Game Theory based Cyber-Insurance to Cover Potential Loss from Mobile Malware Exploitation

LI WANG and S. SITHARAMA IYENGAR, Florida International University, USA
AMITH K. BELMAN, Syracuse University, USA
PAWEŁ ŚNIATALA, Poznań University of Technology, Poland
VIR V. PHOHA, Syracuse University, USA
CHANGSHENG WAN, Southeast University, China

Potential for huge loss from malicious exploitation of software calls for development of principles of cyber-insurance. Estimating what to insure and for how much and what might be the premiums poses challenges because of the uncertainties, such as the timings of emergence and lethality of malicious apps, human propensity to favor ease by giving more privilege to downloaded apps over inconvenience of delay or functionality, the chance of infection determined by the lifestyle of the mobile device user, and the monetary value of the compromise of software, and so on. We provide a theoretical framework for cyber-insurance backed by game-theoretic formulation to calculate monetary value of risk and the insurance premiums associated with software compromise. By establishing the conditions for Nash equilibrium between strategies of an adversary and software we derive probabilities for risk, potential loss, gain to adversary from app categories, such as lifestyles, entertainment, education, and so on, and their prevalence ratios. Using simulations over a range of possibilities, and using publicly available malware statistics, we provide insights about the strategies that can be taken by the software and the adversary. We show the application of our framework on the most recent mobile malware data (2018 IST report—data for the year 2017) that consists of the top five Android malware apps: Malapp, Fakeinst, Premiumtext, Maldownloader, and Simple-locker and the resulting leaked phone number, location information, and installed app information. Uniqueness of our work stems from developing mathematical framework and providing insights of estimating cyber-insurance parameters through game-theoretic choice of strategies and by showing its efficacy on a recent real malicious app data. These insights will be of tremendous help to researchers and practitioners in the security community.

CCS Concepts: • Theory of computation → Network games; Algorithmic game theory; • Security and privacy → Malware and its mitigation; • Computing methodologies → Stochastic games; • Security and privacy → Economics of security and privacy; Software and application security; • Social and professional topics → Malware/spyware crime;

Additional Key Words and Phrases: Game theory, malicious apps, software apps, nash equilibrium, android malware, cyber insurance

This work was supported in part by the US Army Research office grant number “W911NF-19-1-0362” and the US NSF (No. SaTC1527795).

Authors’ addresses: L. Wang (corresponding author) and S. S. Iyengar, School of Computing & Information Sciences, Florida International University, Miami, FL, 33199, USA, emails: [lwang059, iyengar]@cs.fiu.edu; A. K. Belman and V. V. Phoha, Syracuse University, Department of Electrical Engineering and Computer Science, Syracuse, NY, 13244, USA, emails: [akamathb, vvphoha]@syr.edu; P. Śniatala, Institute of Computing Science, Poznań University of Technology, Poznań, Poland; email: pawel.sniatala@put.poznan.pl; C. Wan, School of Information Science and Engineering, Southeast University, Nanjing, Jiangsu, China; email: wan.changsheng@163.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2021 Association for Computing Machinery.
2576-5337/2021/04-ART9 $15.00
https://doi.org/10.1145/3409959

Digital Threats: Research and Practice, Vol. 2, No. 2, Article 9. Publication date: April 2021.
1 INTRODUCTION

The use of mobile devices for business has gained popularity in organizations ranging from a one-person shop to those employing thousands of employees. Because of the prevalence of malicious apps, these devices are vulnerable to compromise putting millions of dollars at risk. The severity and reach of this problem grows with the growing popularity of apps for business and non-business activities. To address such a risk and to prevent loss if a compromise occurs, we propose a rigorous mathematical framework based on game theory, risk analysis, and a cyber-insurance model. We also develop strategies to adjust permissions on apps and to calculate probabilities of potential loss and amounts of insurance needed should a loss occur, to guard against the loss. We show the applications of this model on real scenarios. We make changes to the Gordon Loeb model to adapt it to mobile malware scenario. The Gordon Loeb model is one of the most well accepted analytical models for the economics of cyber security. It has also been empirically tested under various settings and generalised by mathematicians Marc Lelarge and Yuliy Baryshnikov. Our adaptations to the model makes it applicable to the mobile app security landscape for individuals and small and large organizations.

The state of the cyber-insurance market is at a “nascent” stage. The world wide expenditure on cyber insurance in 2017 was around US$4 billion in 2017 and is estimated to double by 2020. Various sources also agree on the cost of data breaches and cybercrimes to be around US$3 trillion to US$6 trillion annually between 2015 to 2021.

Because computing systems are a gateway to security breaches for cyberspace and the resulting potential loss from breaches, our research builds on interactions between computing systems, in particular software applications, and attackers in a game-theoretic adversarial model (Section 4). Computing systems ranging from mobile phones, to desktops, to high-end servers run a host of software applications. These software applications share common computing resources, such as memory, CPU time, I/O devices, and so on. In the case of mobile devices, applications, commonly referred as apps have revolutionized the utility and functionality of these devices. To function effectively these apps use and take permission to use resources such as memory, location services, access to sensors, such as accelerometers, gyroscopes, touch sensors, and so on. However, a conflict exists between allowing the access that will make the apps run effectively but exposing the systems for malicious attacks against adversaries who can exploit the access to steal sensitive information or disrupt the functioning that may make the apps shut down or not run efficiently.

Resource requesting is a typical requirement for software installed in mobile phones. For example, to use email for communicating with customers, the software has to request for email password from the user. And to watch online films, the software has to request for video card from the operating system, and configure it.

By requesting more resources, the software may potentially gain advantage. However, the more resources the software requests, the more risks that it will misuse these resources. For example, the software may misconfigure the video card and damage it, leading to a loss. However, if the software requests more resources, then the adversary may have more chance to establish an efficient attack on the software. For example, if the software only requests video cards, then the adversary can just damage the video card. However, if the software requests both the video card and the email key, then the adversary will be able to not only damage the video card but also send junk mail to customers. At the same time, when attempting to attack the software, the adversary may be caught resulting in loss to the adversary. Therefore, it is desired to analyze the strategies of both the software and the adversary, to develop strategies that maximize the chances of gain or reduce chance of loss for themselves.
In the rest of the article, we use the term \textit{gain} to refer to any type of advantage depending on the context of the conflict. For example, gain may be financial, or gain may lead to competitive advantage of some sort, or gain may refer to numerical measure of success of a malicious activity of an adversary. A loss is the reverse of gain. If an adversary has a gain, then the software has a loss and vice versa. For ease of computation, we normalize gain and loss to be between 0 and 1 (for example, in Figure 2). However, for calculations of case study, we derive loss or gain in dollar figures.

Based on the choice of strategy we can calculate the potential loss (or gain in terms of probabilities (probability of misuse: $p_{mu}$, probability of successful attack: $p_s$ and probabilities of the adversary being caught: $p_c$) using the Gordon Loeb model. These probabilities may be obtained from recent statistics, such as the ISTR reports. The values of potential loss then feed into a cyber-insurance model to calculate cyber-insurance parameters, such as amount to be insured, premiums, and claims. For example, using the 2018 ISTR report, we project a potential loss of $5,400 misuse of resources such as Phone number, Location Information and App Information and $11,200 if attack is successful and for 50% insurance coverage total premium of $4,407 and claim of $2,700 (misuse of resources) and $5,600 if attacked successfully or a total $8,300 if both losses occur. These values provide enough protection considering the chance of the software being compromised.

1.1 Contributions

We make the following contributions in this article:

1. Ours is perhaps the first work to develop a rigorous theory linking risk insurance to vulnerabilities on mobile devices brought about by malicious software.

2. A mathematical framework that links game theory, risk analysis and cyber-insurance to develop strategies for permissions on software and risk premiums.

3. We analyze the resource-requesting problem in smart phone environment, in adversarial environment.

4. We formulate this conflict as a two-person zero sum game and derive Nash equilibrium for this game. We present theoretical analysis and do simulations with various parameter settings to evaluate our theoretical analysis.

5. One of the key contribution of the article is contextualizing resource sharing as a game theory problem and constructing case studies to analyze merits of mobile security.

6. We demonstrate the applications of this framework on real-world data.

7. Using most recent real data of five malicious apps for Android OS, and loss of common information, such as phone number, location information, and app privileges, we analyze the conflict as a two-person zero sum game and present insights that will be of help to security community.

The reminder of this article organized as follows. In Section 2, we survey related work of this field. The concepts, terminology and equations from literature, such as \textit{Security Games}, \textit{Nash equilibrium}, \textit{Gordon Loeb model}, and \textit{Insurance Premiums and Claims} are described in Section 3. In Section 4, we discuss the design goals, system model, and threat model of this problem. Followed by problem formulation and analysis in Sections 5 and 6, respectively. We provide experimental analysis in Section 7. In Section 8, we analyze the five most prevalent Android malicious apps given in the 2018 Symantech ISTR report \cite{33} to find parameters used in our game-theoretic model. In the first part of Section 9, we analyze these five apps from game-theoretic perspective based on our earlier derived theoretical results, we compute and present the financial values of threats and risks. The second part of Section 9 shows computations of insurance premiums and claims for losses that occur under different configurations. Finally, in Section 10 we list the significance of the work and draw our conclusions in Section 11.
Table 1. Comparison of Our Work with Related Work in Literature

|                  | Risk Modeling | Game Theoretic Agents in System | Insurance Formulation | Computation of Financial Value of Threats | Computation of Insurance Premium and Claims | Use Real-World Data | Salient Features and Differences |
|------------------|---------------|---------------------------------|-----------------------|------------------------------------------|---------------------------------------------|---------------------|----------------------------------|
| Lu et al. [19]   | ✓             | x                               | ✓                     | x                                        | x                                           | x                   | Focus on Heterogeneous -Wireless-Networks in event of service outage |
| Meland and Seehusen [21] | ✓             | x                               | ✓                     | ✓                                        | ✓                                           | x                   | Method to help cyber-insurance decisions |
| Khalili et al. [12] | ✓             | ✓                               | ✓                     | x                                        | x                                           | x                   | Method to help insurer pre-screen clients to increase profits |
| Lepoint et al. [17] | x             | x                               | ✓                     | x                                        | x                                           | x                   | Propose a Hyperledger Composer based framework |
| Pal and Hui [28]   | x             | ✓                               | x                     | x                                        | x                                           | x                   | Method to differentiate contracts using Bonachich/eigenvector centrality |
| Vakilinia and Sengupta [36] | ✓             | x                               | ✓                     | x                                        | x                                           | x                   | Different frameworks for Common Platform Insurance |
| Uuganbayar et al. [35] | x             | x                               | ✓                     | x                                        | x                                           | x                   | Analysis on Security Interdependence |
| Meland et al. [22] | ✓             | x                               | ✓                     | x                                        | x                                           | x                   | Assessment work on Cyberinsurance |
| Our Work          | ✓             | ✓                               | ✓                     | ✓                                        | ✓                                           | ✓                   | Additional differences other than the one marked in columns on the left follow. (a) [12, 36] use game-theoretic approach to model agents in system. Whereas, we formulate it as adversary vs. software game. (b) [21] compute value of threats premium and claim with fictitious assumed data. We use real-world ISTR [33] data. |

2 RELATED WORK

Table 1 summarizes the related work on cyber-insurance. As an efficient tool, game theory has been widely used for protecting the security of software. A majority of the works carried out in literature focus on defining new frameworks for cyber-insurance and fall short of analysis with real-world data.

