Risk Assessment for Mobile Systems through a Multi-layered Hierarchical Bayesian Network

Shancang Li, Theo Tryfonas, Gordon Russell, and Panagiotis Andriotis

***Abstract*—Mobile systems are facing a number of application vulnerabilities that can be combined together and utilized to penetrate systems with devastating impact. When assessing the overall security of a mobile system, it is important to assess the security risks posed by each mobile applications (apps), thus gain a stronger understanding of any vulnerabilities present. This work aims at developing a three-layer framework that assesses the potential risks which apps introduce within the android mobile systems. A Bayesian risk graphical model is proposed to evaluate risk propagation in a layered risk architecture. By integrating static analysis, dynamic analysis, and behavior analysis in a hierarchical framework, the risks and their propagation through each layer are well modelled by the Bayesian risk graph, which can quantitatively analyse risks faced to both apps and mobile systems. The proposed hierarchical Bayesian risk graph model offers a novel way to investigate the security risks in mobile environment and enables users and administrators to evaluate the potential risks. This strategy allows to strengthen both app security as well as the security of the entire system.**

***Index Terms*—Bayesian Risks Graphs, Mobile Security, Risk Assessment, Android Malware;**

I. Introduction

The Android-based platform has reached the top of the smartphone market, and claimed nearly 79% of smartphone share in 2013 [1], [2]. In the Google Apps marketplace, more than $1,192,749$ applications (*apps*) are available and this is increasing by 40% monthly (*i.e.* $32,000$ in Mar, 2014) [1]. Although Google has released an exciting security tool called the *application verification service* to protect against harmful Android apps in the market [2], it is still challenging to detect potentially harmful apps before the damage is done. The overall malware detection rate from among $1,260$ samples can be as low as 15.32% even with Google's new application verification service [2], and the overall detection rate utilising its anti-virus tools give an overall detection rates of 20.41% [3], which is significantly lower than the third part anti-virus apps that range from $51.02\% - 100\%$ [1]–[4].

Although awareness is constantly rising, the Android-based mobile systems are exposed to strong and significant security threats and many issues are reported each day [2], [4]–[8]. The peculiarities of mobile systems have been well known and many approaches have been developed to protect them. In general, a security solution for mobile systems begins with risk assessment to determine the threats and loss expectancy. Many risk assessment models have been proposed to model the risks, attacks, and threats in mobile systems, including attack graphs [5], attack tree [6], Bayesian belief model [9] and hybrid Bayesian network [10]. However many of these models fail to consider the causal-relations of risks, and consequently the contributory causes of threats are ignored [3], [11]–[13].

In [5] and [6], Bayesian networks are used to model vulnerability management and potential attack paths, which make it possible to quantatively evaluate the likelihood that an attack happens. In [10], a Bayesian network model is developed that can calculate the posterior probabilities of states of risks. Assessing the risks of apps or mobile systems is a crucial aspect in determining security state, information protection, and related security controlling. The capability of the Bayesian probabilistic model is a significant motivation to investigate risks analysis by analysing the likelihood of app (or system) compromise by chaining Bayesian rules in the proposed three-layered framework.

In this paper, the term 'risk' is used to denote an unsafe state or behavior of an app or system that is related to vulnerabilities or threats. In mobile systems there are two fundamental risks sources: (1) The apps distribution platform, the Google App Store, might be a source of potential security risks. Users generally always accept all permission requests requested during app installation. This can lead to overprivilege authority in apps and can lead to privacy and security-relevant vulnerabilities; (2) The fragmentation of apps, devices, and OS versions. The existence of multiple versions of Android and the huge number of developers may lead to customised modifications of Android, and these could be riddled with security vulnerabilities.

In mobile systems, security assessment generally involves the risk identification, risk analysis, and risk evaluation for both apps and the underlying system [13], [16]–[19]. This paper aims to develop a risk analysis framework to systematically assess the potential risks present in a mobile system. The hierarchical model decomposes risks in an app (or the system) with inter-dependencies into three layers (*static analysis layer, dynamic analysis layer, and behavior analysis layer*). Each layer focuses on specific aspects and needs. The framework then combines all specific risks in a coherent way and captures all possible causes of identified risks by applying a Bayesian Risk Graph.

Manuscript received Oct. 08, 2015; revised Nov. 05, 2015; accepted Feb. 20, 2016.

This work was supported by the ECR2015/2016 funding at Edinburgh Napier University and the European Commission under Grant HOME/2010/ISEC/AG/INT-002.

S. Li and G. Russell are with Computing, Netowrk, and Security Group, School of Computing, Edinburgh Napier University, Edinburgh, UK. (E-mail: s.li@napier.ac.uk, g. russell@napier.ac.uk. Tel: +44-131-455 2822.)

T. Tryfonas and is with the Faculty of Engineering, Bristol Cryptography Group, University of Bristol, Bristol BS8 1UB, UK. (E-mail: theo.tryfonas@bristol.ac.uk.)

P. Andriotis is with the Information Security Group, University College London, WC1E 7JE, UK. (E-mail: mr.panagiotis.andriotis@ieee.org.)

The hierarchical Bayesian risk graph (HBRG) features: (1) A different approach than most existing static analysis, this model integrates the potential risks analysis over: *static risk layer, dynamic risk layer, and behavioral risk layer*; (2) It can help users to find the dominating risk causes; (3) It is able to determine *out-system* cause and effects (*e.g.* for two different mobile systems A and B, it can analyse the probability that B will fail if A fails); and (4) Provide risks scores, so that when evaluating the potential risks in a mobile system, the model will grade each application with an app-related risk-score as well as a score for the whole mobile system.

II. Challenges in Mobile Risks Analysis

In building a secure mobile environment, several key strategic issues should be considered [20]: (1) Develop a risk model that can identify the potential risks of data loss based on a trusted risk assessment; (2) Provide a dynamic risk analysis module, which should be able to keep up with the rapid pace found in the Android ecosystem; (3) Evaluate each app installed using a risk assessment service that assigns them each a risk score. For example, if you install an app with a high-risk score, you should block or disable exploitable resources until the app is removed; (4) Layer the security, so that exploiting a system requires multiple vulnerabilities to penetrate the multiple layers of the security model.

Recently, a number of risk assessment methods have been proposed in the network security domain, such as the *probabilistic risk model* [6], *in-operability input-output models* [9], and *hierarchical holographic models* (HMM) [10]. In general, a complex system can be decomposed with inter-dependencies into several independent views, each of them focuses on different aspects and needs. One such set of views could include: static analysis aspects to model the problem, dynamic analysis to the security planning process, and behavior analysis to the susceptible behaviors analysis for apps. However in the Android mobile systems, the huge number of apps and platform variants makes the risk evaluation difficult, as the probability of each potential risk needs to be considered in security planning. The major challenges in performing an Android risk analysis can be summarized as follows:

1) *Security vulnerabilities are rampant*. As reported in [21], mobile app security concerns are growing in importance and 96% of all tested applications have one or more serious security issue, such as *privacy* (90%), *overprivileges* (80%), *session authentication* (53%), *input validation* (62%), and *infrastructure* (66%). This makes it difficult to find all potential risks when assessing apps and mobile systems [21].
2) *Complex mobile attacks*. An app might be threatened by a variety of security threats. The attackers could combine these multiple threats to form a complex and powerful mobile attack. The potential mobile threat could be classified into four categories: *App-based threats*, *Network threats*, and *Physical threats*. It is a challenging task to protect apps or mobile systems from a complex multi-step and multi-host attack. To deal with complex attacks, it is necessary to perform proactive and consistent app risk evaluation by both performing *efficiently detecting* and *remediating vulnerabilities* across the entire apps ecosystem.
3) *The complexity of risk detection. Early detection* might be the most efficient way to reduce vulnerabilities but it does not come without without weaknesses. Early detection may use **static security analysis** tools to identify vulnerabilities created during apps development, such as issues related to embedded information. In fact, most mobile app vulnerabilities relate to run-time activities. Dynamic risks analysis tools are expected to more accurately discover vulnerabilities that only appear when an app is launched. A **dynamic analysis process** can help in detecting run-time issues. In addition, a **behavior analysis** aspect to testing could be helpful in identifying potentially harmful long-term activities of apps.
4) *Risk communication strategy.* Communicating risks is an important challenge to protect apps and mobile systems from threats, allowing incident intelligence to identify risks before they become a focal point for mobile system attacks. In app-based systems, when an app attack happened, it is crucial to be able to predict the possibility that it will occur in other mobile systems, thus help to reduce the spread of similar attacks and the costs incurred. By communicating with developers, vendors, or app stores, the 'out-system risks analysis scheme' can create a roadmap of risk across multiple systems. To address the anticipated risks, users could for instance review the use of a particular app with the knowledge that it has been reported on another platform.

A mature risk assessment model should provide both real-time analysis and risk trend analysis over time for both apps and mobile systems. This paper meets these challenges by capturing vulnerability interdependencies and measuring security in a similar way that real attackers penetrate the mobile system, providing an assessment framework of overall systems risks as well as that related to individual apps.

III. Mobile Systems Risk Modeling

In risk assessment modelling it is essential to be able to identify what each app is actually doing. A hierarchical risk analysis architecture is able to accurately identify vulnerabilities, while a Bayesian risk graph can capture the interdependencies among the vulnerabilities posed by apps and the mobile systems. In this paper, hierarchical risk analysis is combined with a Bayesian risk graph to accurately model the risk states, propagations, and transitions.

Fig.1 shows a typical Hierarchical Bayesian Risk Graph (HBRG) model, which integrates a hierarchical risk analysis architecture into a Bayesian risk graph. It consists of a three-layer architecture and each layer extracts featured risks to form a directed acyclic graph (DAG). Each edge between nodes in the graph denotes the probabilistic causal dependencies. Based on the DAG, an Bayesian Network model could be created, in which each node maintains a conditional probabilities table (CPT). The parental nodes in the HBRG are assumed

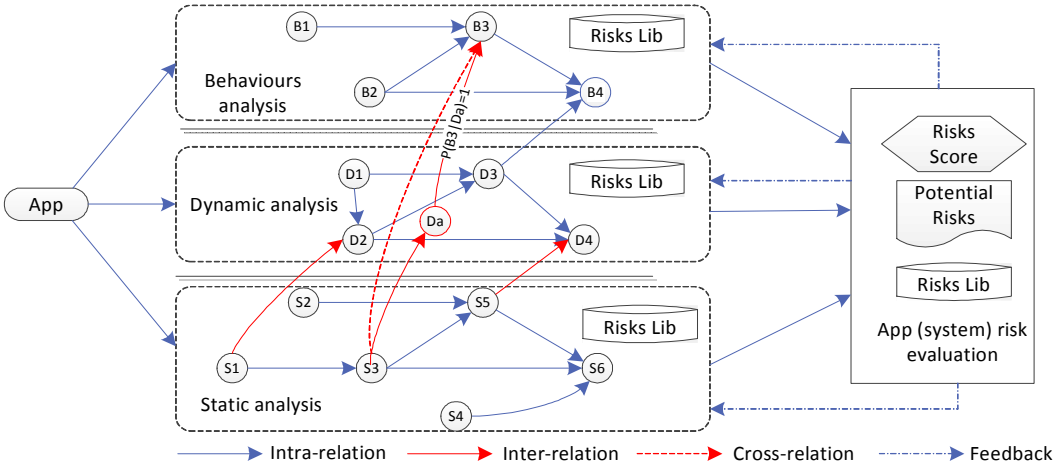


Fig. 1. Hierarchical Bayesian Risk Graph Model

to be marginally independent. The relations in the HBRGs denote the transition probabilities between nodes, which are classified into three aspects: (1) intra-layer relations, which connect the nodes within a layer; (2) inter-layer relations, cover the connections between two adjacent layers, and (3) cross-layer relations, denoting the links which bridge between the behavior layer and static layer. In order to reduce the computation complexity, cross-layer relation can be converted into two inter-layer relations by adding a *virtual node* concept. In Fig.1, the cross-layer relation from $S3$ to $B3$ can be translated into two inter-layer relations, where Da is the newly created *virtual node* and $P(B3|Da) = 1$. The *virtual node* Da can be seen as a copy of node $B3$ in dynamic layer without needing inter-relations in the dynamic layer. By doing this, the complexity of the relation space could be significantly reduced.

In HBRG, the model parameters (CPTs, transition probabilities) are always estimated using statistical learning algorithms [4], [5], [9]. Statistically, CPTs evaluation requires many samples to test whether a risk can or cannot cause other effect or make a contribution to other risks. Fortunately, the Genome dataset released by the *Malware Genome Project* [32] have collected more than $1,200$ malware app samples that cover the majority of existing Android malware families. This dataset has been successfully used to detect most existing malware apps. In this paper, the Genome dataset is used to statistically learn risk features at the static, dynamic, and behavioral layers to build accurate CPTs.

A. Bayesian Risks Graph Model

The process that one or more vulnerabilities propagate to one or more different threats could be defined with a dependence graph. The risk states or its propagation are usually constructed as a DAG and the transitions between nodes could be modelled with local conditional probabilities.

A DAG can be modelled with a Bayesian graph model $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$, in which the nodes $\mathbf{V}$ denotes the variables of risks, and the edges $\mathbf{E}$ denote relations between nodes that can be described with conditional probability distributions. For each variable $S_i \in \mathbf{V}$, a conditional probability distribution $P(S_i|Pa(S_i))$ is used to describe the transition. $Pa(S_i)$ denotes the parent set of S_i in $\mathbf{G}$. The Bayesian model reflects a conditional independence statement, which could significantly reduce the number of parameters needed.

The Bayesian network model needs to be extended to encode the *apps* risk conditions as a *Bayesian Risk Graph (BRG)*. The BRG is a powerful but intuitive tool for modelling, analysing, and predicting risks in mobile systems. It supports both app and mobile system security risk analysis. A unique joint probability distribution over $\mathbf{V}$, could be verified to be

$$P(S_1, S_2, \ldots, S_n) = \prod_{i=1}^{n} P(S_i|\pi_i) \tag{1}$$

where $S = \{S_1, S_2, \ldots, S_n\}$ is a set of Bernoulli random variables and $P(S_i|\pi_i)$ denotes the conditional probability of S_i given the condition of π_i. For example, for a set $S = \{S_1, S_2, S_3, S_4, S_5\}$, the joint probability of all the variables can be derived with Eq. (1)

$$\begin{aligned} P(S_1, S_2, S_3, S_4, S_5) = P(S_1|S_2) \cdot P(S_2|S_3, S_4) \cdot \\ P(S_3|S_4) \cdot P(S_4) \cdot P(S_5) \end{aligned} \tag{2}$$

Definition 1: *Conditional probability table (CPT).* In a BRG, the quantitative parameters must be specified. Each node maintains a CPT to describe the conditional probability distribution for a particular combination of values of its parental nodes.

