Shifting to Mobile: Network-Based Empirical Study of Mobile Vulnerability Market

Keman Huang, *Member, IEEE*, Jia Zhang, *Senior Member, IEEE*, Wei Tan, *Senior Member, IEEE*, and Zhiyong Feng, *Member, IEEE*

Abstract—With the increasing popularity and great economic benefit from vulnerability exploitation, it is important to study mobile vulnerability in the mobile ecosystem. Beyond the traditional technical solutions such as developing technologies to identify potential vulnerabilities, discover the widely available exploitations and protect consumers from attacks, constructing the vulnerability market, a marketplace for vulnerability discovery, disclosure and exploitation, has been considered as an effective approach. Therefore, understanding the mechanism of the vulnerability market for further optimizations is attracting attentions from both academia and industry. Since mobile ecosystem is playing an increasingly important role for the daily life, this paper aims to understand the evolution of the mobile vulnerability market in a data-driven approach, aiming to identify the important issues for further research. Specially, a five-layer heterogeneous network, consisting of the software vendors, products, public disclosed vulnerabilities, hunters, organizations and their relations, is established to formally represent the evolution of the mobile vulnerability market. Based on the data collected from a variety of agencies, including NVD, OSVDB, BID and vendor advisories, a comprehensive empirical analysis is reported, focusing on the growth of the mobile vulnerability market as well as the interactions between mobile and other PCs platforms. Finally, suggestions drawn from the observations, including security evaluation for code reused, data leaking protection and permission overuse identification, hunter's strategy and behavior understanding, information sharing and external workforce hiring, as well as cross-platform vulnerability digging are discussed for further security enhancement.

Index Terms—Mobile vulnerability market, heterogenous network, exploratory empirical study, sociotechnical system

1 Introduction

Information security has become increasingly important for the current business environment supported by the IT/OT system[7]. With the rapid growth of the mobile computing techniques, users nowadays are heavily relying on the mobile ecosystem to perform a variety of tasks, not only for fun but also for their daily life such as accessing social networks, running business operations, dealing with financial services, etc. [23], [57]. Globally, mobile digital media has recently surpassed desktop and other media [31]. Consequently, more and more valuable information assets are stored in the mobile ecosystem [53], [58]. However, due to the inherent vulnerable characteristic from the IT/OT system[45], the potential vulnerabilities for the mobile ecosystem may be discovered and exploited by attackers resulting into huge economic loss or private information leakage. As the golden rule saying *"the more widely a technology is used, the more likely it is to become target of hackers to make the potential security threats a reality"* [33], the increasing popularity and great economic benefit from vulnerability exploitation make the mobile ecosystem more and more attractive to hackers. The Symantecs 2015 annual threat report shows a significant increase of security threat in mobile ecosystem. More than one million malicious Apps, including 46 new families of Android malware, have been identified in 2014 alone. Therefore, the security in the mobile ecosystem is becoming a serious and crucial issue for both academia and industry.

Beyond the technical solutions such as developing technologies to identify potential vulnerabilities, to discover the widely available exploitations and to protect the consumers from attacks [24], [41], [49], [52], constructing an effective vulnerability market, *a marketplace for vulnerability discovery, disclosure, exploitation, sharing and patching* [3], has been recognized as an effective approach to solve the slow technology adoption problem [6], [17], [39]. Therefore, besides that the software vendors will publicly disclose vulnerability and patching information, many agencies have emerged to collect and share vulnerability information, such as 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] etc. Additionally,

- *K. Huang is with the School of Computer Science and Technology, Tianjin University, Tianjin 300072, China, and the Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. E-mail: keman.huang@tju.edu.cn, keman@mit.edu.*
- *J. Zhang is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Moffett Field, CA 94035. E-mail: jia.zhang@ sv.cmu.edu.*
- *W. Tan is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598. E-mail: wtan@us.ibm.com.*
- *Z. Feng is with the School of Computer Software, Tianjin University, Tianjin 300072, China. E-mail: zyfeng@tju.edu.cn.*

Manuscript received 17 May 2016; revised 27 Oct. 2016; accepted 23 Dec. 2016. Date of publication 30 Dec. 2016; date of current version 12 Feb. 2020. Digital Object Identifier no. 10.1109/TSC.2016.2646687

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

due to the effectiveness of hiring external experts to hunt down the vulnerabilities [21], some companies such as Microsoft, Facebook, Google and Twitter etc. are launching bug bounty program to reward the external hunters for reporting their discovered vulnerabilities. As the vendors are altered before the vulnerabilities are publicly aware, they have enough time to fix the flaws before they are exploited. Hence we can observe a rapid development of the vulnerability market nowadays.

Many efforts have been reported to understand the mechanism of the vulnerability market [10], [11], [47], [51], [54]. However, most of the existing researches only focus on the vulnerable ecosystem about the vulnerabilities, affected products or software vendors, while *the discovery ecosystem focusing on vulnerability hunters and organizations is somehow overlooked*. Some recent works start to pay attention to the hunters to understand their behaviors in bug bounty programs [21], [56]. However, *the vulnerable ecosystem and the discovery ecosystem are still discussed separately, while in fact they together make up a comprehensive vulnerability market*. Additionally, since the desktop PCs are still the main target of cyber attacks over the past decades, *most current literature only takes the software industry as a whole and no special attention is paid to the mobile vulnerability market*. Given the fast annual growth rate, the possibility of virus breakouts in the mobile ecosystem is rising [48]. Our previous study observes a rapid growth of mobile vulnerability market [30] and the industry is beginning to notice the importance of the mobile security. For example, Google announced its bug bounty program focusing on the Android system in June 2015.[5] Furthermore, due to the explosive growth of the mobile ecosystem, most app developers are often lack of formal training in software engineer [37] which resulting into potential risk for end-users. Also, unlike the PCs platform, the mobile ecosystem is a typical platform-centric ecosystem that Android and Apple iOS are the two main cores in the ecosystem.

Therefore, the goal of this paper is to investigate the growth patterns in the mobile vulnerability ecosystem to identify the urgent issues for further security enhancement. First, combing the vulnerable ecosystem and the discovery ecosystem, a five layer heterogeneous network is established to formally describe the mobile vulnerability market. Based on the proposed network model, we formally model the growth of mobile vulnerability market as the evolution of the network series. Since the Google Android, Apple iOS and Microsoft Windows Phone are the main platforms in the mobile ecosystem, we collect the publicly disclosure vulnerabilities relevant to these three platforms from various agencies to form a comprehensive data set for the empirical study. Then we conduct an exploratory network-based analysis, focusing on *1) the growth of the mobile vulnerability market to understand the evolution patterns*; and *2) the intersection between mobile and other PCs platforms to understand the migration between mobile and PCs*. Finally, based on the observations, suggestions about the further actions for software vendors, security hunters and researchers are discussed, aiming to improve the security situation in the mobile ecosystem.

In summary, the main contribution of this paper is the first comprehensive empirical study about the mobile vulnerability market for further security improvement, consisting of:

- A heterogeneous network model to formally represent the mobile vulnerability market,
- A systematical analysis to understand the growth of mobile vulnerability market and the migration between mobile and other PCs platform,
- Suggestions drawn from observations for further security enhancement.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 presents the network model and research questions. Section 4 shows the data collection and overview. Section 5 reports the empirical study about the market growth. Section 6 discusses the intersection between mobile and PCs. Section 7 identifies the important issues. Section 8 concludes the paper.

2 Related Works

2.1 Vulnerable Ecosystem

With the debate on whether vulnerability disclosure programs can improve the overall security [32], many works have been presented to understand the activities of vulnerability discovery, patching and exploitation, etc. Based on the 1,675 vulnerabilities in the ICAT[6] dataset, Rescorla showed that there exists very weak or no evidence for vulnerability disclosure benefit [42]. Neuhaus and Plattner collected vulnerability reports for Mozilla, Apache httpd, and Apache Tomcat and their results showed that the vulnerabilities is not declining after the vulnerability disclosure [38]. Bilges research showed that attacks exploiting particular zero-day vulnerabilities increased by as much as 100,000 fold after their disclosure [11].

On the other hand, public disclosure of security information inspires software vendors to repair vulnerabilities and build more secure products [13]. Ozment and Schechter found the social benefit for the vulnerability discovery in the foundational code of OpenBSD [40]. Based on the publicly disclosed vulnerability information between 1996 to 2007, Frei et al. presented a security ecosystem including discovers, markets, criminals, vendors, security information providers and the public, demonstrating the importance of the security information providers [22]. Cavusoglu et al. [14] argued that the responsible vulnerability disclosure policy ensures the release of a patch. Arora et al. further showed that the vulnerability disclosure can speed up the vendors response rate [8]. Shahzad et al. [45] constructed a larger-scale study of the evolution of the vulnerability life cycle based on the vulnerabilities disclosed till 2011, showing the improvement of the security in the whole software industry and the software companies are becoming more agile in responding to the discovered vulnerabilities.

Due to this double-edged sword characteristic that vulnerability disclosure has both positive and negative effects, some researchers turned to study the mechanism of vulnerability disclosure to optimize the social benefit. Arora et al. [9] developed a framework to optimize the disclosure timing. Kannan and Telang showed that some regulatory guidelines are necessary for proper vulnerability disclosure and the payment

5. www.google.com/about/appsecurity/android-rewards/

6. ICAT is now known as the National Vulnerability Database.

for vulnerability discovery can encourage the growth of benign identifiers [32]. From a different perspective, some researchers tried to employ the publicly disclosure information for vulnerability identification and severity assessment [12], [41], [43], [52]. Based on the features extracted from the public disclosure vulnerability information, Bozorgi et al. [12] employed SVM to predit whether the disclosed vulnerability will be exploited, which is helpful for vulnerability assessment. Sabottke et al. [43] introduced the Twitter-based exploit detector, which used the discussion on Twitter to detect the active exploits in the real world.