Though the work by Meland and Seehusen [21] present financial value computations of risks and premiums, use of non-real data from an assumed fictive company and the directions in which these methods are applied are unconnected and simplistic. Khalili et al. [12] take a game-theoretic approach toward cyber-insurance, but the agents defined in their game are two potential clients for cyber-insurance with an interdependence of security. They present a method to maximize profits for the insurer and analyze and transfer risks for the clients. Other
works mentioned in Table 1 focus on various aspects of cyber-insurance like formulation of insurance \cite{35, 36} and proposing a framework. However, there is no work focusing on the strategies when the software wants to request more resources from the operating system, the threats in this scenario and the corresponding insurance framework. Given the nascent stage of the field, only few works are in the same direction of ours, we summarize them below.

Our literature search did not yield specific applications of game theory to mobile software resource allocation in adversarial environment. However, there are works in software security and game theory models in related field that give a basis of comparison and help us build our work. In this section, we first investigate existing software security techniques, and then investigate game models used in related fields.

Investigation of software security techniques: There are a variety of techniques that can be used for protecting software security. For example, the first technique is called dynamic information flow tracking (DIFT) \cite{7, 8, 10, 15}, which can be used for attack detection. In this technique, untrusted information is labeled. During data processing, the DIFT technique will trace this labeled data to find its movement in the system. Once these labeled data are used for executing sensitive commands such as system call, an alert will be generated to warn users.

Sandbox is another technique for protecting software \cite{18, 26, 32, 34, 37}, which can be used for blocking communication between software and the operating system. In this technique, when the software wants to write or generate files, its operation is redirected to a directory managed by the sandbox. By doing so, all the operations of the software are managed by the sandbox and will not affect the operating system.

Software watermarking is the third technique for protecting software from being pirated \cite{11, 30, 31, 39, 40}. In this technique, authentication and copyright protection information is inserted into the software for preventing an unauthorized user from executing the software.

However, these techniques mainly focused on the detection of adversary and protection of software. And there is no technique for discussing the strategy of requesting resources.

Investigation of game models and insurance in related fields: Early in 2002, Neel et al. \cite{24} used game theory for analyzing the role of entities in software radio network. Then, in 2006, Oza et al. \cite{27} modeled the software outsourcing process using game theory. Gao et al. \cite{9} used game theory for configuring detection software. Savani et al. \cite{29} developed a software for helping users playing online games, which was based on game theory. And Mattos et al. \cite{20} analyzed the cooperation in software development using game theory. Related research works that combine some of the concepts used in our work are summarized in Table 1.

Observing this, we aim to (a) analyze the strategies of software when applying for more resources from the operating system, (b) to model the threats and risks as a game, (c) provide a framework for insurance premium and claim computation, and (d) present analysis with real-world data.

Only those works that were similar in research direction as ours are compared and presented in Table 1. Khalili et al. \cite{12} and Pal and Hui \cite{28} used forms of game-theoretic agents in their work, but both of these works have a different focus from our research. Our work addresses cyber-insurance and mobile malware exploitation, whereas Khalili et al. \cite{12} focus on a game-theoretic model to help insurer pre-screen clients increase profits, and Pal and Hui \cite{28} focus on a method to differentiate contracts using Bonachich/eigenvector centrality. Therefore, these two works are not compatible to be analytically compared at a framework level by using the data form IST report that we used for our work. The work by Meland and Seehusen \cite{21} does not propose a game-theoretic approach; however, this work focuses on computation of insurance values for threats and computation of insurance claims. Even though this commonality exists, we cannot analytically compare our work at a framework level, as the work from Meland and Seehusen \cite{21} focuses toward summarizing threat and loss data from multiple sources like Advisen and Fidelity and categorizing them by industrial sectors. The lack of game-theoretic model, common variables, or other tangential points makes it unsuitable to be analytically compared at a framework level in Section 9.
Table 2. Nomenclature of All the Symbols Used in the Following Sections

| Symbol | Description | Used in sections |
|--------|-------------|------------------|
| $V$    | set of all strategies available to a player | 3.2, 4.1, 5.6, 7, 8, 9 and 10 |
| profile | set of sets containing strategy for each player | |
| $R$    | strategy of a software to request more resource | |
| $NR$   | strategy of a software to not request more resources | |
| $NA$   | strategy of an adversary to not attack the software | |
| $p_{mu}$ | probability of misusing a resource | |
| $l_{mu}$ | loss for the software due to misusing a resource | 7, 8, 9 and 10 |
| $e_{uc}$ | gain for the software due to correct usage of a resource | |
| $l_a$  | loss for the software due to a successful attack | |
| $p_c$  | probability of an attack being caught | |
| $l_c$  | loss for the adversary for being caught | |
| $p_s$  | probability of a successful attack | |
| $e_a$  | gain for the adversary due to successful attack | |
| $I_{soft}$ | Gain for the software | |
| $I_{ad}$ | Gain for the adversary | |
| $A$    | strategy of an adversary to attack the software | |
| $v$    | probability of a breach occurring | 3.3 and 9.1 |
| $L$    | potential loss | |
| $z$    | investment in security measure | |
| $t_c$  | threat probability of the category of the app | |
| $ex$   | exposure to network threats | |
| $\varrho$ | probability of applying a security measure | |
| $\rho_t$ | the insurance premium to be paid in the year $t$ | 3.4 and 9.3 |
| $\zeta_t$ | the insurance claims in the year $t$ | |
| $\beta_t$ | all other costs excluding $\rho_t + \zeta_t$ | |
| $i_t$  | interest rate for the year $t$ | |
| $\tau_t$ | equity for the year $t$ | |

3 BACKGROUN
In this section, we briefly describe various concepts from Security Games, Game Theory, Nash equilibrium, Gordon Loeb model, and Insurance that are key to discussions throughout the article.

3.1 Nomenclature
Symbols and variables used in discussing the concepts of Nash equilibrium, Gordon–Loeb model, Cyber Insurance, and claims are briefly described in the nomenclature in Table 2.

3.2 Security Games
In game theory, based on the type of player interactions and assumptions, we come across various game classifications. In the security literature [13], we mainly come across the following; two-player zero-sum game, perfect
information game, complete information game, cooperative game, static game, sequential game, repeated game, and stochastic game. As these game models are very similar to each other in the context of security, they can be classified as Security Games. As defined by Alpcan and Basar [2], security games “study the interaction between malicious attackers and defenders.” Majority of these security game models are non-cooperative two-player models focusing on an attacker’s interactions with a defending system.

Although we make the same assumptions in our work, where we deal with a non-cooperative two-player game, we do not model the players as attacker and defender, we look at a more complex scenario of resource allocation and utilization for software versus adversarial exploitation. We also observe that game-theoretic models adopt a very high level of abstraction when defining Intrusion Detection System models with very few parameters.

3.3 Nash Equilibrium

The Nash equilibrium is a proposed solution of a non-cooperative game with two or more players where each player is assumed to know the equilibrium strategies of the other players, and no player has an incentive to change only their chosen strategy. For every finite game, Nash [23] showed that there exists a Nash equilibrium. A two player game (for example, software vs. adversary) is said to be in Nash equilibrium if both software and adversary make their best decisions while taking into account the other player’s decision, which remains unchanged.

**Strategy Space and Profile:** Strategy space is the set of all strategies available to a player. It can be represented as $V_i$ for player $i$. Strategy profile is a set of sets, with each consisting of a strategy for every player involved in the game. It can be represented as $profile = \{c_1 = (V^1, V^2, \ldots, V^n), \ldots, c_m = (V^1, V^2, \ldots, V^n)\}$, where $n$ is the number of players and $m$ is the number of game conditions.

**Pure and Mixed strategies:** If each player in the game plays one specific strategy, then it is called a pure-strategy equilibrium, whereas a mixed-strategy equilibrium is where at least one player in the game randomizes over some or all of their pure strategies. The randomization is brought by applying a probability distribution over choosing pure strategies. Pure strategies are restricted forms of mixed strategies where the probability of the chosen strategy is 1.

3.4 Gordon–Loeb Model

The Gordon–Loeb model [1] is a mathematical-economic model used to analyze the optimal investment level in information security. We modify the Gordon–Loeb model to better suit our analysis of the gains and losses for the software and adversary. The assumptions made by Gordon–Loeb model can be summarized as follows:

(a) Information sets are vulnerable to attacks. The vulnerability is represented by a probability, that a breach occurs in the current conditions, denoted by $v$ ($0 \leq v \leq 1$).

