SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2013; **6**:33–48
Published online 18 April 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.528

RESEARCH ARTICLE

What you see predicts what you get—lightweight agent-based malware detection

Wei Wang*, Ilona Murynets, Jeffrey Bickford, Christopher Van Wart and Gang Xu

AT&T Security Research Center, 33 Thomas Street, New York, NY, U.S.A.

ABSTRACT

Because of the always connected nature of mobile devices, as well as the unique interfaces they expose, such as short message service (SMS), multimedia messaging service (MMS), and Bluetooth, classes of mobile malware tend to propagate using means unseen in the desktop world. In this paper, we propose a lightweight malware detection system on mobile devices to detect, analyze, and predict malware propagating via SMS and MMS messages. We deploy agents in the form of hidden contacts on the device to capture messages sent from malicious applications. Once captured, messages can be further analyzed to identify a message signature as well as potentially a signature for the malicious application itself. By feeding the observed messages over time to a latent space model, the system can estimate the current dynamics and predict the future state of malware propagation within the mobility network. One distinct feature of our system is that it is lightweight and suitable for wide deployment. The system shows a good performance even when only 10% of mobile devices are equipped with three agents on each device. Moreover, the model is generic and independent of malware propagation schemes. We prototype the system on the Android platform in a universal mobile telecommunications system laboratory network to demonstrate the feasibility of deploying agents on mobile devices as well as collecting and blocking malware-carrying messages within the mobility network. We also show that the proposed latent space model estimates the state of malware propagation accurately, regardless of the propagation scheme.

KEYWORDS

malware detection; malware propagation; mobile malware; latent space model

***Correspondence**

Wei Wang, Security Research Center, AT&T, 33 Thomas Street, New York, NY, U.S.A.
E-mail: wei.wang.2@att.com

1. INTRODUCTION

In recent years, mobile devices have grown increasingly complex and more prominent than typical full-fledged PC systems. They run operating systems and software at the same complexity level as desktop machines. However, few smart phone users adopt anti-malware software on their devices because of unawareness or lack of available options. Because of the worldwide adoption of smart phones today, an increasing number of devices are now vulnerable to various complex attacks.

Over the past decade, different forms of malware such as viruses, worms, and trojans have plagued mobile phones throughout the world. Mobile malware has been evolving rapidly for various mobile operating systems at an exponential rate [1]. In 2004, the first mobile worm was released for the Symbian operating system. Cabir [2] spread via Bluetooth, continuously scanning for other Bluetooth enabled devices. Although it was quite harmless, battery life decreased because of its continuous scanning nature. A variant called Mabir [3] was later released the same year, which spread via not only Bluetooth but also multimedia messaging service (MMS). When an infected phone received a text message, Mabir replied to the originating phone number with the worm attached potentially infecting them as well. More advanced mobile malware, such as Commwarrior [4] and Andriod malware [5], sent propagation messages to all users listed in an infected phone's contact book. Attackers also tend to use short message service (SMS) messages along with social engineering techniques in attempts to infect a device, for example, sending an SMS message with a malicious link to a random contact in the contact book. Users infected with mobile malware can be impacted significantly in various ways such as loss of personal data, financial loss, application and full-system crashing, battery discharge, denial of service attacks on the service provider, and the like. In order to protect users and the mobility network, it is important for a network provider to detect malware propagation in a quick manner and forecast the state of infection within

the network as an overview picture. Such knowledge is the foundation for countermeasures such as patching devices and preventing malware from propagating further.

Various security companies, such as McAfee, Symantec, and Kaspersky, supply mobile versions of commercial antivirus software. These solutions are typically similar to their desktop counterparts and incur a high cost when executed on a resource-constrained device such as a smart phone [6]. Recent research also proposes detection mechanisms to defend against the increasing mobile malware threat [7–12], although some incur a high resource overhead and energy cost. Moreover, a recent trend to perform security checks on backend servers (e.g., the cloud) [11,13–15] also introduces significant energy costs because of data upload. It has been previously shown that heavyweight sophisticated detection algorithms currently adopted in the desktop space cannot simply be deployed on mobile devices because of energy constraints [16]. Orthogonally, malware detection based on data mining algorithms in aggregated network activities or rule-based detection on mobile device metrics [17,18] require statistical differences between malicious behaviors and benign ones, thus may introduce false alarms or miss detection.

Messaging has been used by attackers as a preferred vehicle for malware propagation in the mobility network. Messages can embed text, audio, image, and video, thus, making messaging a very powerful way to spread different types of malware. Also, messaging has the possibility of rapid and widespread outbreak. This paper focuses on messaging-based malware propagation via SMS and MMS. Proximity-based propagation, typically using Bluetooth, limits the impact to a small local area and is out of the scope of this paper. We propose a novel method to detect, predict, and block message-based malware propagation by deploying *lightweight agents* on selected mobile devices. An agent, in essence, is a special phone number inserted into a device's contact book that is hidden to the user but exposed to malware. Because an agent is only an entry in the contact book, there is minimal computational or energy overhead as a byproduct of the system. When a device hosting an agent is infected, malware will select an agent with some probability in an attempt to propagate. We maintain a pool of agent numbers, all of which direct to a backend system within the mobility network. Any message sent to an agent is received by our system for further inspection, potentially generating both a signature for the message and malicious binary. Observing the rate of captured messages provides an insight into the malware spreading state. By using a latent space model, we can further predict the future trend for the propagation.

Our method has three distinct advantages. First, it is generic and independent of malware propagation schemes because the system is solely based on message observations. Therefore, it can be applied to any malware that propagates via messages. Second, because agents are only deployed on a small portion of mobile devices and each selected device is only equipped with a few agents, the system is lightweight and suitable for wide deployment. Lastly, the probability of malware selecting at least one agent increases with time exponentially. Consequently, the method can capture a malware signature within a short period after its outbreak.

The subsequent sections of the paper are organized as follows: In Section 2, we review related works similar to the proposed system, followed by the architecture design in Section 3. The complete system is designed as two components, which reside on the device and within the mobility network separately. We will explain the implementation of these two components and report experimental results in Section 4. In Section 5, we will propose the latent space model that is designed to forecast the malware propagation state based on messages received at deployed agents. The model will be validated with both the simulated contact book network and a network built on the real-world Call Detail Records (CDRs) in Section 6, in terms of forecasting accuracy and detection delay. Finally, we conclude our work in Section 7.

2. RELATED WORK

Unlike proximity-based worms, typically using Bluetooth as a propagation vector, messages can be routed anywhere in the world as well as across multiple network domains. For this reason, mobile malware exploiting messaging functionality for propagation is capable of rapid and widespread outbreak. Previous work has investigated the possibility of malware propagating via messages. Fleizach *et al.* [19] simulated a universal mobile telecommunications system (UMTS) network to discover the possible infection rate of a messaging worm. Meng *et al.* [20] performed propagation simulations on real-world SMS data to gather information about the possible infection rate starting from nodes with different degrees. Bose and Shin [21] revealed that hybrid worms, using both messages and Bluetooth for propagation, can spread rapidly throughout a cellular network. Bose later investigated propagation, detection, and containment of mobile malware spreading via non-traditional vectors such as power-law topologies, messages, and Radio Frequency (RF) channels [22]. Courtney and Sanders [23] presented four illustrative MMS virus scenarios measuring propagation rates in the device population and provided response mechanisms to threats.

Some work has applied honeypot schemes to detect malware propagation over instant messenger (IM) systems. Xie *et al.* [24] deployed HoneyIM accounts on IM clients within an enterprise network to trap IM malware outbreaks. They analyzed the relationship between the honey accounts coverage and the detection delay. Antonatos *et al.* [25] took the same approach one step further by setting up IM accounts and "friending" real-world IM user accounts crawled from the web. They focused on analyzing malicious files and links sent to the honeypot accounts. Xu *et al.* [26] proposed a similar approach in the social networking space by generating an early worm detection mechanism using decoy social network accounts. They compared the correlation of messages from

decoy friends and the ones among normal friends to apply detection.

Because the categories of wireless malware are analogous to biological viruses whose propagation is constrained by geographical proximity and contact patterns, significant research efforts have also been done in modeling malware propagation trends by susceptible–infectious (SI) and susceptible–infectious–recovered (SIR) models [27–29]. Many works have applied epidemic models on computer networks. Khelil *et al*. [30] used the simplest SI model to develop an information diffusion algorithm in MANETs. De and Das [31] discussed potential models in wireless sensor networks to solve the issues on data dissemination, routing algorithms, and broadcast protocols. Chen and Ji [32] focused on modeling the spread of one particular form of malware that propagates based on topology information by using probabilistic graphs and a spatial–temporal random process. Zou *et al*. [33] proposed a two-factor model on the Code Red worm propagation.

