# Securing Android App Markets via Modelling and Predicting Malware Spread between Markets

Guozhu Meng*†‡, Matthew Patrick§, Yinxing Xue¶, Yang Liu‡, Jie Zhang‡
*SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences
†School of Cyber Security, University of Chinese Academy of Sciences, China
‡Nanyang Technological University, Singapore
§University of Michigan
¶University of Science and Technology of China

**Abstract**—The Android ecosystem has recently dominated mobile devices. Android app markets, including official Google Play and other third party markets, are becoming hotbeds where malware originates and spreads. Android malware has been observed to both propagate within markets and spread between markets. If the spread of Android malware between markets can be predicted, market administrators can take appropriate measures to prevent the outbreak of malware and minimize the damages caused by malware. In this paper, we make the first attempt to protect the Android ecosystem by modelling and predicting the spread of Android malware between markets. To this end, we study the social behaviors that affect the spread of malware, model these spread behaviors with multiple epidemic models, and predict the infection time and order among markets for well-known malware families. To achieve an accurate prediction of malware spread, we model spread behaviors in the following fashion: 1) for a single market, we model the within-market malware growth by considering both the creation and removal of malware, 2) for multiple markets, we determine market relevance by calculating the mutual information among them, 3) based on the previous two steps, we simulate a Susceptible Infected (SI) model stochastically for spread among markets. The model inference is performed using a publicly-available well-labeled dataset AndRadar. To conduct extensive experiments to evaluate our approach, we collected a large number (334,782) of malware samples from 25 Android markets around the world. The experimental results show our approach can depict and simulate the growth of Android malware on a large scale, and predict the infection time and order among markets with 0.89 and 0.66 precision, respectively.

**Index Terms**—Malware Spread Prediction, Epidemic Model, Malware Lifecycle, Android Market, Ecosystem Security.

## 1 Introduction

Android has become the largest and prevailing mobile platform since 2011 [1]. Millions of Android applications, hereinafter referred to as apps, provide end users a convenient and swift environment for online education, shopping, entertainment, etc. Meanwhile, Android also attracts a large number of cybercriminals who create malware to harvest users' sensitive data, cause financial loss, and remote-control devices [2, 3]. The stunning growth of Android malware poses a huge threat to users, and this situation has been even exacerbated in recent years since Android malware has become more infectious and disseminated [4]. In the early years of Android development, malware spread mainly relied on SMS/MMS, file duplication with USB, and Internet access [5]. Newly-discovered malware has leveraged increasingly popular Android app markets to spread [6].

Restraining Android malware's propagation and spread has proven to be an effective proactive pathway to reduce the damages it causes [7]. Unfortunately, however, it is non-trivial to study and identify the propagation and spread mechanisms of Android malware. On traditional PC OSes, the malware infection is through the access to multiple media (e.g., emails, URL links, and external storage) [8]. On iOS, due to Apple's rigorous review and inspection of iOS apps, only the certified apps can be put on the shelf for users to download. Different from both the traditional and iOS platforms, Android malware utilizes the app markets for propagation, including the official market GooglePlay and hundreds of unofficial markets. The users who have downloaded malware from these markets may unintentionally get their devices infected. To some extent, to secure the Android ecosystem is to secure Android app markets.

Android market is a digital distribution service that allows users to browse and download apps. There are two types of Android markets—Google Play, the official and generally more regulated Android market that is operated and developed by Google, and third-party Android markets that are owned by either profitable or non-profitable companies. Due to the unfaithful security inspection of Android markets and the continuous emergence of malware variants and zero day malware, Android markets, especially third-party markets, become the dominating venue through which to disseminate malware [9, 10]. Hence, securing Android markets is not an easy task, considering the enormous malware (especially the piggybacked apps [11]) and unofficial markets.

As the problem of third party markets is comparatively new, there is still no clear understanding and vigorous study on the increase and spread of Android malware within or between markets. Existing studies have extensively investigated the epidemic model of traditional malware [12–14], whereas research on Android malware propagation is not adequate to fully understand its propagation among markets. Yu *et al.* [15] have made the first step to study the distribution of malware in terms of networks in three stages—early, final and late. The infected smartphones formed a specific network and were classified by the type of carrying malware. Hence, the existing studies on mobile malware propagation

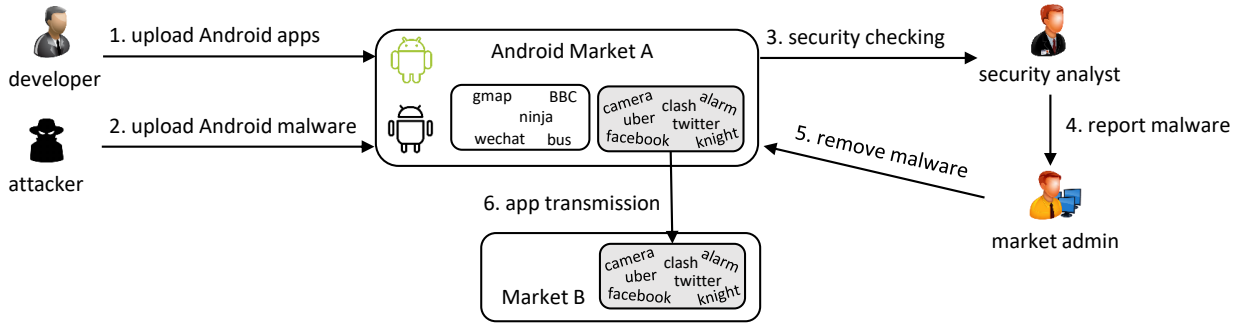



Fig. 1: Malware-related behaviors in the Android ecosystem

and spread center around the end users or smartphones (treating end users as the nodes in a connected network).

Different from above work, we take Android markets as nodes, and form a connected network. Malware, as a pathogen on Android, can proliferate within one market and amplify its destructive effect by markets. By analogy with epidemic diseases, malware gets markets infected in the course of spread. In such cases, an accurate model of malware behaviors can facilitate prediction of malware propagation and spread, and thereby protect Android markets from infection. In particular, we aim to address the following research problems in security engineering of Android markets: Given one market, how to predict the number of malware that resides in it? Given a set of markets and the probabilities of app transmission among them, how to predict the number of infected markets at a certain time for a certain malware family? Succeedingly, how to predict infection order with maximal probability for a certain malware family on the set of markets?

However, we are facing some critical challenges that impede an adequate research on the above problems. First, insufficient data is publicly available for this study — the malware data exists in a variety of Android markets. It impedes the study of malware spread between markets. Second, since the spread mechanism of malware is subject to multiple factors' interference, a comprehensive model is desired to accurately depict Android malware spread among markets, based on the within-market growth model.

We start from inferring a preliminary model using 20,000 well-labelled malware samples from the study [16]. Then, to address the issue of insufficient data, we further collect 334,782 malware samples to evaluate the model. To construct a comprehensive model, we infer the potential connections between markets by calculating the mutual information among markets in a pairwise way. The mutual information computes the percentage of shared apps amongst two markets, and thereby implies the probability of app transitions between markets.

Technically, to provide an accurate model of malware spread, we first identify the social behaviors involved in malware spread (see Section 2). Given a market and a malware family, we model the behaviors on malware creation and removal to build a growth model of that family (see Section 3.1). Next, we model between-market spread patterns of Android malware by combining mutual information and epidemic theory. Specifically, we apply ARACNe [17] to calculate the mutual information among markets. The mutual information, acting as closeness degrees between markets, connect all involved markets to form a network through which malware spreads. We then propose an *SI (Susceptible-Infected)* model by incorporating mutual information and simulate the spread of malware stochastically between markets (see Section 3.2). Lastly, we evaluate the model on 334,782 malware samples from 25 markets and the experimental results show our model can be highly coherent to these malware samples.

To the best of our knowledge, this is the first work to study the market-oriented malware spread model (which covers both within-market and between-market spread) and apply it to predict infection time and order of markets for Android malware.

We make the following contributions in the paper:

- The first attempt to study connections between Android markets from the perspective of malware spread. To the best of our knowledge, previous studies mainly consider how malware infects and spreads from the perspective of end devices or users. Also different from the study on the iOS malware dissemination through the single infection source APP STORE [18], we focus on the impact of third-party or unofficial app markets in Android, which are often the first victims and hosts for infection.
- Building multiple computable models to predict the between-market spread of Android malware. We first model the propagation of a malware family (i.e., the growth model) within a market. Based on that, to model between-market spread, we propose a comprehensive model that combines mutual information and epidemic theory.
- Predicting malware infection time and order for markets on an unprecedentedly large data set from the real world. 334,782 real-world malware samples are used to evaluate our model. The results show the within-market model provides a good fit to the data and the between-market model can effectively predict the infection time and order of malware among markets with 0.89 and 0.66 precision, respectively.

The remainder of the paper is organized as follows: Section 2 states the problems to solve in this study; Section 3 introduces the social behaviors that affect malware propagation and spread, and then proposes theoretical models to depict malware propagation and spread; Section 4 conducts several experiments to validate our theories and models; Section 5 discusses the limitations of our work, and promising applications in the field of security; Section 6 briefly describes the related work on modelling malware propagation and spread; We summarize our work in Section 7; Last, we attach supplementary investigation and results in Section 8.

## 2 PROBLEM STATEMENT

In this section, we first uncover behaviors that affect malware propagation and spread on Android, and then provide a definite

TABLE 1: Notations of symbols in the study

| Notation | Description |
|---|---|
| $I(t)$ | The number of infected apps at time $t$ |
| $I_m(t)$ | The number of infected markets at time $t$ |
| $S_m(t)$ | The number of susceptible markets at time $t$ |
| $T_c(a)$ | The creation time for the app $a$ |
| $T_r(a)$ | The removal time for the app $a$ |
| $T(M)$ | The infection time for the market $M$ |
| $M[i,j]$ | The mutual information between market $i$ and $j$ |

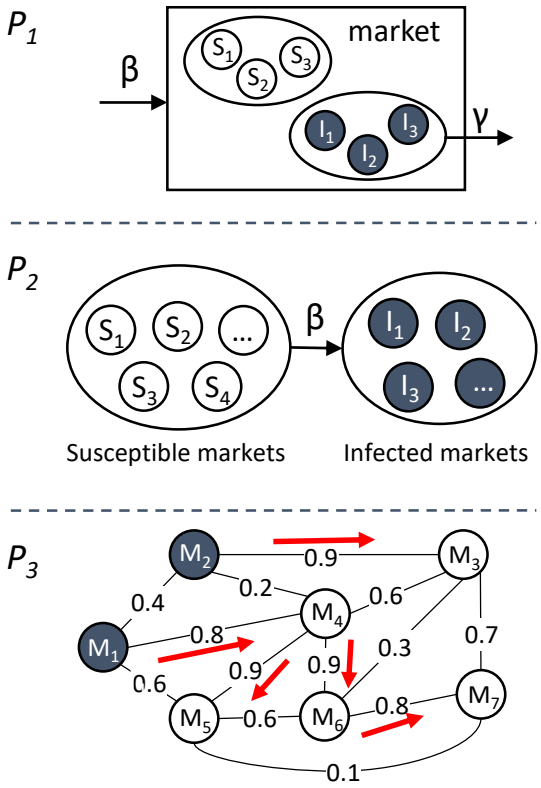



Fig. 2: Research problems in malware spread

statement for problems to solve in this paper. For ease of understanding, we summarize all key symbols used in this paper, and list them in Table 1.