(b) The potential loss, denoted as $L$, is the value of the information in case of a successful attack and can be expressed as a monetary value. $vL$ is the expected loss before investing in more security.

(c) $v$ is reduced by investment in security measures, denoted as $z$. Function $s(z, v)$, provides a revised (possibly reduced) measure of vulnerability.

(d) The function $s(z, v)$ is twice continuous differentiable and strictly convex and reduction in vulnerability increases at a decreasing rate with investment in security.

(e) Investments in security can reduce the probability of breach close to zero, but not zero and therefore some vulnerability always exists.

Two equations from the Gordon–Loeb model [1] are of interest to us:

(a) The *Expected benefits of an investment in information security*, denoted by $EBIS(z) = [v - S(z, v)]L$. This is the reduction in loss due to investment in security.
(b) The Expected net benefits of an investment in information security, denoted by \( \text{ENBIS}(z) = [v - S(z, v)]L - z \). This is the net reduction in loss due to investment in security, accounting for the cost of investment as well.

3.5 Cyber-Insurance and Claims

The concepts and the issues at the intersection of cyber security and insurance have been studied by very few researchers and as a result literature in this particular overlap of concepts is rather scarce. Therefore, we study various insurance premium calculation methods presented by Bühlmann [5, 6] and Wang [38] and adapt them to a scenario with software entity under risk of adversarial attacks.

**Insurance Premium:** In a system with multiple risks of varying degree and the assumption that Premiums, Insurance Claims, and Costs are paid on a yearly basis, we arrive at the insurance premiums from Bühlmann’s [6] work as summarized below:

- Total premium according to **standard deviation principle**
  \[
  \rho_{\text{total}} = E[\zeta] + \sqrt{2i \ln \psi_0 \sigma[\zeta]}. \]

- The **variance principle** is then used to split the premium to individual risks
  \[
  \rho_{\text{ind}} = E[\zeta_{\text{ind}}] + \frac{1}{\tau_0} \ln \psi_0 \text{Var}[\zeta_{\text{ind}}] \]
  with
  \[
  \tau_0 = \sqrt{\frac{\ln \psi_0}{2i}} \sigma[\zeta], \]

where \( \psi_0 \) (generally equal to 1%) is used to select the level at which the probability of ruin is to be controlled by setting \( k = \frac{1}{\tau_0} \ln \psi_0 \). According to a standard formula in risk theory, probability of ruin is at most equal to \( e^{-k\tau_0} \) where \( k > 0 \) such that \( e^{k(r-\beta)} = E[e^{k\zeta}] \).

**Insurance Claims:** In the event of a loss of insured assets, a standard formula to compute the insurance claim can be used. In the event of a loss to the tune of \( L \), the claim \( \zeta_i \) is given by

\[
\zeta_i = \frac{\text{Actual insurance amount}}{\text{Required insurance amount}} \times L. \]

4 DESIGN GOALS, SYSTEM MODEL, AND THREAT MODEL

In this section, we analyze the design goals, the system model, and the threat model.

4.1 Design Goals

As a whole, it should be guaranteed that both the software and the adversary want to increase their gains and reduce their losses. Therefore, the design goal is to point out all possible choices, compute gains and losses of the software and adversary, and then provide strategies for both the software and the adversary.

Specifically, the following two requirements should be satisfied.

- For the software, when selecting a strategy, its gain due to using resources should be more than its losses due to misuse and attack.
- For the adversary, its gains due to a successful attack should be more than its loss due to being caught.

We derive the payoffs for the adversary and the software using the statistics and the probabilities obtained from the ISTR 2018 report, however we use Gordon–Loeb model to project the potential loss or gain (Section 8).
Fig. 1. System model. The conflict arises because of balancing act between allowing more privileges and resources for user speed and convenience but at the same time more privileges and resources allows possibilities of attack. The software can choose between Requesting (R) or Not Requesting (NR) a resource, while the adversary can choose to Attack (A) or Not Attack (NA). The green regions depict increase in ease of use and security, whereas the red regions depict increase in risk and limited functionality. The probabilities of various events affect the choice of strategies. Key probabilities of misusing a resource \( P_{mu} \), of successful attack \( P_s \) and of attack being caught \( P_c \) are shown on the side of the players whose strategies are are affected by them. The risks and liabilities are then considered into the Risk Management Framework that we adopted from the International Association of Insurance Supervisors standard Enterprise Risk Management framework \[25\]. The probabilities \( P_{mu}, P_s, \) and \( P_c \) flow into the risk management framework that determines the insurance and financial risk associated with the software.

4.2 System Model

The system model is shown in Figure 1. There are three components: (1) a game-theoretic model to calculate the payoffs; (2) adaptation of the Gordon–Loeb model to calculate potential loss or gain, and (3) the Cyber-Insurance model. In Figure 1 we combine (1) and (2) under the heading of Software Versus Adversary Game model.

There are two entities (the software and the adversary) involved in two phases (the initialization phase and the requesting phase). Initialization phase: The software has already owned a set of resources, such as hardware, private files, and email key. Using these resources, the software can establish primary transactions, such as selling goods and other online services or communicating with customers. Through these actions, the software can gain advantage. For example, the software can send its private files including secret data such as keys for accessing online services to the customer, and the customer may send back bank information for paying for the service. At the same time, the software may misuse these resources, thus suffering loss. For example, the software may send the secret data to a customer who has not paid for the service. However, the adversary may establish an attack on the software. If it succeeds, then the software may suffer losses, and the adversary may gain advantage. The adversary may be caught, resulting in loss. For example, the adversary may get the secret data by attacking the software and access the online service without paying anything. But the software may detect the adversary and catch it. In this case, the adversary may suffer losses.
Requesting phase: The software wants to request more resources. For example, it may request the root key of the operating system, so that it can manage system configurations. By requesting more resources, the software may gain advantage. However, once it misuses the root key, it may suffer losses. However, the adversary may have better chance to attack the software and get the root key. If it succeeds, then the adversary may modify the system configuration information using the root key to gain advantage, and the software suffers a loss. However, the adversary may be caught leading to a loss for itself.

4.3 Threat Model
We make the following assumptions in the adversarial setup. Common to both the software and malicious adversary is the assumption that they are non-altruistic and each of them wants to maximize their payoffs.

- Software and adversary do not know about each others strategy.
- Software and adversary can calculate the payoffs using publicly available malicious apps figures (for example, from the ISTR report the way we have used in this article). For example, probability of attack, percentage of malicious apps in different usage categories, such as lifestyle, entertainment, books and reference, music and audio, and so on.
- Both the adversary and the software use the preceding year’s publicly available statistics to project next year’s figures to calculate risk.
- In the above scenario, both the software and the adversary want to find their best choices, so that they can increase their gains, and reduce their losses. Moreover, if the adversary gains advantage, then the software may suffer losses and if the software gains advantage, then the adversary may suffer losses. This is a typical two-player game.

In Section 5, we shall formulate this problem using game theory and solve it.

5 PROBLEM FORMULATION
In this section, we formulate the problem described in Section 4 in two parts. In part I, we present the game overview, which defines the players and the rules they play in the game. In part II, we present the game construction, which further defines the parameters in this game, and constructs the overall game.

5.1 Part I: Game Overview
In this subsection, we first define the players and assumptions. Then, we define their strategies. Finally, we give the strategy space.

(1) Players and assumptions. There are two players in our game, namely the software and the adversary. To formulate this game, we assume that both the software and the adversary are rational and know the strategies and parameters used in this game. That is, both players aim to select their best strategies, so that they can reduce their losses and improve their gains.

(2) Strategies. The software has two strategies, namely, to request more resources \((R)\) or not to request more resources \((NR)\). The software can only select one of these two strategies at a time. There are various resources to be requested by the software, such as getting authorization to access system hardware or data, and getting secret keys for protecting communications. If the software misuses these resources, then the adversary may succeed in attacking it, resulting in loss for the software. However, if the software uses these resources correctly, then it may gain advantage from the use of resources.

The adversary has two strategies, namely to attack \((A)\) the software or not to attack it \((NA)\). The adversary can only select one of these two strategies at a time. There are various attacks on the software, such as chosen-ciphertext attack and brute-force attack. If the adversary succeeds in the attack, then it may gain advantage using
these resources, resulting in loss for the software. However, if the adversary fails in the attack, then it may be caught, resulting in loss for the adversary.

(3) Strategy Space. Finally, we get the strategy profile of this game, which includes four conditions: profile = \{c_1 = (R, A), c_2 = (R, NA), c_3 = (NR, A), c_4 = (NR, NA)\}.

5.2 Part II: Game Construction

To further construct the game described under Section 5.1, we need to define the parameters in the strategy profile (profile), and compute the gains and losses of both the software and the adversary.

(1) Parameters. For the software, there are four parameters as described below.

—The resource-misusing probability. This probability is denoted by \( p_{mu} \). If the resource is used correctly, then the software can gain advantage. Otherwise, the software will suffer losses.
—The initial gain of the software is \( E(1|x) \), and the adversary does not establish an attack. In this case, we can compute the gains and losses as follows. The expected gain of the software is \( E(e_{uc}|c_1) = e_{uc} + (1 - p_{mu})e_{uc} \). The expected loss of misusing is \( E(l_{mu}|c_1) = l_{mu} + p_{mu}l_{mu} \). The expected loss of the software due to attack is \( E(l_a|c_1) = p_a l_a \). The expected gain from a successful attack is \( E(e_a|c_1) = p_a e_a \).

For the adversary, there are four parameters as described below.

—The probability of establishing a successful attack. This probability is denoted by \( p_a \). Only when the adversary establishes a successful attack, will it gain advantage.
—The loss if attacked successfully. This loss is denoted by \( l_a \). Only when the software is attacked successfully, will it suffer a loss.

(2) Gains and losses in different strategies. Before analyzing the gains and losses, we assume that the software has already owned a set of resources, and the initial gains and losses are described below. The initial gain of the software is denoted as \( x_{uc} \). The initial loss of the software due to misuse is denoted as \( l_{mu} \). Then, the software considers requesting a new resource \( r \) or not, and the adversary considers establishing an attack or not, using the strategies defined in Section 5.1.