Different modified versions of epidemic models were proposed to characterize epidemic malware propagated via Bluetooth and messages in the mobility network. Zyba *et al*. [34] proposed a signature-based local detection combined with a globally broadcast signature dissemination system for proximity malware spreading in an epidemic pattern. Zheng *et al*. [35] studied mobility variables, such as population density, Bluetooth radius, and device velocity, and proposed an epidemic model of mobile phone virus propagation based on these variables. Yan and Eidenbenz [36] presented that an epidemic model captures not only the behavior of the Bluetooth protocol but also the impact of mobility patterns on the Bluetooth worm propagation. Mickens and Noble [37] introduced a probabilistic queuing model for epidemic spreading in mobile environments. Wang *et al*. [38] modeled the mobility of mobile phone users to study the fundamental spreading patterns characterizing an epidemic mobile virus outbreak.

Different from the previous honeypot work, our system not only detects and analyzes malicious messages but also models the state of malware propagation. We do not aim at improving existing propagation models to better predict a particular malware propagating throughout a mobility network. Instead, we propose a generic methodology to forecast the future propagation state without assuming a priori knowledge of the malware propagation scheme. This method provides a high-level overview of the whole network, which, to our knowledge, has not been studied in any previous modeling work. We accomplish this task by monitoring messages continuously received by our agents and constructing a latent space model solely from these messages. Because all model parameters are determined from the observable messages without making any assumptions about the malware propagation scheme, our method is generic and can be applied to any message-based malware spreading.

3. ARCHITECTURE DESIGN

In this paper, we propose a novel method to detect, predict, and block message-based malware propagation by deploying lightweight agents on selected mobile devices. Architecturally, our system is designed as two components, the *agent host component* stored on a mobile device and the *agent network component* within the mobility network.

An agent, in essence, is a special phone number inserted into the device's contact book. The uniqueness is that it is *invisible* to the user but *visible* to the malware. We define a mobile device that has agents installed in its contact book an *agent host*. An agent host is designed with two features in mind, the ability to hide agents for certain applications and update agents on demand. Different from prior work on instant messaging honeypots [24–26], where the buddy list entries are clearly visible to the user, the feature of hiding agent contacts is out of convenience to the user to avoid the possibility of sending erroneous messages to the agents. Accordingly, storing agents on a device is built out of two components: (i) an Application Programming Interface (API) level modification used to hide agents from specified applications and (ii) a background service that manages the state of agents on a device. Because of this design, in the typical case where a user communicates with friends in his/her contact book, it is guaranteed that messages are not sent to an agent. At the same time, agents are visible to malware in the same manner as normal contacts. For this reason, malware may pick an agent or a normal contact with equal probability while attempting to propagate.

The agent network component contains two modules: the *message filter module* and the *analysis module*. Every message will pass through the message filter module, which is used for blocking malicious messages and forwarding all agent messages to the analysis module. The mobility carrier can register multiple phone numbers as agents such that messages to different agent numbers all terminate within the agent network component. Within the analysis module, two submodules are working in parallel, one is the *analysis engine* and the other is the *model engine*. The analysis engine analyzes the incoming messages from agent hosts to generate a message signature in the malware signature database. Once a malicious message has been obtained, future messages of the same form will be blocked by the message filter module to protect users. At the same time, the model engine will forecast the state of malware propagating throughout the network based on the number of infected agents over time.

Without loss of generality, we illustrate an example scenario to describe the architecture design in Figure 1. We assume that three contacts have been selected by the malware for propagation, "Fridy Agno", "Secret Agent 007", and "John Smith". "SecretAgent 007" is one of the agents in the infected user's contact book. To explain the system, assume that this is the first round of propagation and "Friday Agno" is the first normal user that receives a propagation message, depicted by path 1 in Figure 1. Because this first malicious message has not been detected

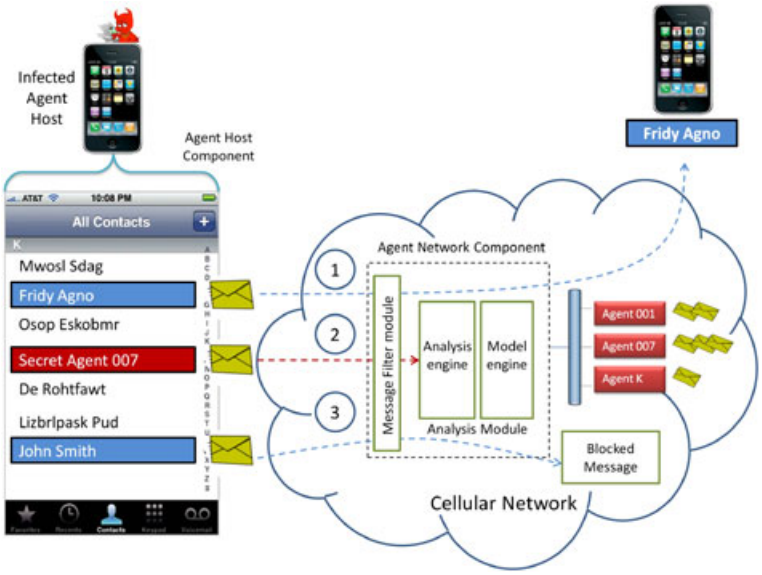

Figure 1. Lightweight agent-based detection system architecture: *agent host component* stored on a mobile device and *agent network component* within the mobility network.

by the system yet, the message successfully passes through the message filter and arrives at "Fridy Agno". The second message is sent to "Secret Agent 007" and is forwarded to the agent network component located within the mobility network, depicted by path 2. Now that the network component has received a single message, detection has occurred and the message is passed to the analysis and model engines. The analysis engine generates a signature for the message to block future propagation messages of the same form. The third message sent to "John Smith" will arrive at the message filter and now match a signature previously generated by the analysis engine. In this case, the message is now blocked, depicted by path 3, and a warning can be sent to the infected sending device notifying the user that his device has been compromised.

4. SYSTEM IMPLEMENTATION

The following section will discuss implementation details for the agent host component, the agent network component, and the prototype system in a UMTS laboratory network.

4.1. Agent host component

Because all major smart phone operating systems, for example, iOS, Android, Blackberry, and Windows Phone 7, include an API to interact with the contact book, OS developers could incorporate a modification to support hiding agent contacts. Typically, each phone uses an internal database to store contact book information. The APIs provided by an OS allow application developers the ability to easily query this contact book database. In order to hide agents, these API level queries can be modified to return only a subset of contacts in the database. Using this method, agent contacts can be hidden for typical default contact applications that most users tend to use.

To add, update, remove, and manage agent contacts, the ability to create a background service is needed. The background application would communicate with the service provider to check if any updates are needed. Android, iOS, and Blackberry support the execution of applications that run in the background. The background service could run at the system level for increased security. For OSes that do not support background applications such as Windows Phone 7, an application can be triggered to update agents via push notifications.

We illustrate the agent host component API modification on the device in Figure 2. When the contact application needs to make a query for contact information (path 1 in Figure 2), the *exposure policy*, located within the API, will verify that the application is the default contact application and therefore returns the contacts with the "Secret Agent 007" filtered. On the contrary, if a piece of malware queries the contact database (path 2), the API will expose "Secret Agent 007".

Because of its open-source nature and rapidly increasing market share, our agent-based system is implemented on the Android OS. We create a prototype implementation of the agent host system above the Android 2.2 OS. We incorporate the functionality to hide agent numbers by making a minor modification to the Android API. Alongside of this, we create a background system level service that contains the ability to update agent numbers.