### 2.1 Malware-related Behaviors

We illustrate malware-related behavior graph for malware spread in Figure 1. This graph indicates the most relevant participants whose behaviors (e.g., creating and disseminating malware) can significantly affect the malware ecosystem according to [19, 20]. There are four main roles involved in malware spread: 1) *developers*, who upload apps into markets (denoted as behavior 1); 2) *attackers*, who upload malware into markets (denoted as behavior 2); 3) *security analysts*, who inspect the uploaded apps (denoted as behavior 3) and report malware to market admin (denoted as behavior 4), and; 4) *market admins*, who remove malware according to detection results (denoted as behavior 5), and transfer apps in bulk from other markets (behavior 6).

In one market, Android malware experiences creation, growth, and removal. When an attacker creates new malware and uploads it into a market, the malware starts its lifecycle [15]. It may undergo a fast development by piggybacking other apps [21] or evolving [22], as the number of malware variants increases dramatically. However, after its outbreak and popularity for a while, malware may be detected by anti-virus software. Then the malware and its variants will be removed from markets gradually. As reported by [16], the malware still exist in some markets after a long period of time due to loose checking procedures in those markets. In Figure 1, behavior 2 increases the amount of malware directly within a market, and behavior 6 transmits apps from one market to another market which may also increase the number of malware. Behavior 5 removes the malware within a market. Note that behavior 1 refers to users' unwitting submission of an app that is actual malware.

According to the study [23], the malware carriers play a significant role in spreading malware over time. As indicated by [11, 16], some markets share a considerable number of replicated apps in between, which implies that an app transmission process exists between these markets. The spread of Android malware comes from two aspects: malware is distributed into different markets, and malware is moved from one market to another. As shown in Figure 1, behaviors 1, 2, 6 can all spread malware between markets. Note that behavior 6 refers to app transmission due to app sharing (or copying) by the admins of markets.

### 2.2 Problem Definition

Without loss of generality, we provide the following definitions for the malware ecosystem:

**Definition 1.** *An app market is a set of susceptible and infected apps $\mathcal{M} = \{S, I\}$. $S$ is the set of susceptible apps (i.e., benign apps) $S = \{a | a \text{ is not } malware\}$, and $I$ is the set of infected apps (i.e., malware) $I = \{a | a \text{ is } malware\}$.*

In the scope of one market, the number of malware is constantly changed due to uploading and removal. Let $T_c(a), T_r(a)$ be the creation and removal time, respectively. To quantify malware propagation along with time, we regard $I(t)$ as the number of malware at time $t$.

We assume that there are $N$ app markets in total, i.e., $N = \{M_1, M_2, ..., M_n\}$. One market $\mathcal{M}$ is infected by malware at time $t$ if $|I(t)| > 0$. Malware spread across markets until all markets are thoroughly infected. Let $I_m(t)$ be the infected markets at time $t$. Then the spread velocity of malware can be computed with the differential equation $\frac{d\, I_m(t)}{dt}$. In the meantime, one thorough infection follows a proper order which is defined below.

**Definition 2.** *Infection order is the infection sequence of app markets as per time $\pi = \langle M_{i^1}, M_{i^2}, ..., M_{i^n} \rangle$. Let $T(M_{i^n})$ be the infection time for market $M$ which is the $n$-th market to be* infected. *Then $T(M_{i^j}) \leq T(M_{i^k})$ when $1 \leq j < k \leq n$.*

It is significant to understand and further predict malware propagation and spread between markets. With an accurate prediction, security analysts and users can take an instant measure to counter an upcoming outbreak of malware. In this paper, we propose three problems that exhibit security concerns during malware spread as shown in Figure 2.

**P1**. Within a market $M$, malware can be created and removed, and the amount is dynamically changed along with time. In this study, we intend to approximate the number of malware samples within the market $M$ at time $t$, i.e., $|I(t)|$.

**P2**. Markets can be infected in the course of malware spread. Let $\beta$ be the probability of market transmitted from state "susceptible" to state "infected", and $N$ be the total number of markets, i.e., $|I_m(t)| + |S_m(t)| = N$. Then, we will explore the number of infected markets $I_m(t)$ at time $t$.

**P3**. There exists an "app flow" between markets as per behavior 6, which reveals the closeness between markets. If some markets have been infected by malware initially, the remaining markets will be infected eventually and the infection order relies on the closeness between markets. Given an arbitrary number $j$ of infected markets $I_m(0)$ where $0 < j < n$, we will identify one infection order $\pi = \{M_{ij+1}, ...M_{in}\}$ with maximal probability.

## 3 Spread Model of Android Malware

In this section, we proposal two models to predict malware propagation and spread between markets. In particular, we demystify the lifecycle of malware within one market and construct models to exhibit its dynamic propagation for solving **P1**. Further, we extend epidemic models to illustrate malware spread between markets, which is akin to how infectious disease spreads in epidemiology. These models are designed to solve **P2** and **P3**. Note that our models are derived from the study on malware-related behaviors in the malware ecosystem and conclusive models in previous studies [15, 24, 25]. We give more details in the following sections.

### 3.1 The Within-market Propagation of Android Malware

The within-market propagation of Android malicious apps is determined by their *creation* and *removal*. Malware is generally created via injecting malicious code into normal apps by attackers — infecting those apps. Once an app is determined to be malware by the market administrators, it as well as its variants will be deleted from the market. However, there is typically a time delay between malware creation and removal. This delay differs depending on the app market (e.g., its security inspection capability) and malware family (e.g., its maliciousness), hence leading to different patterns of malware *growth*.

#### 3.1.1 Simple model of malware growth

As with previous investigations into malware modelling, we start with a simple model that assumes a linear rate of growth [5] (Equation 1). In this model, malware-infected apps are added to a market at a linear rate $\beta I$ (where I is the number of infected apps in the market). In other words, the rate at which infection increases is assumed to be proportional to the number of currently infected apps [15]. Whilst this is a simple model and is unlikely to be accurate in the general case, it is appropriate for the early stages of an infection (i.e., before density dependent factors [26] take effect) [15]. We also show how this model can accurately represent the overall growth of malware in a market, since (in the two datasets we have collected) the overall growth of malware shows no signs of slowing down. We use this as a starting point for more sophisticated models, incorporating the differences between malware families (see Section 3.1.2) and the spread of malware across multiple markets (see Section 3.2.1).

**(Theoretical assumption for the growth model)** This simple malware model may be motivated by looking at Figure 1: newly infected apps from behaviors 1, 2 and 6 are all proportional to the number of currently infected apps. In the absence of density dependence, the rates at which these behaviors occur may be assumed to be linear. In behavior 1, users or developers unwittingly upload more infected apps when the malware is more common; in behavior 2, attackers infect more apps for uploading if seeing its popularity; in behavior 6, market A gets more infected apps if market B has more. Similarly, the rate of malware removal depends on the number of infected apps, since as more apps are infected, it becomes more likely for security analysts to notice and detect the infected apps, and then report them to admins for removal (behaviors 3, 4 and 5). One further reason why density dependence does not occur in the overall growth of malware is because attackers may upload any number of variations of an existing app or of new apps they created. The rate of growth is proportional to the number of currently infected apps, since attackers use and adapt malware that has already been uploaded, but it is not proportional to the number of uninfected apps, since there is no upper limit to the number of apps that can be infected.

$$\frac{dI}{dt} = (\beta - \gamma)I, \ and \ \frac{dR}{dt} = \gamma I \tag{1}$$

In Equation 1, the growth rate of infected apps is the subtraction of the removed number ($\gamma I$) from the newly added number ($\beta I$). After solving these differential equations, we have this analytical solution for $I$:

$$I(t) = I(0)\exp((\beta - \gamma)t) \tag{2}$$

We have collected two datasets denoted as *DS1* and *DS2*, respectively. DS1 contains a number of apps as well as their removal time, and DS2 supplements creation time to apps. Therefore, we compute the creation rate and removal rate of malware separately (using nonlinear least squares estimation and *Bayesian* parameter estimation): the growth rate ($\beta$) is determined according to the cumulative number of malware apps created over time in DS2 (for each market); the removal rate ($\gamma$) is calculated by aligning all the malware-infected apps created (for each market) in DS1 to time $t = 0$ and then recording the time at which each app is removed (see Equation 3 and 4, respectively).

$$I(t)^{DS2} = I(0)^{DS2}\exp(\beta t), \beta > 0 \tag{3}$$

$$I(t)^{DS1} = I(0)^{DS1}\exp(-\gamma t), \gamma > 0 \tag{4}$$

**Remark.** With Equation 3 and 4, we can portray the increasing trend and decreasing trend of malware within one market. It solves problem **P1** by quantifying residual malware as a security measure to market.

#### 3.1.2 Malware-family specific growth in each market

The previous model may be extended to represent the number of infected apps for a particular family of malware (rather than just the total number of malware-infected apps). In order to do this, we introduce a new concept, visibility ($\nu_f$). Visibility is a constant value for each malware family that determines how quickly infected apps are detected by administrators and removed. It may be affected by the number of malware samples, the ability of detection, etc. It can also dissuade attackers from uploading particular families of malware, if they become too easily detectable and are hence less effective. The visibility of a particular malware family ($f$) depends on a large number of factors, such as how well its malicious code is hidden against detection, the impact it has on an app's behavior and the time until that impact takes effect.