There are totally four strategies in the profile. We shall analyze the gains and losses of both the software and the adversary while using these strategies, which is shown below. Note that \( E(x|c_y) \) means the expected value of \( x \) under the condition \( y \), where \( y \in \{c_1, c_2, c_3, c_4\} \).

The \( c_1 = (R, A) \) strategy:

For this strategy, the software requests the new resource \( r \), and the adversary establishes an attack. In this case, we can compute the gains and losses as follows. The expected gain of the software is \( E(e_{uc}|c_1) = e_{uc} + (1 - p_{mu})e_{uc} \). The expected loss of misusing is \( E(l_{mu}|c_1) = l_{mu} + p_{mu}l_{mu} \). The expected loss of the software due to attack is \( E(l_a|c_1) = p_a l_a \). The expected gain from a successful attack is \( E(e_a|c_1) = p_a e_a \).

The \( c_2 = (R, NA) \) strategy:

For this strategy, the software requests the new resource \( r \), and the adversary does not establish an attack. In this case, the expected gain of the software is \( E(e_{uc}|c_2) = e_{uc} + (1 - p_{mu})e_{uc} \). The expected loss of misusing is \( E(l_{mu}|c_2) = l_{mu} + p_{mu}l_{mu} \). The expected loss of the software due to attack is \( E(l_a|c_2) = 0 \), since the adversary
The expected gain from a successful attack is 

\( E(\delta_{\text{ec}} | \delta_{\text{c}}) = e_{\text{ec}} - \mu_{\text{mu}} - \mu_{\text{mu}} - 0.0 \)

\( e_{\text{ec}} - \mu_{\text{mu}} - \mu_{\text{mu}} - 0.0 \)

Table 3. Game Construction

|  | A | NA |
|---|---|---|
| R | \( e_{\text{ec}}(1 - p_{\text{mu}})e_{\text{ac}}l_{\mu 0}p_{\text{mu}}l_{\mu}p_{\text{l}a}p_{\text{c}}l_{\text{c}}p_{\text{e}}e_{\text{a}} \) | \( e_{\text{ec}}(1 - p_{\text{mu}})e_{\text{ac}}l_{\mu 0}p_{\text{mu}}l_{\mu}0.0 \) |
| NR | \( e_{\text{ec}}(1 - p_{\text{mu}})e_{\text{ac}}l_{\mu 0}p_{\text{l}a}p_{\text{c}}l_{\text{c}}p_{\text{e}}e_{\text{a}} \) | \( e_{\text{ec}}l_{\mu 0}0.0 \) |

Here each strategy is composed of a tuple of five elements \( \{ e_{\text{ec}} | e_{\text{ac}} | l_{\mu 0} | p_{\text{mu}} | l_{\mu} \} \) determined by the intersecting rows (R, NR) and columns (A, NA).

Table 4. Gains of the Software and the Adversary

|  | A | NA |
|---|---|---|
| R | \( e_{\text{ec}}(1 - p_{\text{mu}})e_{\text{ac}}l_{\mu 0}p_{\text{mu}}l_{\mu}p_{\text{l}a}p_{\text{c}}l_{\text{c}}p_{\text{e}}e_{\text{a}} \) | \( e_{\text{ec}}(1 - p_{\text{mu}})e_{\text{ac}}l_{\mu 0}p_{\text{mu}}l_{\mu}0.0 \) |
| NR | \( e_{\text{ec}}l_{\mu 0}p_{\text{l}a}p_{\text{c}}l_{\text{c}}p_{\text{e}}e_{\text{a}} \) | \( e_{\text{ec}}l_{\mu 0}0.0 \) |

Here each strategy is comprised of a pair \( \{ E(l_{\text{soft}} | c_{y}) | E(l_{\text{ad}} | c_{y}) \} \), where \( c_{y} \in \{ c_{1}, c_{2}, c_{3}, c_{4} \} \) determined by the intersecting rows (R, NR) and columns (A, NA).

does not establish an attack. The expected loss of the adversary due to being caught is \( E(l_{c} | c_{2}) = 0 \). And the expected gain from a successful attack is \( E(e_{\text{a}} | e_{2}) = 0 \).

The \( c_{3} = (\text{NR, A}) \) strategy:

For this strategy, the software does not request the new resource \( r \), and the adversary establishes an attack. In this case, the expected gain of the software is \( E(e_{\text{ac}} | c_{3}) = e_{\text{ac}} \). The expected loss of misusing is \( E(l_{\mu} | c_{3}) = l_{\mu 0} \). The expected loss of the software due to attack is \( E(l_{a} | c_{3}) = p_{a}l_{a} \). The expected loss of the adversary due to being caught is \( E(l_{c} | c_{3}) = p_{c}l_{c} \). And the expected gain from a successful attack is \( E(e_{\text{a}} | c_{3}) = p_{a}e_{a} \).

The \( c_{4} = (\text{NR, NA}) \) strategy:

For this strategy, the software does not request the new resource \( r \), and the adversary does not establish an attack. In this case, the expected earning of the software is \( E(e_{\text{ac}} | c_{4}) = e_{\text{ac}} \). The expected loss of misusing is \( E(l_{\mu} | c_{4}) = l_{\mu 0} \). The expected loss of the software due to attack is \( E(l_{a} | c_{4}) = 0 \), since the adversary does not establish an attack. The expected loss of the adversary due to being caught is \( E(l_{c} | c_{4}) = 0 \). And the expected gain for the adversary \( E(e_{\text{a}} | c_{4}) = 0 \).

From the above discussion, we can construct the overall game, as shown in Table 3.

6 PROBLEM ANALYSIS

In this section, we shall analyze the game defined in Section 5 in the literature of Nash equilibrium. Before analyzing the game, we define the expected gain of the software as \( E(l_{\text{soft}} | c_{y}) = E(e_{\text{ac}} - l_{\mu} - l_{a} | c_{y}) \), and the expected gain of the adversary as \( E(l_{\text{ad}} | c_{y}) = E(e_{\text{a}} - l_{c} | c_{y}) \), where \( c_{y} \in \{ c_{1}, c_{2}, c_{3}, c_{4} \} \). From Table 3, we can compute \( E(l_{\text{soft}} | c_{y}) \) and \( E(l_{\text{ad}} | c_{y}) \). The results are shown in Table 4.

Obviously, both the software and the adversary want to increase their gain. For the adversary, only the gains from establishing an attack is positive will it prefer to establish an attack. That is, \( p_{a}e_{a} - p_{c}l_{c} > 0 \Rightarrow p_{e_{a}} > p_{c}l_{c} \). For the software, only when the gain from requesting more resources is higher than that of not requesting more resources will it prefer to request more resources. That is, \( (1 - p_{\text{mu}})e_{\text{ac}} - p_{\text{mu}}l_{\mu} > 0 \Rightarrow e_{\text{ac}} > p_{\text{mu}}l_{\mu} / (1 - p_{\text{mu}}) \).

In the following paragraphs, we shall further list several potential conditions and analyze the best choices and the Nash equilibrium of this game.

6.1 Proofs of Nash Equilibrium under Different Conditions

The proofs for mixed and pure strategies under different conditions for \( e_{\text{a}}, e_{\text{ac}}, p_{c}, p_{s}, p_{\text{mu}}, l_{a}, l_{c}, \) and \( l_{\mu} \) for the strategies \( c_{1}, c_{2}, c_{3}, \) and \( c_{4} \) are as follows:
Theorem 1. If \( e_a > p_c l_c / p_s \) and \( e_{uc} > p_{mu} l_{mu} / (1 - p_{mu}) \), then the game in Table 4 admits a pure strategy Nash equilibrium profile \((c_1 = (R, A))\).

Proof. If \( e_a > p_c l_c / p_s \), then the gain of the adversary for establishing an attack is always positive, and the adversary prefers to attack the software. If \( e_{uc} > p_{mu} l_{mu} / (1 - p_{mu}) \), then the gain of the software for requesting more resources is always higher than not requesting, and the software prefers to request more resources. That is, \( c_1 = (R, A) \) is the best choice for the adversary and the software.

Theorem 2. If \( e_a > p_c l_c / p_s \) and \( e_{uc} = p_{mu} l_{mu} / (1 - p_{mu}) \), then the game in Table 4 admits a mixed strategy Nash equilibrium profile \((c_1 = (R, A))\).

Proof. If \( e_a > p_c l_c / p_s \), then the gain of the adversary for establishing an attack is always positive, and the adversary prefers to attack the software. If \( e_{uc} = p_{mu} l_{mu} / (1 - p_{mu}) \), then the gain of the software for requesting more resources is equal to that of not requesting, and the software has no preference. That is, the mixed strategy \( c_1 \) and \( c_3 \) is the best choice for the adversary and the software.

Theorem 3. If \( e_a > p_c l_c / p_s \) and \( e_{uc} < p_{mu} l_{mu} / (1 - p_{mu}) \), then the game in Table 4 admits a pure strategy Nash equilibrium profile \((c_3 = (NR, A))\).

Proof. If \( e_a > p_c l_c / p_s \), then the gain of the adversary for establishing an attack is always positive, and the adversary prefers to attack the software. If \( e_{uc} < p_{mu} l_{mu} / (1 - p_{mu}) \), then the gain of the software for requesting more resources is always lower than not requesting, and the software prefers to not request more resources. That is, \( c_3 \) is the best choice for the adversary and the software.

Theorem 4. If \( e_a = p_c l_c / p_s \) and \( e_{uc} > p_{mu} l_{mu} / (1 - p_{mu}) \), then the game in Table 4 admits a mixed strategy Nash equilibrium profile \((c_1 = (R, A)) \) and \((c_2 = (R, NA))\).

Proof. If \( e_a = p_c l_c / p_s \), then the gain of the adversary for establishing an attack is always zero, and the adversary has no preference. If \( e_{uc} > p_{mu} l_{mu} / (1 - p_{mu}) \), then the gain of the software for requesting more resources is always higher than not requesting, and the software prefers to request more resources. That is, the mixed strategy \((c_1, c_2)\) is the best choice for the adversary and the software.