Modern mobile operating systems provide APIs that developers use to create third-party applications. These APIs give developers the ability to access information stored in a database on the device, for example, a user's contact book. In the case of Android, data are stored and

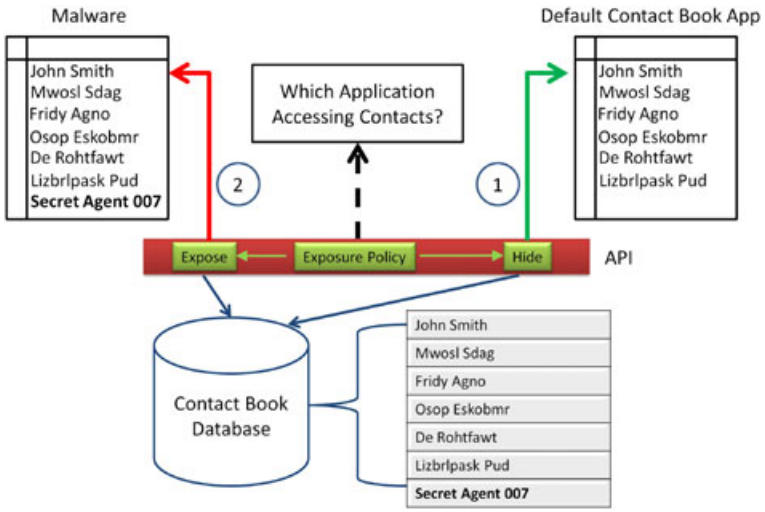

Figure 2. The agent host component API modification on the device.

retrieved through an object called a `ContentProvider`. Each `ContentProvider` provides an interface, called a `ContentResolver`, for querying, adding, updating, and deleting data. To identify the data being handled, every `ContentResolver` method is passed a uniform resource identifier (URI).

A URI uniquely identifies a data set in Android. For example, the URI associated with contact book data is of the form "`content://com.android.contacts/...`". Any application making a query for contact information will pass a URI of this form on to the `ContentResolver`. All queries in Android end up calling the same query function, a method within the `ContentResolver` class. The main arguments we are interested in is the URI argument and the selection clause. This selection clause is essentially an SQL WHERE clause to determine which rows to return.

To hide agent numbers from a contact application, we can modify a query's selection clause to remove agents. To remove an entry from a query, we append a string to the selection argument of the form `id !=n`, where n is the database id number of an agent. The proper agent `id` fields can be received from the system level service managing the agent contacts. We only trigger this functionality when the default contact application is querying the contact database. Any other application querying the contact book will observe agents as normal contacts.

To manage agent contacts on the device, we implement a background service that is in communication with the network component. Mainly, this service will maintain the addition, deletion, and modification of agent numbers. In order to manage the agent numbers, the service keeps track of the SQL database `id` field for each agent. The service provides a function that returns an integer array of ids for all agents on the system. This function is called at query time to determine which contacts to remove from a query. Because of the Android security model, no normal application can call this function in an attempt to obtain the list of agents.

For security reasons, only the default contact application will hide agents from the user. We can potentially support hiding agents for multiple applications by maintaining a white list of certified applications. These applications would go through a review process similar to the Apple App Store. The main difference would be that our review process focuses on security and only certifies contact applications. We could also use previously published techniques to certify applications [7,8].

Because the default contact application is the only application that hides agent numbers, the agents will be exposed to all other applications. Because of Android's API model, all applications query a database from its own context. This means that a piece of malware cannot query the contact database on behalf of the default contact book in an attempt to avoid the agent numbers. It follows that the only possible way to identify the agent contacts is through manual inspection. A hacker can use a non-default contact application and compare the output with the default one to identify the agents.

To address this problem, we have maintained a large agent pool, and each device is only deployed with a very small number of agents randomly drawn from this pool. Therefore, identifying agents on one device will not give the hacker the capability of coding an agent-avoidance rule for all devices and all agents. Moreover, because we can update agents on demand and periodically, the attacker is unable to predict agents on other devices.

4.2. Agent network component

We separate the agent network component into two modules, the *message filter module* and the *analysis module*. The function of the message filter is twofold. First, it must be able to filter messages based on the content and

forward messages for delivery if the message is found not to be malicious. Second, it must be able to route messages destined to agent numbers to the analysis module. The analysis module contains two engines: the *analysis engine* and *model engine*. The analysis engine generates malware-carrying message signatures from messages sent to agent numbers, whereas the model engine continuously feeds the number of infected agents into a latent space model to forecast the dynamic trend of the total number of infected devices within the network. We will first briefly explain the messaging system in a UMTS network and then explain the implementation in a UMTS laboratory network.

4.2.1. Short message service architecture.

In the mobility network, SMS messages are sent and received across the mobility network as well as the standard circuit switched network. Mobile user equipment (UE) will typically be connected to a base station, called a NodeB in the UMTS network, at any given time. Multiple base stations are managed by a radio network controller (RNC), which governs the radio access network. In current network deployments, it is very common that voice calls and SMS messages are routed to the circuit switched network using elements from the GSM core network. One of these such elements is the mobile switching center (MSC), responsible for both voice calls and SMS messages. The serving GPRS support node (SGSN) is in parallel with the MSC and is the bridge between the mobility network and the packet switched network. The SGSN is responsible for data messages such as MMS messages and WAP Push messages. Messages are passed from the MSC or the SGSN to the short message service center (SMSC). The job of the SMSC is to route messages to their destination. Routing is determined by the location of the destination user using both the home location register and the visitor location register or by routing rules specified in the MSC itself. If the recipient is not available, the SMSC stores the message until the end user connects to the network. From the SMSC, the message is routed through the signaling system no. 7 (SS7) network to the MSC in which the destination phone is associated with. The SS7 network handles signaling for basic call setup, management, and tear down. It is also a global network that performs the function of routing messages between SMSCs and between network providers. Once the message is sent to an MSC that the destination handset is currently associated with, the message is delivered again through the RNC and NodeB to the UE as a recipient.

4.2.2. Network component architecture.

We implement a prototype of the message filtering module and the analysis module within a UMTS laboratory network. This laboratory network is fully functioning and contains all components of the network explained in Figure 3. Figure 4 shows the design of the network component architecture. We make routing changes to the SMSC to route all incoming messages to the message filter module. The message filter module is implemented using Kannel, an open-source SMS gateway [39]. Within Kannel, all aspects of incoming messages can be inspected as well as forwarded back to an SMSC for delivery.

When messages arrive at the message filter, they are first checked against the agent list. The agent list is a list of unused numbers within the mobility network provider's pool of phone numbers and does not correspond to any normal user. The provider has the ability to update and add numbers periodically as seen fit, protecting against a potential compromise. When the destination address of an incoming message corresponds to an agent number in the database, this message will be passed along to the analysis module. If the destination address does not correspond to an agent, the message body is checked against the message signature database. If there is no match, the message is forwarded back to an SMSC for delivery and the destination receives the message as normal. If there is a match, the message filter blocks the message and can return a warning message back to the original sender, informing that a malicious message has been detected originating from his device.

The analysis module consists of two pieces, the analysis engine and the model engine. We implement a simple

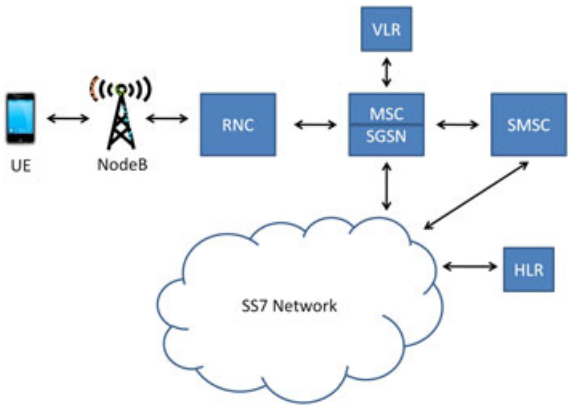

Figure 3. Universal mobile telecommunications system SMS-related architecture.

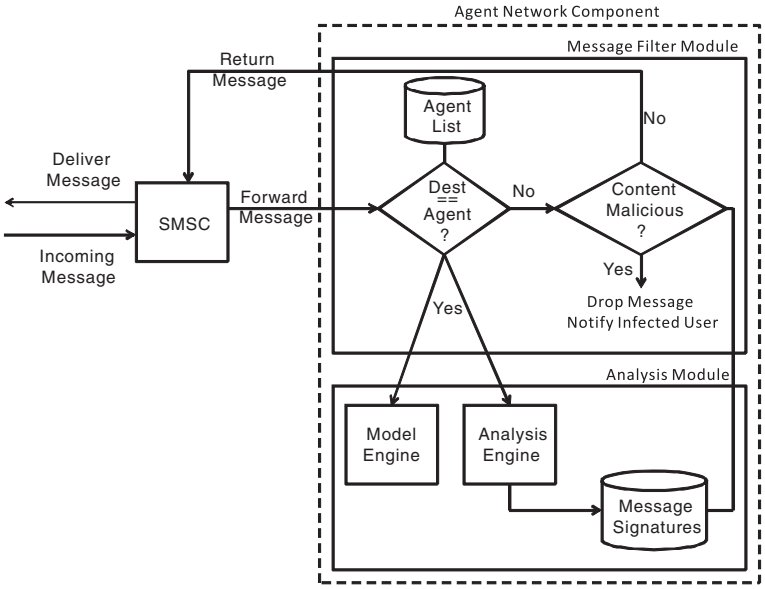

Figure 4. Network component architecture with two modules: the *message filter module* and the *analysis module*.

analysis engine that generates a message signature of the incoming message as a proof of concept to show that malicious messages can be filtered within the network. The message signature is an Secure Hash Algorithm (SHA)-1 hash of the message and is inserted into a message signature database, used to filter future incoming messages. We acknowledge that sophisticated malware can easily defend against this type of simple hashing technique. We leave to future work a more complex message signature generation scheme to protect against complex techniques, such as polymorphism within propagation messages.