Definition 2: *Hierarchical Risk Template.* A hierarchical-risk-template is a generic property of the structured risk-template that at least includes one of three basic properties:

1) *Behavior Risk*, which includes the potential threats that are based on the behavior analysis;
2) *Dynamic Risk*, which involves potential risks based on the dynamic analysis, such as app activities, network activities, etc.
3) *Static Risk*, which includes the potential risks and threats based on the static analysis, such as malware, size analysis, permission analysis, virus matching, etc.

The *hierarchical risk template* provides BRG a template that describes the basic potential risks of an app or a mobile

system. It can help us categorize the properties of risks in mobile systems. For example, 'accessing the contact list' is an instance of dynamic risk.

Definition 3: *Risk Attribute*. This is a binary attribute representing the state of an instance of a risk. A risk S is associated with a state $(S = 1, true)$ or $(S = 0, false)$. The probability that risk S happens is $Pr(S = 1)$. When an app (or system) is compromised it means the risk state of the app (or system) is $true(S = 1)$. Therefore

$$Pr(\neg S) = 1 - Pr(S) \tag{3}$$

is the probability of the state being $S = 0$.

Definition 4: *Atomic Risk.* Atomic risk is a risk that cannot be further broken down into smaller parts and retain their meaning in the context of an risk. Typical example is vulnerability identifiers. Let $\mathbf{S}$ denote a set of risks. $\mathcal{A} : \mathbf{S} \times \mathbf{S} \rightarrow [0, 1]$ denotes a set of relationships in $\mathbf{S}$. Then given $S_{pre}, S_{post} \in S$, and thus $a : S_{pre} \mapsto S_{post}$ can be identified as an atomic risk if

1) $S_{pre} \neq S_{post}$
2) given $S_{pre} = 1$, $S_{post} = 1$ with probability $\mathcal{A}(S_{pre}, S_{post}) > 0$
3) $\nexists S_1, \cdots, S_j \in S - \{S_{pre}, S_{post}\}$.

Hence, $\mathcal{A} - S$ allows an action to compromise the risk S_{post} from S_{pre} with a nonzero probability. Thus S_{pre} is the stage before the risk occurred, and S_{post} is the state after the risk occurred.

Definition 5: *Bayesian Risks Graph (BRG)*. A BRG is a tuple $(S, \xi, \mathcal{T})$, in which $S = R_B \cup R_D \cup R_S$ denotes the set of attributes from three risk layers. R_B, R_D, and R_S denote the risks from the *behavior layer*, the *dynamic layer*, and the *static layer*, respectively.

$\xi \subseteq S \times S$ denotes the space of risks. A risk states pair $(S_{pre}, S_{post}) \in \tau$ if $S_{pre} \mapsto S_{post} \in \mathcal{A}$. Furthermore, for $S_i \in S$, the set $Pa[S_i] = \{S_j \in S | (S_j, S_i) \in \xi\}$ is called the parent set of S_i. Here $\mathcal{T}$ denotes conditional probability table (CPT).

Fig. 2 shows a simple BRG example, in which the risk r_a could be caused by sub-risks r_b and(or) r_c, and iteratively r_b could be redivided into r_e and r_d, where r_d, r_e, and r_c are atomic risks and thus could not be further divided. The basic idea of the BRG is to calculate the probability of r_a according to its parental conditional probabilities.

Definition 6: *Risks Combination*. For a risk S_j in a BRG, $S_j \in S_R \cup S_D \cup S_B$, then assuming that $S_i \in Pa[S_j]$ and $S_i \mapsto S_j$, then there are two kinds of possibilities to consider for risks combination: AND-combination and OR-combination. As shown in Fig. 3, the AND-combination denotes each sub-risk is a distinct event and the chance of r_a to be $true$ depends on the success of each individual subrisk (r_b and r_c). For the OR-combination, the chance of r_a to be $true$ depends on the success of at least one of individual subrisks (r_b or r_c). The probability could be defined as

$$Pr(S_i | Pa[S_i]) = \begin{cases} Pr(\cap_{S_i=1}(a_i)), & AND \\ Pr(\cup_{S_i=1}(a_i)), & OR \end{cases} \tag{4}$$

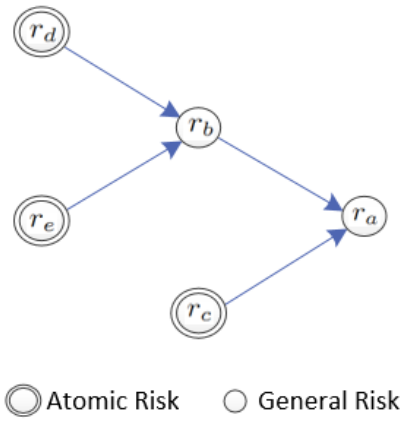


Fig. 2. A simple example of BRG

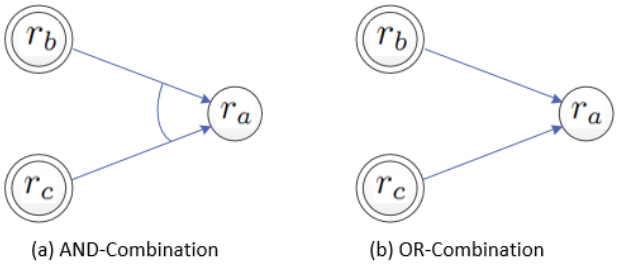


Fig. 3. Risks Combination in BRG

B. Hierarchical BRG

The HBRG is a hierarchical model formed by a three layer Bayesian risk graph. Fig. 1 shows a typical HBRG model that consists of a BN (Bayesian Network) model at each layer. A risk-evaluation process performs at each layer and a compound risk evaluation will be given for each app (or system).

A HBRG can be represented with a double combination of $\mathbf{G}_{\text{HBRG}}$ and a relationships set $\mathbf{R}$ as $< \mathbf{G}_{HBRG}, \mathbf{R} >$.

1) $\mathbf{G}_{\text{HBRG}} = \{\mathbf{G}_S, \mathbf{G}_D, \mathbf{G}_B\}$ denotes the BRGs model, in which $\mathbf{G}_S$ denotes the BRG at the static layer, $\mathbf{G}_D$ denotes the BRG at the dynamic layer, and $\mathbf{G}_B$ denotes the BRG at the behavioral layer, respectively. The relationships among random variables at each node are constrained by the model structure and a directed acyclic graph (DAG) obeying the usual conditional independence assumptions. Specifically, the arcs between the BRG nodes represent probabilistic causal dependencies. For each BRG, the joint probability distribution function is defined as in Eq.(1).
2) Relation $\mathbf{R}$, a key concept of the HBRN model, provides the bridge among the layered BRG. $\mathbf{R}$ is a set of associations $\{R_{ij} | (i = 1, 2, \ldots, m); j \in (1, 2, \ldots, n)\}$ with $R_{ij} = 1$ implying that node S_i is assigned to state $true$ of relationship with node S_j.