2.2 Vulnerability Discovery Ecosystem

As discussed in [14], an early discovery of vulnerability can increase social benefits. Based on a code review experiment on a small web application with 30 subjects, Edmundson et al. showed that a large group of white hats can discover the potential vulnerabilities in a more effective way [25]. By analyzing the Google Chrome vulnerability reward program and the Mozilla Firefox vulnerability reward program, Finifter et al. [21] confirmed the effectiveness of hiring external experts to discover the potential vulnerabilities. Therefore, we can observe that more and more companies have built security response team or launched the bug bounty program to reward the outside security researchers for the discovered vulnerabilities. Consequently, some third-party bug bounty platforms such as HackerOne,[7] BugCrowd,[8] Wooyun,[9] Sobug,[10] Vulbox[11] etc. are built to host the bug bounty programs for different companies to attract white hats to dig potential vulnerabilities. Zhao et al. [56] collected the publicly available data from Wooyun and HackerOne to study the white hats characteristics, trajectory and impact. Their results showed that the continuously growing white hat communities are providing significant contributions to organizations from different sectors and the monetary incentive plays an important role to attract the white hats.

2.3 Mobile Vulnerability Market

Though some works have been presented to understand the vulnerability ecosystem, most of them take the software industry as a whole and no special attention is paid to the mobile vulnerability market. Due to the rapid growth of the mobile ecosystem, researchers start to study its patterns. Zhang et al. [54] first uncovered the mysterious Android root providers to understand and characterize the risk of the well-engineered exploits from the root providers. Our previous work [30] showed an overview of the Android-relevant vulnerability market, especially about the Android's vulnerable ecosystem. Wang et al. [50] technically demonstrated the threats that mobile devices can be infected by the connected compromised PCs so that the PC-side exploits can become dangerous for the mobile ecosystem.

In this paper, unlike these works discussing the vulnerable ecosystem and discovery ecosystem separately, based on the data we collected, we combine them to formally study the mobile vulnerability market, focusing on its evolution:

7. https://hackerone.com/
8. https://bugcrowd.com/
9. http://en.wooyun.org/
10. https://www.sobug.com/
11. https://www.vulbox.com/

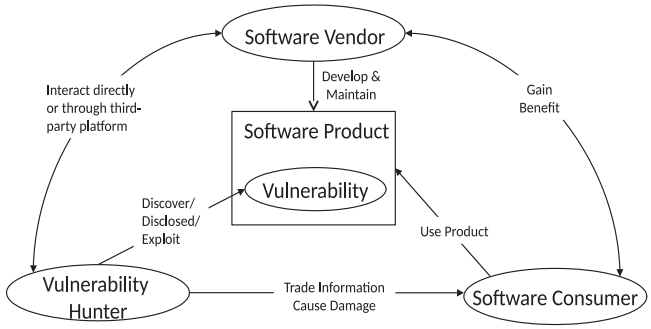


Fig. 1. Overview of mobile vulnerability market.

- Q1: How does the mobile vulnerability market grow over time? How do the relations change?
- Q2: How does the mobile vulnerability market interact with other PCs platforms, especially how do the intersected products/hunters migrate?

3 Network Model and Research Question

3.1 Mobile Vulnerability Market

As shown in Fig. 1, the same to the traditional software, in the mobile vulnerability market, the vendors release software products to fulfill specific requirements. The potential vulnerabilities may be discovered by the vulnerability hunters, who could be workforces belonging to the software vendors, third-party experts or hackers. Hunters can alert the software vendors directly or through third-party platform so that vendors can fix the vulnerability by offering patching before the public disclosure. On the other hand, hunters may sell the vulnerability in the black market where exploitations are developed to attack the consumers resulting into damage. Also, hunters can publicly disclose the vulnerability through third-party platform, intending to speed up the vendor's response and patching developing progress. Therefore, the mobile vulnerability market is a dynamic marketplace driven by the interactions among vendors, software products, vulnerability and hunters. We need to combine the discovery ecosystem and the vulnerable ecosystem to understand the insight of this market. Additionally, hunters can be individual researchers or belong to some security organizations. Therefore, as shown in Fig. 2, we can formally construct a heterogeneous network model consisting of the vendors, software products, vulnerabilities, hunters and organizations as well as their relations to present the mobile vulnerability market:

Definition 1 (Heterogeneous Network Model for Vulnerability Market). *Mobile Vulnerability Market can be modeled as a heterogeneous network* $G = \{S, P, V, H, O, R_{sp}, R_{pv}, R_{vh}, R_{vo}, R_{ho}\}$ *where* S *refers to the vendors,* P *refers to the software products,* V *refers to the vulnerability set,* H *refers to the hunters,* O *refers to the organizations.* R_{sp} *refers to release relations between vendors and products,* R_{pv} *refers that the products are allocated with vulnerabilities,* R_{vh} *means that the vulnerability is discovered by a hunter,* R_{vo} *represents the vulnerability is discovered by an organization and no specific hunter is allocated,* R_{ho} *refers that the hunter belongs to an organization.*

Due to the complexity of the software product, the discovered vulnerability may come from different flaws.

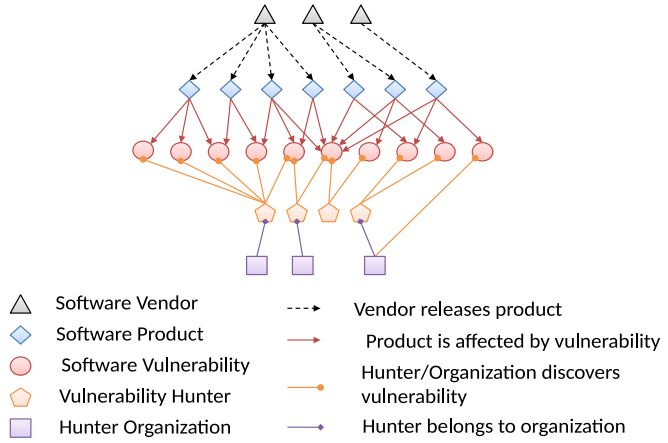


Fig. 2. Heterogeneous network model for mobile vulnerability market.

Therefore, in this paper, we identify the different types of vulnerabilities based on the Common Weakness Enumeration Specification (CWE)[12] and then construct their hierarchical structure, as shown in Fig. 3. The red boxes represent the CWEs being used in this paper. The CWEs located at higher level will cover the CWEs located at the deeper level. It can be seen that there exist four main categories of weaknesses:

- *CWE-19*: Weaknesses in this category are typically found in functionality that processes data.
- *CWE-361*: The improper management of time and state in an environment that supports simultaneous or near-simultaneous computation by multiple systems, processes, or threads.
- *CWE-254*: Concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management.
- *CWE-398*: The code has features that do not directly introduce a weakness or vulnerability, but indicate that the product has not been carefully developed or maintained.

Straightforwardly, the vulnerabilities in categories CWE-19 and CWE-361 are from the weaknesses found in software products' functionality, so that we name them as *functional vulnerability* for description convenience. The vulnerabilities from CWE-254 and CWE-398 are from the improper management of the codes or the security features, so that we name them *management vulnerability*. Note that in this paper, we will not discuss the severity of the vulnerabilities because the widely used Common Vulnerability Scoring System (CVSS)[13] exists some shortages to indicate the actual severity and the exploited possibility of a discovered vulnerability [5],[27].

Based on these definitions, in the following sections, we can formally define the questions about the market growth and its intersection with other PCs platforms.

3.2 Mobile Vulnerability Market Growth

To understand the growth of the market, we first organize all the vulnerabilities based on the disclosed date. Then given a series of consecutive time intervals $1, 2, \ldots t-1, t$,

12. http://cwe.mitre.org/data/published/cwe_v2.9.pdf
13. https://www.first.org/cvss/cvss-guide

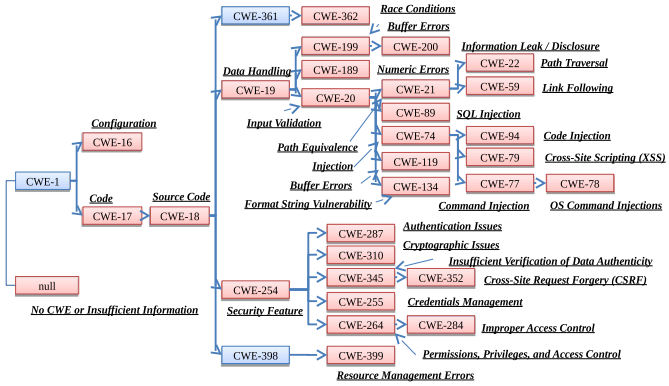


Fig. 3. Vulnerability type taxonomy for mobile vulnerability market.

we use the vulnerabilities disclosed during the given time interval to construct the network, resulting into a snapshot network series $G_{s,1} \ldots G_{s,t}$. Finally we can get the cumulative network series as follow:

$$G_{c,t} = \bigcup_{i=1}^{t} G_{s,i}. \tag{1}$$

Therefore, each cumulative network $G_{c,t}$ refers to the ecosystem no later than given time interval t. Given two consecutive networks $G_{c,t-1}, G_{c,t}$, based on whether the nodes in $G_{c,t}$ exist in $G_{c,t-1}$, we can separate $G_{c,t}$ into two parts:

- *Incremental Network* $G^i_{c,t}$: the nodes in the incremental network don't exist before

$$v \in G^i_{c,t} \leftarrow v \in G_{c,t} \& v \notin G_{c,t-1}, \tag{2}$$

where v can be the vendor, product, vulnerability, hunter or organization.