Theorem 5. If \( e_a = p_c l_c / p_s \) and \( e_{uc} = p_{mu} l_{mu} / (1 - p_{mu}) \), then the game in Table 4 admits a mixed strategy Nash equilibrium profile \((c_1 = (R, A)) \), \((c_2 = (R, NA))\), \((c_3 = (NR, A))\), and \((c_4 = (NR, NA))\).

Proof. If \( e_a = p_c l_c / p_s \), then the gain of the adversary for establishing an attack is always zero, and the adversary has no preference. If \( e_{uc} = p_{mu} l_{mu} / (1 - p_{mu}) \), then the gain of the software for requesting more resources is always equal to that of not requesting, and the software has no preference, either. That is, the mixed strategy \((c_1, c_2, c_3, c_4)\) is the best choice for the adversary and the software.

Theorem 6. If \( e_a = p_c l_c / p_s \) and \( e_{uc} < p_{mu} l_{mu} / (1 - p_{mu}) \), then the game in Table 4 admits a mixed strategy Nash equilibrium profile \((c_3 = (NR, A)) \) and \((c_4 = (NR, NA))\).

Proof. If \( e_a = p_c l_c / p_s \), then the gain of the adversary for establishing an attack is always zero, and the adversary has no preference. If \( e_{uc} < p_{mu} l_{mu} / (1 - p_{mu}) \), then the gain of the software for requesting more resources is always lower than not requesting, and the software prefers to not request more resources. That is, the mixed strategy \((c_3, c_4)\) is the best choice for the adversary and the software.

Theorem 7. If \( e_a < p_c l_c / p_s \) and \( e_{uc} > p_{mu} l_{mu} / (1 - p_{mu}) \), then the game in Table 4 admits a pure strategy Nash equilibrium profile \((c_2 = (R, NA))\).

Proof. If \( e_a < p_c l_c / p_s \), then the gain of the adversary for establishing an attack is always negative, and the adversary prefers to not attack the software. If \( e_{uc} > p_{mu} l_{mu} / (1 - p_{mu}) \), then the gain of the software for requesting more resources is always lower than not requesting, and the software prefers to not request more resources. That is, the mixed strategy \((c_2)\) is the best choice for the adversary and the software.
requesting more resources is always higher than not requesting, and the software prefers to request more resources. That is, \( c_2 \) is the best choice for the adversary and the software.

**Theorem 8.** If \( e_a < p_c l_c / p_s \) and \( e_{uc} = p_{mu} l_{mu} / (1 - p_{mu}) \), then the game in Table 4 admits a mixed strategy Nash equilibrium profile \((c_2, c_4) = (R, NA)\) and \( c_4 = (NR, NA)\).

**Proof.** If \( e_a < p_c l_c / p_s \), then the gain of the adversary for establishing an attack is always negative, and the adversary prefers to not attack the software. If \( e_{uc} = p_{mu} l_{mu} / (1 - p_{mu}) \), then the gain of the software for requesting more resources is always equal to that of not requesting, and the software has no preference. That is, the mixed strategy \((c_2, c_4)\) is the best choice for the adversary and the software.

**Theorem 9.** If \( e_a < p_c l_c / p_s \) and \( e_{uc} < p_{mu} l_{mu} / (1 - p_{mu}) \), then the game in Table 4 admits a pure strategy Nash equilibrium profile \((c_4 = (NR, NA))\).

**Proof.** If \( e_a < p_c l_c / p_s \), then the gain of the adversary for establishing an attack is always negative, and the adversary prefers to not attack the software. If \( e_{uc} < p_{mu} l_{mu} / (1 - p_{mu}) \), then the gain of the software for requesting more resources is always lower than not requesting, and the software prefers to not request more resources. That is, \( c_4 \) is the best choice for the adversary and the software.

Combining the above results, we can have the following conclusions.

1. The software and the adversary may select the \( c_1 \) strategy in the following condition: \( e_a \geq p_c l_c / p_s \) and \( e_{uc} \geq p_{mu} l_{mu} / (1 - p_{mu}) \).
2. The software and the adversary may select the \( c_2 \) strategy in the following condition: \( e_a \leq p_c l_c / p_s \) and \( e_{uc} \geq p_{mu} l_{mu} / (1 - p_{mu}) \).
3. The software and the adversary may select the \( c_3 \) strategy in the following condition: \( e_a \geq p_c l_c / p_s \) and \( e_{uc} \leq p_{mu} l_{mu} / (1 - p_{mu}) \).
4. The software and the adversary may select the \( c_4 \) strategy in the following conditions: \( e_a \leq p_c l_c / p_s \) and \( e_{uc} \leq p_{mu} l_{mu} / (1 - p_{mu}) \).

7 EXPERIMENTAL RESULTS

In Section 6, we have analyzed all the conditions and pointed out the best strategies for both the software and the adversary. In this section, we shall provide experimental results for this analysis.

Since the probabilities in our game \((p_{mu}, p_s, p_c)\) are in the range \([0, 1]\), we shall analyze the maximum and minimum gains of both the software and the adversary in all the four strategies listed in Table 4.

**Maximum gain of the software:** When \( p_{mu} = 0 \) and \( p_s = 0 \), the software will have the maximum gain, and it should request more resources. The maximum gain is \( e_{uc0} + e_{uc} \). This is the ideal condition for the software. In this case, the gain of the adversary is \(-p_c l_c\). That is, the adversary will never gain an advantage, but will be caught. The best strategy for such an ideal system is \( c_2 \).

**Minimum gain of the software:** When \( p_{mu} = 1 \) and \( p_s = 1 \), the software will have the minimum gain. In this case, if the software requests more resources, then the minimum gain is \( e_{uc0} - l_{mu0} - l_{mu} - l_a \). And the software should not request more resources. However, the gain of the adversary is \( e_a - p_c l_c \). In this case, the adversary should establish an attack only when \( e_a \geq p_c l_c \). The best strategy for such a system is \( c_3 \) for \( e_a \geq p_c l_c \) and \( c_4 \) for \( e_a \leq p_c l_c \).

**Maximum gain of the adversary:** When \( p_s = 1 \) and \( p_c = 0 \), the adversary will have the maximum gain, which is \( e_a \). And its best choice is to attack the software. However, the gain of the software is \( e_{uc0} + (1 - p_{mu}) e_{uc} - l_{mu0} - p_{mu} l_{mu} - l_a \) when requesting for more resources, and \( e_{uc0} - l_{mu0} - l_a \) when not requesting for more resources. The best strategy is \( c_1 \) or \( c_3 \).

**Minimum gain of the adversary:** When \( p_s = 0 \) and \( p_c = 1 \), the adversary will have minimum gain. In this case, if the adversary establishes an attack, then the gain will be \(-l_c\). And its best choice is not to attack the software.
However, the gain of the software is $e_{uc0} + (1 - p_{mu})e_{uc} - l_{mu0} - p_{mu}l_{mu}$ when requesting for more resources, and $e_{uc0} - l_{mu0}$ when not requesting for more resources. The best strategy is $c_2$ or $c_4$.

Then, to further analyze the strategies of both the software and the adversary, we provide an example where $e_{uc0} = 1$, $l_{mu0} = 0.5$, $e_{uc} = 1$, $l_{mu} = 0.5$, $l_e = 0.5$, and $l_c = 0.5$. For the software, its gain depends on the probabilities of misusing and successful attack as illustrated below.

First, to analyze the influence of misusing resources, we set $p_s = 0.3$, and draw the gains of the software in the four strategies as shown in Figure 2(a).

Second, to analyze the influence of successful attack, we set $p_{mu} = 0.5$, and draw the gains of the software in the four strategies as shown in Figure 2(b).

For the adversary, its gains depend on the probabilities of successful attack and being caught. First, to analyze the influence of successful attack, we set $p_c = 0.5$, and draw the gains of the adversary in the four strategies as shown in Figure 2(c). Next we set the value of $p_s = 0.25$ and plot the adversary’s gain for all four strategies as shown in Figure 2(d).

8 GAME-THEORETIC ANALYSIS OF 2017 MOBILE MALWARE THREATS
8.1 Key Statistics
The global smartphone shipments worldwide are projected to add up to 1.7 billion units by 2020. The annual smartphone shipments in North America are 201.3 M. Global smartphone sales are $478.7 billion and are 84 billion in North America (see www.statista.com). Currently more than 100 million users in the United States access the Internet via tablet or smartphone. Android is perhaps the most popular smartphone operating system followed by Apple’s iOS. The Apple App Store (2M apps) and Google Play (3.8 M apps) are most popular and account for over 90% of worldwide App revenue. Google Play’s market share is steadily increasing and has doubled within the previous year. During the last four quarters about 5.72 billion malicious installation packages on mobile devices were reported (Q2’17 1.32 M; Q3’17 1.60 M; Q4’17 1.32; Q1’18 1.32 M). In 2017 the amount of damage caused by cyber crime as reported to Internet Crime Complaint Center amounted to $1.42 billion. ($1.45 billion in 2016; $1.07 billion in 2015; $ 800 million in 2014; and $782 million in 2013).

These staggering statistics provide us a compelling reason to choose malware infection in Android apps for the analysis of our game-theoretic model.

8.2 App Selection and the Financial Model
We searched over 50 sources for the latest statistics related the most prevalent mobile malicious apps. Of these the 2018 report [33] (Internet Security Threat Report, Volume 23) by Symantec Corporation contained the most comprehensive data and most important this report is freely downloadable thus providing data (and thus analysis) that can be reproduced. We briefly discuss the methodology used by Symantec to collect the statistics.

8.3 Description of Data
Symantec Global Intelligence Network comprises more than 126.5 million attack sensors [33]. It monitors threat activities over 175 million endpoints located in 157 countries. Symantec maintains comprehensive vulnerability databases with more than 95,800 recorded vulnerabilities from 25,000 vendors representing 78,700 products. Symantec Endpoint Protection Mobile (SEP Mobile) is used to provide mobile threat intelligence. SEP Mobile uses massive crowd-sourced threat intelligence, in addition to device-based and server-based analysis. From the plethora of mobile malicious apps data, we chose the various statistics for the following five apps for Android OS.