**(Theoretical assumption for the family specific growth)** Visibility may be interpreted in terms of cooperation/competition

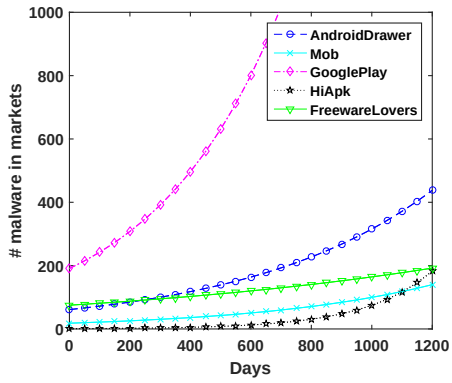



Fig. 3: The creation curve in 5 markets

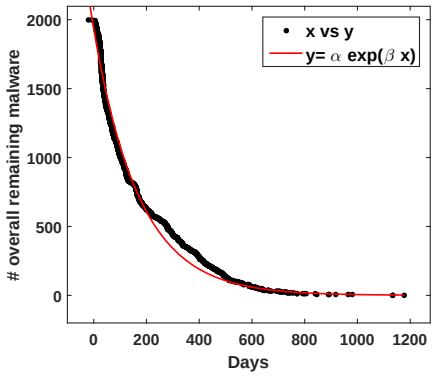



Fig. 4: The overall removal curve of malware

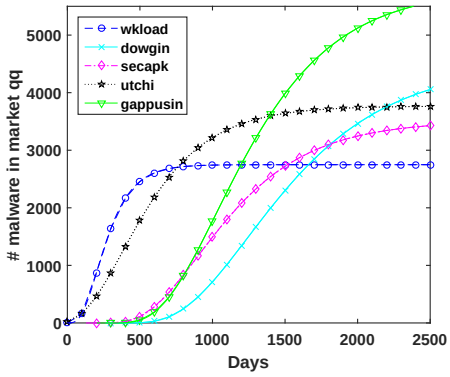



Fig. 5: The growth curves of malware in qq

between malware: malware families that are not highly visible to security analysts may encourage more attackers to upload apps of the same family, whereas the more infected apps there are of a visible family, the more likely these apps are to be removed. To represent this behavior, we turn to the Hill Equation [27]. Hill Equation is widely used to represent the degree of cooperative binding between ligands and the macromolecules in biochemistry. This degree is often enhanced if there are already other ligands present on the macromolecules. Visibility of malware exhibits the similar characteristic, i.e., the lower the visibility is, the lower probability of malware being removed, and thereby the more motivated the malware authors are to upload more samples. Instead of binding affinity, we use the Hill Equation to represent how the rate at which malware is added/removed is affected by the number of malware-infected apps currently on the market, according to the visibility parameter ($\nu_f$). To achieve this, we use a generalized form of the Hill Equation [28]:

$$I(t) = \frac{\kappa}{[1 - J\exp(-\beta t)]^{1/\theta}} \tag{5}$$

In this model, $\kappa$ denotes the curve's maximal value, and is calculated as $\kappa = \frac{1}{\nu_f}$; $J$ denotes a constant for the curvature, and is calculated as $J = 1 - \left(\frac{\kappa}{I_0}\right)^{\theta}$; $\theta$ is a constant denoting the steepness of the curve ($\theta = 5$).

**Remark**. With Equation 5, we can calculate the immediate number of malware of a certain family residing in one market. By complementing the single model aforementioned, it solves problem **P1** by accurately depicting the characteristics of malware families during propagation within one market.

### 3.2 The Between-market Spread of Android Malware

As shown in Figure 1, *app transmission between markets* (behavior 5) influences the spread of malware greatly. According to [16], replicas of malware apps extensively exist in alternative markets. To measure the transmission between markets, we compute their mutual information. We then present a deterministic epidemic model to depict malware spread between markets.

#### 3.2.1 Mutual information between markets

App transmission is a directed action from one market to another. We can predict how likely malware is to spread between particular markets by calculating their mutual information.

**(Theoretical assumption for applying mutual information)** Unlike linear measurements of inter-dependence (e.g. the Pearson Correlation Coefficient), mutual information takes into account all forms of dependence (linear and non-linear). In this scenario of Android markets, the interdependency does not only exist between two markets. Instead, app transmission may occur amongst more than two markets, which implies that the mutual information is more suitable to measure the relativity between markets. Mutual information can be described as the amount of information that can be obtained from one market about the behavior of another. For $N$ markets, we define a symmetric $N \times N$ matrix $M$, where $M[i,j](1 \leq i,j \leq N)$ denotes the mutual information between market $i$ and $j$, and can be used as an approximation for the likelihood of apps in market $i$ transmitting into market $j$.

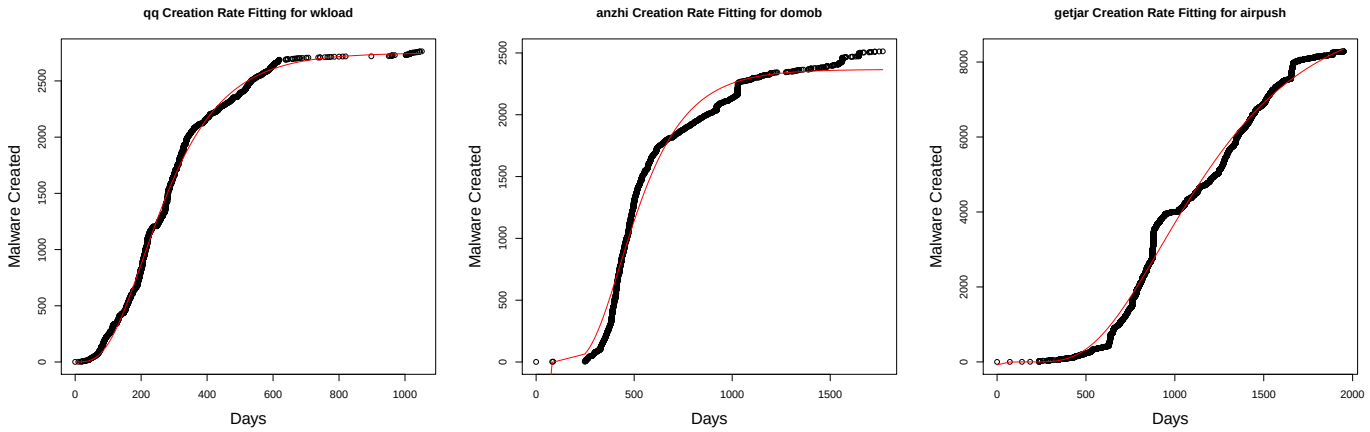
We calculate mutual information using ARACNe (Algorithm for the Reconstruction of Accurate Cellular Networks [17]). ARACNe is typically applied to identify complex interactions between genes, according to differences in gene expression (i.e., RNA) over time. In our application, we replace genes with markets and gene expression with the number of malware apps uploaded each month (from a specific malware family). A Gaussian kernel estimator (see Equation 6) is used, where $P$ is the set of time points (in our case, months) and $x_i$ and $y_i$ are the values (number of infected apps) at each time point $i$; $\hat{f}(\_)$ is the marginal probability density, $\hat{f}(\_,\_)$ is the joint probability density and $N$ is the sample size (i.e., number of markets). The kernel density is tuned by maximising the posterior probability by Bayes theorem, according to cross validation.

$$M[i,j] = \frac{1}{N}\sum_i \frac{\hat{f}(x_i, y_i)}{\hat{f}(x_i)\hat{f}(y_i)} \tag{6}$$

We use the resulting matrix $M$ to simulate the spread of malware between markets (see Section 3.2.3). The higher the mutual information between an infected market $i$ and an uninfected market $j$, the more likely it is that $j$ will be infected by malware spreading from $i$.

#### 3.2.2 Epidemic (SI) model for predicting between-market malware spread

Epidemic models have been used extensively to understand and theorise about the spread of various forms of smartphone malware [5]. The particular epidemic model we use is known as the SI (i.e., Susceptible-Infected) model. This model has been applied before in a theoretical study on the spread of mobile phone viruses




(a) The growth curve of malware *wkload* in market qq (b) The growth curve of malware *domob* in market anzhi (c) The growth curve of malware *airpush* in market getjar

Fig. 6: Three samples to illustrate the growth model for a specific malware family. The x axis denotes the number of days since the infection, and the y axis denotes the cumulative number of created malware of a specific family in this market.

over Bluetooth and MMS [29]. Variations of this model have also been used to understand the dynamics of a wide range of biological diseases, including white-nose syndrome in bats [30] and bovine tuberculosis in cattle [31].

**(Theoretical assumption for applying SI model)** The SI model is suitable for simulating the spread of malware across markets because complete extinction of a family is uncommon. In theory, malware eventually becomes extinct once it fails to be effective, but we did not find any evidence of this happening during the time frame of our study (Android malware is a relatively new phenomenon). Although detected malware variants of each family are removed, new variants are continually added. Hence, we use the SI model to understand how new families of malware spread across the markets, from a single point of infection. Our model can help predict how long it will take from the creation of a new malware family (in the future) until that family infects each market.

The SI model contains two mutually exclusive compartments: $S$ (the number of markets that are susceptible to the disease but not yet infected) and $I$ (the number of markets that are infected). Susceptible markets become infected at rate $\beta SI$, but infected markets remain infected and cannot become susceptible again. Therefore, $I$ is small at first and $S$ is large, but over time, $S$ decreases and $I$ increases. This behavior can be described by a pair of differential equations (see Equation 7).

The analytical solution of the SI model is given in Equation 9, where $I_0$ is the number of markets infected at time $t = 0$ (in our case, we assume this is 1), $I_t$ is the number of markets infected at time $t$ (as predicted by our model) and $N$ is the total number of markets ($N = S + I$). Having an analytical solution to the model, allows us to fit it to our data (DS2) using nonlinear least squares estimation (see Section 3.1). Once the (deterministic version) of the model has been fit, this gives us a value for $\beta$, which we can then use to simulate the model stochastically (see Section 3.2.3 for stochastic simulation details).

$$\frac{dS}{dt} = -\beta SI \tag{7}$$

$$\frac{dI}{dt} = \beta SI \tag{8}$$

$$I_t = \frac{NI_0}{I_0 + (N - I_0)exp(-\beta t)} \tag{9}$$

**Remark.** Given one set of markets and the initial number of infected markets, we can compute the number of infected markets at a certain time with Equation 9. With solving problem **P2** , this model reveals the fierceness of malware spread between markets, and enables all stakeholders to percept the security of the current Android ecosystem.

### 3.2.3 *Stochastic simulation for predicting infection order*

Stochastic models are employed to describe non-deterministic behavior and account for sources of error that are not fully known [32, 33]. In our work, we want to know when markets will become infected and which market will be infected next. Both these prediction goals are made more challenging because of natural uncertainty in the spread of malware over time. We also need to take into account the connections between markets, in terms of their mutual information (see Section 3.2.1). Hence, we simulate the spread of malware between markets (according to our SI model) stochastically, using the Gillespie algorithm [34]. The Gillespie algorithm is a suitable technique for this simulation, since it is based on the well-established theory of Monte Carlo methods.

The Gillespie algorithm samples the next infection event to occur and the time until that event takes place iteratively, according to a series of propensity values $a_1 \dots a_n$. Propensity values represent the relative rate at which each event is expected to occur, depending upon the current state. At each step, the Gillespie algorithm generates two random values between 0 and 1. The first random value ($r_1$) is used to determine the next market to be infected $\mu$ by sampling markets according to their propensity (i.e., mutual information with the currently infected markets), where $\sum_{j=1}^{\mu-1} a_j < r_0 \sum_{j=1}^{n} a_j < \sum_{j=1}^{\mu} a_j$. The second random value ($r_2$) is used to sample the time ($\tau$) until the next infection occurs from an exponential distribution, with rate $\sum a_j$, where $a_j$ is the propensity of event $j$ (given the particular state). This means infections occur at a Poisson rate, which is appropriate because infections occur independently of the series of timings that took place before, depending only on the current state (i.e., Markov

chain). Inverse transform sampling is used to sample from the exponential distribution, so $\tau = \frac{1}{\sum a_j} \ln(\frac{1}{r_2})$. It is important to update the propensities at each step, as depending on the current state, the rate at which events occur may change (e.g., as more markets become infected, there are fewer markets left to infect, so infections occur more slowly).