- *Depositing Network* $G^d_{c,t}$: The nodes in the depositing network exist before

$$v \in G^i_{c,t} \leftarrow v \in G_{c,t} \& v \in G_{c,t-1}. \tag{3}$$

Therefore, based on these definitions, we can formally consider the growth of the mobile vulnerability market as:

Definition 2. (Mobile Vulnerability Market Growth). *Given the consecutive cumulative network series $G^v_{c,i}$ for the whole software industry, $G^m_{c,i}$ for the mobile vulnerability market, $i = 1, 2, \ldots t-1, t$, what are the evolution patterns of the network topology in $G^m_{c,i}$, including the network scale and the relations evolve?*

3.3 Intersection between Mobile and PCs

As shown in Fig. 4, based on whether the vulnerable products in $G^m_{c,t}$ are affected by the vulnerabilities that don't exist in $G^m_{c,t}$, we can separate the vulnerable products into two groups:

- *Pure Mobile Product* P^d_{pm}: The products' allocated vulnerabilities are all mobile-related

$$p_i \in P_{pm} \leftarrow \forall v_j \in R^t_{pv}(p_i), v_j \in G^m_{c,t}. \tag{4}$$

- *Intersected Product* P^d_{im}: The products' allocated vulnerabilities belong to both mobile and other PC platforms

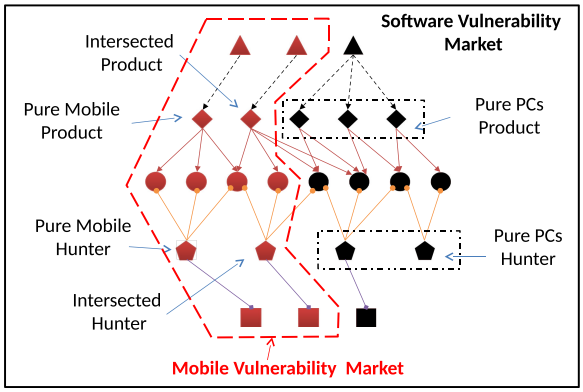


Fig. 4. Intersection between mobile and other PCs platforms.

$$p_i \in P_{im} \leftarrow \exists v_j, v_k \in R^t_{pv}(p_i), v_j \neq v_k, v_j \in G^m_{c,t}, v_k \notin G^m_{c,t}. \quad (5)$$

Here $R^t_{pv}(p_i)$ refers to all the allocated vulnerabilities for product p_i. Similarly, we can separate the hunters in $G^m_{c,t}$ into two parts:

- *Pure Mobile Hunter* H^d_{pm}: All the discovered vulnerabilities are mobile-related

$$h_i \in H_{pm} \leftarrow \forall v_j \in R^s_{vh}(h_i), v_j \in G^m_{c,t}. \quad (6)$$

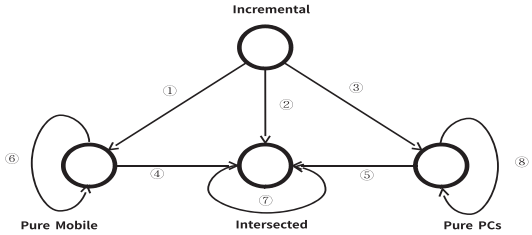
- *Intersected Hunter* H^d_{im}: the hunters' discovered vulnerabilities belong to both mobile and other PC platforms

$$h_i \in H_{im} \leftarrow \exists v_j, v_k \in R^s_{vh}(h_i), v_j \neq v_k, v_j \in G^m_{c,t}, v_k \notin G^m_{c,t}. \quad (7)$$

Here $R^s_{vh}(h_i)$ refers to all the discovered vulnerabilities by hunter h_i. Therefore, as shown in Fig. 5, the interaction between the mobile vulnerability market and the other PCs platforms can be formally defined as the state transitions between different types of products and hunters:

Definition 3 (Intersection between Mobile and Other PCs). *Given the consecutive cumulative network series $G^m_{c,t-1}, G^m_{c,t}$, how do the products and hunters grow into intersected state?*

Until now, we have presented how to construct the network models so that we can formally understand the growth of mobile vulnerability market as well as its intersection with the other PCs platforms. In the following sections, we will show how we empirically study these two research questions based on the constructed network models.



① Incremental Pure Mobile Products/Hunters
② Incremental Intersected Products/Hunters
③ Incremental Pure PCs Products/Hunters
④ From pure mobile product/hunter to intersected product/hunter
⑤ From pure PCs product/hunter to intersected product/hunter
⑥ Remain pure mobile product/hunter
⑦ Remain intersected product/hunter
⑧ Remain pure PCs product/hunter

Fig. 5. State transition diagram for products and hunters.

TABLE 1
Basic Statistics for Collected Dataset

	VMD	MVMD	SSLF
Vulnerability	73,104	1,419	1,390
Product	30,156	379	5,381
Vendor	14,343	190	2,839
Hunter	/	601	/
Organization	/	257	/
Average Products per vendor	2.102	1.995	/
Average Vulnerability per product	3.592	7.216	/
Average Products per vulnerability	1.482	1.927	/
Average Vulnerability per hunter	/	2.84	/
Average Hunters per organization	/	1.74	/

4 Data Collection and Overview

To empirically study the mobile vulnerability market, we collect the vulnerability information from the National Vulnerability Database and then extract the meta-data including the CVE-ID, disclosure date, the affected products, the reference resources and the CWE-ID. Additionally, we fetch the OSVDB-ID, relevant CVE-ID, hunter and affiliation information from OSVDB and BID. Finally, we leverage the CVE-IDs to aggregate the vulnerability data from these three sources and get the *software vulnerability market* dataset, named as VMD for short in this paper.

As known, Android, Apple iOS and Windows Phone are the main platforms in the mobile business ecosystem, hence we use the keywords *Android, Apple iOS, Windows Phone* as the query content on January 6th, 2016 and get 2,095 Android, 710 Apple-iOS, 6 Windows-Phone vulnerabilities from NVD, forming the *mobile relevant vulnerability market (MVMD)* dataset studied in this paper. Furthermore, since OSVDB fails to update the vulnerability's information since May 22nd, 2015, we collect the hunter and affiliation information from BID and the vendor's advisory manually. Note that during September 2014 and October 2014, we observed an outbreak of vulnerabilities for the android applications because of the same *failure of dynamic SSL validation testing* discovered by the same hunter *Will Dormann* from *CERT*. Due to the huge number (23,644) of affected applications, these vulnerabilities are organized into the *Android App SSL Failures* spreadsheet,[14] named as *SSLF* in this paper. It was kept up to date with newly-discovered vulnerable applications while these newly-discovered vulnerabilites are assigned with a CVE-ID, resulting into 1,390 distinst CVE-IDs in SSLF. Therefore, we removed all these 1,390 SSLF vulnerabilities from the original VMD, and MVMD. In the following discussions, the VMD and MVMD both refer to the dataset without these SSL failures vulnerabilities.

Table 1 reports the basic statistics for these three datasets. It can be seen that comparing with the global software industry, the mobile vulnerability market has a slightly small *average products per vendor*. However, its *average vulnerability per products* is almost doubled, which means that the

14. https://docs.google.com/spreadsheets/d/1t5GXwjw82SyunALV Jb2w0zi3FoLRIkfGPc7AMjRF0r4/edit?usp=sharing

TABLE 2
Basic Statistics of Different Niche Ecosystem

		V	P	S	H	O
NE	Android	0	0	0	1	2
	Apple iOS					
	Windows Phone					
	Android	0	2	3	2	0
	Windows Phone					
	Android	2	7	8	31	39
	Apple iOS					
Platform	Android	703	322	167	230	72
	Apple iOS	708	45	11	335	142
	Windows Phone	6	3	1	2	2
	Apple iOS	710	52	19	367	183
	Android	705	331	178	264	113
	Windows Phone	6	5	4	5	4
	Total	1,419	379	190	601	257

products in mobile ecosystem have a more serious security situation so that the community should pay more attention to the mobile ecosystem. Also the *average products per vulnerability* in MVMD is higher than VMD, which means that *comparing with the whole software industry, each vulnerability in MVMD will affect more software products*.

5 Mobile Vulnerability Market Growth

5.1 Scale Growth

As discussed above, the scale growth of the mobile vulnerability market can be modeled as the evolution of nodes in the constructed network $G^m_{c,t}$. Note that there exist three main platforms *Android, Apple-IOS and Windows-Phone*. Before we dig into the growth of the market, we will first understand the *niche ecosystem* for each platform, including the related products, vendors, hunters and organizations

$$NE(p_j) = \bigcup_{v_i \in R^t_{pv}(p_j)} NE(v_i) \tag{8}$$

$$NE(s_j) = \bigcup_{p_i \in R^t_{sp}(s_j)} NE(p_i) \tag{9}$$

$$NE(h_j) = \bigcup_{v_i \in R^s_{hv}(h_j)} NE(v_i) \tag{10}$$

$$NE(o_j) = \left(\bigcup_{h_i \in R^s_{ho}(o_j)} NE(h_i) \right) \bigcup \left(\bigcup_{v_i \in R^s_{vo}(o_j)} NE(v_i) \right). \tag{11}$$

Here $NE(y)$ refers to the nodes $y \in <S, P, V, H, O>$ for the given niche ecosystem presented in Table 2. For $R^z_x(y), z \in \{t, s\}, x \in \{pv, sp, hv, ho, vo\}$ refers to the operation to get the source or target of the given type of relations for the given node.

As shown in Table 2, it can be seen that *Apple iOS* and *Android* are no doubt the main stream in the mobile vulnerability market while *Window Phone* is far behind the other two. This is consistent with the market share for the three platforms,[15] reported by the International Data Corporation (IDC). It provides an evidence for the well-known quote "*the more widely a technology is used, the more likely it is to become the target of hunters that more potential vulnerabilities are discovered*" [33] and "*Given enough eyeballs, all bugs are shallow*" [36]. Actually, no matter whether the potential vulnerabilities are publicly known, they exist in the products resulting into risk for consumers. If the vulnerabilities can be discovered by the workforce belonging to vendors or white hat hackers and be fixed on time, the whole platform-centric niche ecosystem can become more secure and trustworthy for the consumers. Since the niche ecosystem for Window Phone is very tiny, in the following section, we will only discuss the Android and Apple iOS niche ecosystems.

15. http://www.idc.com/prodserv/smartphone-os-market-share.jsp

5.1.1 Node Scale Growth

First, we consider *how different nodes in the network grow overtime*. Straightforwardly, we set the time interval as one year and calculate the following metrics:

- Network Node Size $|G^m_{s,t}(y)|$: Refers to the size of different nodes during each time interval.
- Incremental Proportion $|G^{m,i}_{c,t}(y)|/|G^m_{s,t}(y)|$: Refers to the proportion of new nodes during the given time interval.
- Mobile-aware Proportion $|G^m_{c,t}(y)|/|G^v_{s,t}(y)|$: Refers to the proportion of mobile related nodes in the whole software ecosystem in the given time interval.