8.3.1 Brief Description of the Five Most Prevalent Mobile Malicious Apps. We chose the five most prevalent malicious apps [33], namely Malapp, Fakeinst, Premiumtext, Maldownloader, and Simplelocker. These apps
Fig. 2. Gains of the software and adversary with various combinations of attack, misuse, and when the adversary is caught.

Malapp: This is a VoIP app added to the Play Store, puts contact and communications data at risk given its capabilities and correlations to known bad actors. Even though it matched no prior signature nor engaged in overtly malicious behavior, this is a pre-crime threat.

Fakeinst: This is an example of a Trojan; it uses randomization techniques to evade detection. Variants in the Fakeinst malware family are installers for various (possibly legitimate) applications. While active however, the Trojans send SMS messages to premium rate phone numbers or a subscription-based paid service. The core functionalities of the variants are very similar, with only slight variations in the implementation and code. For instance, one variant uses single SMS sending, whereas a later variant may use asynchronous and threaded SMS sending.

Premiumtext: Premiumtext is a Trojan horse on the Android platform that sends SMS texts to premium-rate numbers. These Trojans will often be repackaged versions of genuine Android software packages, often distributed outside the Android Marketplace. The Trojan generally arrives within a repackaged .apk file from a legitimate application.

Maldownloader: This corresponds to a wide variety of individual but unclassified threats. One illustration is a ransomware where it changes the device’s PIN to a random value, thus locking the user out of their phone. The PIN is neither stored on the device nor sent to any server. It can only be reset remotely by the attacker once the ransom is paid. It may also be used to encrypt all the files on the device’s primary storage using AES encryption algorithm and rename the files to have the extension “.cryeye”. The user is then shown a screen in which the attacker demands a ransom for decrypting the user’s file. The ransom is
0.013 Bitcoins and it must be paid within 24 hours. If the ransom is not paid, then the data would remain encrypted and would not be deleted.

- **Simplelocker**: Simplelocker represents a ransomware. It is a Trojan horse for Android devices that may encrypt files on the compromised device. It then asks the user to pay to decrypt these apps or files on the compromised devices.

9 MOBILE MALWARE ANALYSIS AND RESULTS

In this section, we provide financial analysis using a modified version of The Gordon–Loeb model [1] and using real data from the 2018 Symantec ISTR report [33].

For a better understanding of the scenario discussed throughout Section 9, it is crucial to look at some of the statistics from the 2018 ISTR report [33] and the variables they translate to in our analysis. They are as follows:

- The ISTR report presents the list of top ten most frequently blocked Android malware in 2017 [33], of which we consider the top five for our analysis. Apps and the rate at which they are blocked are as follows: (a) Malapp, 20.2%; (b) Fakeinst, 16.4%; (c) Premiumtext, 11.2%; (d) Maldownloader, 10.0%; and (e) Simplelocker, 8.8%. These values translate to the probability of the adversary being caught, \( p_c \), as 0.2, 0.16, 0.11, 0.1, and 0.08, respectively, in our analysis. The probability of a successful attack, \( p_s \), is derived by simply computing \( 1 - p_c \).
- The report also presents the percentage of apps that leak sensitive information of three categories [33]: Phone number, 63%, Location information, 37%, and Installed app information, 35%, which is factored into the resource misuse probability \( p_{mu} \), by multiplying the probabilities with \( p_c \).
- For simplicity in discussion and analysis, we also make the assumption that the gain or from correct usage of the software \( euc \) and resources is $14,000 and the gain for the adversary \( e_a \) is $20,000, which are reasonable even for small and medium sized organizations.

With these basic values, the equations from Section 5 were used to populate the fields in Table 5.

9.1 Enhanced Gordon–Loeb Model for Cyber-Insurance and Security of Mobile Apps

As discussed briefly in Section 3.4, the Gordon–Loeb model is a mathematical economic model that helps in analyzing the optimal investment level in information security by quantizing the gain or loss if a particular security measure was to be implemented. However, the Gordon–Loeb model is a generic take on information security measures that needs to be adapted to the mobile apps security paradigm to be truly useful for individuals and businesses in estimating feasible expenses on security measures. We present our adaptation to the Gordon–Loeb model, by considering the additional levels or threats and vulnerabilities shown in the ISTR report [33] and introducing suitable variables to represent them.

Two equations from the Gordon–Loeb model [1] are of interest to us: The *Expected benefits of an investment in information security*, denoted by \( EBIS(z) = [v - S(z, v)]L \). This is the reduction in loss due to investment in security. The *Expected net benefits of an investment in information security*, denoted by \( ENBIS(z) = [v - S(z, v)]L - z \). This is the net reduction in loss due to investment in security, accounting for the cost of investment as well.

We also need to consider other factors and statistics from the ISTR report [33] in addition to those that were discussed at the beginning of Section 9, like the category of the app (Lifestyle, Music and Audio, Books and Reference, etc.), rate of risk in devices (1 in 107 devices were identified as high-risk [33]), and cumulative exposure to network threats over time. Tables 6, 7, and 8 present summarized statistics about the same.

To factor in the various levels of threats and vulnerabilities, we change the model as follows: Net gain for the software by implementing a security measure \( z \) is given by \( \Pi(z) = (t_c v - p_c - S(z, t_c, v, p_c, ex, \varrho))L - z \), where \( t_c \) is the threat probability of the category of the app (Lifestyle : .27, Music and Audio : .20, so on), \( v \) is the vulnerability or the probability of a breach occurring (1/107; 1 in 107 devices were identified as high-risk [33]),
Table 5. Financial Parameters Calculated from 2017 Statistics (See Symantec ISTR 2018 Report [33]) See Section 9.1 for the methodology of how we came up with these numbers

|                        | (a)  | (b)  | (c)  | (d)  | (e)  |
|------------------------|------|------|------|------|------|
| Probability of being   | 0.2  | 0.16 | 0.11 | 0.1  | 0.08 |
| caught \( p_c \)        |      |      |      |      |      |
| Phone number            | 2520 | 2016 | 1386 | 1260 | 1008 |
| Adversary               |      |      |      |      |      |
| Loss if caught \( l_c \) |      |      |      |      |      |
| Location info           | 1480 | 1184 | 814  | 740  | 592  |
| Installed app info      | 1400 | 1120 | 770  | 700  | 560  |
| Probability of successful attack \( p_s \) | 0.8 | 0.84 | 0.89 | 0.9 | 0.92 |
| Earnings if successful  | 16,000 | 16,800 | 17,800 | 18,000 | 18,400 |
| \( e_a \) (in $)        |      |      |      |      |      |

Resource misuse probability \( p_{mu} \)

|                  | (a)  | (b)  | (c)  | (d)  | (e)  |
|------------------|------|------|------|------|------|
| Phone number     | 0.126 | 0.1008 | 0.0693 | 0.063 | 0.0504 |
| Location Info    | 0.074 | 0.0592 | 0.0407 | 0.037 | 0.0296 |
| Installed app info | 0.07 | 0.056 | 0.0385 | 0.035 | 0.028 |

Software

|                  | (a)  | (b)  | (c)  | (d)  | (e)  |
|------------------|------|------|------|------|------|
| Financial loss if misused \( l_{mu} \) (in $) | 5400 | 4320 | 2970 | 2700 | 2160 |
| Earnings when resource used correctly \( e_{uc} \) (in $) | 14,000 | 14,000 | 14,000 | 14,000 | 14,000 |
| Financial loss if attacked successfully \( l_a \) (in $) | 11,200 | 11,760 | 12,460 | 12,600 | 12,880 |

The apps are represented columnwise: (a) Malapp, (b) Fakeinst, (c) Premiumtext, (d) Maldownloader, and (e) Simplelocker.

Table 6. Top 5 App Categories for Malware

| Category         | Malware % |
|------------------|-----------|
| Lifestyle        | 27%       |
| Music & Audio    | 20%       |
| Books & Reference| 10%       |
| Entertainment    | 6%        |
| Tools            | 6%        |

Table 7. Percentage of Apps Leaking Information

| Type of Information | Apps % |
|---------------------|--------|
| Phone Number        | 63%    |
| Location Info       | 37%    |
| Installed App Info  | 35%    |

Table 8. Cumulative Exposure to Network Threats over Time

| Time (in months) | % of devices |
|------------------|--------------|
| +1               | 21%          |
| +2               | 31%          |
| +3               | 38%          |
| +4               | 44%          |

Condensed Statistics from Various Locations in the ISTR Report [33].

\( p_c \) is the probability of blocking the malware or attack (Malapp: 20, Fakeinst: 16, and so on), \( L \) is the loss, and \( S \) is a function to obtain the revised measure of vulnerability after implementing the security measure \( z \). The function \( S \) also takes into account the exposure to network threats \( ex \), which is directly proportional to the time of device usage (1 month: 21%, 2 months: 31%, and so on), and inversely proportional to the probability of applying other security measure \( \varrho \) such as OS upgrades and patches.

9.2 Game Theory–based Analysis of Existing Software and Five Malware Apps

To elaborate on the scenario presented in Table 5, we revisit the four conclusions at the end of Section 5, pointing out the conditions and the corresponding best strategies for the software and the adversary. We have presented the financial parameters for software installed on smartphones, as shown in Table 5. In this subsection, we
Table 9. Parameters That Affect the Choice of Strategy and Corresponding Values (in $) from the Latest ISTR Report [33]

| Parameters               | (a) Malapp | (b) FakeInst | (c) Premiumtext | (d) Maldownload | (e) Simplelocker |
|--------------------------|------------|--------------|-----------------|-----------------|-----------------|
| $e_a$                    | 16,000     | 16,800       | 17,800          | 18,000          | 18,400          |
| $p_{clc}/p_s$            | 1,350      | 823          | 367             | 300             | 188             |
| $e_{uc}$                 | 14,000     | 14,000       | 14,000          | 14,000          | 14,000          |
| $p_{mu}l_{mu}/(1 - p_{mu})$ | 1997       | 1190         | 517             | 421             | 261             |

compute the parameters that allow us to analyze and choose a specific strategy $c_1, c_2, c_3, c_4$ from the base data given by Table 5. The results are shown in Table 9.