IV. Hierarchical Risk Assessment with BRGs

The Android security model is primarily based on a *sandbox* and *permissions* mechanisms, in which an app is running in a specific Dalvik virtual machine with a unique user ID. In general, each app runs in isolation from all others apps, and so each app cannot access to other app's resources. An app is signed using a certificate by the developer, to confirm the integrity of the app. To conserve resources, apps with the same ID can also arrange to run in the same process, sharing the same VM. Apps can be granted permissions, and these are displayed to the user for approval before the app is installed. However, users tend to ignore the permissions that an app requests, and there are many reasons for this to happen [22]–[25].

Forensically, the risk detection, finding, and evaluation are important for providing users a safe mobile platform. The hierarchical risk assessment framework is able to conduct three layers of in-depth risk analysis for both apps and mobile systems.

A. Static Risk Analysis

A number of static risk analysis methods have been developed to examine the potential risks that an app can exhibit [1], [2], [26]. However, most of them are constrained to certain performance capabilities due to their inherent limitation of not actually executing the code. In Android systems, the static analysis is mainly focused on disassembling the binary file (APK) to check source code (e.g. `classes.dex, smali, AndroidManifest.xml`). This permits the static risk analysis to scan an app's permission requests, API calls, and check discrepancies between the two, including the following stages: (1) app-decompile into smali format; (2) information extractor; (3) API calls parse; (4) permission analysis; (5) grouping of critical API calls; (6) embedded sources; (7) log monitor and parser. Commonly used analysis tools have been developed, such as *apktool, baksmali, fiwalk,* and *icat* [25].

The static analysis can also provide insights into the third part software development kits (SDKs), libraries, API calls, and app functions that are employed within an app. Most of the existing mobile security schemes are using static risk analysis to identify malicious apps [1], [2], [26]. For example, in [27] Zhou and Jiang found that the BOOT_COMPLETED event registration was used in over 83% of the malware apps sample dataset, and given the assumption that this event registration would occur less frequently in legitimate applications it thus could be used as an indicator for identifying malicious applications. The limitation of static analysis code mapping is that unable to effectively evaluate any permissions abuse which may occur in the Android system. It also cannot cover dynamic and forgery bytecode calls that can generate false negative in terms of permissions requested [28].

The main goal of static analysis is to statistically find ways to trigger potentially malicious or risky behaviors before the app runs. Using static risk analysis, one can identify main static risks in an app. The output of static analysis could be used to guide the dynamic analysis, or even help parametrise the dynamic analysis directly. This can significantly reduce the parameters needed when performing dynamic analysis and thus improve its performance.

B. Dynamic Risk Analysis

After completing the static analysis stage, the framework performs a real-time analysis, which evaluates apps under run-time conditions on instrumented emulators or modified hardware devices. The dynamic analysis module could run over a server and perform concrete execution of apps while extracting certain details related with security. The analysis includes the execution of *Dalvik bytecode* and contains data structures related to a single execution flow through an app, while other methods also contain a stack to temporarily retain data about each method that is called at run time. As the execution encounters method calls and returns, this stack grows and shrinks. In this layer, the framework mainly focuses on detecting the following risks and exploits:

(1) *Intent-based risks*. Each app is composed of several components (`Activity, Service, Broadcast Receiver, Content Provider`) that can communicate between each other using `Intent` message (for both inter and intra app communication). However, wrong information exchange between activities could leak information. The Intent-based risks include: *Unauthorized Intent Receipt*, *Broadcast Theft*, *Activity Hijacking*, *Service Hijacking*, and *Special Intent*.

(2) *Intent Spoofing*. Malicious apps are able to launch an Intent spoofing attack by sending an Intent to an exported component that is not expecting Intents from that app. This kind of risks include *malicious broadcast injection* and malicious activity/service launch.

(3) *Component Analysis*. The main goal of this component is to check susceptible risks for each components in an app by examing the app's *manifest file, Dalvik instructions, etc*.

(4) *Android network sniffing*. It includes detection of risky network activities, such as *networking vulnerability scanning/search, illegal login, http POST/GET, https hijacking, etc*.

In dynamic risks analysis, a number of tools have been developed such as ComDroid [29], DroidMOSS [30], and Paranoid [31]. The output of this layer includes: (1) components analysis, (2) inter-app communication, (3) network connection analysis, (4) method call graph, (5) control flow graph, and (6) a list of relevant in-app behaviors, includes *execution of binaries, execution of commands, network events detection, loading of libraries, I/O accessed,* and *dynamic class-loading*. These outputs can be used to determine vulnerabilities or guide further behavior risk analysis.

These results could easily be modified to search for and record any functionality or behavior deemed of interest. Dynamic analysis could check the proactive risks, such as *auto-updating, auto-downloading,* and *repacking*.

C. Behavior Risk Analysis

The behavior layer identifies risk behaviors and measures an app's actual activity over time. By leveraging the risks found in the static, dynamic, and behavior layers, the framework can discover hidden, risky behaviors within a mobile app or

system. It can then combine all risks to provide an overall risk score, and thus create custom mobile policy. The main goal in this layer is to identify risk behaviors that might bring risks to an app or the mobile system. This process collects both security and privacy information on how an app performs its potentially risky run-time activities. Typical tests in this layer include:

- *Auto exercises apps at run-time*. Each app is auto-loaded into a propriety emulator designed to test each app as it would run in its native environment.
- *Perform credential auto-login*. This test replicates login activity by auto-loading necessary credentials for full execution of the app's functionality. This ensures that HBRG can test all aspects of the app's functionalities beyond the login screen.
- *Collect network activity for both SSL and non-SSL activities*. Tests designed to monitor any network activity, essentially to see if the app is accessing any third party networks. By doing this, one can at a minumum identify whether such traffic was properly encrypted or not.
- *Determine if app launches other apps during run-time*. This security test checks to see if an app sends requests to access/launch other apps during run-time which may allow malicious or risky activity to occur. This type of exposure may allow malicious apps to be 'silently' installed on the user's devices as well as giving them the ability to auto-execute apps with malicious intent.
- *Monitor read/write access to device directory and hardware*. This test monitors access to device hardware (i.e. the camera) and also monitors which system files the app can access and how encryption is used for saving information.
- *Abnormal behaviors detection*. This could be undertaken by checking for abnormal behaviors, such as privacy collections, regularly information uploading, using HTTP POST methods, receiving commands, etc.

The typically behavioral analysis methods used in this layer include (1) *Machine learning classifier*: based on previous training experience and after testing and weighing the security issue, the classifier decided the potential risks that a behavior may bring; and (2) *Feature selection*: the risky features selection methods are applied for behavior datasets. The behavioral analysis is an effective solution to use characteristic and behavioral-based methods to detect malware by observing the static and/or dynamic behavior and features of mobile systems.

D. Risk Score

The causal risks could be a compound of a set of sub-risks produced by the joint effect of other risks. By using the Bayesian theorem, a numerical analysis can be performed to find the probability for each causal-risk. By refining the CPTs, the HBRG can determine (i) the probability that an attack occurred on an app (or system), (ii) the probability that an app (or system) will fail.

Given the risk values for each app (or system), a weighted risk score for each app (or system) can be easily calculated

$$R_S^w = wP(S), (R_S^w) \in [0, 3] \tag{5}$$

in which $P(S) \in [0, 1]$. An app may be threatened by multiple risks, and each risk may lead an individual threat. In this case, the risk level can be calculated with weighted probabilities of each risk [5]. The $Risk_Score$s are classified into three categories: $Low : [0.0 - 1.0)$, $Medium : [1.0 - 2.0)$, and $Severe : [2.0 - 3.0]$.