As shown in Fig. A in the supplymentary, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSC.2016.2646687, it can be seen that the scale of the mobile vulnerability market is growing rapidly since 2009, with an outbreak in 2012. It also shows that comparing with the whole software vulnerability market, the mobile related market is increasing overtime with an explosion in 2012. Furthermore, we can see that the hunters and organizations are increasing rapidly and their incremental proportion is relative high. More than 85 percent of the hunters and more than 70 percent of the organizations are new for the mobile vulnerability market. Additionally, since 2010, the incremental proportions for the hunters and organizations are decreasing, which means that more and more hunters and organizations are participating in the mobile vulnerability digging in a sustained style, not discovering the vulnerability by chance. Therefore, it is very straightforward for us to draw the first conclusion:

Observation 1. Mobile Vulnerability Market is growing rapidly and it is attracting more and more attentions from the community in a sustained way.

Second, comparing the Android and Apple iOS niche ecosystems, it can be seen that the Android platform owns a higher proportation in software vendors and vulnerable products since 2009. The number of vulnerabilities discovered in the Android platform is higher than the Apple iOS until 2014 except 2010. However, it can be seen that since 2012, the proportation for the Apple iOS is increasing and it exceeds the Android platform in 2015. Also, the proportaion of the organizations and hunters for Apple iOS is larger than that for the Android platform.

Observation 2. Comparing the Android and Apple iOS, the research community, especially the organizations, is

TABLE 3
Pearson Correlation between Nodes

		S	P	H	O
V	**MVMD**	0.6156 (0.0776)	0.6315 (0.0681)	**0.9793 (0)**	**0.9815 (0)**
	Android	**0.8781 (0.0018)**	**0.8736 (0.0021)**	**0.9281 (0.0003)**	**0.9648 (0)**
	Apple iOS	0.214 (0.5804)	0.2427 (0.5292)	**0.9933 (0)**	**0.9836 (0)**

playing more and more attentions to the Apple iOS platform.

Third, we consider the correlation among the vendor, product, vulnerability, hunter and organization over year. The result shows that the vendor and product have a strong positive correlation with $r = 0.9785, p << 0.01$ while the hunter and organization have a strong positive correlation with $r = 0.9826, p << 0.01$. However, the correlation between vendor/product and hunter/organization is not significant. Therefore the hypothesis that *more products can attract more hunters/organizations* should be rejected. For the vulnerability, as reported in Table 3, it has a strong correlation with hunter and organization, which means that the *more hunters and organizations invoke in the vulnerability hunting, the more potential vulnerabilities will be discovered*. Additionally, when considering the whole mobile vulnerability market or the Apple iOS platform, the vulnerability number has no significant correlation with the product/vendor because $p > 0.01$. However, for the Android platform, the correlations are significant which means the more products considered by the hunters, the more vulnerabilities will be discovered.

Observation 3. The more hunters/organizations involving in the market, the more vulnerabilities will be discovered. For the Android platform, the more products the hunters work for, the more vulnerabilities will be found. However, for the Apple iOS platform, the number of vulnerability and the number of product have no significant correlation.

5.1.2 Vulnerable Type Evolution

As discussed in Section 3, based on the CWE, we can identify the vulnerable type and group for each vulnerability in VMD and MVMD. As shown in Fig. 6, it can be seen that the functional vulnerabilities are still the mainstream in MVMD. However, comparing with the whole software vulnerability market, the management group has a larger proportion than the functional group, indicating that *the management vulnerability are more significant in the mobile vulnerability market*.

For the top vulnerability type, "*CWE-119: Buffer Errors*" and "*CWE-264: Permissions, Privileges and Access control*" are the main challenges in MVMD, while CWE-119 is in the functionality group and CWE-264 belongs to the management group. Actually, for the mobile ecosystem, the permission problem could be much more serious because of the overuse of permissions by mobile applications [24] and users incomprehension of permission request [51]. Note that "*NULL*" means that no CWE-ID is assigned with the vulnerability. It can be seen that it occupies the top three positions in the mobile vulnerability market. Therefore there is still a long way to go for the community to identify the vulnerabilities. Good news is that its proportion decreased from 75 percent

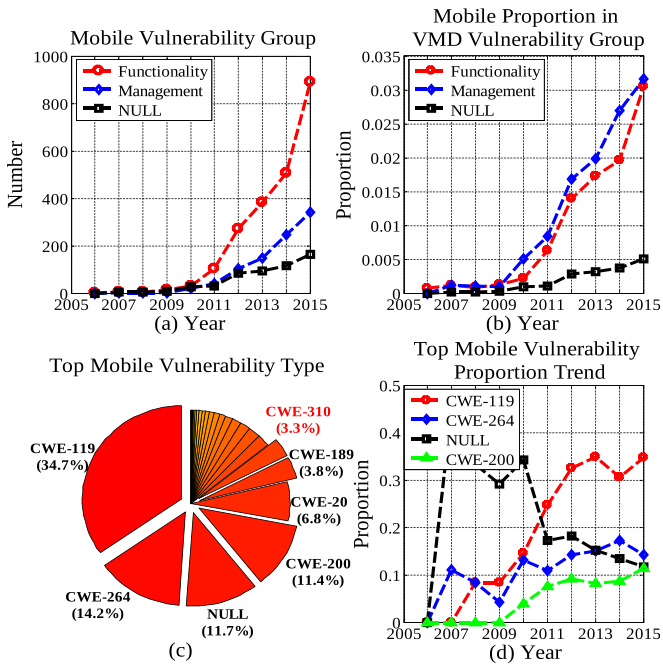


Fig. 6. Type evolution over year in mobile vulnerability market.

in 2007 to about 11.7 percent in December 2015. "*CWE-200: Information Leak / Disclosure*" is the top 4 vulnerability in MVMD, with an 11.4 percent in total. Actually we can observe a significant increase for this type of vulnerability in 2015. Note that we discount the 1,390 SSLF vulnerabilities which all belong to "*CWE-310: Cryptographic Issues*", "CWE-310" still occupy the top 7 in MVMD. Considering the trend that more and more users nowadays are relying on the mobile ecosystem to perform a variety of business tasks and more and more sensitive information is stored in the mobile platform, the impact of these privacy relevant vulnerabilities will become much more important.

Observation 4. Functional vulnerability is still the main stream in mobile vulnerability market and the information leak/disclosure related vulnerability is rapidly increasing. Comparing with whole market, management vulnerability is more important in mobile ecosystem. Specially, the permissions, privileges and access control related and cryptographic issues related vulnerability are the top ones.

5.2 Relation Evolution

Based on the relations in the networks presented in Section 3, we will focus on the combination between the vulnerable ecosystem and the discovery ecosystem.

5.2.1 Vulnerability versus Products

To understand how the vulnerability affect different products, we design the following two metrics in $G^m_{c,t}$:

- *Average Vulnerability Number per Product*: $|R_{pv}|/|P|$ refers to the average number of vulnerabilities in each product, which represents how product is affected by vulnerability.
- *Average Affect Product Number per Vulnerability*: $|R_{pv}|/|V|$ refers to the average number of vulnerable products for each vulnerability, representing how vulnerability distributes across products.

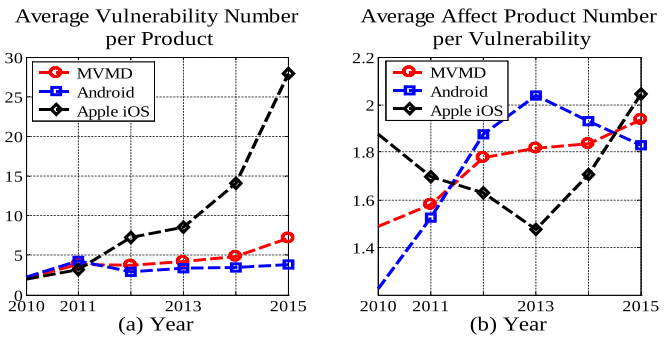


Fig. 7. Vulnerability-product distribution.

As reported in Fig. 7, it can be seen that the average vulnerability number per product for the mobile vulnerability market is increasing since 2010, especially for the Apple iOS platform. However, for the Android platform, the average vulnerability per product increases slightly since 2012. We can also observe that the average number of vulnerable products per vulnerability in the whole mobile vulnerability market is increasing. For the Android platform, it increased between 2010 and 2013 with a slight reduction from 2013 to 2015. On the contrary, for the Apple iOS, after the decrease from 2010 to 2013, we can observe a rapid increase after that. This means that the affected scope of the vulnerability is increasing, especially for the Apple iOS platform.

Observation 5. In average, more potential vulnerabilities per product are discovered and the affected scope is increasing. However, for Android, the average vulnerabilities per product is growing slightly while the affected scope is decreasing since 2013.

Additionally, to understand how the products offered by the same vendor share the vulnerability, we separate the vulnerable products into three types:

- *Tree-Modular-Product (TMP)*: All the associated vulnerabilities are not shared with the products offered by the same vendor.
- *Spindle-Modular-Product (SMP)*: All the associated vulnerabilities are shared with the products offered by the same vendor.
- *Combine-Modular-Product (CMP)*: The other cases about the vulnerabilitiy sharing among products offered by the same vendor.

Obviously, the lower proportion the TMP is, the higher the vulnerabilities are shared in the products from the same vendor. Therefore, we have removed the vendors with only one product and keep 62.27 percent of the products, and then calculate the proportion of TMP, SMP, and CMP in the mobile vulnerability market. From Table 4, it can be seen that overall 32.63 percent of the products belongs to TMP, which means that 67.37 percent vulnerabilities are shared in the products from the same vendors. This phenomenon is more significant in the Apple iOS platform. For the 75 percent products which is offered by the same vendor with the other product, the TMP only occupies 5.13 percent. Therefore, the products offered by the same vendor will have a high possibility to share the same vulnerability. One reason for this is that these products sometimes will share the same software development environment, such as the same framework, library, or code sharing.