**Remark.** With stochastic simulation, we can predict approximately real infection order of a certain malware family whilst it is spreading between markets. With solving problem **P3**, the model can pinpoint which markets are most likely to be infected next in terms of probability, and direct market administrators to take measures against malware infection.

## 4 Model Validation

In this section, we examine our theoretical models using a large-scale malware dataset in the real world. For the proposed models, we aim to answer:

- How does malware propagate within market?
- How is the mutual information between different markets?
- Can our spread model predict the infection of a specific malware between markets effectively?

In order to evaluate the quality of models, we provide two metrics to measure the discrepancy between the real data and the estimation models. *Residual sum of squares (RSS)* [35] is the accumulation of squares of residuals as follows:

$$RSS = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

where $n$ is the size of the data set, $y_i$ is the actual value of the $i$th variable, and $f(x_i)$ is the predicted value. The less the RSS value is, the more suitable the model can fit the data. To avoid unbias, we leverage *residual standard error (RSE)* [36] instead which can be computed as below:

$$RSE = \sqrt{\frac{RSS}{d}}$$

where $d$ is the degrees of freedom that counts the numbers of independent pieces of data involved in the estimation [37]. We use RSE to evaluate the models in Section 4.2 and 4.4.1.

On the other hand, Pearson correlation coefficient [38] is used to measure the precision of predicted infection time and order in Section 4.4, which is commonly used to measure the linear correlation between two variables. Assuming $X$ is the predicted data, while $Y$ is the actual data, the Pearson correlation coefficient $\rho$ is calculated as follows:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \ \sigma_Y} \tag{10}$$

where $cov(X,Y)$ is the covariance of $X$ and $Y$, and $\sigma_X, \sigma_Y$ are the standard deviations of $X$ and $Y$, respectively. $\rho_{X,Y} \in [-1,1]$, and the larger $\rho_{X,Y}$ denotes that our spread model can predict the infection order more accurately.

### 4.1 Android App Dataset

Our models are evaluated on two sources of Android apps as described below.

**DS1. AndRadar**. Lindorfer *et al.* [16] tracked over 20,000 apps in 16 Android markets. They recorded the creation time and removal time for the app in each market and the detection time for malware by anti-virus software. Hence, owing to this detailed information, it is a suitable dataset for model deduction.

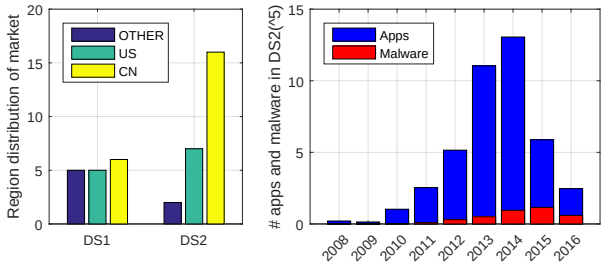



Fig. 7: The statistics of datasets. The left figure shows the region (including China, USA, and others) from which the apps were collected. The x-axis of the right figure is the years since the malware is created, and the y-axis denotes the number ($\times 10^5$) of apps (in blue) and malware (in red).

**DS2. Apps crawled by ourselves**. We have collected over 2 million apps between Sep 2013 to July 2016 from 25 Android markets [39]. Among these apps, there are 334,782 malware samples from 1,149 malware families. For each app, we assess its creation time, belonging market, and detection result (e.g., whether it is malware and if so, which family it comes from) by VirusTotal[1]. Since the detection results for one malware come from 57 integrated commercial anti-virus software in VirusTotal which may vary very much, we leverage AVClass [40] to identify the most likely family name. We publish our dataset and more detailed analysis results at this link [41].

#### 4.1.1 Statistics of Malware

Apps in DS1 are from 16 markets including GooglePlay, SlideME, AppChina, Wandoujia, Lenovo, etc. The markets are evenly distributed in three regions: China, US, and other countries. DS2 includes GooglePlay and other 24 famous third party markets in multiple countries and languages. For example, 16 markets are from China, and 7 markets are from the US. Note that many markets in DS1 are included in our investigated markets. The variety of markets facilitates the study of malware spread.

Figure 7 shows the statistics of apps and malware inside these two datasets. The left figure presents the regional distribution of app markets to analyze, with both sets both have the largest number of markets from China. The right figure shows the number of apps and malware contained in the 25 markets that have been created during the period from 2008 to 2016. Since our crawled apps cover a wide range of collection times (2013 to 2016) and creation times (2008 to 2016), we are able to characterize Android malware development and spread in general.

Table 2 presents the top 10 markets and malware families that contain the largest number of Android malware, respectively. Additionally, we elaborate these markets with their regions and the day range of malware existing inside, and malware families with the day range from the day of first detection to the last observation day in DS2.

### 4.2 Evaluation of Malware Propagation Model — Prediction for P1

We fit the propagation model using *nonlinear least squares* and *Bayesian* parameter estimation and evaluate it with RSE.

1. http://www.virustotal.com, an online malware scanner

TABLE 2: Top 10 of markets and families containing the largest number of malware samples in DS2.

| No | Android Market | | | | Malware Family | | |
|---|---|---|---|---|---|---|---|
| | Name | Region | Number | Day Range | Name | Number | Day Range |
| 1 | GOOGLEPLAY | U.S. | 81,968 | 2,997 | KUGUO | 25,002 | 3,003 |
| 2 | QQ | China | 74,265 | 3,012 | AIRPUSH | 17,135 | 2,995 |
| 3 | ANZHI | China | 59,550 | 2,958 | DOWGIN | 16,818 | 3,006 |
| 4 | GETJAR | Euro | 22,334 | 2,965 | SMSREG | 10,737 | 3,003 |
| 5 | XIAOMI | China | 18,257 | 2,924 | SECAPK | 9,885 | 3,003 |
| 6 | MUMAYI | China | 16,974 | 2,934 | GAPPUSIN | 9,452 | 3,004 |
| 7 | EOEMARKET | China | 11,344 | 2,937 | REVMOB | 8,731 | 2,987 |
| 8 | HIAPK | China | 11,083 | 3,008 | LEADBOLT | 6,636 | 2,976 |
| 9 | APPCHINA | China | 8,656 | 2,975 | YOUMI | 5,919 | 2,974 |
| 10 | APK20 | U.S. | 5,241 | 2,999 | DOMOB | 5,718 | 2,973 |

TABLE 3: The statistics for two models in Section 4.1: overall malware growth in top 10 markets hosting the most malware; family-specific malware growth in a specific market with top 10 pairs of {family, market} that contain the most malware.

| Market | Creation Model | | | Family & Market | Growth Model | | | |
|---|---|---|---|---|---|---|---|---|
| | $\alpha$ | $\beta$ | $RSE$ | | $\alpha$ | $\beta$ | $\gamma$ | $RSE$ |
| GOOGLEPLAY | 1.91E+02 | 2.39E-03 | 1345 | KUGUO & QQ | 4.30E+04 | 1.98E+00 | 1.52E-03 | 1839 |
| QQ | 3.03E+03 | 1.27E-03 | 6535 | DOWGIN & ANZHI | 1.23E+04 | 1.58E+00 | 2.03E-03 | 487.8 |
| ANZHI | 1.78E+03 | 1.49E-03 | 2371 | AIRPUSH & GETJAR | 9.39E+03 | 1.38E+00 | 2.10E-03 | 300.0 |
| GETJAR | 1.20E+06 | 1.25E-03 | 1476 | REVMOB & GOOGLEPLAY | 1.06E+04 | 1.67E+00 | 2.02E-03 | 159.1 |
| XIAOMI | 8.72E+01 | 2.39E-03 | 324.7 | AIRPUSH & GOOGLEPLAY | 4.09E+04 | 1.50E+00 | 8.65E-04 | 277.4 |
| MUMAYI | 1.36E+03 | 1.09E-03 | 1335 | KUGUO & ANZHI | 1.47E+05 | 1.38E+00 | 6.72E-04 | 740.7 |
| EOEMARKET | 8.06E+01 | 2.10E-03 | 395.3 | GAPPUSIN & QQ | 5.81E+03 | 1.79E+00 | 2.14E-03 | 228.8 |
| HIAPK | 8.56E-01 | 4.47E-03 | 497.6 | LEADBOLT & GOOGLEPLAY | 3.41E+05 | 8.79E-01 | 2.40E-04 | 44.01 |
| APPCHINA | 4.04E+01 | 2.01E-03 | 321.6 | SECAPK & ANZHI | 2.25E+04 | 1.55E+00 | 9.88E-04 | 157.8 |
| APK20 | 2.74E+01 | 2.14E-03 | 168.0 | DOWGIN & QQ | 4.51E+03 | 1.85E+00 | 1.79E-03 | 1839 |

**Creation rate $\beta$ (Equation 1).** We fit $\beta$ for each of the 25 markets in DS2. The first 4 columns of Table 3 show the parameters of the fitted model for the 10 markets with the most malware. The mean and median RSE values for all 25 markets are 201.0 and 662.5, respectively, which suggests the exponential distribution provides a good fit for the creation rate in each market. In addition, we plot the cumulative number of malware apps created for 5 markets in Figure 3. GOOGLE PLAY and ANDROIDDRAWER have the largest creation rates, while MOB and FREEWARELOVERS have the least ones. To some extent, it can unveil the popularity of Android markets to attackers. In particular, the largest app market GOOGLE PLAY, once being compromised, can lead to very serious consequences to its users. However, markets like FREEWARELOVERS rarely appeal to attackers due to its scarce users. In despite of varying curvatures, these curves have a high consistence with the data in MCAFEE's 2016 security report [42].

**Removal rate $\gamma$ (Equation 1).** Since only DS1 contains information regarding when malware is removed by markets, we evaluate the removal rate fitting merely on DS1. There are only six markets and 8,339 samples shared between DS1 and DS2, so we do not provide an individual removal rate for each market. Instead, we compute an overall removal rate for Android malware (5.792E-3 apps per infected app per day). The fitted curve is shown in Figure 4. It indicates removal rate fitting was highly effective on the data (RSE=76.4). The removal rate complies with an inverted sigmoid curve, indicating that malware becomes more imperceptible over time. It can imply in reality that market administrators can receive more either explicit or implicit notifications when a relatively larger number of malware samples are residing in the markets, and thereby set about checking and removing them.

**Family-specific growth model (Equation 5).** The family-specific malware growth model reveals the markets on which each family is most successful. We fit the model using DS2, which contains detailed information on creation dates and malware families. We identified the top 10 pairs of malware family and market that contain the most malware, and fit the model on this data. The last 5 columns in Table 3 show the detailed parameters fitted for this model (mean RSE=439.0). We plot three of them in Figure 6, i.e., malware WKLOAD in market QQ, malware DOMOB in market ANZHI, and malware AIRPUSH in market GETJAR (all these malware families have at least 1000 samples, and exist for at least 1000 days). As the curves show, the model can simulate the growth of malware in a specific family accurately. We also compute the RSE values for all 40 pairs of markets and families that have at least 1000 samples, of which the mean is 163.3, the median is 79.7, 1st quartile is 42.4, 2nd quartile is 79.7, 3rd quartile is 149.5, and 4th quartile is 1839.