From Table 9, we can draw the following conclusions.

Given the numbers, $c_1$ is the best strategy. Since $e_a \geq p_{clc}/p_s$ and $e_{uc} \geq p_{mu}l_{mu}/(1 - p_{mu})$ for all five software, the $c_1$ strategy is the best strategy for the adversary using these apps and the software requesting more resources.

That is, these software should request resources, and the adversary should establish an attack.

However, we have used adversary earnings as $20,000 and the probability of successful attack is $>0.8$. If either of these two values decreases, then strategy $c_1$ will no longer be the best. For example, if $p_s$, then the probability of successful attack was to be 0.1 (1 in 10) and the earnings were $5000 then $e_{uc} \geq p_{mu}l_{mu}/(1 - p_{mu})$ is no longer valid in case of Malapp and FakeInst and strategy $c_3$ will be better. Payoff is so less that it is better to not request new resource but it is advantageous for the attacker to attack. Similarly, we assume earnings of the software, as $14,000, which is reasonable for an organization. Earnings of less than $1,000 will present an entirely different scenario.

The financial loss of misusing a resource is relatively lower than that of malicious attacks. Tables 5 and 9 show that the probability of misusing is quite low, and the financial loss due to misusing is relatively lower than that of successful attacks. We also observe that the probability of being caught is quite low, but the earnings of successful attack is quite high. Therefore, the adversary always prefer to establishing an attack. And software companies should invest more on security to prevent malicious attacks. Table 9 provides a guiding principle of when to invest more to ensure safety relative to the payoff.

9.3 Insurance, Premiums, and Claims

In this section we use the financial values that we arrived at in Section 9.1 to compute the insurance premiums for a company or individual assuming that they have the five apps ((a) Malapp, (b) Fakeinst, (c) Premiumtext, (d) Maldownloader, and (e) Simplelocker) on their portfolio. The risks involved from the software’s perspective can be derived from the financial parameters shown in Table 5. The financial loss if misused ($l_{mu}$) and financial loss if attacked successfully ($l_a$) are the values considered to arrive at the insurance premiums in the discussions that follow. For example, if all five apps were considered as a group, then, first, the mean risk for each app is computed as $(l_{mu} + l_a)/2$ separately and then averaged over all five apps, which yields the mean value of 7,845 in our calculations as a group. Whereas, further down in our calculations where the five apps were analyzed individually, we simply compute $(l_{mu} + l_a)/2$ for each app yielding the risk means of 8,300, 8,040, 7,715, 7,650, and 7,520 for (a) Malapp, (b) Fakeinst, (c) Premiumtext, (d) Maldownloader, and (e) Simplelocker, respectively.

9.3.1 Premium Calculation. Assuming the five apps to be on a company’s or individual’s portfolio as a group, we use the formulas covered in Section 3.5 to calculate the standard deviation principle and the insurance premiums as shown below:
As a standard practice we choose $\psi_0 = 1\%$, $\therefore |\ln \psi_0| = 4.6052$

We can then calculate:

| Interest  | Portfolio | Optimum $\tau_0$ | Premium $\rho$ in $\$ for risks in portfolio |
|-----------|-----------|-------------------|---------------------------------------------|
| Rate      | Premium   | $\tau_0 = \sqrt{\frac{|\ln \psi_0|}{2i}\sigma^2}$ | Optimum $k$ |
| $i = 2\%$ | 43712.79  | 112194.79         | 4.1 $\times 10^{-5}$                      |
| $i = 5\%$ | 46320.82  | 70958.21          | 6.4 $\times 10^{-5}$                      |
| $i = 10\%$| 49260.00  | 50175.03          | 9.1 $\times 10^{-5}$                      |

If the five apps were analyzed individually, again using the formulas covered in Section 3.5, then the insurance premiums for each app are as follows:

| No. of Risks | Mean | Variance | Variance Principle | Std. Dev. Principle |
|--------------|------|----------|--------------------|---------------------|
| (a) Malapp   | 1    | 8,300    | 16,820,000         | $1 + \frac{k}{2}$  |
| (b) FakeInst| 1    | 8,040    | 27,676,800         | $1 + \frac{k}{2}$  |
| (c) Premiumtext| 1   | 7,715    | 45,030,050         | $1 + \frac{k}{2}$  |
| (d) Maldownloader| 1  | 7,650    | 49,005,000         | $1 + \frac{k}{2}$  |
| (e) Simplelocker| 1 | 7,520    | 57,459,200         | $1 + \frac{k}{2}$  |
| portfolio    | 39,225 | 195,991,050 | 39225 + $\sqrt{2i |\ln \psi_0|}$ | 13999.68 |

We again choose $\psi_0 = 1\%$, $\therefore |\ln \psi_0| = 4.6052$

We can then calculate individual premiums:

| Interest  | Portfolio | Optimum $\tau_0$ | Premium $\rho$ in $\$ for risk |
|-----------|-----------|-------------------|-----------------------------|
| Rate      | Premium   | $\tau_0 = \sqrt{\frac{|\ln \psi_0|}{2i}\sigma^2}$ | (a) (b) (c) (d) (e) |
| $i = 2\%$ | 45233.58  | 150214.67         | 3.06 $\times 10^{-5}$      |
| $i = 5\%$ | 48725.41  | 95004.10          | 4.84 $\times 10^{-5}$      |
| $i = 10\%$| 52660.60  | 67178.04          | 6.85 $\times 10^{-5}$      |

We observe that the yearly insurance premium, $\rho$, would approximately range from $8,800 to $11,500 when the malicious apps are considered separately and from $43,500 to $49,500 when considered as a group. In both cases, the premium amounts are reasonable for a company or an individual with financial dependencies on the software.

9.3.2 Insurance Claims. In the event of a loss to the tune of amount $L$, several possibilities arise depending on the amount of insurance that the company or individual chose to protect their assets with. We continue with our calculations for the five apps described in the previous sections, a loss $L$ may occur either due to misuse of...
resources by the software or due to successful attack by the adversary. The financial values for both cases have been presented in Table 5. Using the formula described in Section 3.5, values for $L$ from Table 5 and three different values for an insured’s choice of coverage (50%, 75%, and 100%); we can calculate the value of a claim $\zeta$ (filed by the insured when loss occurs) as follows:

| Probability of successful attack | (a) | (b) | (c) | (d) | (e) |
|----------------------------------|-----|-----|-----|-----|-----|
| Phone number                     | 0.126 | 0.1008 | 0.0693 | 0.063 | 0.0504 |
| Location info                    | 0.074 | 0.0592 | 0.0407 | 0.037 | 0.0296 |
| Installed app info               | 0.07 | 0.056 | 0.0385 | 0.035 | 0.028 |
| Resource misuse probability      |      |      |      |      |      |
| Financial Loss (in $)            |      |      |      |      |      |
| Resource misused                 | 5,400 | 4,320 | 2,970 | 2,700 | 2,160 |
| Attack successful                | 11,200 | 11,760 | 12,460 | 12,600 | 12,880 |
| Required insurance premium for $i=2\%$ (in $) | 8815.6 | 8888.5 | 9095.5 | 9152.3 | 9281.5 |
| If the insured chose 50% coverage (in $) | 4407.8 | 4444.25 | 4576.15 | 4640.75 |      |
| Actual premium, $\rho$           | 2,700 | 2,160 | 1,485 | 1,350 | 1,080 |
| if resource misused               | 5,600 | 5,880 | 6,230 | 6,300 | 6,440 |
| if attack successful              | 8,300 | 8,040 | 7,715 | 7,650 | 7,520 |
| Claim $\zeta$                     |      |      |      |      |      |
| If the insured chose 75% coverage (in $) | 6611.7 | 6666.3 | 6821.6 | 6864.2 | 6961.1 |
| Actual premium, $\rho$           | 4,050 | 3,240 | 2227.5 | 2,025 | 1,620 |
| if resource misused               | 8,400 | 8,820 | 9,345 | 9,450 | 9,660 |
| if attack successful              | 12,450 | 12,060 | 11572.5 | 11475 | 11,280 |
| Claim $\zeta$                     |      |      |      |      |      |
| If the insured chose 100% coverage (in $) | 8815.6 | 8888.5 | 9095.5 | 9152.3 | 9281.5 |
| Actual premium, $\rho$           | 5,400 | 4,320 | 2,970 | 2,700 | 2,160 |
| if resource misused               | 11,200 | 11,760 | 12,460 | 12,600 | 12,880 |
| if attack successful              | 16,600 | 16,080 | 15,430 | 15,300 | 15,040 |

10 SIGNIFICANCE OF THE WORK

The deliberate choice of the game-theoretic adversarial model with only strategies of attack and no-attack and request and no-request gives the software and the attacker a clear idea of likelihood of the success (or failure) of the attack, so that the software can make a decision to ask for more resources (or permissions) or not. For example, if the success is more likely with a greater potential or for large loss, then an organization can set the policy to restrict allowing more resources (or permissions). It has has the advantage that the output of the game theory model feeds seamlessly into the financial calculations (Gordon–Loeb model) and the cyber-insurance calculations. These strategies and the payoffs for the adversarial decisions may be calculated directly from the publicly available figures such as the ISTR report or other published data. This approach helped us build a theoretic model that has immediate practical application and at the same time develops a foundation to build multi-user multi-strategy formulation and more complex game scenarios such as when the strategy space is exponential.