E. Contributory Causes Tracking

1) Independent Contributory Causes Tracking: Forensically, when a risk occur, it is important to find all contributory causes or the probability of causes that make it happened. Assume $\mathbf{s} = \{s_1, s_2, \ldots, s_k\}$ denotes a group of contributory causes of an attack. Thus

$$p(\mathbf{s}|a) = \sum_{i=1}^{N_s} \alpha_k(i) \tag{6}$$

in which N_s denotes the number of nodes that cause the attack a; the α_ks are the forward variables. Using the likelihood ratio:

$$\begin{aligned} P(a|\mathbf{s}) &= \frac{P(\mathbf{s}|a)P(a)}{P(\mathbf{s})} \\ &= \frac{P(\mathbf{s}|a)P(a)}{P(\mathbf{s}|a)P(a) + P(\mathbf{s}|\overline{a})P(\overline{a})} \end{aligned} \tag{7}$$

where $P(a)$ represents the prior probability of a. The posterior probability is the node's belief on the existences of $\mathbf{s}$ based on the observation of a. It is the probability of causes that allowed the attack to happen. For each cause in $\mathbf{s}$, it is possible to calculate the probability that contributed to a.

$$P(s_i|a) = \frac{P(\mathbf{s}|a)P(a)}{P(\mathbf{s})} = \frac{P(\mathbf{s}|a)P(a)}{P(\mathbf{s}|a)P(a) + P(\mathbf{s}|\overline{a})P(\overline{a})} \tag{8}$$

2) Multiple Contributory Causes Tracking: If an attack happened, the probability of any individual risk could be calculated by applying above approach. Similarly, the model is able to calculate the probability of any individual risk group assumed to cause the attack. The group causes tracking method enables the identification of the group causes of an attack after it happened.

$$P(a_1|\mathbf{s}) = P(a_1\overline{a}_2|\mathbf{s}) + P(a_1a_2|\mathbf{s}) \tag{9}$$

$$P(a_2|\mathbf{s}) = P(\overline{a}_1a_2|\mathbf{s}) + P(a_1a_2|\mathbf{s}). \tag{10}$$

Then, we have

$$P(a_1a_2|\mathbf{s}) = \frac{P(\mathbf{s}|a_1a_2)P(a_1a_2)}{P(\mathbf{s})} \tag{11}$$

in which,

$$\begin{aligned} P(\mathbf{s}) &= P(\mathbf{s}|a_1a_2)P(a_1)P(a_2) + P(\mathbf{s}|a_1\overline{a}_2P(a_1)p(\overline{a}_2) \\ &\quad + P(\mathbf{s}|\overline{a}_1\overline{a}_2)P(\overline{a}_1)P(\overline{a}_2) \\ &= P(\mathbf{s}|a_1a_2)P(a_1)P(a_2) + P(\mathbf{s}|a_1\overline{a}_2P(a_1)p(\overline{a}_2) \end{aligned} \tag{12}$$

F. Determining out-system causal effects

An app could be installed by many users. When an app has failed (or been compromised) on a mobile system, it is important to predict the possibility that the same app will fail (or will be compromised) on other mobile systems. In

fact, the observation that an app failed could increase the possibility that an attack will occur on other systems. This prediction could be helpful for refining the CPTs for other systems. Accordingly, for two mobile system $\mathbf{S}_1$ and $\mathbf{S}_2$,

$$P(\mathbf{S}_2) = P(\mathbf{S}_1 \cap \mathbf{S}_2) + P(\overline{\mathbf{S}}_1 \cap \overline{\mathbf{S}}_2) \quad (13)$$

we further have

$$\begin{aligned} P(\mathbf{S}_2) &= P(\mathbf{S}_1 \cap \mathbf{S}_2) + P(\overline{\mathbf{S}}_1 \cap \mathbf{S}_2) \\ &= P(\mathbf{S}_2|\mathbf{S}_1)P(\mathbf{S}1) + P(\mathbf{S}_2|\overline{\mathbf{S}}_1)P(\overline{\mathbf{S}}_1). \quad (14) \end{aligned}$$

Hence, when an app fails on a mobile system, it implies the increase of the probability that the app will fail if it is installed on other mobile systems. Similarly, when an attack happened on a mobile system, the probability that an attack with same contributory causes will happen on other systems will also increase.

V. Risks Analysis with HBRG - A Case Study

As discussed in above sections, the HBRG model is able to analyse risks associated with apps or mobile system by integrating the static analysis, dynamic analysis, and behavior analysis into a Bayesian risk graph. One possible real-world risk is that posed by the Fake app patch, which is an issue which can be found in the Android environment.

For instance, Google published an app 'Android Market Security (AMS)' to undo the side effects caused by Android. *Rootcager* is a Trojan horse that steals information from Android device. This app was automatically pushed to devices of users who had downloaded and installed an infected app. However, a repackaged version of AMS was found on third party app markets, and for discussion purposes the authors name it rAMS. The rAMS might threaten the users' privacy information and may cause other issues. As an example, HBRG was used to analyse rAMS to demonstrate how HBRG can be used in a risk analysis.

In risks evaluation of an app or system, the statistically learned CPTs require many samples to test whether one can or cannot cause other effect or make contribution on other risks. To characterise existing Android risky malware, particularly within static and dynamic analysis, most known risky features can be collected and classified that cover the majority of existing Android malware families. This helps to build a risks library for efficient evaluation of mobile security. Fortunately, the *Malware Genome Project* has collected more than $12,000$ malware samples that cover the majority of existing Android malware families. It has been successfully used to detect most existing malware apps. In this paper, CPTs have been evaluated based on statistic results with the reported harmful features of apps.

TABLE I
Static Analysis (size)

apps	Total (KB)	Apps (KB)	Data (KB)	Cache(KB)
AMS	140	140	0.00	0.00
rAMS	208	196	12.00	0.00

In static analysis layer, the following features are first checked: (1) Size, (2) Permission requested by AMS and rAMS, (3) Package structure. This is shown in Table. I and Table. II. The size of rAMS is bigger than that of AMS, which might be caused by repacking risks. It can be identified that the requested permissions are different and the rAMS requests extra permissions that could bring risks to the app and system. After checking the package structure, it is found that the structure for AMS and rAMS are different. AMS includes a 'google.android' package that contains three sub-packages `googleapps` and `googlelogin`, however in rAMS, the `google.android` package was obviously repackaged as `mss.bg`, which includes three new packages: `transaction, ui, util`. The *eighteen* potential static risks in rAMS are analysed in Fig.4, which involves *permissions, internet accessing, storage,* and *privacy*, which are organized as 18 BRG nodes from $S1$ to $S18$ as shown in Fig.5. All risks combinations in this example are assumed to be OR-combinations for simplicity.

TABLE II
Static Analysis (Permissions)

permissions	AMS	rAMS
Storage	Y	Y
Location	N	Y
Network	Y	Y
Phones Call	Y	Y
Default	Y	Y
Message	N	Y
Money	N	Y
Systems	N	Y

In the dynamic layer, further analysis of the dynamic risks can be carried out. By de-assembling the app it was found that the rAMS is collecting the following information: `(imei, version, smscentre, handled, pid, install time, sysversion, author, fare, and phonenumber)`. The collected information was saved in a hidden file `upload.xml`.

