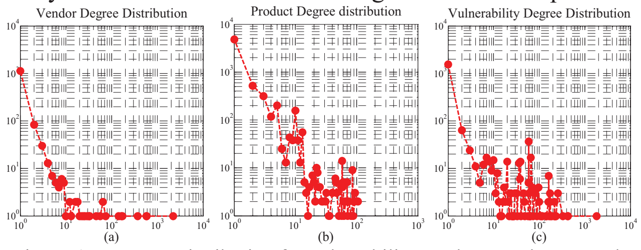


Figure 6. Degree Distribution for vulnerability-product-vendor network.

As shown in Figure 6, the vendor degree distribution fits the power-law distribution but with a long tail; the product degree distribution is similar to the truncated power-law; while the vulnerability degree distribution cannot fit the scale free property. There is, however, a significant power-law phenomenon when the whole vulnerability market is taken into account [13]. Therefore, it can conclude that the Android mobile vulnerability market is different from that of the whole industry [2, 13]. It is necessary to analyze its patterns specifically, which is one motivation of this research.

IV. General Trend Analysis

This section examines our study of the evolution of the vulnerabilities in the Android mobile vulnerability market.

A. *Vulnerability Disclosure Trend*

To evaluate the evolution of the disclosed vulnerability, we get the disclosure date of each CVE vulnerability, and then calculate the disclosed vulnerabilities each year since 2008. Furthermore, we fetch all the vulnerabilities disclosed in NVD and then calculate the percentage of Android mobile vulnerabilities per year. As shown in Figure 7, the disclosed Android-relevant vulnerabilities increased from 2008 to 2012. After a slight reduction in 2013, there was an outbreak of vulnerabilities in 2014. The outbreak owed to the fact that multiple Android software products failed to properly validate SSL certificates provided by HTTPS connections. The 10 tested library products are all vulnerable and 23.2% (=320/1379) of the tested Android applications are due to the vulnerability of the library.

Meanwhile, the percentage of the Android vulnerabilities has been increasing since 2008. Therefore, it can conclude that the security situation in the Android ecosystem is worsening over year. One possible reason is that the number of applications released for consumption is increasing with the rapid development of mobile Internet, but most of the developers are not professional enough [14]. Additionally, the Android ecosystem is creating tremendous revenues that it attracts increasing attentions from the security community.

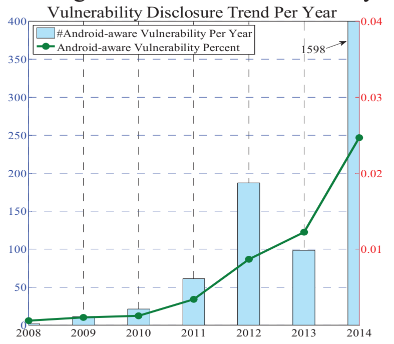


Figure 7. Vulnerability Disclosure Trend Per Year.

B. *Vulnerability Type Trend*

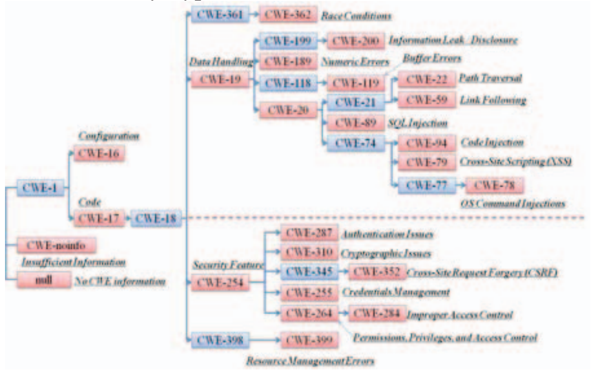


Figure 8. Vulnerability Type Taxonomy for android-aware vulnerability.

For the Android vulnerabilities, we identify 25 different types of vulnerabilities based on the CWE. Then considering these CWE types as the leaf nodes, we construct the sub-tree structure based on the hierarchical relations among CWEs. Finally, we can get the vulnerability taxonomy, as shown in Figure 8. The boxes with red color refer to the vulnerability types for the Android-aware vulnerabilities and the other refers to the relevant CWEs generated based on their relations. The vulnerabilities above the dash line are from the weaknesses found in software products' functionality such as data processing or time and state management. Here we name them as ***functional vulnerability*** for describe convenience. The vulnerabilities below the dash line are not from the functionality but from the improper management of the codes or the security features, and we name them as ***management vulnerability***.