Because we capture the malicious message within the analysis engine, a live sample of the malware can be potentially obtained to analyze the malware itself. This analysis can be used to generate malware signatures, which can be supplied to existing anti-malware systems. Different sophisticated detection schemes, such as behavioral analysis, automated URL browsing, and content-based detection [11,10,9], can be applied. For example, if an MMS message is captured, behavioral analysis can be applied to the attachment for signature generation. If a typical SMS is captured, a link could point to a malicious website triggering a download of the malicious application. By following the link, the analysis engine can potentially obtain the malicious application for further inspection.

Within the analysis module, a model engine whose function is to forecast the dynamic state of infectious devices within the network is also managed. Network providers continuously monitor the state of their networks, and malware infection could be monitored in the same manner. The more information a network provider has, the better suited they are to defend against attacks. By observing messages arriving at the network component over time, such information is passed to the model engine to forecast how many actual devices are infected by the malware. Moreover, the model engine can be used to check the effectiveness of the message signature generation scheme by continuously observing the rate at which messages arrive at the agents.

4.3. Experimental results

To test our prototype, we ran two separate experiments with two malware samples on an Android device connected to our UMTS laboratory network. An agent is installed on one device, and the network component is initialized to handle this specific agent. A successful experimental run consists of running a malware sample on the agent host, a message being sent to an agent, a message signature being generated, and blocking subsequent messages produced by the same malware executing on another Android device.

The first malware sample was developed in-house and is disguised as a simple game. When the game starts, in the background, propagation messages are sent to all contacts in a user's contact book. The message contains a link that attempts to trick another user into downloading the malicious application. The second malware sample is an actual real-world sample called Walk and Text [5]. On execution of this application, a predefined message is sent to all contacts. Although it is not a propagation message, it serves the same purpose to test the system. In both cases, the agent-based system can successfully detect and block the malicious messages.

Because we are connected to a laboratory network, we observed that no detectable overhead is caused by the agent network component. We consider the performance impact on the basic message system with the addition of the agent network component. In a real network handling millions of messages per hour, the load of the message

filter can be distributed across multiple machines. Messages can also be routed across multiple machines based on the current overall load of the system. Even better, with the help of an SMSC manufacturer, the message filtering can be integrated directly alongside the SMSC for minimal overhead and optimized functionality.

5. MOBILE MALWARE PROPAGATION MODEL

This section presents a latent space model for forecasting the total number of infected mobile devices in the network by the model engine given an up-to-date information of the number of infected agents over time. All model parameters are determined from the observable messages without making any assumptions about the malware propagation scheme; thus, our method is generic and can be applied to any message-based malware spreading.

5.1. Background epidemic susceptible–infectious–recovered model

Susceptible–infectious–recovered model is a deterministic compartmental model to mathematically describe the progress of an epidemic in a large population, comprising many different individuals in various fields. The main idea is to reduce a population diversity to a few key characteristics that are relevant to the infection under consideration and accordingly divide the population into three compartments: *susceptible*, those who are susceptible to the disease; *infectious*, those who are already infected; and *recovered*, those who either are immune to the epidemic or have recovered and gained immunity.

The speed of disease propagation is determined by the two model parameters: β and v, the transition rate from *susceptible* to *infectious* and from *infectious* to *recovered*, as shown in Figure 5, respectively. Because the recovered population is immune to the disease, there is no transition from *recovered* to *susceptible*.

5.2. Latent space model for malware propagation

The SIR model was originally designed for an epidemic of a human (or animal) disease. However, the message-based malware spread is significantly different from human disease in terms of spreading speed and vehicle. Therefore, applying SIR model to the message-based malware propagation in mobile networks requires certain adaption. First, human epidemic spreads through physical contact, e.g., food, air, or blood, which significantly limits the speed of propagation. The message-based malware propagates electronically at a very high speed [33,40,23]. Furthermore, unlike human body, mobile devices cannot automatically get recovered from a disease. Unless the malware's signature is identified, which is usually a relatively slow process, the device remains infectious. Consequently, combination of the high speed propagation and the slow process of recovery makes the transition rate from *infectious* to *recovered*, v, a very small number in the message-based malware SIR model. Hence, we assume $v = 0$ in our model. Note that it is not appropriate to drop the *recovered* state from the model (which becomes an SI model). This is because any computer malware is designed only for a certain type of devices, operating systems, and applications. For those devices with different OSes and applications, they are inherently immune, so that they belong to the *recovered* state.

Therefore, the model we consider assumes that mobile devices may be only in three states: *susceptible* or *infectious*, whose population changes over time, and *recovered*, which size is constant during the malware spreading process. Similarly, the deployed agents can also be in three states: agents that receive an SMS from an infected mobile device are *infectious*, those deployed on *susceptible* devices but do not receive any malware messages yet are *susceptible*, and agents deployed on insusceptible devices are *recovered* because the malware is incompatible with the device OS. Note that infected agents do not participate in further malware propagation; they only enable malware-carrying messages to be captured by the malware filter and forwarded to the agent network component for further analysis.

Table I lists all frequently used notations and explanations. $\mathcal{S}, \mathcal{I}$ and $\mathcal{R}$ represent the set of susceptible, infectious, and recovered devices, and $\mathcal{S}^A, \mathcal{I}^A$ and $\mathcal{R}^A$ are the counterpart agents on agent hosts. $S(t)$, $I(t)$ and $S^a(t)$, $I^a(t)$ denote the fractions of *susceptible* and *infectious* in the population of mobile devices and agents at time t, respectively, whereas R and R^a represent the fractions of *recovered*, which are constant over time. The percentage of the total number of *susceptible* devices to the total device population N is fixed during the malware epidemic, and such devices are those with compatible OS. When the first malware-carrying message is sent to one of the agents, we can identify the OS the malware is targeting. Thus, by knowing the targeted OS's market share μ, the fraction of insusceptible devices R can be then determined. Mathematically, in such a case, the relation of *susceptible* and *infectious* can be expressed as $S(t) + I(t) = \mu, S^a(t) + I^a(t) = \mu$ for any time t.

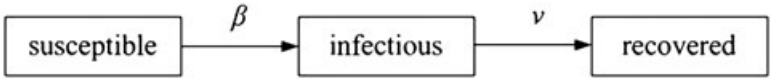

Figure 5. Epidemic propagation model. $\boldsymbol{\beta}$ and $\boldsymbol{\nu}$ are the transition rates between compartments.

Table I. Notation.

Notation	Explanation
$\mathcal{S}$	Set of susceptible devices
$\mathcal{I}$	Set of infectious devices
$\mathcal{R}$	Set of insusceptible/recovered devices
$\mathcal{S}^{\mathcal{A}}$	Set of susceptible agent hosts
$\mathcal{I}^{\mathcal{A}}$	Set of infectious agent hosts
$\mathcal{R}^{\mathcal{A}}$	Set of insusceptible/recovered agent hosts
$S(t)$	Fraction of $\mathcal{S}$ in the population of devices
$I(t)$	Fraction of $\mathcal{I}$ in the population of devices
R	Fraction of $\mathcal{R}$ in the population of devices
$S^a(t)$	Fraction of $\mathcal{S}^{\mathcal{A}}$ in the population of agents
$I^a(t)$	Fraction of $\mathcal{I}^{\mathcal{A}}$ in the population of agents
R^a	Fraction of $\mathcal{R}^{\mathcal{A}}$ in the population of agents
N	Total device population
μ	Device market share
β	Transition ratio from $\mathcal{S}$ to $\mathcal{I}$
γ	Transition ratio from $\mathcal{S}^{\mathcal{A}}$ to $\mathcal{I}^{\mathcal{A}}$

Our goal is to infer the malware propagation state in real time and generate a model to predict its future spreading trend. To achieve this goal, we partition the problem space into two planes as shown in Figure 6: (i) the latent device plane of unobservable $S(t)$ and $I(t)$ and (ii) the observable message plane of malware-carrying messages. The population of susceptible and infected agents is completely observable because malware-carrying messages sent to agents will all be forwarded and terminated at the agent network component, whereas the general population of mobile devices, their infectious states, and the malware propagation pattern remain latent. Once a device becomes infectious, the malware may send messages to any devices, which belong to $\mathcal{S}$, or $\mathcal{I}$ or $\mathcal{R}$, but only the susceptible devices will potentially increase the total number of infectious devices. We define a message process $\{h(t)\}$, such that in each time interval $[t, t+dt)$, $h(t)$ is the total number of malware-carrying messages that have been sent to the agents. From $\{h(t)\}$, the total number of infected agents can be determined as the number of distinct agents that receive at least one message.