TABLE 4
Vulnerability Sharing Among Same-Vendor Products

	TMP	SMP	CMP	Total
Apple iOS	**2 (5.13%)**	**32 {82.05%}**	**5 (12.82%)**	**39 (75%)**
Android	**73 (36.86%)**	**102 {51.52%}**	**23 {11.62%}**	**198 (59.82%)**
Total	**77 (32.63%)**	**134 (56.78%)**	**25 {10.59%}**	**236 (62.27%)**

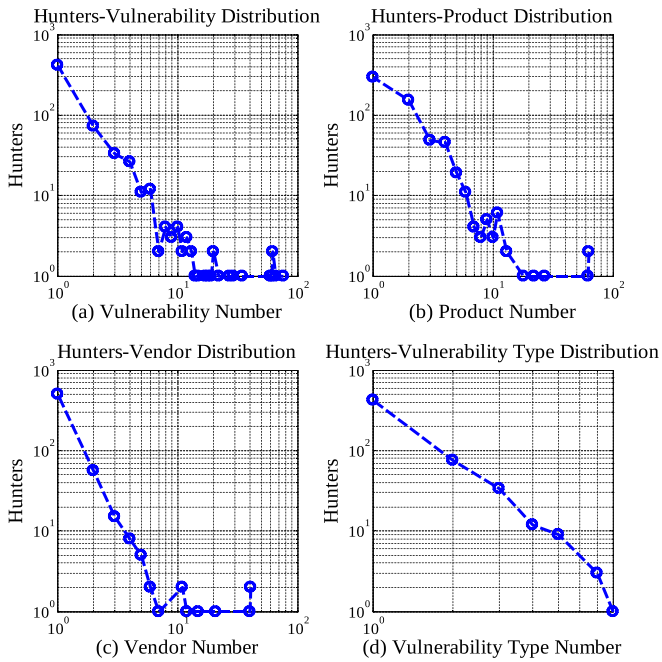


Fig. 8. Hunter performance: Productivity, diversity and skill.

Observation 6. Products offered by the same vendors will have a high possibility to share the same vulnerability, especially in the Apple iOS platform.

5.2.2 Hunter versus Product

To understand how hunters discover vulnerabilites in different products, we can consider the following metrics to evaluate hunters' performance:

- *Hunter Vulnerability Distribution (HVD)*: Refers to the number of hunters who discover a given number of vulnerability, representing the hunters' productivity.
- *Hunter Product Distribution (HPD)*: Refers to the number of hunters who work for a given number of products, representing the hunters' diversity.
- *Hunter Vendor Distribution (HVD)*: Refers to the number of huntes who work for a given number of vendors, representing the hunters' diversity.
- *Hunter Vulnerability Type Distribution (HVtD)*: Refers to the number of hunters who discover a given number of vulnerability types, representing the hunters' skills.

As shown in Fig. 8, it is easy to observe that for the productivity, the HPD fits the long-tail power-law distribution which means most of hunters only discover a few vulnerabilities while a few top hunters contribute great efforts for the

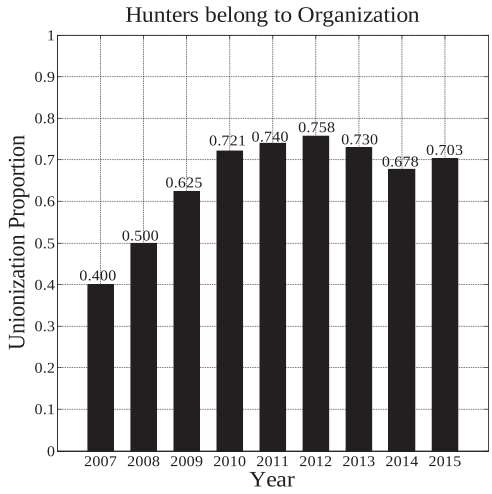


Fig. 9. Hunter unionization rate over year.

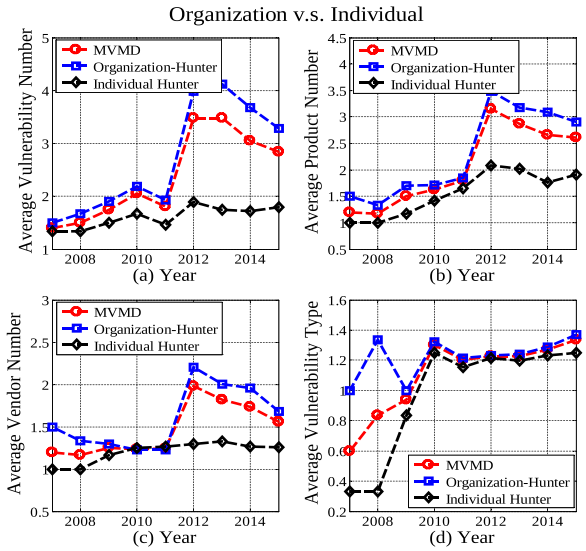


Fig. 10. Performance comparing between organizationed and individual hunters.

whole community. For the diversity, the long-tail power-law distribution refers that most of hunters work in only a few products from a few vendors, and a few top hunters succeed in digging vulnerabilities in many products from different vendors. For the skill, only a few hunters have the skills to discover many different types of vulnerabilities.

Observation 7. *The mobile vulnerability discovery ecosystem has a large number of hunters with low productivity, low diversity and single skill while a few experienced experts with high productivity, high diversity and different skills contribute most efforts in the market.*

5.2.3 Individual versus Organizationed Hunters

In the discovery ecosystem, some hunters are individual researchers, some may form a team to work together on vulnerability hunting, while some belong to companies or organizations. To understand whether the unionization will affect the hunters' performance, we first separate the hunters into two groups:

- *Organizationed Hunters (OH):* Refers to the hunters who are allocated with an organization, a team or a company

$$h_j \in OH \leftarrow R^t_{ho}(h_i) \neq \phi. \tag{12}$$

- *Individual Hunters (IH):* Refers to the hunters who don't belong to any organizations

$$h_j \in OH \leftarrow R^t_{ho}(h_i) = \phi. \tag{13}$$

Then for the constructed network $G^m_{c,t}$, we further calculate the following metrics to understand the effect from the unionization:

- *Unionization Rate*:$|OH|/|H|$, refers to the proportion of the hunters who belong to an organization.
- *Average Vulnerability Number*: Refers to the average number of vulnerabilities discovered by the given hunters.
- *Average Product Number*: Refers to the average number of products the hunters ever work on.
- *Average Vendor Number*: Refers to the average number of vendors the hunters ever work for.
- *Average Vulnerability Type Number*: Refers to the average number of vulnerability types the given hunters ever discovered.

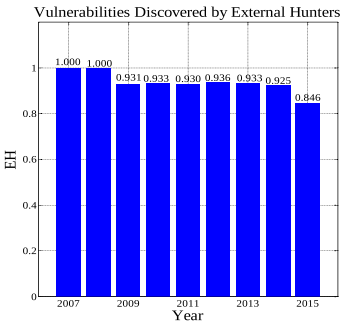
As shown in Fig. 9, it can be seen that before 2012, the unionzation rate is increasing from 40 to 75.8 percent, which means that more and more hunters in the mobile vulnerability market are from ogranizations or they form an organization to regulate the vulnerability hunting behavior, not just motivated by personal interests. After a slight decreasing during 2013 and 2014, the unionization rate reaches 70.4 percent in 2015. Furthermore, we can compare the performance between the individual hunters and organizationed hunters. Fig. 10 prominently shows that the organizationed hunters have a better performance in vulnerability hunting with a larger average vulnerability number, average number of products, average number of vendors, and average vulnerability types.

Additionally, though the skills for both types of hunters are still inceasing slightly after 2012, the productivity and diversity for the organizationed hunters are decreasing after 2013. This means that the organizationed hunters are becoming more concentrated on fewer products and vendors. However, for a given products, the more vulnerabilities are discovered, the less potential vulnerabilities exist so that it would become more difficult to find new vulnerability [40], [56].

Observation 8. Hunters are intending to work together as an organization for the vulnerability hunting. Though the productivity and diversity for the organizationed hunters are decreasing since 2012, these hunters have a better performance than the individual hunters, with a higher productivity, diversity and different skills.

5.2.4 External versus Internal

As reported in [21], the evidence from Google Chrome and Mozilla Firefox bug bounty programs shows that the external experts can effectively discover the potential vulnerabilities. In order to quantify this phenomenon in the mobile-related niche ecosystem, based on the constructed network



Top Vendor with lowest EH

Vendor	EH
IBM	66.67%
Google	69.02%
Mozilla	73.53%
Apple	75.28%
Code Aurora Forum	85.71%
Linux	87.50%
Adobe	98.39%
Apache	100.00%
Qualcomm	100.00%
Go for Android	100.00%

Fig. 11. External and internal effort for vulnerability disclosure.

$G^m_{c,t}$ presented in Section 3, we can get the relations between vendor and organization which represents how many vulnerabilities in the vendors' products are discovered by the given organizations. If the vendor and organization are the same, then the vulnerabilities are discovered by the inside workforce. Otherwise they are discovered by the external hunters. Therefore, we can formally define this proportion to evaluate the effectiveness of the external experts

$$EH(V) = \frac{\sum_{s_i=o_j} |R_{so(s_i,o_j)}|}{\sum |R_{so}(s_i,o_j)|}. \quad (14)$$

Here $|R_{so}(s_i,o_j)|$ refers to the number of vulnerabilities in vendor s_i discovered by the organization o_j.

As reported in Fig. 11, it can be seen that the vulnerabilities discovered by the external hunters is extremely high. Before 2008, 100 percent of the publicly disclosure vulnerabilities are discovered by external hunters. During 2009 and 2014, this rate slightly reduces to about 93 percent. In 2015, this rate significantly reduces to 84.6 percent. Though this rate is still quite high, indicating that most publicly disclosure vulnerabilities are still discovered by the external hunters, some high-technology companies such as IBM, Google, Mozilla, Apple etc, are sharing the vulnerability discovered internally.