Figure 5 shows the growth curves of five malware families in market QQ. WKLOAD is more infectious and can reach its peak in a shorter time, while GAPPUSIN is less infectious with a slow increase, but can reach a relatively higher peak. This implies that GAPPUSIN is more likely to evade the security inspection.

**Significance.** The lifecycle model can benefit the security community threefold: 1) predicting the growth (either increase or decline) of malware and thereby helping to restrain new outbreaks of malware in its early phase; 2) assessing the resistibility of Android markets in terms of the growth rate of malware (from Figure 3)

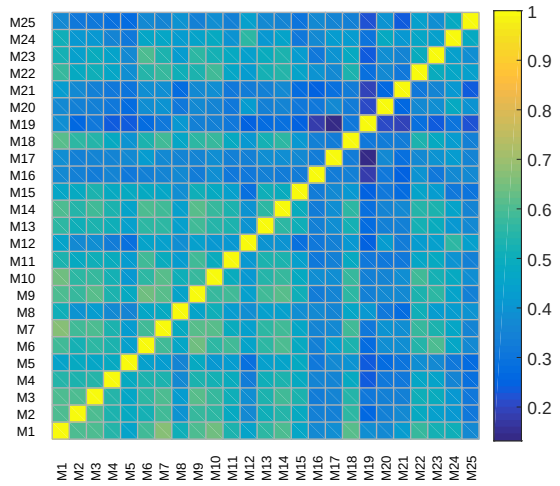



Fig. 8: Mutual Information between markets. In particular, the 25 markets are in sequence: GOOLGEPLAY, QQ, ANZHI, GETJAR, MUMAYI, XIAOMI, APK20, HIAPK, EOEMARKET, APPCHINA, COOLAPK, APKMIRROR, FLYME, GFAN, CNMO, BAIDU, ANDROIDDRAWER, WANGYI, ANRUAN, FDROID, FREEWARELOVERS, MOB, WANDOUJIA, APKPURE, AND CHINAMOBILE

to suggest further improvements in security, and; 3) identifying family-specific malware growth in a specific market. The variety of growth rates can facilitate the study of malware categorization and characteristics as well as the weaknesses of security detection techniques.

### 4.3 Mutual Information between Markets

We examine mutual information between the 25 Android markets in DS2, calculated using ARACNe (see Section 3.2.1). The results are shown as a matrix in Figure 8, where mutual information is in the range [0, 1] and cells with a larger value have a lighter color. The larger mutual information between two markets indicates that they share more same apps (as well as malware) in between.

We list the top 5 pairs of markets with highest mutual information value in Table 4 and the top 5 pairs of markets with lowest mutual information value in Table 5. These results indicate a relatively large rate of app replicas between the markets based on the analysis on DS2. The pair with the highest mutual information is APK20 and GOOGLEPLAY. The reason is that APK20 claims to have the "Top 100,000 Play Store Apps Available To Download" in its website, implying that APK20 has crawled a large number of apps from GOOGLEPLAY. Consequently, malware from GOOGLEPLAY is highly likely be transmitted into APK20. The pair with the lowest mutual information is ANRUAN and FDROID. ANRUAN is a commercial application repository that employs TENCENT anti-virus software to detect potential malware, and provides a variety of Android apps for its users. FDROID is an application repository hosting thousands of open-source Android apps, which asks its developers to upload source code for apps. This open source policy impedes many developers in ANRUAN to share their commercial products in FDROID. By examining the malware in these two markets, we found that no malware is shared in between. Therefore, the mutual information is only 0.2 between these two markets.

**Significance.** Behavior 6 in Figure 1 facilitates the formation of an implicit underlying network amongst markets. This experiment confirms our theory by revealing markets often share a considerable amount of mutual information, and indicates the hidden topology of the underlying network for these markets. Information gathered from these results can be used to: 1) study the direction and velocity of malware spread between markets combining the growth model within a market. 2) control, or even cut off, the channels of spread of malware before it is widely spread.

TABLE 4: Top 5 pairs of markets with the highest correlation.

| No | Market Pair | Correlation |
|---|---|---|
| 1 | {GOOGLEPLAY, APK20} | 0.66 |
| 2 | {GOOGLEPLAY, APPCHINA} | 0.64 |
| 3 | {XIAOMI, EOEMARKET} | 0.64 |
| 4 | {APK20, APPCHINA} | 0.62 |
| 5 | {GFAN, EOEMARKET} | 0.62 |

TABLE 5: Top 5 pairs of markets with the lowest correlation.

| No | Market Pair | Correlation |
|---|---|---|
| 1 | {ANRUAN, FDROID} | 0.20 |
| 2 | {ANRUAN, WANDOUJIA} | 0.23 |
| 3 | {ANRUAN, MUMAYI} | 0.23 |
| 4 | {ANRUAN, GETJAR} | 0.24 |
| 5 | {ANRUAN, CNMO} | 0.24 |

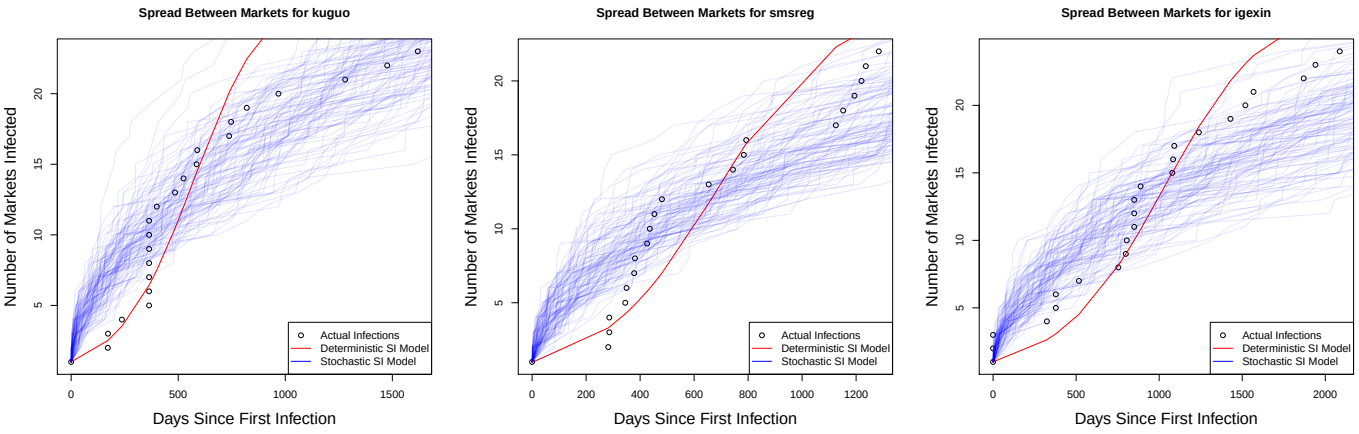
### 4.4 Malware Spread Prediction between Markets — Prediction for P2 and P3

As described in Section 3.2, we use an SI model to help us understand the spread of malware between markets. In this section, we evaluate our spread model on DS2 from two aspects: the spread velocity of malware across markets, and the spread order of malware between markets.

#### 4.4.1 *Spread velocity of malware — Prediction for P2*

We aim to evaluate our spread model on infection velocity of malware across markets, i.e., how fast one malware family spreads to all of the markets. First, we fit a deterministic form of the model using least squares estimation and then we simulate that model stochastically, taking into account the mutual information between markets. Figure 9 shows the spread models for three malware families—KUGUO, SMSREG, and IGEXIN. The three families vary in the number of contained malware samples, in particular, KUGUO has 25,002 samples within the range of [20,000, $\infty$), SMSREG contains 10,737 samples in the range of [10,000, 20,000), and IGEXIN owns 3,926 samples in the range of [0, 10,000). As shown above, $\beta_{kuguo}$=5.810E-3, $\beta_{smsreg}$=4.732E-3, $\beta_{igexin}$=3.189E-3 are the market infection rates per day of these three families, respectively. To evaluate the performance of the deterministic model on DS2, we compute the RSE values for 100 families that has the most number of malware. Of all computed RSE values, the average is 2.24, 1st quartile is 1.23, 2nd quartile is 2.06, 3rd quartile is 3.15, and 4th quartile is 6.48. The results show that the deterministic model can closely approximate the real world data.

In addition, we employ the Pearson correlation coefficient to quantify and assess the prediction results. For Equation 10, we feed $X$ with the predicted infection time of markets where $x_i$ is the predicted number of days to infect $i$ markets, and $Y$ with the actual infection time of markets where $y_i$ is the actual number of




(a) The spread model of malware *kuguo* (b) The spread model of malware *smsreg* (c) The spread model of malware *igexin*

Fig. 9: Three samples to illustrate the spread model for a specific malware family. The x axis denotes the number of days since the first infection, and the y axis denotes the number of infected markets by this malware.

days to infect $i$ markets. We focus on the top 100 malware families that exist in all 1,149 Android markets, and plot the distribution of correlations in Figure 11a. The overall correlations achieve an average of 0.89 and a median of 0.91. Of these families, there are 27 families that contain at least 1,000 samples plotted as "$\geq$ 1000", and the remaining 73 families with less 1,000 samples plotted as "$<$ 1000". The two sets of families have the very close mean-median results with $0.89, 0.90$ and $0.89, 0.92$, respectively.

To some extent, the infection rate of malware reveals its infectivity and popularity in the lifecycle. Taking the malware KUGUO and IGEXIN as an example, malware KUGUO has a larger infection rate than malware IGEXIN, i.e., KUGUO spends less days to infect all these markets as shown in Figure 9a and 9c. According to the security reports [43, 44], KUGUO with "High" risk impact is more aggressive than IGEXIN with "Low" risk impact. Besides the malicious behaviors of stealing users' sensitive information that IGEXIN also contains, KUGUO can carry out a malicious promotion of other apps for making profits or malware spread. It leads that KUGUO is more likely to be used by cybercriminals to achieve their goals. Consequently, KUGUO is observed to have a faster growth and spread between markets.

#### 4.4.2 *Spread order between markets — Prediction for P3*

As well as predicting the times at which markets will be infected, we can also predict the order of infection. By fixing the first five markets to be infected and simulating infection of the remaining markets (averaged over 100 trials), we can assess how similar the predicted order of infection is to the actual data. We use the Pearson correlation coefficient to quantify and assess the prediction results. For Equation 10, we feed $X$ with the predicted infection order of markets by our spread model where $x_i$ is the predicted infection order for i-th market, and $Y$ with the actual infection order of markets where $y_i$ is the actual infection order for i-th market.

We select the top 100 malware families that present in all 1,149 Android markets, and plot the distribution of correlations in Figure 11b. The overall correlations achieve an average of 0.66 and a median of 0.70. We also plot the distribution of correlations for the 27 families that contain at least 1,000 samples (i.e., $\geq$ 1000), and the remaining 73 families with less 1,000 samples (i.e., $<$ 1000). As shown in Figure 11, the model is more suited to the families with more samples: the mean and median are 0.74 and 0.76, respectively, in the families with at least 1000 samples, while they are 0.64 and 0.66, respectively, for the families with less samples. Specifically, for malware KUGUO, the largest malware families in DS2, the Pearson correlation coefficient between the simulated and actual order was 0.801 (a strong correlation) as shown in Figure 10.