In the dynamic layer, it was found that the rAMS used the HTTP POST method and the related URI was identified as `http://www.youlubg.com:82/Coop/request3.php`. This means that the rAMS was able to send the collected information onwards to a remote server. Based on the analysis at static layer, the following potential risks were identified: *repacking, sending SMS, receiving SMS, http post, http get, block sms*, which are organized as BRG nodes $D1$ to $D7$. Two virtual nodes Da and Db are created and mapped to $B3$ and $B4$, respectively.

Similarly, at the behavior layer, behavioral risks could be identified based on matching with the *behavioral risk lib*. It is found that the rAMS is able to interact with the above identified server. By further checking the network activities of rAMS, it was found that the rAMS could receive commands from the reply generated by the POST request and save the commands into a hidden file `serverinfo.xml`. This identified a risk that the remote attackers are able to send SMS messages from the compromised device. rAMS also has the capability to block incoming SMS messages. Based on the behavior features extracted from *Genome Project*, four behavior risks are recognized from rAMS: *B1: Costs; B2:*

Regularly connect third part Server; B3: Collecting Privacy Data; B4: System Leakage as shown in Fig.4.

A HBRG could be built as shown in Fig.5, in which the blue edges denote the Intra-relations and the red edges denote Inter-relations. Two *virtual nodes*, Da and Db, are created to reduce the relation space. Then is trivial to transform the HBRG model to a BN model, where the probabilities and CPTs are pre-specified according to the statistical learning data. Fig.6 (a) shows the risk analysis at the static layer for rAMS, in which the joint probabilities are shown in the table. As an example, Fig.6 (b) shows the CPT maintained by node 'LOG_PRV'. For rAMS, the static layer analysis shows there is only a 25.7% chance of rAMS at risk.

TABLE III
RISK ANALYSIS FOR RAMS USING HBRG

Node	Risk	Layer
S1	Size.Total	Static Layer
S2	Size.Data	Static Layer
S3	Size.Application	Static Layer
S4	Permision.RECEIVE_BOOT_COMPLETE	Static Layer
S5	Permision.RECEIVE_SMS	Static Layer
S6	Permision.SEND_SMS	Static Layer
S6	Permision.ACCESS_NETWORK_STATE	Static Layer
S7	Permision.CHANGE_NETWORK_STATE	Static Layer
S8	Permision.INTERNET	Static Layer
S9	Permision.SYSTEM_TOOLS	Static Layer
S10	Permision.WRITE_EXTERNAL_STORAGE	Static Layer
S11	Permision.ACCESS_COARSE_LOCATION	Static Layer
S12	SMS Access	Static Layer
S13	Storage	Static Layer
S14	SYStem Version	Static Layer
S15	IMEI	Static Layer
S16	Install time	Static Layer
S17	Location	Static Layer
S18	Log reading	Static Layer
S19	Static Risks	Static Layer
D1	Repackage	Dynamic Layer
D2	HTTP Request	Dynamic Layer
D3	HTTP Response	Dynamic Layer
D4	Send SMS	Dynamic Layer
D5	Receives SMS	Dynamic Layer
D6	Block SMS	Dynamic Layer
D7	Dynamic risks	Dynamic Layer
Da	Collecting Privacy	Dynamic/Behavior
Db	System Leakage	Dynamic/Behavior
B1	Costs	Behavior Layer
B2	Connection third part Server	Behavior Layer
B3	Collection Privacy	Behavior Layer
B4	System Leakage	Behavior Layer
B5	Behavior risks	Behavior Layer

Similarly, based on HBRG, the app probability that rAMS is in risk was found to be 88.5% at dynamic layer and 46% at behavior layer. Fig. 7 shows the overall risk of rAMS is 77.7% and the *risk score* is $2.325(\in [2.0 - 3.0])$, which means rAMS is considered to be at the *severe* risk level.

In practice, if a risk occurred, it is possible to analyse the possible contributory causes and determine out-system causal effects as discussed in above sections.

VI. SUMMARY AND FUTURE WORK

This paper proposed a HBRG model as a risk evaluation framework in Android mobile systems. The HBRG can provide users with a friendly risk evaluation framework, which is able to show users where the potential causes of risks exist. It will lead users to select apps with lower risk and help to know how secure their apps or systems are. It is also possible that this model will allow people to well understand the potential risks in their system and create an incentive for developer to create lower-risk apps that do not contain invasive ad networks and avoid over-requesting permissions. This study is not the last word on the question of how to best present risk information, but future work will continue to investigate hidden risk mitigation and propagation in Android systems.