Observation 9. External hunters are still the main contributors for the publicly disclosure vulnerabilities while some High-tech companies are participating to share the internally discovered vulnerabilities.

6 Intersection between Mobile and PCs

Until now we have already shown the results about the growth of the mobile vulnerability market. Since mobile vulnerability market can be considered as a niche ecosystem from the whole software vulnerability market, in this section, we will study its interaction with the other desktop PCs, focusing on the product and hunter level.

6.1 Product Intersection

Based on the definitions presented in Section 3, first, for each $G^m_{c,t}$, we consider the following two indications:

- *Intersected Product Proportion*: Represents the scale of the intersected products.
- *Mobile-aware Vulnerability Proportion for Intersected Products*: Represents the proportion of the vulnerabilities which belong to the mobile vulnerability market for the intersected products.

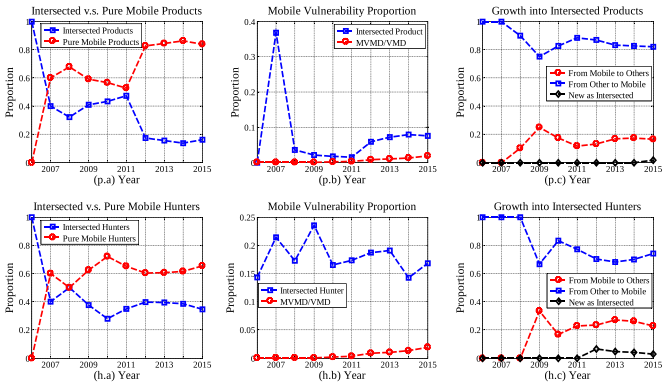


Fig. 12. Intersection products and hunters between mobile vulnerability market and other desktop PCs platforms.

As reported in Fig. 12p.a, the intersected products proportion increases between 2008 and 2011. After a decreasing from 2012 to 2014, we observe a slight growth in 2015. Additionlly, Fig. 12p.c shows that the proportion of the mobile vulnerabilities for the intersected products is higher than the whole software industry and the gap is increasing since 2011. This means that these intersected products are more related to the mobile-aware platform. Furthermore, to understand how the vulnerable products grow into intersected, given the network $G^m_{c,t}$ and $G^v_{c,t}$, the intersected products come from three transitions:

- *From Mobile to Intersected (MtI)*: Means that the vulnerability in mobile platform is discovered first and then the other PCs platforms.
- *From Others to Intersected (OtI)*: Means that the vulnerability in other desktop PCs platforms is discovered first and then the mobile platform.
- *New as Intersected (NtI)*: Means that the vulnerability is discovered in both the mobile and other PCs platforms in the same time interval.

The result shown in Fig. 12p.c reads that 81.97 percent of the intersected products come from other PCs platforms while 16.37 percent come from the mobile platform. Additionally, we can observe that the MtI proportation increases from 2009 to 2011 and it decreases since then while the OtI proportation is converse. Note that in 2015, the NtI proportation is increasing which means the vulnerabilities discovered in both PCs and mobile are increasing. This is consistent with the fact that: most of the products have the PCs versions at the very beginning then the vendors begins to migrate the PC version to the Mobile version. However, as the rapid growth of the mobile ecosystem, some vendors will offer the mobile versions directly and then develop the PC version later. For example, the *wechat* is running on the mobile devices first and then Tecent develops the PCs version after its success. Nowadays, vendors are intending to offer the mobile and PCs version at the same time.

Observation 10. For the intersected products, at the very beginning, most of them find the PCs related vulnerability first than the mobile-related. However, this process is turn converse. Additionally, the rate for discovering vulnerability on both platforms is increasing nowadays.

6.2 Hunter Intersection

Similarly, for each $G^m_{c,t}$ we consider the percentage of the intersected hunters who work across mobile and other PCs platform, and the proportion of the mobile vulnerability that the intersected hunters discover.

Fig. 12h.a shows that the proportion of the pure mobile hunters is larger than the intersected hunters since 2009. Additionally, the proportion of the mobile vulnerability for these intersected hunters is significantly larger than the whole software industry. This means that most of the hunters in the mobile vulnerability market are mainly focusing on the mobile platform.

Furthermore, we also identify three paths by which the hunters grow into intersected:

- *From Mobile to Intersected*: Means that hunters work in the mobile platform first and then discover vulnerability in other PCs platforms.
- *From Others to Intersected*: Means that hunters work in other PCs platforms and then turn to the mobile ecosystem.
- *New as Intersected*: Means that hunters have the ability to discover vulnerability in both PCs and mobile at the same time interval.

As shown in Fig. 12h.c, it can be seen that 74.27 percent intersected hunters migrate from PCs to mobile and about 22.82 percent can start with mobile ecosystem and then turn to other PCs platform. About 2.91 percent can find the vulnerability acrossing mobile and PCs at the same time.

Observation 11. Hunters in the mobile vulnerability market are mainly new and only focusing on the mobile platform. For the platform-acrossing hunters, most of them migrate from PCs to mobile, some can turn from mobile to PCs while only a few can discover the vulnerability in both mobile and PCs at the same time.

7 Discussion and Future Issues

Based on the collected data and the presented heterogeneous network, we have presented an exploratory empirical study about the growth of the mobile vulnerability market and its interaction with the other desktop PCs. It is no doubt that comparing with the whole software vulnerability market, the mobile vulnerability market is rapidly growing with the flourishing of the mobile ecosystem. The more hunters, organizations are involving in this market, the more potential vulnerability are discovered and fixed to enhance the security situation of the ecosystem. Based on our empirical observations, we can further identify the following issues that further researches and solutions are necessary:

7.1 Security Evaluation for Code Reused

As shown above, comparing with the whole software industry, the average number of vulnerable products for each vulnerability in the mobile ecosystem is higher and this value is increasing over time. Additionally, we also find that the vulnerability has a larger possibility to share among the products offered by the same vendors. This is consistent with the observation that software reuse is quite significant in the mobile ecosystem [37]. Actually, if the vulnerability hits the common used libraries or codes, it will affect most of the products which reuse them. Taking the SSLF dataset as an example, we can observe 80.21 percent vulnerable applications due to the employment of the vulnerable library.

Additionally, since most of the app developers are often lack of formal training in software engineering, they are not able to figure out whether there exist some security issues in the reused codes or public libraries. Furthermore, these days developers are using a search engine for help when they encounter unfamiliar issues, and they are often led to online forums such as Stack Overflow. Reusing the codes or libraries directly from these online forums will make the application vulnerable for attacks [18]. For example, Acar et al. [2] found that Stack Overflow contains many insecure answers that Android developers relying on this resource are likely to create less secure code though they can get a quick solutions for their issues.

Therefore, *how to evaluate the security situation, or risk for the public codes and libraries is an urgent issue for the community to help the developers to reduce the risk from code reusage*. For example, adding the mechanism for rating the security of the provided answers in the online forum, recommending the low risk codes or library for developers can be two promising approaches.

7.2 Data Leaking and Permission Overuse

The analysis of the vulnerable types shows that comparing with the whole software products, the management vulnerabilities are more significant in the mobile ecosystem, although the functional vulnerabilities are still the main stream. Specailly, the main functional vulnerability is the "*Information Leak / Disclosure*". Note that these days most users are relying on the mobile ecosystem for their daily life including business operations that more and more valuable privacy data will be stored in the mobile devices. However, current research shows that the private data is still being leaked without the user's permission, especially through the third-party advertisement libraries [46]. How to identify the potential data leak and help the end-user to protect from data leak [4], [34], [55] remains an main issue for the community.

Additionally, permission system is developed to implement security mechanism for the mobile ecosystem, which is also supposed to be able to inhibit the data leak. However, we can observe that the permission related vulnerabilities are increasing these years. Due to the complexity of the permission framework, the dialogs based permission framework for end users have been proved invalid [1], [51] because most end users will just accept and approve the permission request [19] while many mobile applications will request unneccessary permissions [20], no matter the applications are malicious or not. Base on the hypothesis that *the application's functionality should map with the request permissions* as well as *applications with similar functions should request similar permissions*, a two-phase skewness-based methodology [28] is presented to reduce the effect from the permission over-privilege. However, *how to optimize the permission system, help the developer to configure permissions, identify the risk of the permission request for the end-user still needs a long way to go for the community*.

7.3 Hunters' Strategy and Behavior

Our empirical study shows that the unionization rate of the hunters in the mobile community is increasing.

Issues	Supported Observations	Suggestion
Data Leaking and Permission Overuse	Affect scope in the mobile ecosystem is much higher and this value is increasing, especially in Apple iOS platform	Evaluate the security situation or risk for the public codes and library; risk-aware codes or library recommendation
Data Leaking and Permission Overuse	Functional vulnerabilities are still the main stream, and information leak/disclosure related vulnerability occupy the top position	Identify the potential data leak for protection
	Data Leaking and Permission Overuse	Permission over-privilege identification
Hunters' Strategy and Behavior	Organizationed hunters are more productive, more skillful and more diverse in vulnerability hunting	Digging vulnerability in an organization style
	Concentration is good for Apple iOS, while diversification is better for Android platform	Understanding the exploration-exploitation trade-off
		Pay attentions to the vulnerability in apps for Apple iOS niche ecosystem
		Deal with fragmentation problem in Android system
Information Sharing and External Workforce Hiring	More hunters, more vulnerability discovered	Bug bounty program to hire external experts, involving in information sharing
	High external rate	Motivate the insider workforce
		Develop effective information sharing framework in security domain, especially for the vulnerability information
Cross-platform Flow	Double-direction flow between mobile and other PCs platform in vulnerability. Intending to offer products in both mobile and PCs	Develop cross-platform vulnerability digging framework
	Data Leaking and Permission Overuse	

Fig. 13. Identified issues and suggestions supported by the empirical study about the mobile vulnerability market.