We also calculate the percentage of the vulnerabilities with different CWEs in each month, to explore their evolution patterns over time (as shown in Figure 9). The timeline can be separated into three periods:

- *Period A (2008/1~2010/8)*: most vulnerabilities are the functional vulnerabilities. This is because the Android-aware ecosystem was still in its early stage and the vulnerability market was focusing on the functionality.
- *Period B (2010/9~2013/12)*: the management vulnerabilities begin to appear, especially the "CWE264" (*Permissions, Privileges, and Access Control*). The functional vulnerabilities, however, still dominate.
- *Period C (2014/1~2015/1)*: the management vulnerabilities become the mainstream, especially the "*CWE264*" and "*CWE310*." Both belong to the security features.

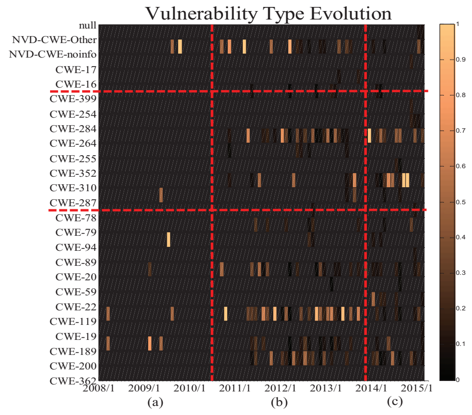


Figure 9. Vulnerability Type Evolution over time.

Therefore, it is reasonable to conclude that the types in the Android vulnerability market are transferring from the functional vulnerabilities to the management vulnerabilities.

C. Vulnerability Severity Trend

To analyze the vulnerability severity pattern in the market, we have studied the following two cases:

- ***ALL Market***: all the vulnerabilities in the software industry are taken into account.
- ***Android Market***: only the 1,943 Android relevant vulnerabilities are used.

We have identified three corresponding categories based on the CVSS score:

- High: $CVSS \geq 7$
- Medium: $4.0 \leq CVSS < 7$
- Low: $CVSS < 4$

Figure 10 shows the comparison of the severity trend between the whole market and the Android market from 2008/1 to 2014/8. Note that we did not consider the vulnerabilities from 2014/9 to 2015/1, because the outbreak happened in 2014/9 and 2014/10, which made the result meaningless.

1) CVSS score evolution:

As shown in Figure 10(a), the average risk score for the Android market is higher than that of the whole market for most of the time. The average risk score for the Android market is higher than 7, which lies in the high risk range. The average risk score for the whole market is between 6 and 7, which is in the medium area and it is decreasing over time. Furthermore, as shown in Figure 10 (b), the percentage of the high risk vulnerabilities of the Android market is substantially higher than those of the whole market most of the time. Hence, it can be concluded that *the security situation for the Android market is worse than that of the whole market*.

2) Exploitability Analysis:

The exploitability subscore of the CVSS is an indicator of "likelihood to be exploited" of a vulnerability [15] in terms of the access vector, access complexity and the authentication. As shown in Figure 10(c), the remotely exploitable vulnerabilities for the whole market are around 90% with a slight decrease over time. The Android market vulnerabilities that are exploited through network, however, are increasing over time and more than 90% vulnerabilities are remotely exploitable since January 2012. For the access complexity, the low complexity vulnerabilities for the whole market are decreasing, indicating that hackers have to use more sophisticated techniques to exploit the vulnerabilities. On the contrary, the percentage of the low complexity vulnerabilities in the Android market is increasing. Furthermore, the percentage of the exploitable vulnerabilities without authentication for the whole market is decreasing. Almost all of the vulnerabilities in the Android market, however, can be exploited without authentication.