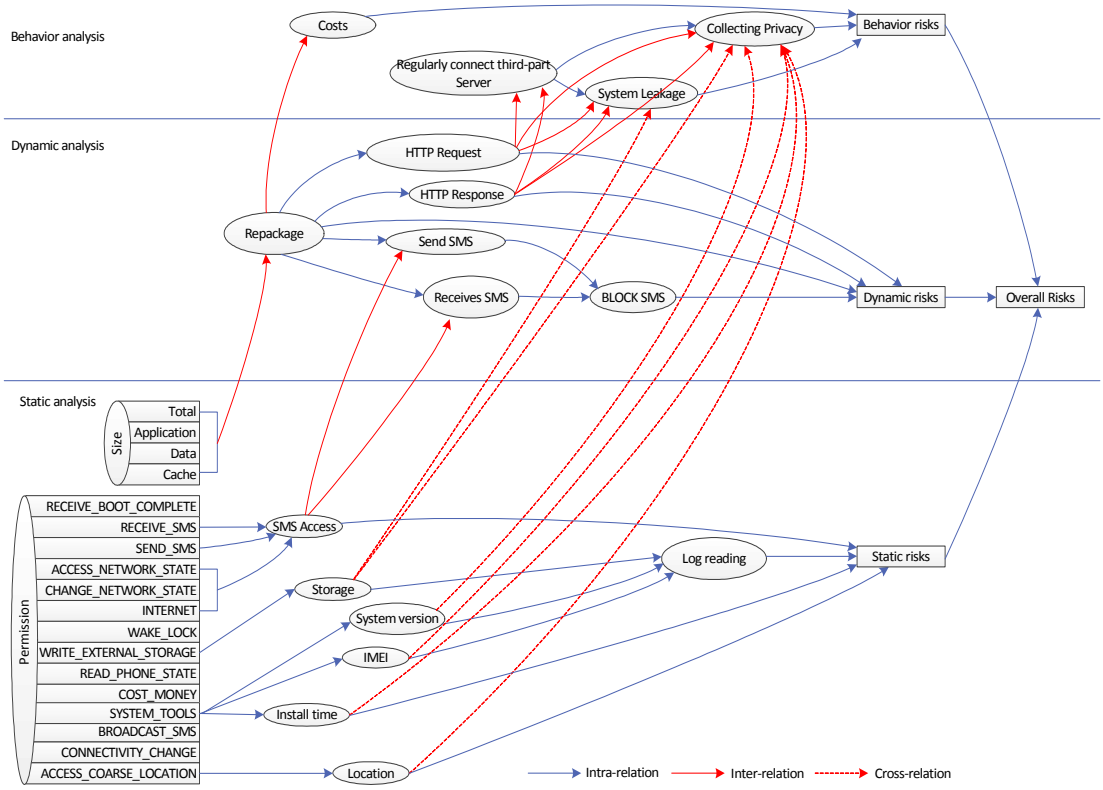


Fig. 4. Risks analysis for rAMS using HBRG

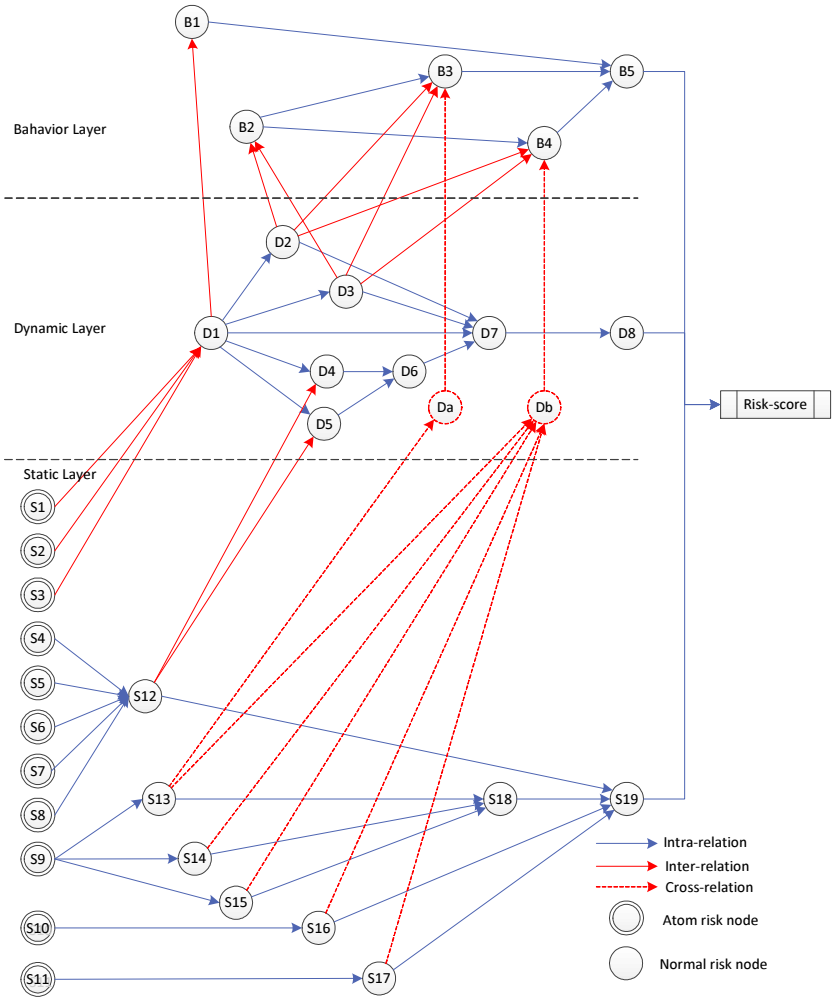
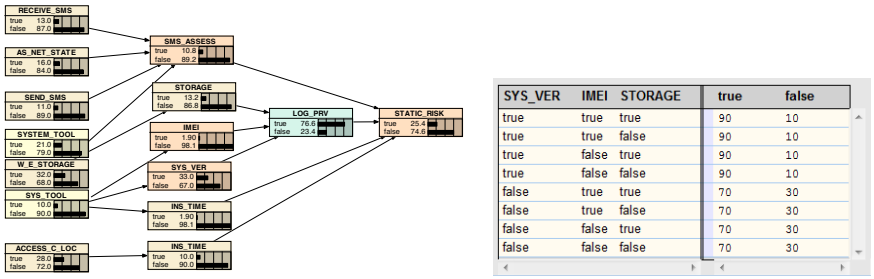


Fig. 5. HBRG model for rAMS



(a) Risks analysis at Static Layer for rAMS

SYS_VER	IMEI	STORAGE	true	false
true	true	true	90	10
true	true	false	90	10
true	false	true	90	10
true	false	false	90	10
false	true	true	70	30
false	true	false	70	30
false	false	true	70	30
false	false	false	70	30

(b) CPT at node LOG_PRV

Fig. 6. Risks analysis on Static Layer for rAMS

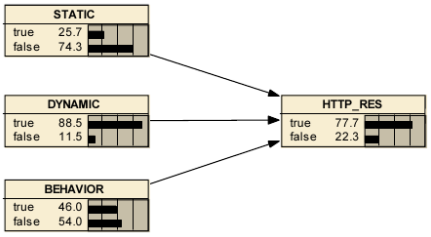


Fig. 7. Risks analysis on Static Layer for rAMS

REFERENCES

[1] V. Rastogi, Y. Chen, and X. Jiang, "DroidChameleon: Evaluating Android anti-malware against transformation attacks, *in Proc. ACM ASIACCS* , pp.329-334, May 2013,

[2] Vaibhav Rastogi, Yan Chen, Xuxian Jiang, "Catch Me If You Can: Evaluating Android Anti-Malware Against Transformation Attacks", *IEEE Transactions on Information Forensics and Security*, vol.9, no.1, pp.99-108, Jan. 2014.

[3] Xuxian Jiang, " An Evaluation of the Application ("App") Verification Service in Android 4.2", *http://www.cs.ncsu.edu/faculty/jiang/appverify/*, available on 04 Nov, 2015.

[4] Yajin Zhou, Xuxian Jiang, "Dissecting Android Malware: Characterization and Evolution", *IEEE Symposium on Security and Privacy*, pp.95-109, 2012.

[5] Poolsappasit, N., Rolla, MO, USA ; Dewri, R. ; Ray, I., "Dynamic Security Risk Management Using Bayesian Attack Graphs", *IEEE Transactions on Dependable and Secure Computing*, vol.9, no.1, pp.61-74, Jan. 2012.

[6] Yu Liu and Hong Man, "Network Vulnerability Assessment Using Bayesian Networks",*Proceedings of the SPIE*, vol.5812, pp.61-71, 2005.

[7] D Wu, D. L. Olson, John Birge. "Operational Research in Risk Management", *Computers & Operations Research*, vol.39, no.4, pp.751-752, 2012.

[8] D Wu, David L. Olson, John R. Birge. "Introduction to Special Issue on Enterprise risk Management in Operations", *International Journal of Production Economics*, vol.134, no.1, pp.1-2, 2011.

[9] Haiying Tu, Allanach, J., Singh, Satnam, Pattipati, K.R., "Information integration via hierarchical and hybrid bayesian networks", *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol.36, no.1, pp.19-33, Jan. 2007.

[10] Luis M. de Campos, Juan M. Fernndez-Luna, Juan F. Huete, "A Layered Bayesian Network Model for Document Retrieval", *LNCS Advances in Information Retrieval*, vol.2291, pp.169-182, 2002.

[11] Desheng Dash Wu, "Selling to the Socially Interactive Consumer: Order More or Less?", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol.45, no.3, pp.399-410, 2015.

[12] D. Wu, C. Luo, D. Olso. "Efficiency Evaluation for Supply Chains Using Maximin Decision Support", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol.44, no.8, pp.1088-1097, 2014.

[13] Panagiotis Andriotis, George Oikonomou, Theo Tryfonas, and Shancang Li, "Highlighting Relationships of a Smartphones Social Ecosystem in Potentially Large Investigations", *IEEE Transactions on Cybernetics*, (DOI: 10.1109/TCYB.2015.2454733), 2015.