The agents create a mapping from the unobservable latent space $I(t)$ to the observable space $I^a(t)$. Intuitively, $I^a(t)$ is largely determined by $I(t)$. Besides $I(t)$, $I^a(t)$ depends also on the method that the malware picks the targets, and the device idiosyncratic property, for example, the contact book size. However, the device idiosyncratic property is not only unknown but also dynamic because the device owner can change it at any time. The malware propagation scheme is also unknown until the malware is completely understood. Thus, to infer the latent $I(t)$ and $S(t)$ from the observable $I^a(t)$, we model propagation of the malware by the following latent space model of two differential equations, one for the latent space and the other for the observation space:

$$dI(t) = \beta(\mu - I(t))I(t)dt, I(0) = I_0 \tag{1a}$$

$$dI^a(t) = \gamma(\mu - I^a(t))I(t)dt, I^a(0) = I_0^a \tag{1b}$$

These two equations show the dependency between $I^a(t)$ and $I(t)$, which is different from the traditional epidemic model. Equation (1a) is adopted from the typical epidemic model and describes propagation of the malware in the population of mobile devices. Here, a change in the fraction of infected devices d$I(t)$ at time t is modeled as proportional to the fraction of infected devices $I(t)$ and to the fraction of susceptible devices $S(t) = \mu - I(t)$. The speed of malware propagation, in other words, the transition rate from *susceptible* devices to *infected* devices, is determined by the model parameter β. Equation (1b) describes propagation of the malware in the population of agents. It assumes that a change in the fraction of infected agents d$I^a(t)$ at time t proportional to the fraction of infected mobile devices $I(t)$ and to the fraction of susceptible agents $\mu - I^a(t)$ remained. Note that the change in the fraction of infected agents at time t does not depend on the fraction of infected agents at time t, because infected agents do not participate in malware propagation, but depends on the fraction of infected mobile devices, which directly account for the malware propagation. Parameter γ represents the speed of transition from *susceptible* agents to *infected* agents. Initial

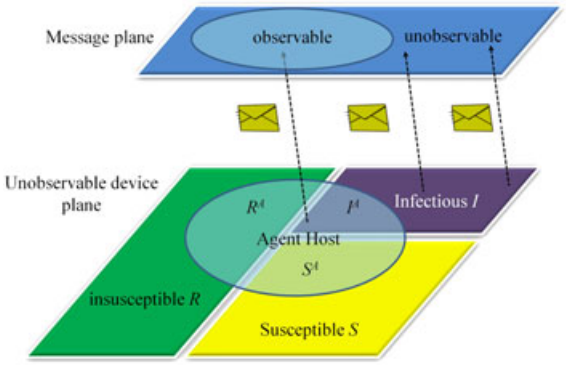

Figure 6. Applying the observable message process in the message plane to infer the *susceptible* and *infectious* population in the unobservable device plane.

fractions of infected mobile devices and agents I_0 and I_0^a are unknown, because we do not have any prior knowledge about the exact time when the malware propagation starts.

The differential Equation (1a) has a closed-form solution

$$I(t) = \mu - (\mu - I_0)/\left(\mu - I_0 + I_0 e^{\beta t}\right) \tag{2}$$

Substituting Equation (2) in Equation (1b) obtains the following closed-form solution of Equation (1b):

$$I^a(t) = \mu - \left(\mu - I_0^a\right)\left(\frac{\mu + e^{I_0}}{e^{I_0} + e^{\beta t}}\right)^{\frac{\gamma}{\beta}} \tag{3}$$

Given the discrete observations of infected agents $\left\{\tilde{I}^a(t)\right\}_{t=t_0}^{t_0+dt}$ for the period $(t_0, t_0 + \mathrm{d}\,t)$, where t_0 is the time when the first message is received at one of the agents and $\mathrm{d}t$ is the observation period, parameter estimation on γ, β, I_0, and I_0^a can be obtained via the least squares procedure, minimizing the total quadratic error as

$$\underset{\beta,\gamma,I_0,I_0^a}{argmin} \sum_{t=t_0}^{t_0+dt} \left(I^a(t) - \tilde{I}^a(t)\right)^2 \tag{4}$$

Thus, the forecasted number of infected devices at any given time t can be estimated as

$$N\hat{I}(t) = N\left(\mu - \left(\mu - \hat{I_0}\right)/\left(\mu - \hat{I_0} + \hat{I_0} e^{\hat{\beta}} t\right)\right) \tag{5}$$

where N is the total number of devices in the network. As a result of Equation (5), we can successfully leverage the observable $I^a(t)$ in the agent population as a proxy to determine the model parameters and eventually estimate $I(t)$ in the latent device population.

6. MODEL VALIDATION AND PERFORMANCE EVALUATION

In this section, we validate the accuracy and evaluate the performance of the proposed model by using both simulated and real-world data sets. In order to do this, a priori knowledge of device connectivity in the mobility network is required. In other words, the spread of malware-carrying messages within the mobility network depends on the social relation among mobility users, which is determined by contact book connectivity. We define the network generated by this social relation as the *contact book network*.

Throughout this paper, we consider the case of malware spreading via messages by choosing contacts from the user's contact book using a selection scheme designed by the hacker. In order to use the proposed model to forecast the state of malware propagation throughout the mobility network, the contact book network is required. However, because of privacy concerns and lack of knowledge, it is impractical to collect contact book information for all users in the mobility network.

An alternative approach to generating a social network of mobility users is to use CDRs to infer the social relation among them. In the real world, when a text message occurs between two users, the mobility network generates a CDR of this exchange for billing purposes. We define the social relation among users using these data as the *who-talks-to-whom network*.

The contact book network and who-talks-to-whom network are two different networks in nature. Connections in the who-talks-to-whom network depend on the users' communication patterns. For this reason, the who-talks-to-whom network is typically a small subset of the contact book network. A person with a large contact book, a node with high connectivity within the contact book network, may send and receive messages rarely, thereby appearing as having low connectivity in the who-talks-to-whom network. Even by generating a network using all CDR records over a long period, the result is not the contact book network because of the high possibility that users do not communicate with every person in their contact books. Because of these, it is impractical to infer the contact book network from the CDR who-talks-to-whom network.

To verify that message-based malware propagation is a threat to the mobility network, we use a simulation over the who-talks-to-whom network generated by real-world CDRs. In Section 6.1, we show that malware can propagate across the whole data set and can be detected in a quick manner using the agent-based scheme proposed in this paper. In a real-world outbreak, message-based malware propagation depends on the contact book network connectivity; the verification of the model requires the use of this network. As previously mentioned, it is impractical to generate this network even when using real-world CDR data; therefore, we use a simulated contact book network to verify the accuracy of the model in Section 6.2.

6.1. Call detail records-based validation

Various attributes of a CDR include the time stamp of communication, the origin and destination number, the type and duration of connection, international mobile equipment identity, and the like. To construct a who-talks-to-whom network, we extracted origin and destination identifiers from each CDR record. We use anonymized CDR data produced in the cellular network of a large US service provider. The CDR data we used is 20 gigabytes in size and is composed of 50 million unique users with approximately 150 million unique social pairs on April 11, 2011. The graph of the who-talks-to-whom network is composed of nodes, representing unique users within the mobility network, and edges, representing directed message exchanges between nodes.