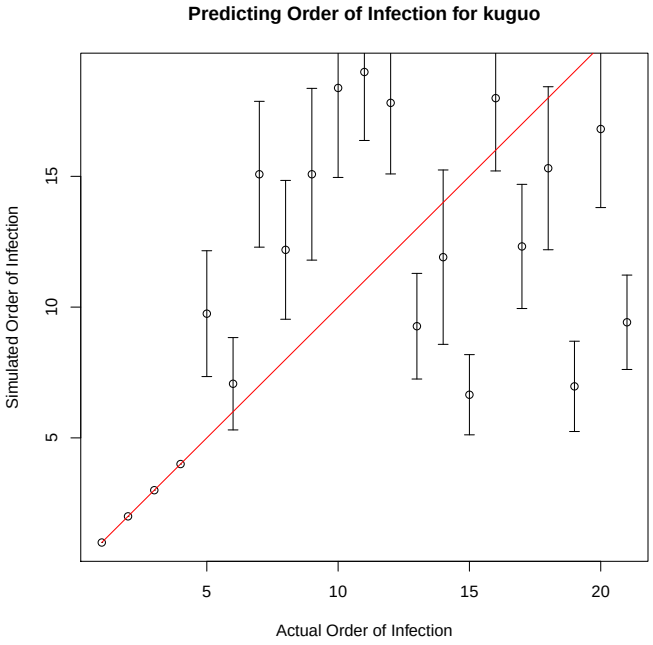



Fig. 10: The infection order of malware *kuguo* between markets. Each bar illustrates its mean and standard deviation of the prediction order for the corresponding market.

The infection order of markets reveals the dissemination of malware and the app transmissions between markets (behavior 2, and 6 in Figure 1). The predicted infection order can help the market administrators prepare in advance to take necessary measures to prevent the spread of malware.

**Significance.** The SI model provides a good representation of the spread between markets and can be used to predict future infections. By applying information learned from one family to simulate infection in others, we show our approach can be used to make predictions about new malware families.

## 5 Discussion

This section discusses the applicable areas of our work, and its limitations.

### 5.1 Applications

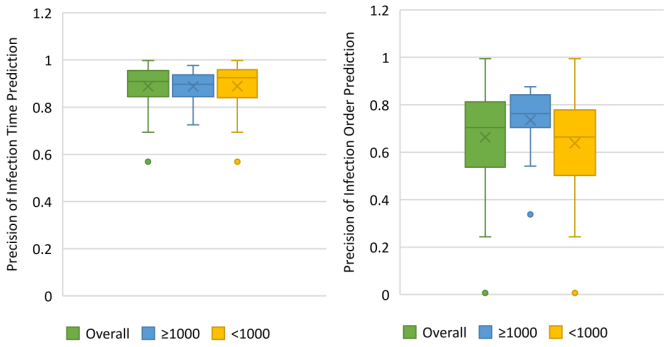
**Prediction of malware propagation and spread.** Understanding malware propagation and spread accurately can benefit to both app markets and Android users. Our models, which portray malware's dynamics, can be leveraged by market administrators to predict the outbreak and proliferation of malware [45]. Endeavors and resources (e.g., man power) are subsequently designated against upcoming threats. The prediction results can also make normal Android users aware of the security risk of installing apps in one market. As a result, our models along with conclusive insights can aid in protecting the Android ecosystem, and reducing the damages caused by malware.

**Security assessment for Android markets.** Android markets adopt diverse vetting processes to detect malware among uploaded apps, and alleviate the damages caused to users. For instance, Google launched Bouncer [46] in 2012 to automatically scan Android apps in Google Paly, and then engaged a team of security experts to identify the violations of uploaded apps [47] in 2015. Besides, third-party markets rely on the scanning results of commercial anti-virus software to eliminate malware from markets. However, the capabilities of these protection mechanisms are not clear. Therefore, it is desirable to provide a security metric to evaluate these markets. The curvature of the malware propagation curve in Section 3.1 shows to some extent the resistance of Android markets to malware. The higher the curvature is, the more susceptible the market is to new variants of malware. As shown in Figure 3, although Google Play is recognized as one of the most secure Android markets taking rigorous security inspections to apps, it is still susceptible to new malware due to its largest number of users in the Android world. Hence, before applying intensive malware detection (e.g., ICCDetector [48] and others [22, 49, 50]) on the market side, a security assessment with our approach is desired.

**Facilitate the understanding of malware.** The propagation of Android malware within a market varies from the belonged malware families. Our overall malware propagation model can reveal at least two sorts of malware characteristics: aggressiveness of malware, i.e., how much harm the malware can cause. In general, more aggressive malware can cause more severe damages and loss; the evasiveness of malware, i.e., how likely the malware can evade the detection of anti-virus software. There exist many techniques to prevent the detection, such as obfuscation [51, 52], dynamic code loading and execution [53], and app packing [54].

### 5.2 Limitations

**Family labeling accuracy**. In this paper, we propose a family-specific growth model and spread model between markets. Therefore, the fitness of models to some extent, relies on the accuracy of family labeling of Android malware. Given one malware samples, off-the-shelf anti-virus software probably reports different family names. Therefore, AVClass is used to normalize malware families in our study. Although its evaluated precision is high with 87.2%-95.3%, there are some cases in which it is incapable to differentiate family version, and it introduces noise and disturbance into the model and causes inaccuracies that are hard to measure. Moreover, finer-grained family labels necessitate manual confirmation in most cases [55, 56]. We have alleviated this problem by focusing on markets and families that have a large number of samples, so as to minimize the inaccuracy of mis-labeling a small number of samples.




(a) Infection Time (b) Infection Order

Fig. 11: Performance measurement for infection time and order prediction. "Overall" depicts the quartile for the top 100 malware families, "≥1000" for the 27 malware families with more than, or equal to, 1000 samples, and "<1000" for the 63 malware families with less than 1000 samples.

**Insufficiency of removal data**. We used the data in AndRadar to construct our removal model. AndRadar tracked more than 1,500 app deletions across 16 markets over a period of three months. However, it has only 6 markets in common with the data we collected, and a duration of three months, which is not long in the active life of Android apps since 2008. Therefore, the removal data may lack generality which would cause inaccuracy of the extracted model. However, this drawback could be mitigated by collecting more removal data in more popular markets with a longer duration.

**Assumption validity in the model**. We have minimized the limitations of our model as far as possible. The first (simple) version of the model we evaluated did not take into account heterogeneities in behaviour between different families of malware - hence we proposed a family-specific growth model. However, this version of the model still assumes the "law of mass action" [57], i.e., the rate at which infections occur is directly proportional to the number of malware-infected apps. Whilst this is true in the asymptotic sense (i.e. over an infinite number of trials), the actual result may be a little different. To address this, we extended our model to consider how app markets are connected in terms of the spread of malware, and simulated the distribution of possible infection events using a stochastic model. By applying a stochastic model, we acknowledge it is impossible to perfectly predict the order and frequency at which malware will spread between app markets (due to the inherent randomness of malware growth). However, we are able to show with considerable accuracy, how likely different infection scenarios are to occur.

## 6 Related Work

In this section, we summarize relevant research works that model the propagation and spread behaviors for (Android) malware, and discuss the differences with our work.

*Epidemic Model* characterises the spreading features of malware. Thommes and Coates [58] propose a deterministic epidemic model for a P2P virus that facilitates protection from viruses. Kang and Prakash [23, 59] propose an ESM model to present the infection pattern of malware across 1.4 million hosts. Zou *et al.* [60] and Liu *et al.* [24] leverage the epidemic model to simulate and model the propagation of Internet email worm and malware, respectively. In this study, Epidemic Model (EM) is also applied to model the between-market malware spread. However, our study is not a simple application of EM or fitting the model with the data. We consider the within-market propagation and incorporate EM with the connection between markets (the mutual information among them). It is a crucial difference with these studies that we consider the mobility of apps between Android markets, i.e., the mutual information in Section 3.2.

*Markov Chains* are widely used to model the spreading characteristics of malware in networks. Garetto *et al.* [12] present a modelling methodology based on Interactive Markov Chains that can capture the impact of the underlying topology on the spreading characteristics of malware. Mieghem and Omic [61, 62] employ $N$-intertwined Markov Chains to depict the transition of viruses in networks. Chen and Ji [63] derive a spatial-temporal random process by combing an independent model and a Markov model to identify the statistical dependence of malware propagation. In the dataset of malware samples, since we have no actual correlation probabilities among markets, for any app (no matter benign or malicious) on a certain market, there is no way to test the probabilities that it will be transited to the other markets. Simply put, without the mobility factors or topology among markets, it is extremely hard to model the between-market propagation with Markov chains.

*Information Diffusion* is a field encompassing technique for social media mining [64]. Yang and Leskovec [65] develop a linear influence model to depict the spread of information via social media. Information diffusion models the phenomena in which an idea or behavior gets popular due to the influence of others. It has also been applied to establish a spread model for malware [23], which considers the malware as the information that is spread among networks (e.g., social networks). In this study, information diffusion is not adopted, as the topology of and mobility factors between markets are not explicitly known. Instead, the mutual information among markets is calculated to serve as the potential connection between any two markets.

Recently, there have been some studies on Android malware spread. Yu *et al.* [15] formulate the problem of malware propagation as a network, and establish a two-layer epidemic model from network to network. They report several distributions of given malware in different stages of malware growth. However, this study uses an obsolete dataset MALGENOME [2] collected from 2010 to 2011, which cannot characterize the spread of the latest malware. Peng *et al.* [5] summarize several Susceptible-Infected (SI) models as well as existing problems. They suggest that a social network and human behavior are critical components to resolve the malware spread modelling problem. Hence, Yu *et al.* [15] and Peng *et al.* [5] apply SI models to depict the malware propagation among end devices in mobile networks. In other words, these two studies do not provide information on the market-oriented propagation.

In this study, we propose an approach based on an epidemic model, as the network topology among market servers is unclear and social media data is not available. Different from previous study [15], we focus on factors of malware spread within and between Android markets, rather than network topology. Inspired by GLEaM (Global Epidemic and Mobility Modeler) [66] and [5], malware spread can be affected by the mobility of apps between different Android markets. Therefore, we first investigate the mutual information among markets, and then build a spread model based on the mutual information.

## 7 Conclusion

In this paper, we conducted the first attempt to explore the mechanism of malware propagation and spread. Based on the understanding, we studied the spread of Android malware within and between markets from a huge number of Android apps infected with malware, and proposed comprehensive models to simulate the spread behavior of Android malware. The model benefits the prediction of Android malware, and is able to warn markets when to take security measures to resist to emerging malware. We carried out a comprehensive experiment to evaluate our proposed models. The results show that our models can provide robust and accurate predictions for the outbreak of Android malware and the propagation between different markets.