Our work provides a basis for developing more precise models of cyber-insurance and game-theoretic analysis of choices between software and adversary. Conditions to achieve Nash equilibrium shown through proofs in Theorem 1 through Theorem 9 provide us with a theoretical foundation for choosing one of the four strategies. Conditions to achieve Nash equilibrium in complex multiplayer games with multiple strategies is a direct extension of our model. Development of cyber-insurance methods provides a firm basis to build more sophisticated methods in special situations and adapted to specific needs, for example small organizations may choose to address only specific apps; may set policies for their employees to give privileges to apps installed based not only
on risk. The amount of insurance and the premiums can help them in financial planning and risk management policies.

Analysis of the most recent real-world malware apps (2018 ISTR report) provides us substantiation of the potential applicability of our approach.

Using these strategies, a novice user can guesstimate the chance of an attack, and possible earnings from the software she wants to have more resources from and use the default parameters to get detailed guideline of whether it would be beneficial to ask for more resources. A more sophisticated user or an enterprise or a business can play sophisticated games (see Table 9) with different combination of parameters to get a detailed analysis of if–then scenarios of asking for more resources. The if–then scenarios can help in calculating parameters for insuring software based on the current and projected usage of software, life-styles, the type of app used, and prevalence and threat of malicious attacks.

11 CONCLUSION

We presented a method of calculating parameters for insuring software against the potential of compromise because of attack by malicious apps. By focusing on software installed on smart phones we developed Nash equilibrium by analyzing the possible choices of the software and the adversary, and pointed out the best choices for them under various adversarial conditions. Experimental results on real-world data show that our analysis is reasonable, and can be deployed in practice. For various choices, we then built a cyber-insurance framework to present potential monetary value of insurance, possible loss, and the premiums under varying conditions. We posit that our work will be of value to individual, both small and large businesses, government organizations, and insurance companies to develop insurance models and practice the craft of insuring software against malicious attacks.

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their detailed comments, which have greatly improved the article.

REFERENCES

[1] Lawrence A. Gordon, Martin Loeb, and Lei Zhou. 2016. Investing in cybersecurity: Insights from the gordon-loeb model. J. Inf. Secur. 7, 1 (2016), 49–59. DOI: https://doi.org/10.4236/jis.2016.71006

[2] Tansu Alpcan and Tamer Başar. 2010. Network Security: A Decision and Game-theoretic Approach. Cambridge University Press.

[3] W. S. Baer and A. Parkinson. 2007. Cyberinsurance in IT security management. IEEE Secur. Priv. 5, 3 (May 2007), 50–56. DOI: https://doi.org/10.1109/MSP.2007.57

[4] Yuliy Baryshnikov. [n.d.]. IT Security Investment and Gordon-Loeb’s 1/e Rule.

[5] Hans Bühlmann. 1980. An economic premium principle. ASTIN Bull. 11, 1 (1980), 52–60.

[6] Hans Bühlmann. 1985. Premium calculation from top down. ASTIN Bull. 15, 2 (1985), 89–101.

[7] Dorothy E. Denning and Peter J. Denning. 1977. Certification of programs for secure information flow. Commun. ACM 20, 20 (1977), 504–513.

[8] Antonio M. Espinoza, Jeffrey Knockel, Pedro Comesanaafaro, and Jedidiah R. Crandall. 2016. V-DIFT: Vector-based dynamic information flow tracking with application to locating cryptographic keys for reverse engineering. In Proceedings of the International Conference on Availability, Reliability and Security. 266–271.

[9] Xing Gao, Weijun Zhong, and Shue Mei. 2013. A game-theory approach to configuration of detection software with decision errors. Reliab. Eng. Syst. Safety 119 (2013), 35–43.

[10] Bozuan Gu, Xinfeng Li, Gang Li, A. C. Champion, Zhezhe Chen, Feng Qin, and Dong Xuan. 2013. DzTaint: Differentiated and dynamic information flow tracking on smartphones for numerous data sources. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM’13). 791–799.

[11] William G. Horne, Umesh Maheshwari, Robert E. Tarjan, James J. Horning, Olin W. Sibert, Lesley R. Matheson, Andrew K. Wright, and Susan S. Owicki. 2017. Systems and methods for watermarking software and other media. Patent No. US7770016B2, Filed April, 19th 2007, Issued March, 8th, 2010.
[12] M. M. Khalili, P. Naghizadeh, and M. Liu. 2018. Designing cyber insurance policies: The role of pre-screening and security interdependence. *IEEE Trans. Inf. Forens. Secur.* 13, 9 (Sep. 2018), 2226–2239. DOI: https://doi.org/10.1109/TIFS.2018.2812205

[13] Christophe Kienzert, Ziad Ismail, Hervé Debar, and Jean Leneutre. 2018. A survey on game-theoretic approaches for intrusion detection and response optimization. *Comput. Surv.* 51, 8 (2018), 1–31. DOI: https://doi.org/10.1145/3232848

[14] N. Khatri. 2018. The economics of cyber-insurance. *IT Prof.* 20, 6 (Nov. 2018), 9–14. DOI: https://doi.org/10.1109/MITP.2018.2874210

[15] Jinyong Lee, Ingoe Heo, Yongje Lee, and Yunheung Paek. 2015. Efficient dynamic information flow tracking on a processor with core debug interface. In *Proceedings of the Design Automation Conference*. 79.

[16] M. Lelarge. 2012. Coordination in network security games: A monotone comparative statics approach. *IEEE J. Select. Areas Commun.* 30, 11 (2012), 2210–2219.

[17] T. Lepoint, G. Ciocarlie, and K. Eldefrawy. 2018. BlockCIS—A blockchain-based cyber insurance system. In *Proceedings of the 2018 IEEE International Conference on Cloud Engineering (IC2E’18)*. 378–384. DOI: https://doi.org/10.1109/IC2E.2018.00072

[18] Yanlin Li, Jonathan McCune, James Newsome, Adrian Perrig, Brandon Baker, and Will Drewry. 2014. MiniBox: A Two-Way Sandbox for x86 Native Code. In *USENIX Annual Technical Conference (USENIX ATC’14)*. USENIX Association, 409–420.

[19] X. Lu, D. Niyato, H. Jiang, P. Wang, and H. V. Poor. 2018. Cyber insurance for heterogeneous wireless networks. *IEEE Commun. Mag.* 56, 6 (June 2018), 21–27. DOI: https://doi.org/10.1109/MCOM.2018.1700504

[20] Eduardo Maciel Da Cunha Mattos, Marcos Vieira, Eber Assis Schmitz, and Antonio Juarez Alencar. 2014. Applying game theory to the incremental funding method in software projects. *J. Softw.* 9, 6 (2014).

[21] Per Hakon Meland and Finn Seehusen. 2018. When to treat security risks with cyber insurance. In *Proceedings of the 2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA’18)*. 1–8. DOI: https://doi.org/10.1109/CyberSA.2018.8551456

[22] P. H. Meland and F. Seebusen. 2018. When to treat security risks with cyber insurance. In *Proceedings of the 2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA’18)*. 1–8. DOI: https://doi.org/10.1109/CyberSA.2018.8551456

[23] Rahul Savani and Bernhard von Stengel. 2015. Game theory explorer: Software for the applied game theorist. *Comput. Manage. Sci.* 12, 1 (2015), 5–33.

[24] John F. Nash. 1950. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences* 36, 1 (1950), 48–49. DOI: 10.1073/pnas.36.1.48

[25] James Neel, Virginia Mprg, Tech, Jeffrey H. Reed, Mprg, Virginia Tech, and Robert P. Gilles. 2002. The role of game theory in the analysis of software radio networks. *Rev. Med. Chile* 118, 3 (2002), 313–319.

[26] International Association of Insurance Supervisors (IAIS). 2018. Insurance Core Principles.

[27] Yossef Oren, Vasileios P. Kemerlis, Simha Sethumadhavan, and Angelos D. Keromytis. 2015. The spy in the sandbox: Practical cache attacks in JavaScript and their implications. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS’15)*. Association for Computing Machinery, 1406–1418. DOI: 10.1145/2810103.2813708

[28] Nilay V. Oza. 2006. Game theory perspectives on client/vendor relationships in offshore software outsourcing. In *Proceedings of the 2006 International Workshop on Economics Driven Software Engineering Research*. 49–54.

[29] R. Pal and P. Hui. 2013. On differentiating cyber-insurance contracts a topological perspective. In *Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM’13)*. 836–839.

[30] Rahul Savani and Bernhard Von Stengel. 2015. Game theory explorer: Software for the applied game theorist. *Comput. Manage. Sci.* 12, 1 (2015), 5–33.

[31] Milad Jafari Barani, Peyman Ayubi, Milad Yousefi Valandar, and Behzad Yousefnezhad Irani. 2020. blind video watermarking algorithm robust to lossy video compression attacks based on generalized Newton complex map and contourlet transform. *Multimedia Tools and Applications* 79, 3 (2020), 2127–2159. DOI: 10.1007/s11042-019-08225-5

[32] J. Xuexia. 2010. Digital watermarking and its application in image copyright protection. In *2010 International Conference on Intelligent Computation Technology and Automation*, Vol. 2. 114–1147. DOI: 10.1109/ICTA.2010.625

[33] Michael Spreitzenbarth, Felix Freiling, Florian Echtler, Thomas Schreck, and Johannes Hoffmann. 2013. Mobile-sandbox: Having a deeper look into android applications. In *Proceedings of the ACM Symposium on Applied Computing*. 1808–1815.

[34] Shaun Wang. 1996. Premium calculation by transforming the layer premium density. *ASTIN Bulletin: The Journal of the IAA* 26, 1 (1996), 71–92.
[39] Zhang Yanqun and Wang Qianping. 2011. Digital watermarking algorithm of vector map based on feature points. In Proceedings of 2011 Cross Strait Quad-Regional Radio Science and Wireless Technology Conference, Vol. 2, 1430–1433. DOI: 10.1109/CSQRWC.2011.6037234

[40] Pedro Aaron Hernandez-Avalos, Claudia Feregrino-Uribe, and Rene Cumplido. 2012. Watermarking using similarities based on fractal codification. Digital Signal Processing 22, 2 (2012), 324–336. DOI: https://doi.org/10.1016/j.dsp.2011.10.012

Received June 2019; revised May 2020; accepted July 2020