Figure 7 shows the distribution of the outdegree for all nodes in the graph. Observing the distribution determines that in one day, a user on average sends SMS messages to one or two other unique users. On the other hand, a very

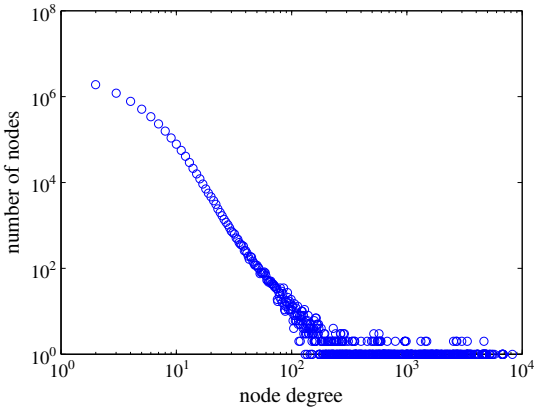

Figure 7. The outdegree distribution for 50 million nodes in the who-talks-to-whom network.

small number of users communicate with more than 100 unique users throughout the course of one day. Because a user typically has many more than two entries in his contact book, the average outdegree of all nodes in the contact book network would be much higher. Thus, the who-talks-to-whom network is indeed a small subset of the contact book network. By validating that malware can propagate through the who-talks-to-whom network, it follows that malware will be able to propagate throughout the contact book network.

We simulate propagation by starting at a random node in the graph and traversing nodes that are reachable from this node. On average, more than one million nodes are infected for a single propagation. Figure 8 is a visual representation of a connected component generated by one propagation example. The average clustering coefficient of the connected component in the graph is 0.4. From the heavy-tailed distribution of the node degree and the high clustering coefficient, it follows that the who-talks-to-whom network is a "small world" network, which is consistent with the observations by Meng *et al*. [20]. In such a small world network with "six degrees of separation", malware can spread to many other users quickly via messages.

Because we validate that message-based malware can propagate widely throughout the who-talks-to-whom network, we measure the performance of our agent-based system to identify the time required to detect an outbreak. The *detection delay* is defined as the total number of infected devices in the network when the first malware-carrying message received the agent network component. This metric represents the impact of the malware before the agent system detects an outbreak. Figure 9 shows multiple detection delays when three agents are deployed on 20% of all devices within the who-talks-to-whom network. Because the probability of selecting an agent is dependent on the size of a user's contact book, during each propagation stage, we select an agent with a probability based on a simulated contact book size. We simulate the contact book size for each node following a particular power-law distribution, with a minimum contact book size of 50 and an average of 100. We analyze the detection delay starting with nodes of different degrees (ranging from 1 to 20) as the first infected node. For each node degree, 1000 experiments are run to calculate the statistics of the detection delay. We can see from Figure 9 that although the degrees of the first infected node are different, the average detection delay is about the same, ranging from 50 to 100 infected devices. These results show that the detection delay is not dependent on the outdegree of the initial infected node, and therefore, the performance of our detection system is robust. If the mobility service provider requires a lower detection delay, the performance can be improved by increasing the deployment ratio. On the other hand, in order to reduce the agent footprints and consume less network and device resources, it is ideal to keep the deployment ratio low. Therefore, for the agent deployment ratio, there is a tradeoff between the detection delay and the resource footprint, but it is an adjustable parameter.

6.2. Simulation-based validation

In this section, we validate performance of the model presented in the previous section through simulations due to the inability to generate a contact book network using real-world CDR data. We assume that the network size N and the device market share μ are the only a priori knowledge, because these details are known by the mobility service provider. Malware propagation scenarios contain many parameters, in particular, the malware's target selection algorithm. To demonstrate that our method is generic and independent of these factors, we generate different simulation scenarios by varying these parameters.

Without loss of generality, the model is initialized with a social network formed using 1 million simulated mobile devices. The contact book network topology is a directed graph where each node represents a mobile device, and a directed edge between two node represents the case where the destination node is located within the source node's contact book. Therefore, the outdegree of a node indicates the number of contacts in its contact book.

To make the scenario more realistic, we consider several variables for the simulation. Previous work uses the power-law distribution to model networks generated by email address books [41,42,19], which are similar or even synced to contact books on mobile devices. Therefore, we use the Barbasi and Albert model to generate the distribution of contact book sizes for each node in the simulated network. Once a susceptible device receives a malware-carrying message, there is a probability that this device will get infected. Although the infection probability is different for each device, previous work chose to use a Gaussian distribution in a similar case [24]. When a susceptible device becomes infected, the malware may wait a certain amount of time before propagating. The malware propagation delay is determined for each device

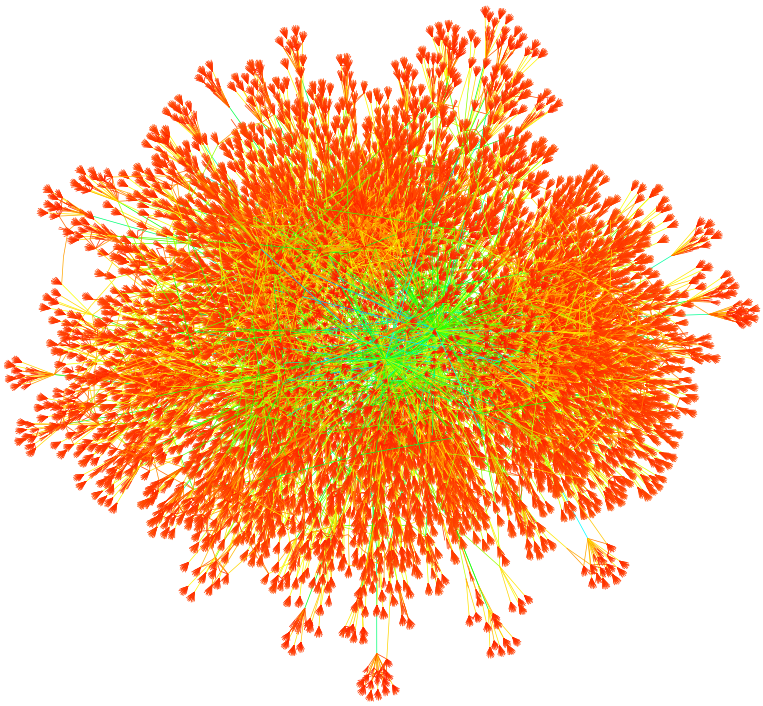

Figure 8. Visual representation of malware propagation starting at a single node.

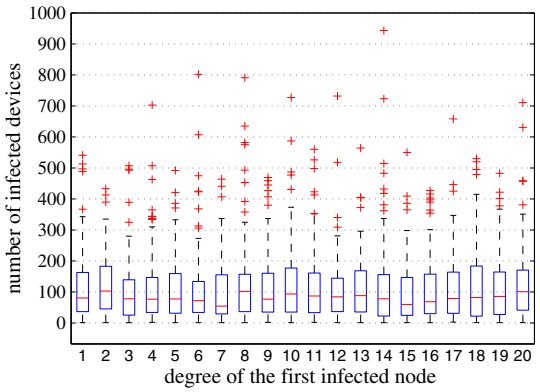

Figure 9. The detection delay for random nodes with various outdegrees in the who-talks-to-whom network.

using an exponential distribution, which is also used in Xie's work [24].

The simulation of the malware propagation starts from a single infected device in the simulated contact book network. We simulate propagation of the malware in Mathematica and record discrete observations of the fraction of infected agents $\tilde{I}^{\mu}(t)$ at each simulation step t. These discrete observations are used to estimate parameter estimates $\hat{\gamma}$, $\hat{\beta}$, $\hat{I}_0$, and $\hat{I}_0^a$ as described in equation Equation (4). Using the estimated parameters along with the network size N and the device market share μ, we forecast the total number of infected devices $N\hat{I}(t)$ in the network at any given time t by Equation (5).

In all simulations, we simulate a network with 1 million devices for various market shares, agent deployment ratio, and propagation selection schemes. Because agents are only deployed on a small portion of mobile devices, α is used to denote the agent deployment ratio within the mobility network. The number of agents deployed per agent host is fixed to be three for all cases, whereas α is adjusted from 10% to 20%. The market share is varied between 25% and 40%, which is representative of the actual mobile OS market shares [43]. The number of contacts selected by the malware for propagation m is chosen to be either a constant or a quantized number or proportional to the contact book size. Different simulation scenarios are generated by adjusting the parameters α, μ, and m to show that our model forecasts the total number of infected devices in the network accurately regardless of propagation scenarios.

Table II lists the various cases that we consider. First, we vary the agent deployment ratio $\alpha = 10\%, 15\%, 20\%$ and fix the market share $\mu = 30\%$ and the number of contacts that the malware chooses to spread $m = 3$ (listed in Table II, cases 1–3). Then we choose different market shares $\mu = 25\%, 35\%, 40\%$ and fix $\alpha = 15\%$ and $m = 3$ (listed in Table II, cases 4–6). And finally we select the number of contacts that the malware chooses to spread, m, as a function of the contact book size while fixing $\alpha = 10\%$ and $\mu = 30\%$ (listed in Table II, cases 7–9).