We believe that our work contributes significantly to the security community on malware prevention and prediction by studying its characteristics and trends of increase and spread. In addition, we envision that more works can be conducted to further describe the dynamics of Android malware in future. For example, similar with the cyclical contagions of an infectious disease [67], Android malware may also present a periodical increase in its lifecycle due to the emergence of new malware variants and the upgrade of detection techniques. This study can unveil the underground industry of Android malware, and evaluate the resistance of detection techniques to these variants.

## Acknowledgments

We appreciate the anonymous reviewers for their valuable comments. This work is supported in part by International Cooperation Program on CyberSecurity, administered by SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences, China (No. SNHTBH-2017110681). Dr. Yinxing Xue is supported by CAS Pioneer Hundred Talents Program of China.

## References

[1] M. Isaac, "Android OS Now World's Leading Smartphone Platform," https://www.wired.com/2011/01/android-os-leading-smartphone/, 2017.

[2] Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution," in *IEEE Symposium on Security and Privacy*, 2012, pp. 95–109.

[3] G. Meng, Y. Liu, J. Zhang, A. Pokluda, and R. Boutaba, "Collaborative security: A survey and taxonomy," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, pp. 1:1–1:42, Jul. 2015.

[4] K. Chandrasekar, G. Cleary, O. Cox, H. Lau, B. Nahorney, B. O. Gorman, D. OBrien, S. Wallace, P. Wood, and

C. Wueest, "Internet Security Threat Report," Symantec Inc., California, USA, Tech. Rep. ISTR-22-2017, 2017.

[5] S. Peng, S. Yu, and A. Yang, "Smartphone Malware and Its Propagation Modeling: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 2, pp. 925–941, 2014.

[6] CheckPoint, "The Judy Malware: Possibly the largest malware campaign found on Google Play," https://blog.checkpoint.com/2017/05/25/judy-malware-possibly-largest-malware-campaign-found-google-play/, 2017.

[7] G. Zyba, "Mobile Malware Propagation and Defense," Ph.D. dissertation, School of Computer Science and Engineering, University of California, San Diego, San Diego, California, USA, 2013.

[8] G. Serazzi and S. Zanero, *Computer Virus Propagation Models*. Springer Berlin Heidelberg, 2004.

[9] J. Kirk, "Android root malware widespread in third-party app stores," https://www.pcworld.com/article/3032296/security/android-root-malware-widespread-in-third-party-app-stores.html, 2016.

[10] D. Steele, "Third Party App Stores Blamed for Malware Infections," https://www.androidheadlines.com/2016/01/third-party-app-stores-blamed-for-malware-infections.html, 2016.

[11] L. Li, D. Li, T. F. Bissyandé, J. Klein, Y. L. Traon, D. Lo, and L. Cavallaro, "Understanding android app piggybacking: A systematic study of malicious code grafting," *IEEE Trans. Information Forensics and Security*, vol. 12, no. 6, pp. 1269–1284, 2017.

[12] M. Garetto, W. Gong, and D. F. Towsley, "Modeling Malware Spreading Dynamics," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2003.

[13] H. Okamura, H. Kobayashi, and T. Dohi, "Markovian Modeling and Analysis of Internet Worm Propagation," in *ISSRE*, 2005, pp. 149–158.

[14] C. Nowzari, V. M. Preciado, and G. J. Pappas, "Analysis and Control of Epidemics: A survey of spreading processes on complex networks," *IEEE Control Systems Magazine*, 2015.

[15] S. Yu, G. Gu, A. Barnawi, S. Guo, and I. Stojmenovic, "Malware Propagation in Large-Scale Networks," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 27, no. 1, pp. 170–179, 2015.

[16] M. Lindorfer, S. Volanis, A. Sisto, M. Neugschwandtner, E. Athanasopoulos, F. Maggi, C. Platzer, S. Zanero, and S. Ioannidis, "AndRadar: Fast Discovery of Android Applications in Alternative Markets," in *DIMVA*, 2014, pp. 51–71.

[17] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. D. Favera, and A. Califano, "ARACNE: An Algorithm for the Reconstruction of Gene Regulatory Networks in a Mammalian Cellular Context," *BMC Bioinformatics*, vol. 7, no. S-1, 2006.

[18] C. Szongott, B. Henne, and M. Smith, "Evaluating the threat of epidemic mobile malware," in *8th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2012, Barcelona, Spain, October 8-10, 2012*, 2012, pp. 443–450.

[19] C. Yang, J. Zhang, and G. Gu, "Understanding the market-level and network-level behaviors of the android malware ecosystem," in *37th International Conference on Distributed Computing Systems*, 2017.

[20] L. H. Newman, "How Malware Keeps Sneaking Past Google Play's Defenses," https://www.wired.com/story/google-play-store-malware/, 2017.

[21] W. Zhou, Y. Zhou, M. Grace, X. Jiang, and S. Zou, "Fast, Scalable Detection of "Piggybacked" Mobile Applications," in *CODASPY*, 2013.

[22] G. Meng, Y. Xue, Z. Xu, Y. Liu, J. Zhang, and A. Narayanan, "Semantic modelling of android malware for effective malware comprehension, detection, and classification," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, ser. ISSTA 2016. New York, NY, USA: ACM, 2016, pp. 306–317.

[23] C. Kang, N. Park, B. A. Prakash, E. Serra, and V. S. Subrahmanian, "Ensemble Models for Data-driven Prediction of Malware Infections," in *The International Conference on Web Search and Data Mining*, 2016, pp. 583–592.

[24] B. Liu, W. Zhou, L. Gao, H. Zhou, T. H. Luan, and S. Wen, "Malware Propagations in Wireless Ad Hoc Networks," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, pp. 1–11, 2017.

[25] M. R. Faghani and U. T. Nguyen, "Modeling the Propagation of Trojan Malware in Online Social Networks," *CoRR*, vol. abs/1708.00969, 2017. [Online]. Available: http://arxiv.org/abs/1708.00969

[26] M. A. Hixon and D. W. Johnson, "Density Dependence and Independence," *Encyclopedia of Life Sciences*, dec 2009.

[27] A. V. Hill, "The possible effects of the aggregation of the molecules of hæmoglobin on its dissociation curves," *The Journal of Physiology*, vol. 40, pp. i–vii, January 1910.

[28] J. Giraldo, N. M. Vivas, E. Vila, and A. Badia, "Assessing the (a)symmetry of concentration-effect curves: empirical versus mechanistic models," *Pharmacology & Therapeutics*, vol. 95, pp. 21 – 45, 2002.

[29] P. Wang, M. C. González, C. A. Hidalgo, and A. L. Barabási, "Understanding the spreading patterns of mobile phone viruses," *Science*, vol. 324, pp. 1071–1076, 2009.

[30] S. P. Maher, A. M. Kramer, J. T. Pulliam, M. A. Zokan, S. E. Bowden, H. D. Barton, K. Magori, and J. M. Drake, "Spread of white-nose syndrome on a network regulated by geography and climate," *Nature Communications 3*, no. 1306, 2012.

[31] E. Brooks-Pollock, G. O. Roberts, and M. J. Keeling, "A dynamic model of bovine tuberculosis spread and control in Great Britain," *Nature 511*, pp. 228–231, 2014.

[32] K. Borovkov, *Elements of Stochastic Modelling*. World Scientific Publishing Co Inc, 2014.

[33] M. Patrick, A. P. Craig, N. J. Cunniffe, M. Parry, and C. A. Gilligan, "Testing stochastic software using pseudo-oracles," in *ISSTA*, 2016, pp. 235–246.

[34] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *The Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.

[35] J. O. Rawlings, S. G. Pantula, and D. A. Dickey, *Applied Regression Analysis*. John Wiley, 1998.

[36] C. J. Willmott and K. Matsuura, "On the use of dimensioned measures of error to evaluate the performance of spatial interpolators," *International Journal of Geographical Information Science*, vol. 20, no. 1, pp. 89–102, 2006.

[37] H. M. Walker, "Degrees of freedom," *Journal of Educational Psychology*, vol. 31, no. 4, pp. 253–269, 1940.

[38] Rogers and Nicewander, "Thirteen Ways to Look at the Correlation Coefficient."

[39] G. Meng, Y. Xue, J. K. Siow, T. Su, A. Narayanan, and Y. Liu, "AndroVault: Constructing Knowledge Graph from Millions of Android Apps for Automated Analysis," *CoRR*, vol. abs/1711.07451, 2017. [Online]. Available: http://arxiv.org/abs/1711.07451

[40] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "AVclass: A Tool for Massive Malware Labeling," in *Research in Attacks, Intrusions, and Defenses - 19th International Symposium, RAID 2016, Paris, France, September 19-21, 2016, Proceedings*, 2016, pp. 230–253.

[41] M. Dataset, "A Large-scale Real-world Android Malware," https://sites.google.com/site/malwarepropagation/, 2017.

[42] McAfee Inc., "Mobile Threat Report: What's on the Horizon for 2016," Tech. Rep., 2016.

[43] Symantec, "Android.kuguo," https://www.symantec.com/security_response/writeup.jsp?docid=2014-040315-5215-99, 2015.

[44] ——, "Android.igexin," https://www.symantec.com/security_response/writeup.jsp?docid=2015-032606-5519-99, 2015.

[45] S. Shin, G. Gu, N. Reddy, and C. P. Lee, "A Large-Scale Empirical Study of Conficker," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 676–690, April 2012.

[46] H. Lockerheimer, "Android and Security," http://googlemobile.blogspot.sg/2012/02/android-and-security.html, 2012.

[47] E. Kim, "Creating Better User Experiences on Google Play," https://android-developers.googleblog.com/2015/03/creating-better-user-experiences-on.html, 2015.

[48] K. Xu, Y. Li, and R. H. Deng, "Iccdetector: Icc-based malware detection on android," *IEEE Trans. Information Forensics and Security*, vol. 11, no. 6, pp. 1252–1264, 2016.

[49] A. Narayanan, G. Meng, Y. Liu, J. Liu, and L. Chen, "Contextual weisfeiler-lehman graph kernel for malware detection," in *2016 International Joint Conference on Neural Networks (IJCNN)*, July 2016, pp. 4701–4708.

[50] A. Narayanan, M. Chandramohan, L. Chen, and Y. Liu, "A multi-view context-aware approach to android malware detection and malicious code localization," *Empirical Software Engineering*, vol. 23, no. 3, pp. 1222–1274, Jun 2018.

[51] V. Rastogi, Y. Chen, and X. Jiang, "DroidChameleon: Evaluating Android Anti-malware Against Transformation Attacks," in *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, 2013, pp. 329–334.

[52] G. Meng, Y. Xue, M. Chandramohan, A. Narayanan, Y. Liu, J. Zhang, and T. Chen, "Mystique: Evolving android malware for auditing anti-malware tools," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (AsiaCCS)*, 2016, pp. 365–376.

[53] Y. Xue, G. Meng, Y. Liu, T. H. Tan, H. Chen, J. Sun, and J. Zhang, "Auditing Anti-Malware Tools by Evolving Android Malware and Dynamic Loading Technique," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 7, pp. 1529–1544, July 2017.