Additionally, the hunters belonging to an organization or team are more productive, more skillful and more diverse. This is because that the organizationed hunters will consider the vulnerability hunting as a regular and sustained work, which makes it possible for them to dig more potential volunerability. Additionally, it is easier for the hunters in the same organization to share knowledge about the vulnerabilities. The collaboration among hunters will make the vulnerability discovery, an extremely uncertain, time consuming and high skill intensive job much easier. Therefore, *it is good for hunters to join an organization or team, especially the one focusing in security domain*.

Additionally, it can be seen that for the Apple iOS platform, the diversity of products and vendors has no significant relation with the productivity of the vulnerability. Actually, we can observe that the average vulnerability number per product in the Apple iOS niche ecosystem is rapidly increasing. This indicates that the hunters in the Apple iOS are concentrating in some specific products to dig more vulnerabilities. Since Apple iOS platform is closed-source and only a few open-source products are released, hunters have to work on these 52 different products from only 19 vendors. However, nowadays, the apps running over the Apple iOS are an important part to offer end-to-end security for the end-users. Therefore, for the Apple iOS hunters, they also need to pay attention to the vulnerabilities in the end-user apps.

However, for the Android niche ecosystem, the behavior pattern is totally different. Hunters are working for many different products to find vulnerabilities, while the average discovered vulnerabilities per product are slightly increasing. Note that the Android operation system is open-source that many different products may use different android versions, resulting into a serious android fragmentation problem [35]. Hunters need to spread their resources over different fragments to dig different vulnerabilities. *How to deal with this fragmentation problem in the vulnerability discovery domain and help the hunters to concentrate their resources is an important issue*.

Furthermore, recently many developer issues have been identified [15], [16]. However, the understanding about the vulnerability hunters is still rare. For the current mobile vulnerability market, we already shows that the hunters in the Apple iOS niche ecosystem are using the concentration strategy and the hunters employ the diversification strategy for the Android vulnerability digging. Note that the tradeoff between the exploration and exploitation strategy is considered as one of the fundamental and complicated challenges for behavior understanding [26]. Based on the study of the bug bounty programs, our preliminary result [29] shows a migration between the concentration and diversification strategy. Therefore, *taking a step further to deeply understand the mechanism of this exploration-exploitation trade-off can offer insight for the hunters' behavior to enhance the security in the ecosystem*.

7.4 Information Sharing and External Workforce Hiring

The correlation between the hunters and vulnerabilities shows that the more hunters work in the community, the more potential vulnerabilities will be discovered. This observation supports the famous quote "*Given enough eyeballs, all bugs are shallow*" [36] and "*Wisdom of crowds*" [44] in vulnerability discovery. Therefore, considering the effectiveness of hiring external experts for vulnerability hunting [17], organizations should consider how to hire these external experts to strengthen their security workforce. However, due to the potential ethic issue, how to integrate the external experts is still an important issue. One promising approach might involve launching a bug bounty program, no matter self-running or held by third-party platform. Another might be to work together with other organizations and share information about vulnerability due to the increase of affected scope for the vulnerability.

Furthermore, the high external rate indicates the fact that the software vendors don't effectively motivate the internal workforces to dig the vulnerability, or they intend to keep the vulnerabilities found internally as a secret to the community. For the first case, how to motivate the internal researchers for the vulnerability hunting and balance the external and internal is becoming an important issue for the community. One effective approach is to form the red-team for vulnerability digging in the company. For the second case, it is due to the concern that most consumers fail to patch the vulnerability on time and publicly disclosure information will enable the exploitation. Actually, the OSVDB has been shut down since April 5, 2016.[16] Therefore, the community should consider how to make the patching simpler and more convenient for consumers to fix the vulnerability. Furthermore, *a more effective information sharing framework to speed up the patching development and adoption as well as inhibit the exploited effect will be extremely valuable for the community*.

7.5 Cross-Platform Vulnerability Discovery

Based on the analysis about the intersection between mobile and other PCs platforms, we can observe a significant double-direction flow between them. Some products will uncover the vulnerability in the PCs version and then the mobile version, some discover the vulnerability in mobile and then PCs, while we can see some products find vulnerability in mobile and PCs at the same time interval. Actually, the recent research [50] shows that the mobile ecosystem may be affected by the PC's flaw. Therefore, the hunters should also pay attention to the vulnerability in the PCs platform and how to deal with the

16. https://blog.osvdb.org/2016/04/05/osvdb-fin/

vulnerability propagation from PCs to mobile is an important issue for the community.

However, it can be seen that currently most hunters only focus on the mobile ecosystem and only a few hunters working in other platforms can migrate to the mobile vulnerability market. Also we can see that only a few hunters can discover vulnerability in both mobile and PCs at the same time. Therefore, considering the double-direction flow between the mobile and PCs, as well as the fact that nowadays, most vendors intend to offer mobile and PCs versions for the products at the same time, how to enable the hunters to dig vulnerability in both ends becomes an important issue. An cross-platform vulnerability digging framework can be a valuable contribution for the community.

8 Conclusion

Due to the rapid popularity of the mobile ecosystem, its security has become an important issue for the industry and academia. To understand the patterns of the mobile vulnerability market to share insight for further issues, we constructed a five-layer heterogeneous network to formally study its evolution. Based on the data collected from a variety of agencies such as NVD, OSVDB, BID and software vendors' advisories, we have presented an exploratory empirical research, focusing on the market's growth including the scale growth and relation pattern, as well as its intersection with other PCs platforms. Based on the observations, we have identified five issues and their recent progresses for further security improvement: the security evaluation of codes or library for code reuse, methodology for data leaking protection and permission overuse identification, understanding about hunters' behavior and strategy, hiring external workforce and effective information sharing framework to enhance the security capability for the organizations, as well as the cross-platform vulnerability digging approaches.

In the future research, based on this empirical study, we will further focus on these identified data-driven issues to enhance the security for the mobile ecosystem.

Acknowledgments

The authors would like to thank all the reviewers for their valuable comments. Zhiyong Feng is the corresponding author. This work is supported by the National Natural Science Foundation of China grant 61502333, 61572350, 61373035 and the MIT Interdisciplinary Consortium for Improving Critical Infrastructure Cybersecurity, also known as MIT-(IC)^3.

References

[1] Y. Acar, M. Backes, S. Bugiel, S. Fahl, P. Mcdaniel, and M. Smith, "SoK: Lessons learned from android security research for appified software platforms," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 433–451.

[2] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky, "You get where you're looking for: The impact of information sources on code security," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 289–305.

[3] A. M. Algarni and Y. K. Malaiya, "Software vulnerability markets: Discoverers and buyers," *Int. J. Comput. Elect. Autom. Control Inf. Eng.*, vol. 8, no. 3, pp. 480–490, 2014.

[4] A. Ali-gombe, G. G. Richard III, I. Ahmed, and V. Roussev, "Don't touch that column: Portable, fine-grained access control for android's native content providers," in *Proc. 9th ACM Conf. Secur. Privacy Wireless Mobile Netw.*, 2016, pp. 79–90.

[5] L. Allodi and F. Massacci, "Comparing vulnerability severity and exploits using case-control studies," *ACM Trans. Inf. Syst. Secur.*, vol. 17, no. 1, 2014, Art. no. 1.

[6] R. Anderson, "Why information security is hard-an economic perspective," in *Proc. IEEE 17th Annu. Comput. Secur. Appl. Conf.*, 2001, pp. 358–365.

[7] R. Anderson and T. Moore, "The economics of information security," *Science*, vol. 314, no. 5799, pp. 610–613, 2006.

[8] A. Arora, R. Krishnan, R. Telang, and Y. Yang, "An empirical analysis of software vendors' patch release behavior: Impact of vulnerability disclosure," *Inf. Syst. Res.*, vol. 21, no. 1, pp. 115–132, 2010.

[9] A. Arora, R. Telang, and H. Xu, "Optimal policy for software vulnerability disclosure," *Manage. Sci.*, vol. 54, no. 4, pp. 642–656, 2008.

[10] T. August and M. F. Niculescu, "The influence of software process maturity and customer error reporting on software release and pricing," *Manage. Sci.*, vol. 59, no. 12, pp. 2702–2726, 2013.

[11] L. Bilge and T. Dumitras, "Before we knew it: An empirical study of zero-day attacks in the real world," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 833–844.

[12] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: Learning to classify vulnerabilities and predict exploits," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 105–114.

[13] H. Cavusoglu, H. Cavusoglu, and S. Raghunathan, "Emerging issues in responsible vulnerability disclosure," in *Proc. 4th Workshop Econ. Inf. Secur.*, 2004, pp. 1–31.

[14] H. Cavusoglu, H. Cavusoglu, and S. Raghunathan, "Efficiency of vulnerability disclosure mechanisms to disseminate vulnerability knowledge," *IEEE Trans. Softw. Eng.*, vol. 33, no. 3, pp. 171–185, Mar. 2007.

[15] E. Chen, Y. Pei, S. Chen, Y. Tian, R. Kotcher, and P. Tague, "OAuth demystified for mobile application developers," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 892–903.

[16] E. Chin, A. Felt, K. Greenwood, and D. Wagner, "Analyzing inter-application communication in Android," in *Proc. 9th Int. Conf. Mobile Syst. Appl. Services*, 2011, pp. 239–252.

[17] D. E. Denning, "Toward more secure software," *Commun. ACM*, vol. 58, no. 4, pp. 24–26, 2015.

[18] S. Fahl, M. Harbach, H. Perl, M. Koetter, and M. Smith, "Rethinking SSL development in an appified world," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 49–60.

[19] A. Felt, E. Ha, S. Egelman, and A. Haney, "Android permissions: User attention, comprehension, and behavior," in *Proc. 8th Symp. Usable Privacy Secur.*, 2012, pp. 1–16.

[20] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *Proc. 18th ACM Conf. Comput. Commun. Secur.*, 2011, pp. 627–638.

[21] M. Finifter, D. Akhawe, and D. Wagner, "An empirical study of vulnerability rewards programs," in *Proc. 22nd USENIX Secur. Symp.*, 2013, pp. 273–288.