Figures 10 to 12 compares the accuracy of the forecast for cases 1–9. The number of infected agents recorded at the model engine is represented by a *blue dotted line*, which is located lower than other curves. Observations of

Security Comm. Networks 2013; **6**:33–48 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/sec

Table II. Simulation parameters.

Case	α	μ	m
1	10%	30%	$m=3$
2	15%	30%	$m=3$
3	20%	30%	$m=3$
4	15%	25%	$m=3$
5	15%	35%	$m=3$
6	15%	45%	$m=3$
7	10%	30%	$m=3$ (constant)
8	10%	30%	$m(k_i)=\begin{cases}2 & k_i\le 20\\ 3 & 20<k_i\le 50\\ 5 & 50<k_i\le 100\\ 10 & 100<k_i\le 200\\ 20 & 200<k\le 50\end{cases}$ k_i is the contact book size of device i
9	10%	30%	$m(k_i)=\lceil k_i/15\rceil$ k_i is the contact book size of device i and $\lceil\cdot\rceil$ denotes a ceiling function

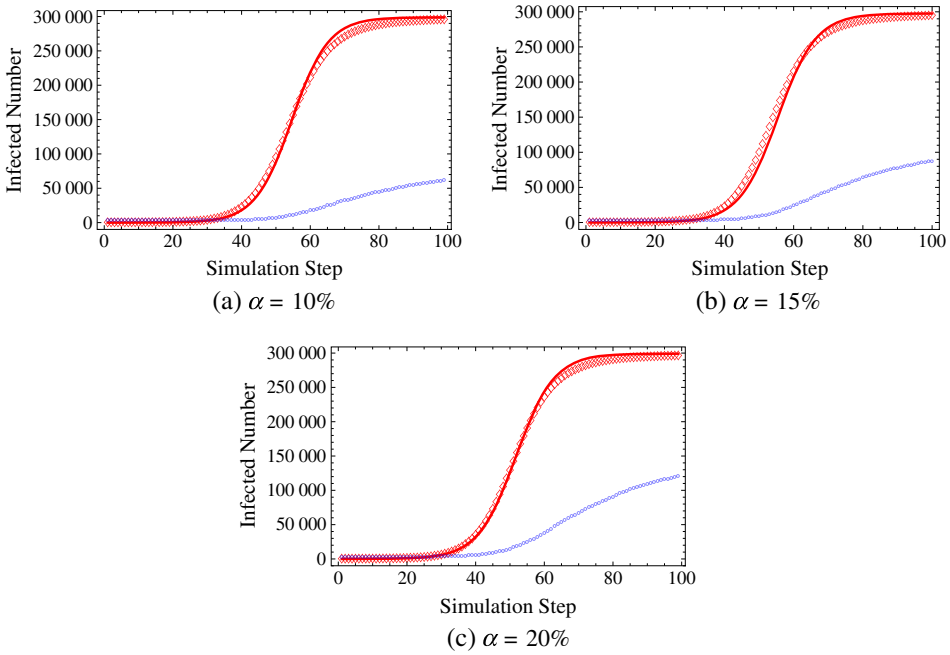

Figure 10. Case 1–3: model closely forecasts the simulated $I(t)$ with various agent deployment ratio $\boldsymbol{\alpha}$, fixed market share $\boldsymbol{\mu}$ = 30%, and m = 3.

the total number of infected devices in the network are denoted by a *red diamond line*, and the forecasted number of the infected devices is presented by a *solid red line*. Figure 10 plots the accuracy of the forecast under different agent deployment strategies. We intuitively expect that deploying more agents in the entire population will improve the estimation of $I(t)$. However, as illustrated in Figure 10, the results do not show significant improvement in accuracy with increasing agent deployment ratio. This may be because given the large device population, even at a low deployment ratio, for example, 10%, the number of agents is sufficient to make an accurate estimation. Figure 11 compares the accuracy of the forecast generated for malware targeting different types of mobile OSes. The increasing rate in the number of infected devices shows that the higher the market share is, the more rapid the spread of malware. However, its impact on model accuracy is indiscernible. Figure 12 compares the accuracy of the forecast generated in different malware propagation scenarios. Results demonstrate that our forecasts consistently approximate the true number of infected devices regardless of the malware propagation scheme.

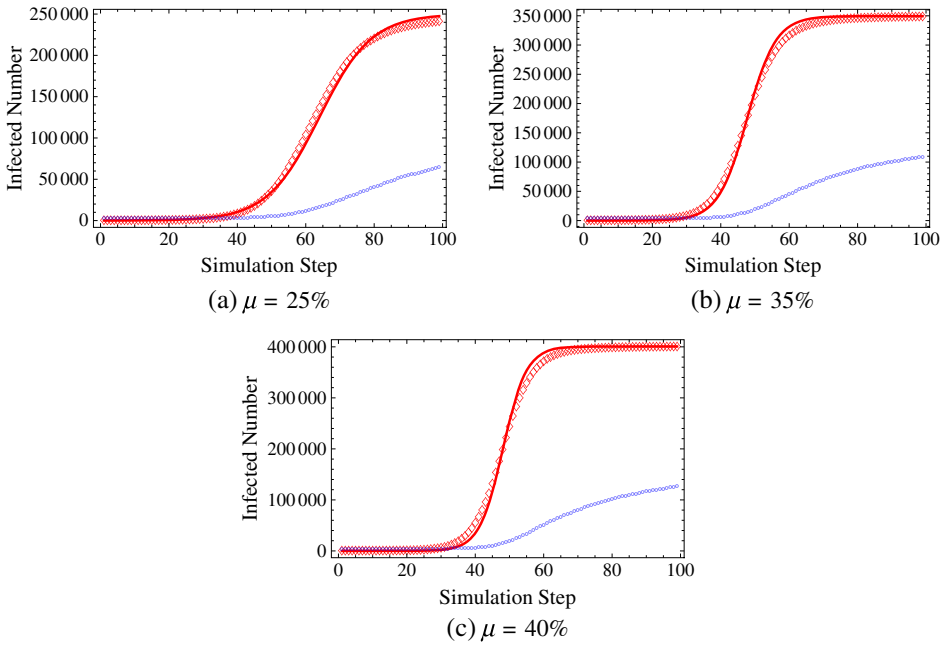

Figure 11. Case 4–6: model closely forecasts the simulated $I(t)$ with various market share $\boldsymbol{\mu}$, fixed agent deployment ratio $\boldsymbol{\alpha}$ = 15%, and m = 3.

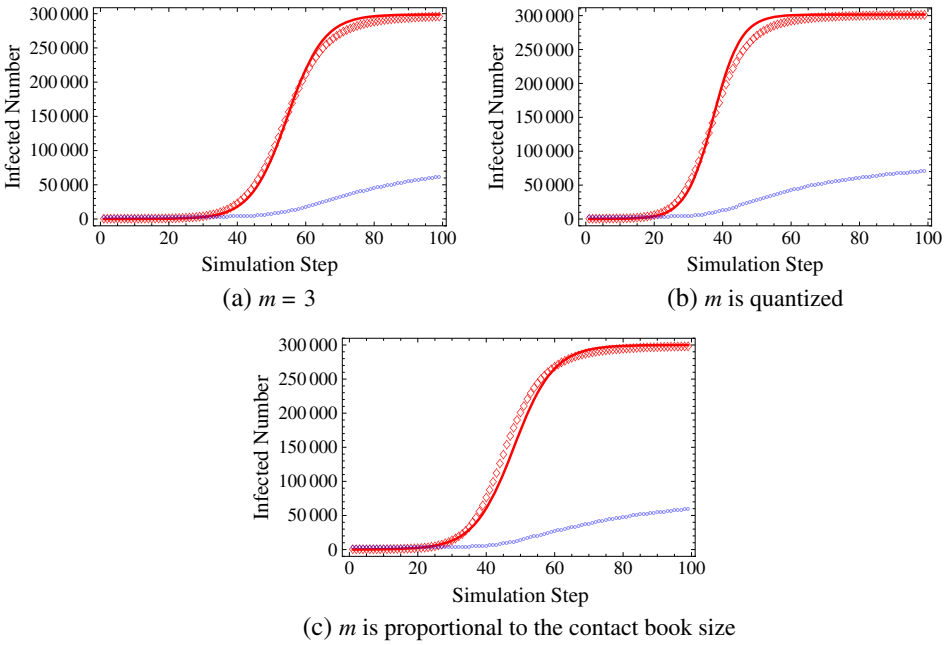

Figure 12. Case 7–9: model closely forecasts the simulated $I(t)$ with various malware selection scheme m, fixed agent deployment ratio $\boldsymbol{\alpha}$ = 10%, and market share $\boldsymbol{\mu}$ = 30%.