[54] L. Xue, X. Luo, L. Yu, S. Wang, and D. Wu, "Adaptive Unpacking of Android Apps," in *Proceedings of the 39th International Conference on Software Engineering*, ser. ICSE '17, 2017, pp. 358–369.

[55] M. Hurier, G. Suarez-Tangil, S. K. Dash, T. F. Bissyandé, Y. L. Traon, J. Klein, and L. Cavallaro, "Euphony: Harmonious Unification of Cacophonous Anti-virus Vendor Labels for Android Malware," in *Proceedings of the 14th International Conference on Mining Software Repositories*, ser. MSR '17. Piscataway, NJ, USA: IEEE Press, 2017, pp. 425–435. [Online]. Available: https://doi.org/10.1109/MSR.2017.57

[56] F. Wei, Y. Li, S. Roy, X. Ou, and W. Zhou, "Deep Ground Truth Analysis of Current Android Malware," in *Detection of Intrusions and Malware, and Vulnerability Assessment*. Cham: Springer International Publishing, 2017, pp. 252–276.

[57] E. A. Guggenheim, "Textbook errors IX: More About the Laws of Reaction Rates and of Equilibrium," *Journal of Chemical Education*, vol. 33, pp. 544–545, 1956.

[58] R. W. Thommes and M. Coates, "Epidemiological Modelling of Peer-to-Peer Viruses and Pollution," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2006.

[59] B. A. Prakash, "Prediction Using Propagation: From Flu Trends to Cybersecurity," *IEEE Intelligent Systems*, vol. 31, no. 1, pp. 84–88, 2016.

[60] C. C. Zou, D. Towsley, and W. Gong, "Modeling and Simulation Study of the Propagation and Defense of Internet E-mail Worms," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 2, pp. 105–118, 2007.

[61] P. V. Mieghem, J. Omic, and R. E. Kooij, "Virus Spread in Networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 1–14, 2009.

[62] P. Van Mieghem, "The N-intertwined SIS Epidemic Network Model," *Computing*, vol. 93, no. 2, pp. 147–169, 2011.

[63] Z. Chen and C. Ji, "Spatial-temporal Modeling of Malware Propagation in Networks," *IEEE TNN*, vol. 16, no. 5, pp. 1291–1303, 2005.

[64] R. Zafarani, M. A. Abbasi, and H. Liu, *Social Media Mining: An Introduction*. Cambridge University Press, 2014.

[65] J. Yang and J. Leskovec, "Modeling Information Diffusion in Implicit Networks," in *IEEE International Conference on Data Mining (ICDM)*, 2010, pp. 599–608.

[66] D. Balcan, B. Gonçalves, H. Hu, J. J. Ramasco, V. Colizza, and A. Vespignani, "Modeling the spatial spread of infectious diseases: The GLobal Epidemic and Mobility computational model," *Journal of Computer and Security*, vol. 1, no. 3, pp. 132–145, 2010.

[67] F. Córdova-Lepe, G. Robledo, M. Pinto, and E. González-Olivares, "Modeling pulse infectious events irrupting into a controlled context: A SIS disease with almost periodic parameters," *Applied Mathematical Modelling*, vol. 36, no. 3, pp. 1323 – 1337, 2012.

[68] L. H. Newman, "How Malware Keeps Sneaking Past Google Play's Defenses," https://www.wired.com/story/google-play-store-malware/, September 2017.

# 8 Appendix

## 8.1 Security Protocols of Android Markets

To further evaluate our experimental results and verify the conclusions drawn from the analysis, we investigated the security protocols employed by different Android markets. In particular, we demystify the processes of app uploading (related to behavior 1, 2, 6), removal (related to behavior 5), and backend engines for

malware detection (related to behavior 3, 4). All the investigation results are presented at Table 6.

**App Uploading**. Authorized developers are allowed to upload Android apps into markets. In terms of security concerns, different markets have adopted varying regulations and restrictions for developers. The restrictions can be categorized into three levels from low to high: *free* level does not impose any restrictions to developers or attackers, which means anyone can upload anything into the markets. Markets like GERJAR and FREEWARELOVERS have relied on this restriction; *verification* level only allows authorized developers to upload apps. For example, most of markets located in China (e.g., ANZHI, MUMAYI and APPCHINA) forces developers to provide valid and lawful certifications. Generally, individual developers have to provide their identity cards (sometimes one photography is mandatory in which the developer without nothing covering face needs to hold his/her identity card). Enterprise developers have to provide the details of the legal person of the company as well as the business licenses acquired from the local government. In such case, it raises the difficulties of attackers circulating malware stealthily. Specifically, GOOGLE PLAY necessitates a bank card bound to the developer account. In addition, *Safety Copyright Service Platform*[2] is an official agent for mobile applications providing a unified certification service. Developers who obtain certifications from it can upload apps into many partner markets such as APPCHINA, GFAN and Mumayi; *prohibition* level does not receive apps from developers. Instead, editors or administrators of markets put apps on the shelves themselves. For example, markets APK20 and MOB duplicate apps in GOOGLE PLAY or other markets. Markets ANDROIDDRAWER and APKPURE accept requests of developers publishing their products which are already existing in GOOGLE PLAY, whereas all apps are selected and thereby published by market editors.

**App Removal**. Apps are prone to being compulsively removed upon violating the regulations of markets. Generally, the regulations stems from four aspects: the quality of apps cannot meet the requirements, such as recurring crashes, low-resolution images and unresponsive GUIs; apps contains malicious code for example privacy harvesting, privilege escalation, and aggressive advertising; apps contain illegal content that violates the local laws or copyrights of specific works. On the other hand, some markets allow developers to withdraw their apps following a predesigned procedure. Although malware may be removed because of other reasons (e.g., low quality or violation of laws) rather than its maliciousness. In this study, we avoid to model the affect of each factor on app removal, but approximate the removal ratio of one malware sample in a specific market statistically.

**Security Checking**. In our investigation, most of markets have employed their own security inspection. For example, GOOGLE PLAY has equipped with one comprehensive and scalable built-in malware detection system that reduces the ratio of potential harmful apps considerably. Even though, few of malicious apps can still impact millions of Android users [68]. In addition, the app scanning engine periodically checks on-the-shelf apps in case of misses. Some markets resort to specialized anvi-virus engines for malware detection. For example, market EOEMARKET relies on four anti-virus engines: 360 Safeguard, Anguanjia, Tencent and KingSoft Antivirus, while MUMAYI relies on 360 Safeguard and Tencent. In addition, we found that markets GETJAR and FREEWARELOVERS do not have explicit security scanning engines running behind. Markets APK20, ANDROIDDRAWER and APKPURE presumably overlook additional security check for the apps which are from GOOGLE PLAY. It is worthy mentioning that a number of Chinese Android markets ask developers to get trustworthy certificates for their products from two official agents — *China National App Administration Center (CNAAC)*[3], *ANVA* White List[4]. CNAAC issues a trustworthy certificate for each app that passes security test, and ANVA certifies benign apps according to detection results by security vendors.

**Remark.** This investigation has explored all behaviors occurring amongst developers, attackers, and app markets. It serves as an important evidence for the correctness of Figure 1. More specifically, developers and attackers can upload apps or malware into markets conforming to the regulations of target markets (behaviors 1, 2). The market editors can also transfer some apps from other markets into their own (e.g., APK20 and ANDROIDDRAWER), which confirms the existence of behavior 6. Moreover, markets commonly adopt security detection techniques to identify malware (behaviors 3, 4) and subsequently get rid of them (behavior 5).

2. http://www.safebq.com/

3. http://www.cnaac.org.cn/

4. https://white.anva.org.cn/

TABLE 6: All markets to study in DS2.

| Market | Website | Region | App Uploading | App Removal | Security Checking |
|---|---|---|---|---|---|
| GOOGLEPLAY | https://play.google.com/store?hl=en | US | developer (card)[2] | owner[8], editor | Built-in system |
| QQ | http://sj.qq.com/myapp | CN | developer (id)[3] | owner (request),[9] editor | AV (Tencent),[10] human[11] |
| ANZHI | http://anzhi.com | CN | developer (id), editor [4] | owner (request), editor | AV(Tencent, 360, KingSoft, etc) |
| GETJAR | http://www.getjar.com | LT | developer | owner, editor | – |
| MUMAYI | http://www.mumayi.com | CN | developer (safebq)[5] | owner(request), editor | AV(360, Tencent) |
| XIAOMI | http://app.mi.com | CN | developer (id) | owner(request), editor | AV, human |
| APK20 | http://www.apk20.com | US | editor (markets)[6] | owner (request), editor | AV |
| HIAPK[5] | http://www.hiapk.com | CN | – | – | – |
| EOEMARKET | http://www.eoemarket.com | CN | devloper (id), editor | owner (request), editor | AV (360, Anguanjia, Tencent, KingSoft) |
| APPCHINA | http://www.appchina.com | CN | developer (safebq) | owner (request), editor | ANVA White List,[12] CNAAC[13] |
| COOLAPK | http://coolapk.com | CN | developer (id) | – | – |
| APKMIRROR | http://www.apkmirror.com | US | editor, developer | editor | signature protection[14] |
| FLYME | http://app.flyme.cn | CN | developer (id) | owner (request) | AV (AVL Mobile Security, etc) |
| GFAN | http://apk.gfan.com | CN | developer (safebq) | owner (request), editor | AV (360, Tencent, LBE) |
| CNMO | http://app.cnmo.com/ | CN | – | – | – |
| BAIDU | http://shouji.baidu.com | CN | developer (id) | owner (request), editor | ANVA White List |
| ANDROIDDRAWER | http://www.androiddrawer.com | –[1] | editor (google play)[7] | editor | – |
| WANGYI | http://m.163.com/android/index.html | CN | – | – | – |
| FDROID | https:/f-droid.org | US | developer | – | – |
| FREEWARELOVERS | http://www.freewarelovers.com/android | DE | developer | – | – |
| MOB | http://mob.org | US | developer, editor (google play) | – | – |
| WANDOUJIA | http://www.wandoujia.com/apps | CN | developer (id) | owner (request), editor | AVNA White List |
| APKPURE | https://apkpure.com | US | developer, editor (google play) | editor | signature protection |
| CHINAMOBILE | http://mm.10086.cn | CN | developer (id) | owner (request), editor | AVNA White list |

[1] –: not available;
[2] developer (card): developers have to be verified by providing a valid bank;
[3] developer (id): individual developers are verified by identity card, and corporate developers verified by business license and legal person;
[4] editor: the editors of market;
[5] developer (safebq): developers that are verified by Safety Copyright Service Platform;
[6] editor (markets): the editors download apps from other markets;
[7] editor (google play): the editors fetch apps from GOOGLE PLAY;
[8] owner: the owner of the app;
[9] owner (request): the owner of the app has to send a request for the withdrawal;
[10] AV (*): Apps undergo the detection by anti-virus software;
[11] human: Apps undergo human inspection;
[12] ANVA White List: Apps have to be evaluated in the ANVA white list;
[13] CNAAC: Apps have to be verified in China National App Administration Center;
[14] signature protection: apps that are verified by its cryptographic signature;