Hence, it can be concluded that *the vulnerabilities in the Android market are more exploitable and easier to exploit over time than those in the whole market. The most effective methodology to secure the Android market is to strengthen the authentication mechanism.*

3) Impact Analysis:

The impact subscore of the CVSS evaluates the impact to the system if the vulnerability is successfully exploited by hackers, including the confidentiality impact referring to the scope that attackers can access, the integrity impact referring to the scope of what attackers can affect, and the availability impact referring to the affection to the accessibility of the information resources. For the vulnerabilities in the Android market, most of them own a complete confidentiality impact, a complete integrity impact and the availability impact. This is because that almost all the software products require all the access authorities whatever the authorities are needed or not. The percentage of the vulnerabilities with a complete impact is higher than that of the whole market. Hence, the impact of the vulnerability exploitation in the Android market is worse than that of the whole market.

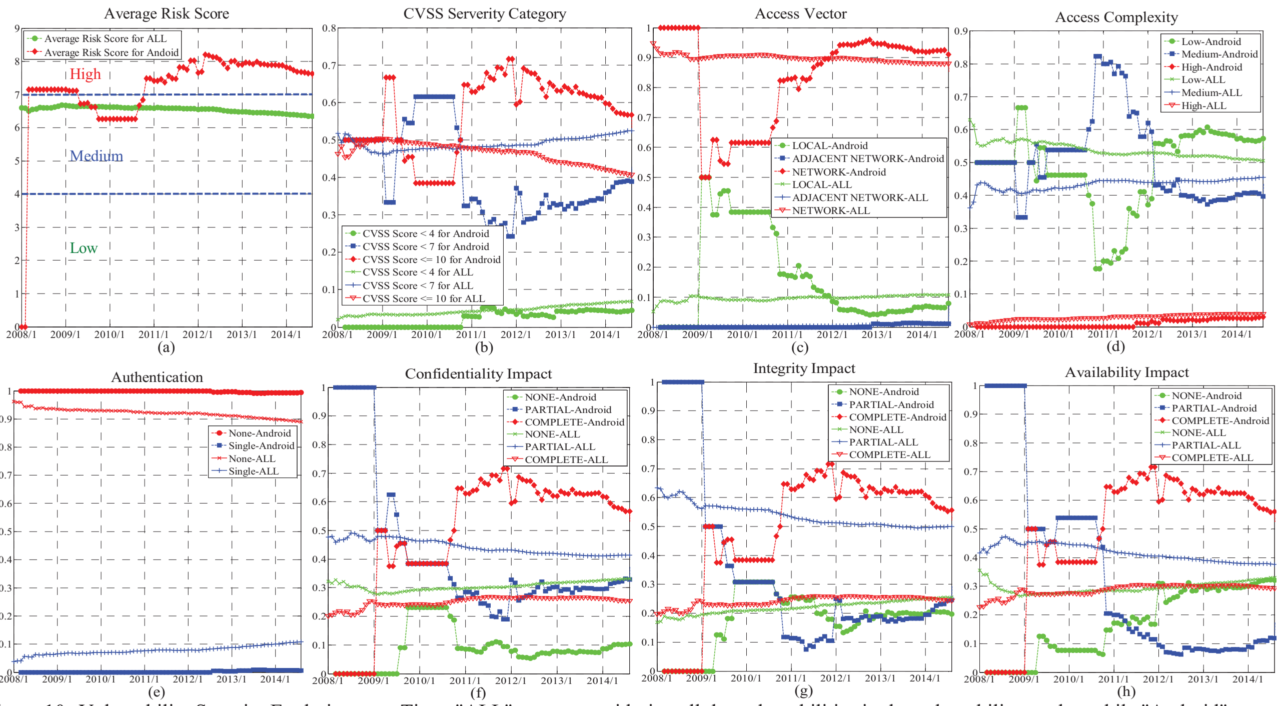


Figure 10. Vulnerability Severity Evolution over Time. "ALL" means considering all the vulnerabilities in the vulnerability market while "Android" means only the Android-aware vulnerabilities are taken into account. For (b)~(h), the two lines in the same color refers to the same category in the two cases. For the authentication, as the percent of the multiple authentications for both cases are nearly 0 that we don't draw them in the figure.

Therefore, the vulnerabilities in the Android market are more exploitable than in the whole market; and the exploitation impact is also more serious based on the analysis of the vulnerability CVSS. Furthermore, though the security environment in the whole market has been improving, the Android market has been deteriorating recently. The possible methodologies to secure the Android market are to strengthen the authentication mechanism, especially for the remotely assessment, and to limit the unnecessary requirements of access authorities for the software products.