7. CONCLUSIONS

In this paper, we proposed a novel system to detect, analyze, and forecast message-based malware propagation by deploying lightweight agents on selected mobile devices. The system contains three parts, which are (i) detection to capture messages sent from malicious applications; (ii) analysis to generate a message signature for filtering

future messages; (iii) capture of the current dynamics and forecasting the future state of malware propagation within the mobility network.

By adding agents to mobile devices within the mobility network, it is possible to detect malicious malware propagation messages sent by infected devices quickly after an outbreak. Once detected, the agent network component can analyze messages to generate a message signature using the analysis engine. Using these signatures, future propagation messages can be blocked, inhibiting future attempts at propagation. By analyzing the observed messages, we can apply a latent space model to capture the dynamics of the malware propagation and predict its future trend. The model is generic and independent of malware propagation schemes because the method is solely based on messages observed at the agent network component. We validate the model through various malware propagation schemes and types of mobile devices. Results demonstrate that the model forecasts the number of infected devices accurately regardless of the malware propagation scheme.

REFERENCES

1. Securelist. Mobile malware evolution: an overview. Available at: http://www.securelist.com/en/analysis?pubid=204792080
2. F-Secure. Available at: http://www.f-secure.com/v-descs/cabir.shtml 2004
3. F-Secure. Available at: http://www.f-secure.com/v-descs/mabir.shtml 2004
4. F-Secure. Available at: http://www.f-secure.com/v-descs/commwarrior.shtml 2005
5. Skype. Shame on you: pirated android app really shameware. Available at: http://hothardware.com/News/Shame-on-You-Pirated-Android-App-Really-Shameware
6. Oberheide J, Jahanian F. When mobile is harder than fixed (and vice versa): demystifying security challenges in mobile environments. Proceedings of the Workshop on Mobile Computing Systems and Applications, 2010.
7. Enck W, Gilbert P, Chun B, Cox LP, Jung J, McDaniel P, Sheth AN. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. Proceedings of the USENIX Symposium on Operating System Design and Implementation, 2010.
8. Enck W, Ongtang M, Mcdaniel P. On lightweight mobile phone application certification. ACM Conference on Computer and Communications Security, 2009.
9. Kim H, Smith J, Shin KG. Detecting energy-greedy anomalies and mobile malware variants. Proceedings of the 6th Conference on Mobile Systems, Applications and Services, 2008.
10. Bose A, Hu X, Shin KG, Park T. Behavioral detection of malware on mobile handsets. Proceedings of the 6th conference on mobile systems, applications, and services (Mobisys), 2007.
11. Portokalidisi G, Homburg P, Anagnostakis K, Bos H. Paranoid android: versatile protection for smartphones. Proceedings of the 26th Annual Computer Security Applications Conference, 2010.
12. Fuchs AP, Chaudhuri A, Foster JS. SCanDroid: automated security certification of android applications. Manuscript, University of Maryland. Available at: http://www.cs.umd.edu/~avik/papers/scandroidascaa.pdf.
13. Cuervo E, Balasubramanian A, Cho D, Wolman A, Saroiu S, Chandra R, Bahl P. MAUI: making smartphones last longer with code offload. Proceedings of the 8th Conference on Mobile Systems, Applications and Service, 2010.
14. Chun B, Ihm S, Maniatis P, Naik M, Patti A. CloneCloud: elastic execution between mobile device and cloud. Proceedings of the 6th Conference on Computer Systems (EuroSys), 2011.
15. Oberheide J, Veeraraghavan K, Cooke E, Flinn J, Jahanian F. Virtualized in-cloud security services for mobile devices. Proceedings of the Workshop on Virtualization in Mobile Computing, 2008.
16. Bickford J, Lagar-Cavilla H, Varshavsky A, Ganapathy V, Iftode L. Security versus energy tradeoffs in host-based mobile malware detection. Proceedings of the 9th Conference on Mobile Systems, Applications and Services (MobiSys), 2011.
17. Gu G, Perdisci R, Zhang J, Lee W. BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection. Proceedings of the 17th USENIX Security Symposium (Security), 2008.
18. Gu G, Porras P, Yegneswaran V, Fong M, Lee W. BotHunter: detecting malware infection through ids-driven dialog correlation. Proceedings of the 16th USENIX Security Symposium (Security), 2007.
19. Fleizach C, Liljenstam M, Johansson P, Voelker GM, Méhes A. Can you infect me now? Malware propagation in mobile phone networks. Proceedings of the ACM Workshop on Recurring Malcode (WORM), 2007.
20. Meng X, Zerfos P, Samanta V, Wong S, Lu S. Analysis of the reliability of a nationwide short message service. Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM), 2007.
21. Bose A, Shin KG. On mobile viruses exploiting messaging and bluetooth services. 2nd IEEE International Conference on Security and Privacy in Communication Networks (SecureComm), 2006.
22. Bose A. Propagation, detection, and containment of mobile malware. PhD Thesis, Ann Arbor, MI, USA 2008. AAI3328771.
23. Courtney T, Sanders WH. Quantifying the effectiveness of mobile phone virus response mechanisms. Proceedings

of the 37th IEEE/IFIP International Conference on Dependable Systems and Networks, 2007.
24. Xie M, Wu Z, Wang H. HoneyIM: fast detection and suppression of instant messaging malware in enterprise-like networks. Proceedings of the 23rd Annual Computer Security Applications Conference (ACSAC), 2007.
25. Antonatos S, Polakis I, Petsas T, Markatos EP. A systematic characterization of IM threats using honeypots. Proceedings of the 17th Network and Distributed System Security Symposium (NDSS), 2010.
26. Xu W, Zhang F, Zhu S. Toward worm detection in online social networks. Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC), 2010.
27. Kephart JO, White SR. Directed-graph epidemiological models of computer viruses. Proceedings of the IEEE Symposium on Security and Privacy, 1991.
28. Kephart JO, Chess DM, White SR. Computers and epidemiology. IEEE Spectrum, 1993.
29. Kephart JO, White SR. Measuring and modeling computer virus prevalence. Proceedings of the IEEE Symposium on Security and Privacy, 1993.
30. Khelil A, Becker C, Tian J, Rothermel K. An epidemic model for information diffusion in MANETs. MSWiM, 2002.
31. De P, Das SK. In *Epidemic models, algorithms, and protocols in wireless sensor and ad hoc networks*, Wiley (ed). Algorithms and Protocols for Wireless Sensor Networks: Hoboken, NJ, USA, 2008.
32. Chen Z, Ji C. Spatial-temporal modeling of malware propagation in networks. IEEE Transactions on Neural Networks: Special Issue on Adaptive Learning Systems in Communication Networks, 2005.
33. Zou CC, Gong W, Towsley D. Code red worm propagation modeling and analysis. ACM Conference on Computer and Communications Security (CCS), 2002.
34. Zyba G, Voelker GM, Liljenstam M, Mehes A, Johansson P. Defending mobile phones from proximity malware. Proceedings of INFOCOM, 2009.
35. Zheng H, Li D, Gao Z. An epidemic model of mobile phone virus. Proceedings of SPCA, 2006.
36. Yan G, Eidenbenz S. Modeling propagation dynamics of bluetooth worms (extended version). *IEEE Transactions on Mobile Computing* 2009; **8**(3): 353–368.
37. Mickens JW, Noble BD. Modeling epidemic spreading in mobile environments. ACM Workshop on Wireless Security (WiSe 2005), 2005.
38. Wang P, Gonzalez MC, Hidalgo CA, Barabasi AL. Understanding the spreading patterns of mobile phone viruses. *Science* 2009; 1071–1076.
39. Kannel. Kannel: open source WAP and SMS gateway. Available at: http://www.kannel.org
40. Thommes R, Coates M. Epidemiological modeling of peer-to-peer viruses and pollution. Proceedings of IEEE INFOCOM, 2006; 1–12.
41. Zou CC, Towsley D, Gong W. Email worm modeling and defense. Proceedings of the International Conference on Computer Communication and Networks, 2004.
42. Newman M, Forrest S, Balthrop J. Email networks and the spread of computer viruses. *Physical Review E* 2002; **66**.
43. Wikipedia. Mobile operating system. Available at:http://en.wikipedia.org/wiki/Mobile_operating_system