[22] S. Frei, D. Schatzmann, B. Plattner, and B. Trammell, "Modeling the security ecosystem—The dynamics of (In)Security," in *Economics of Information Security and Privacy*. Berlin, Germany: Springer, 2010, pp. 79–106.

[23] A. Ghose and S. P. Han, "Estimating demand for mobile applications in the new economy," *Manage. Sci.*, vol. 60, no. 6, pp. 1470–1488, 2014.

[24] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, "Checking app behavior against app descriptions," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 1025–1035.

[25] K. Harrison and G. White, "An empirical study on the effectiveness of common security measures," in *Proc. 43rd Hawaii Int. Conf. Syst. Sci.*, 2010, pp. 1–7.

[26] T. T. Hills, P. M. Todd, D. Lazer, A. D. Redish, I. D. Couzin, and C. S. R. Group, "Exploration versus exploitation in space, mind, and society," *Trends Cogn. Sci.*, vol. 19, no. 1, pp. 46–54, 2015.

[27] H. Holm and K. K. Afridi, "An expert-based investigation of the common vulnerability scoring system," *Comput. Secur.*, vol. 53, pp. 18–30, 2015.

[28] K. Huang, J. Han, S. Chen, and Z. Feng, "A skewness-based framework for mobile app permission recommendation and risk evaluation," in *Proc. 14th Int. Conf. Service-Oriented Comput.*, 2016, pp. 252–266.

[29] K. Huang, M. Siegel, S. Madnick, X. Li, and Z. Feng, "Diversity or concentration? Hackers' strategy for working across multiple bug bounty programs," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 1–2.

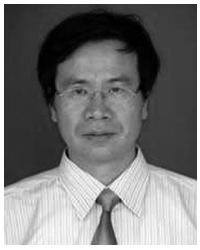
[30] K. Huang, J. Zhang, W. Tan, and Z. Feng, "An empirical analysis of contemporary android mobile vulnerability market," in *Proc. IEEE Int. Conf. Mobile Services*, 2015, pp. 182–189.

[31] Internet Society, "Internet society global internet report 2015: Mobile evolution and development of the internet," Internet Society, Reston, Virginia, USA, Tech. Rep. 2, 2015.
[32] K. Kannan and R. Telang, "Market for software vulnerabilities? Think again," *Manage. Sci.*, vol. 51, no. 5, pp. 726–740, 2005.
[33] N. Leavitt, "Mobile security: Finally a serious problem?" *Comput.*, vol. 44, no. 6, pp. 11–14, 2011.
[34] L. Li, D. Octeau, and J. Klein, "DroidRA: Taming reflection to support whole-program analysis of android apps," in *Proc. 25th Int. Symp. Softw. Testing Anal.*, 2016, pp. 318–329.
[35] X. Lu, et al., "PRADA: Prioritizing android devices for apps by mining large-scale usage data," in *Proc. 38th Int. Conf. Softw. Eng.*, 2016, pp. 3–13.
[36] T. Maillart, M. Zhao, J. Grossklags, and J. Chuang, "Given enough eyeballs, all bugs are shallow? Revisiting eric Raymond with bug bounty programs," in *Proc. Workshop Econ. Inf. Secur.*, 2016, pp. 1–19.
[37] 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," *IEEE Softw.*, vol. 31, no. 2, pp. 78–86, Mar./Apr. 2014.
[38] S. Neuhaus and B. Plattner, "Software security economics: Theory, in practice," in *Proc. Econ. Inf. Secur. Privacy*, 2013, pp. 75–92.
[39] A. Ozment and S. E. Schechter, "Bootstrapping the adoption of internet security protocols," in *Proc. 5th Workshop Econ. Inf. Secur.*, 2006, pp. 1–19.
[40] A. Ozment and S. E. Schechter, "Milk or wine: Does software security improve with age?" in *Proc. 15th USENIX Secur. Symp.*, 2006, pp. 93–104.
[41] H. Perl, et al., "VCCfinder: Finding potential vulnerabilities in open-source projects to assist code audits," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 426–437.
[42] E. Rescorla, "Is finding security holes a good idea?" *IEEE Secur. Privacy*, vol. 3, no. 1, pp. 14–19, Jan. 2005.
[43] C. Sabottke, O. Suciu, and T. Dumitras, "Vulnerability disclosure in the age of social media: Exploiting Twitter for predicting real-world exploits," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 1041–1056.
[44] N. Savage, "Gaining wisdom from crowds," *Commun. ACM*, vol. 55, pp. 13–15, 2012.
[45] M. Shahzad, M. Z. Shafiq, and A. X. Liu, "A large scale exploratory analysis of software vulnerability life cycles," in *Proc. IEEE 34th Int. Conf. Softw. Eng.*, 2012, pp. 771–781.
[46] A. Short and F. Li, "Android smartphone third party advertising library data leak analysis," in *Proc. 11th IEEE Int. Conf. Mobile Ad Hoc Sensor Syst.*, 2015, pp. 749–754.
[47] K. Soska, N. Christin, K. Soska, and N. Christin, "Measuring the longitudinal evolution of the online anonymous marketplace ecosystem," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 33–48.
[48] P. Wang, M. C. González, C. A. Hidalgo, and A.-L. Barabási, "Understanding the spreading patterns of mobile phone viruses," *Science*, vol. 324, no. 5930, pp. 1071–1076, 2009.
[49] R. Wang, et al., "EASEAndroid: Automatic policy analysis and refinement for security enhanced android via large-scale semi-supervised learning," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 351–366.
[50] T. Wang, Y. Jang, Y. Chen, S. Chung, B. Lau, and W. Lee, "On the feasibility of large-scale infections of iOS devices," in *Proc. 23rd USENIX Secur. Symp.*, 2014, pp. 79–93.
[51] P. Wijesekera, B. Columbia, A. Baokar, A. Hosseini, S. Egelman, and D. Wagner, "Android permissions remystified: A field study on contextual integrity," in *Proc. 24th USENIX Secur. Symp.*, 2015, pp. 499–514.
[52] F. Yamaguchi, N. Golde, D. Arp, and K. Rieck, "Modeling and discovering vulnerabilities with code property graphs," in *Proc. IEEE Symp. Secur. Privacy*, 2014, pp. 590–604.
[53] Z. Yang, M. Yang, Y. Zhang, G. Gu, P. Ning, and X. S. Wang, "Appintent: Analyzing sensitive data transmission in android for privacy leakage detection," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 1043–1054.
[54] H. Zhang, D. She, and Z. Qian, "Android root and its providers: A double-edged sword," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1093–1104.
[55] M. Zhang and H. Yin, "Efficient, context-aware privacy leakage confinement for android applications without firmware modding," in *Proc. 9th ACM Symp. Inf. Comput. Commun. Secur.*, 2014, pp. 259–270.
[56] M. Zhao, J. Grossklags, and P. Liu, "An empirical study of Web vulnerability discovery ecosystems," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1105–1117.
[57] J. Yin, X. Lu, C. Pu, Z. Wu, and H. Chen, "JTangCSB: A cloud service bus for cloud and enterprise application integration," *IEEE Int. Comput.*, vol. 19, no. 1, pp. 35–43, 2015.
[58] J. Yin, X. Lu, X. Zhao, H. Chen, and X. Liu, "BURSE: A bursty and self-similar workload generator for cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 3, pp. 668–680, 2015.

Keman Huang received the dual BS degrees from the Department of Automation and School of Economics and Management, Tsinghua University, China, in 2009, and the PhD degree from the Department of Automation, Tsinghua University, China, in 2014, respectively. He is currently an assistant professor in the School of Computer Science and Technology, Tianjin University and a postdoctoral associate in the Sloan School of Management, MIT. His research interests include service ecosystem, mobile service, and Semantic Web. He has published more than 30 journal and conference proceedings papers. He received the Best Paper Runner-up Award from IEEE SCC 2016, Best Student Paper Award from IEEE ICWS 2014, and ICSS 2013. He is currently an associate editor of the *International Journal of Services Computing*. He was in the program committees of many conferences and the publicly chair of IEEE ICWS/SCC/MS/BIGDATA Congress 2016. He is a member of the ACM and the IEEE.

Jia Zhang received the BS and MS degrees in computer science from Nanjing University, China, and the PhD degree in computer science from the University of Illinois at Chicago. She is currently an associate professor in the Department of Electrical and Computer Engineering, Carnegie Mellon University. Her recent research interests center on service oriented computing, with a focus on scientific workflows, net-centric collaboration, and big data. She has co-authored one textbook titled "Services Computing" and has published more than 140 refereed journal papers, book chapters, and conference papers. She is currently an associate editor of the *IEEE Transactions on Services Computing* and the *International Journal of Web Services Research*, and editor-in-chief of the *International Journal of Services Computing*. She is a senior member of the IEEE.

Wei Tan received the BS and PhD degrees from the Department of Automation, Tsinghua University. He is currently a research staff member with the IBM T. J. Watson Research Center, New York. From 2008 to 2010, he was a researcher in the Computation Institute, University of Chicago and Argonne National Laboratory. At that time, he was the technical lead of the caBIG workflow system. His research interests include GPU accelerated machine learning, NoSQL, big data, cloud computing, service-oriented architecture, business and scientific workflows, and petri nets. He has published more than 70 journal and conference papers, and a monograph "*Business and Scientific Workflows: A Web Service-Oriented Approach*" (272 pages, Wiley-IEEE Press). He received the IEEE Peter Chen Big Data Young Researcher Award (2016), Best Paper Award from ACM/IEEE CCGrid (2015), Best Student Paper Award from IEEE ICWS (2014), Best Paper Award from IEEE SCC (2011), the Pacesetter Award from the Argonne National Laboratory (2010), and caBIG Teamwork Award from the National Institute of Health (2008). He is member of the ACM and senior member of the IEEE.



Zhiyong Feng received the PhD degree from Tianjin University. He is currently a full professor in the School of Computer Science and Technology, Tianjin University, China. He is the author of one book, more than 130 articles, and 39 patents. His research interests include knowledge engineering, services computing, and security software engineering. He is a member of the IEEE and the ACM.