D. *Vulnerable Product Type*

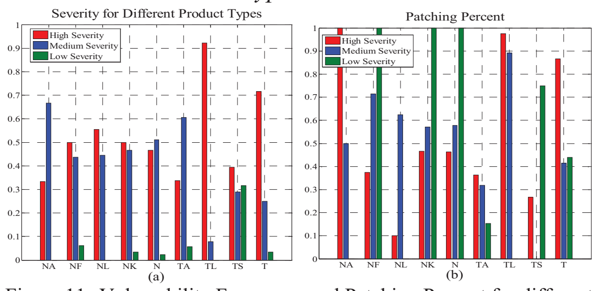


Figure 11. Vulnerability Frequency and Patching Percent for different product types with different severity level. "N" refers to all the native products including "NA","NF","NL","NK"; "S" refers to all the third-party products including "TA","TF","TS".

Without considering the outbreak vulnerabilities owed to Android SSL verification, we classify the vulnerabilities into different groups based on the vulnerable product's type. We then get the vulnerability number and patching percentage of different groups. As shown in Fig 11(a), the percentage of high severity risk in third-party products is larger than the one in native products. More than 90% of the vulnerabilities in TL are in high severity risks. The community in third-party, however, is more activated. 86.68% high risk vulnerabilities of the community in third-party are fixed, while only 46.34% high risk vulnerabilities for the native products are fixed. Additionally, patches are available for 73.92% of the third-party vulnerabilities, while only 53.41% of the native vulnerabilities are dealt with.

V. RELATED WORK

To support the development of vulnerability discovery models, some work has been conducted to understand the patterns of the vulnerability disclosures in software.

Frei etc. [16] examine how vulnerabilities are discovered, disclosed, exploited, and patched since 1995 till 2005. Telang and Wattal [17] investigate the relations between the disclosed vulnerability and the vendor's stock price. Dumitras studies the vulnerabilities patterns and identifies the potential zero-day vulnerabilities [7]. Based on the study of the vulnerability's patching and exploited, Shahzad etc. conclude that the response of vendors has been improving [2]. Algarni etc. [4] identify the whole software vulnerability market and study the motivation and methods of discoverers. Our previous work discusses the scale-free property in the software vulnerability market [13].

All these papers, however, consider the software vulnerability market as a whole and most of them only examine the popular software products such as Microsoft or

Adobe. The mobile ecosystem is somehow overlooked. Due to the differences between the mobile ecosystem and the general software products in terms of growing pattern, ecosystem structure and consumer's behavior patterns, the conclusions of these papers cannot be directly applied in the mobile ecosystem. Therefore, this paper focuses on the Android mobile vulnerability market to uncover the unique vulnerability patterns for the Android mobile ecosystem.

VI. Discussions & Conclusions

Android mobile ecosystem has been playing an important role in consumers' daily life. The security situation in the ecosystem plays an increasingly important role in the current business environment. In order to investigate the patterns of the Android mobile vulnerability to promote the ecosystem's security, we have introduced an information metadata model comprising two major components: a life cycle model based on the vulnerable status transaction and a heterogeneous network model which represents the relationships between vulnerabilities, products and vendors. We have presented a test bed of Android mobile vulnerability data collected from a variety of sources, including the NVD, OSVDB, VN-CERT, BID, SCP as well as vendors' websites. On the established test bed, we have conducted an empirical analysis to analyze the evolution patterns in the *Android mobile vulnerability market (AMVM)*. Our preliminary findings reveal that the AMVM is significantly different from the whole industry from four significant aspects:

- The Android vulnerabilities are increasing over year and the AMVM attracts more and more attentions from the community;
- The dominant type of vulnerability in AMVM is transferring from the functional vulnerability into the management vulnerability. Hence the developers need to take additional steps to enhance the security features such as *Permissions, Privileges, and Access Control;*
- The severity of vulnerabilities in AMVM is higher than that of the whole industry. The vulnerabilities in the Android market are also more exploitable than those of the whole market and the vulnerabilities in android market are becoming easier to exploit over time. The most effective methodology to secure the Android market is to strengthen the authentication mechanism, especially for the remote assessment, and to limit the unnecessary requirements of access authorities for the software products.
- Comparing with the native products, the vulnerabilities for the third-party products are worse. The third-party community, however, is more activated than the native one and the third-party community also has more available patches for the disclosed vulnerabilities.