[14] Shancang Li, George Oikonomou, Theo Tryfonas, Thomas Chen, L Xu, "A Distributed Consensus Algorithm for Decision Making in Service-Oriented Internet of Things", *IEEE Transactions on Industrial Informatics*, vol.10, no.2, pp.1461-1468, 2014.

[15] Suleyman Kondakci, "Network security risk assessment using Bayesian belief networks", *IEEE International Conference on Social Computing and Privacy Security Risk and Trust*, pp.952-960, 2010.

[16] Jian Li, Xiaolong Li, Bin Yang, Xingming Sun, "Segmentation-based Image Copy-move Forgery Detection Scheme," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 507-518, Mar. 2015.

[17] Zhihua Xia, Xinhui Wang, Xingming Sun, and Qian Wang, "A Secure and Dynamic Multi-keyword Ranked Search Scheme over Encrypted Cloud Data," *IEEE Transactions on Parallel and Distributed Systems*, (DOI: 10.1109/TPDS.2015.2401003), 2015.

[18] Bin Gu, Victor S. Sheng, Keng Yeow Tay, Walter Romano, and Shuo Li, "Incremental Support Vector Learning for Ordinal Regression," *IEEE Transactions on Neural Networks and Learning Systems*, (DOI: 10.1109/TNNLS.2014.2342533), 2015.

[19] L Xu, H. Wu, S. Li, "Internet of Things in Industries: A Survey ", *IEEE Transactions on Industrial Informatics*, vol.10, no.4, pp.2233-2243, 2015.

[20] InfoWord, "A clear-eyed guide to Android's actual security risks", online: *http://www.infoworld.com/d/mobile-technology/clear-eyed-guide-androids-actual-security-risks-232034?page=0,2*, available on 25 Jun, 2014.

[21] Cenzic Inc, "Application vulnerability trends report", online: *http://www.cenzic.com/downloads/*), available on Aug 2014.

[22] Rr1. E. Chin, A.P. Felt, V. Sekar, and D. Wagner, Measuring User Confidence in Smartphone Security and Privacy, *Proc. Eighth Symp. Usable Privacy and Security (SOUPS 12)*, pp. 1-16, 2012.

[23] A.P. Felt, K. Greenwood, and D. Wagner, "The Effectiveness of Application Permissions, *Proc. Second USENIX Conf. Web Applica- tion Development (WebApps 11)*, 2011.

[24] P.G. Kelley, S. Consolvo, L.F. Cranor, J. Jung, N. Sadeh, and D. Wetherall, "A Conundrum of Permissions: Installing Applications on an Android Smartphone, *Proc. Workshop Usable Security (USEC 12)*, Feb. 2012.

[25] Christopher S. Gates, Jing Chen, Ninghui Li, and Robert W. Proctor, "Effective Risk Communication for Android Apps", *IEEE Transactions on Dependable and Secure Computing*, vol.11, no.3, pp.252-266, May. 2014.

[26] Mark Guido, Jared Ondricek, Justin Grover, David Wilburn, Thanh Nguyen, Andrew Hunt, "Automated identification of installed malicious Android applications", *Digital Investigation*, vol.10, pp.96-104, 2013.

[27] Xuxian Jiang, "An Evaluation of the Application ("App") Verification Service in Android 4.2", online available, *http://www.csc.ncsu.edu/faculty/jiang/appverify/*, Jul. 2014.

[28] Naser Peiravian and Xingquan Zhu, "Machine Learning for Android Malware Detection Using Permission and API Calls", *IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI)*, Herndon, pp.300-305, Nov. 2013.

[29] Erika Chin Adrienne Porter Felt Kate Greenwood David Wagner, "Analyzing Inter-Application Communication in Android", *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pp.239-252, New York, 2011.

[30] Portokalidis, P. Homburg, K. Anagnostakis, and H. Bos, "Paranoid Android: Versatile protection for smartphones", *in Proc. 26th Annual ACM Computer Security Applications Conference*, Austin, USA, pp.347-356, 2010.

[31] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, "Hey, you, get off of my market: Detecting malicious apps in official and alternative Android markets", *in Proc. 19th Annual Network and Distributed System Security Symposium*, San Diego, USA, 2012.

[32] Yajin Zhou, Xuxian Jiang , "Dissecting Android Malware: Characterization and Evolution", *Proceedings of the 33rd IEEE Symposium on Security and Privacy (Oakland 2012)*, pp.95-109, San Francisco, CA, May 2012.

[33] Zhaohui Wang, Ryan Johnson, Rahul Murmuria, and Angelos Stavrou, "Exposing Security Risks for Commercial Mobile Devices", *LNCS Computer Network Security*, vol.7531, pp.3-21, 2012.

[34] Guanglei Liu and Chuanyi Ji, "Resilience of All-Optical Network Architectures under In-Band Crosstalk Attacks: A Graphical Model Approach", *IEEE Journal on Selected Areas in Communications*, vol.25, no.4, pp.1-16, Aprl. 2007.

Shancang Li received the B.Sc. and M.Sc. degrees in mechanics engineering and the Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 2001, 2004, and 2008, respectively.

He is currently a lecturer with school of computing and a member of the Institute for Informatics and Digital Innovation, Edinburgh Napier University, Edinburgh, Scotland, UK. His current research interests include digital forensics for emerging technologies, network security, cyber attacks, wireless sensor networks, the Internet of Things, and the lightweight cryptography in resource constrained devices. He is a member of the British Computer Society.

Theo Tryfonas received the B.Sc. degree in computer science from the University of Crete, Rethymno, Greece, in 1996, and the M.Sc. degree in information systems and the Ph.D. degree in informatics from the Athens University of Economics, Athens, Greece, in 1998 and 2003, respectively.

He is an expert in cybersecurity and systems engineering with research work focused on assurance and resilience of critical infrastructures including transportation, utilities, healthcare, and government. He researched in particular on systems for maritime safety and port security, public transport security, protection of Unmanned Aerial Vehicles, telecom revenue and system assurance, information security risk analysis as well as assisted in the investigation of computer crimes. His current research interests include modeling cyber-capability with system dynamics and applications of game theory to the analysis of cyber attacks.

Gordon Russell Gordon Russell was born in Scotland, UK, in 1969. He received his BSc in computing and electronics in 1990, and his PhD in computing in 1995, both from the University of Strathclyde, Scotland.

In 1995, he joined the computing department of Edinburgh Napier University as a Lecturer, and became a Senior Lecturer in 2008. As part of his role, he leads the computer systems and networking subject group, and is a member of the cyber security research group. His research specialisms include computer security and networking, with particular interests in cloud technology and computer forensics. Dr Russell is a member of the British Computer Society and the Institute of Engineering and Technology, as well as being a fellow of the Higher Education Academy.

Panagiotis Andriotis received the B.Sc. degree in Mathematics from the National Kapodistrian University of Athens, Greece, in 2006, and the M.Sc. (with distinction) and Ph.D. degrees in Computer Science from the University of Bristol, U.K., in 2011 and 2016, respectively.

He is currently a Research Associate with the University College London, U.K. At the past he was with the banking sector and also a Mathematics Teacher in Greece. His current research interests include digital forensics with a special focus on the Android OS, data mining, data hiding, steganalysis, social network analysis, and human aspects of information security, privacy, and trust.