Based on the observations discussed above, our future research intends to study the patching and systematically exploit patterns and behaviors in AMVM, to predict potential vulnerabilities for the new released products and to identify the unnecessary access authorities to improve the security of the ecosystem.

Acknowledgment

This work is supported by the National Natural Science Foundation of China grant 61070202, 61173155 and National High-Tech Research and Development Program of China grant 2013AA013204.

References:

[1] A. Ghose and S. P. Han, "Estimating demand for mobile applications in the new economy," *Management science: journal of the Institute for operations research and the management sciences,* vol. 60, pp. 1470-1488, 2014.

[2] M. Shahzad, M. Z. Shafiq and A. X. Liu, "A large scale exploratory analysis of software vulnerability life cycles," in *Proceedings of the 34th International Conference on Software Engineering*, Zurich, Switzerland, 2012, pp. 771-781.

[3] S. Ransbotham, S. Mitra and J. Ramsey, "Are markets for vulnerabilities effective?" MIS Quarterly, vol. 36, pp. 43-64, 2012.

[4] A. Algarni and Y. Malaiya, "Software vulnerability markets: Discoverers and buyers," International Journal of Computer, Information Science and Engineering, vol. 8, pp. 71-81, 2014.

[5] R. Anderson and T. Moore, "The economics of information security," Science, vol. 314, pp. 610-613, 2006.

[6] A. Arora, R. Krishnan, R. Telang, and Y. Yang, "An empirical analysis of software vendors' patch release behavior: impact of vulnerability disclosure," Information Systems Research, vol. 21, pp. 115-132, 2010.

[7] T. Dumitras, "Before we knew it: an empirical study of zero-day attacks in the real world," in Proc. of the 2012 ACM conf. on Computer and communications security, pp. 833--844, 2012.

[8] T. August and M. F. Niculescu, "The Influence of Software Process Maturity and Customer Error Reporting on Software Release and Pricing," Management Science, vol. 59, pp. 2702-2726, 2013.

[9] Y.S. Baker, R. Agrawal and S. Bhattacharya, "Analyzing security threats as reported by the United States Computer Emergency Readiness Team (US-CERT)," in Intelligence and Security Informatics (ISI), 2013 IEEE International Conference on, Seattle, WA, 2013, pp. 10-12.

[10] Y. Wu, H. Siy and R. Gandhi, "Empirical results on the study of software vulnerabilities (NIER track)," in Proceedings of the 33rd International Conference on Software Engineering, Waikiki, Honolulu, HI, USA, 2011, pp. 964-967.

[11] A. S. Yuksel, A. H. Zaim and M. A. Aydin, "A Comprehensive Analysis of Android Security and Proposed Solutions," International Journal of Computer Network and Information Security, vol. 6, pp. 9-20, 2014.

[12] M. E. Smoot, K. Ono, J. Ruscheinski, P. L. Wang, and T. Ideker, "Cytoscape 2.8: new features for data integration and network visualization," Bioinformatics, vol. 27, pp. 431-2, 2011-02-01 2011.

[13] K. Huang, Z. Feng, J. Li, and X. Li, "System thinking of the Software Vulnerability Market via Complex Network Theory," in 36th IEEE Symposium on Security and Privacy FAIRMONT, SAN JOSE, CA, 2015.

[14] I. J. Mojica, B. Adams, M. Nagappan, S. Dienst, T. Berger, and A. E. Hassan, "A large-scale empirical study on software reuse in mobile apps," Software, IEEE, vol. 31, pp. 78-86, 2014.

[15] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: learning to classify vulnerabilities and predict exploits," in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, Washington, DC, USA, 2010, pp. 105-114.

[16] S. Frei, M. May, U. Fiedler, and B. Plattner, "Large-scale vulnerability analysis," in Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense, Pisa, Italy, 2006, pp. 131-138.

[17] R. Telang and S. Wattal, "An empirical analysis of the impact of software vulnerability announcements on firm stock price," *Software Engineering, IEEE Transactions on,* vol. 33, pp. 544-557, 2007.