# Towards Smart Anti-Malwares for Battery-Powered Devices

Hajer Al Housani, Hadi Otrok, Rabeb Mizouni
Departement of Electrical and Computer Engineering
Khalifa University of Science, Technology and Research
Abu Dhabi, UAE
{hajer.alhousani,hadi.otrok,rabeb.mizouni}@kustar.ac.ae

Jean-Marc Robert
Ecole de technologie supérieure, Département de génie logiciel et des TI,
Montréal, QC,
jean-marc.robert@etsmtl.ca

Azzam Mourad
Department of Computer Science and Mathematics,
Lebanese American University,
Beirut, Lebanon
azzam.mourad@lau.edu.lb

***Abstract*—Anti-malware programs (AM) are well-known powerful security tools that protects against malicious activities. Behaviour-based AMs are seen as suitable candidates to detect unknown malwares in mobile devices. Usually, AM offers different levels of security namely high, medium, and low. The low level offers minimal basic checks in case of low probability of attacks while the high level offers intensive checks in situations of high probability of attacks. Despite their efficiency, behaviour-based anti-malwares are still considered as resource consumer especially in high detection level which reduces considerably the lifetime of handheld devices. In addition, current anti-malwares are not smart to decide when to shift across those levels to reduce both resources consumption and false alarms without scarifying the security level required. To alleviate this problem, we propose a smart anti-malware that can shift between different security levels according to the assets value and the battery status of the resource-constrained device. Such decision is made possible using a nonzero-sum non-cooperative game theoretical model.**

***Index Terms* — smart anti-malware, game theory, mobile devices, threshold calculation, resource consumption.**

## I. INTRODUCTION

With the new capabilities of mobile devices, these handsets have an increased risk and exposure to malicious programs [1]. Malware is general name for any malicious software designed to cause damage without user consent such as viruses, worms, spyware, keystroke logging [2]...etc. The malware can be used to initiate different forms of attacks. These attacks can vary from violation of confidentiality or privacy to corrupting memory and system files. In addition, spreading worms can have a serious impact on performance of the mobile network due to illegal consumption of bandwidth causing financial loses [3].

Due to the massive continuity of creating malwares of different types for different purposes and severity of damage targeting attacks in mobile devices [4], researchers as well as companies are competing to produce effective anti-malwares [3, 5] for mobile devices. Anti-malware (AM) is a software program than can detect malwares and prevent them from causing further damages. There exist a number of fairly good techniques and products to counter malwares in PCs. However, because mobile devices and PCs are different in terms of architecture, operating systems and available resources, these AM do not necessary fit into mobile devices because of their relatively high resource consumption.

Even with their increasing capabilities, battery-powered devices, such as smart phones, are still considered limited in resources compared to PCs. They still have a limited battery lifetime and developing battery-aware applications is of high value in ubiquitous computing to help extending the lifetime span of the device and avoid its disconnection.

To reduce the power consumption and processor usage as well reducing false alarms of AM in mobile devices, some tools such as ThreatFire was designed based on behaviour and not signature that requires more computations. ThreatFire provides different detection levels according to the sensitivity required by the user. Playing with the levels of detection might cause some problems. In low level, AM may be unable to detect a malware, while in high level unacceptable resources might be consumed without the presence of a malware. In other words, in order to have a trade-off between security level and acceptable resources consumption of the AM, the question that we address in this paper is: *how can the AM switch between the levels to provide an adequate protection and to preserve the device battery?*

To address this problem, and keeping in mind the considerations for malware detection on a mobile device we propose an approach to enable the introduction of smartness to behaviour based AM by allowing it to switch into higher level of security when needed and vice-versa. We solve the problem by using non-zero sum non-cooperative game theoretical model to analyse the interaction between malwares and anti-malware programs running in battery-powered systems. We, formulate our model using three levels of detection, namely *low*, *medium* and *high*. The solution of the game will enable AM to identify appropriate thresholds for changing the current security level as needed and detected by the belief function.

In summary, the contribution of this paper is twofold: (1) modelling the problem of efficient AM in battery-powered devices using game theory and (2) identification of thresholds that enable smart AM to switch at the right moment into a different level of security. The challenges of what constitutes normal and malicious behaviours in behavioural detection that AMs encounter are beyond the scope of this paper.

## II. ANTI-MALWARE GAME MODEL

This section describes the game definition model where different elements of the game are identified. Then, we present the solution of the game where optimal strategy of the players is calculated. Finally, we present the approach of using our obtained results in AM decision strategy to move from lower level to higher level.

### A. *Game Definition*

We formulate our game as nonzero–sum non-cooperative Bayesian game where the type of the player is unknown in prior to other players, and players will not gain and lose the same amount. In fact, it is a Bayesian game because there is an incomplete information about the running application. The game has two players: *j* and *i*, where *j* is the AM (Anti-malware) and *i* is the running program. Obviously, AM does not know if the running application is a normal software or a malware program while the type of the AM is known. Consequently, Player *i* can select his type from the following set Θ = {Malware (M), Normal software (N)} while player *j* is of type M.

In order to construct the game players' strategies, payoffs must be defined. The AM chooses its strategy from the set {*High*, *Medium*, *Low*}. Player *i* selects its strategy from the set {Active, Not Active} if the program type is Malware and {Not Active} if its type is Normal Software. Table I represents all players' Payoffs where:

Table 1: ANTI-MALWARE BAYESIAN GAME

| Strategy | | High | Medium | Low |
|---|---|---|---|---|
| Active | M | $\bar{E}_h V - C_a$ | $\bar{E}_m V - C_a$ | $\bar{E}_l V - C_a$ |
| | AM | $E_h V - C_h$ | $E_m V - C_m$ | $E_l V - C_l$ |
| Not-Active | M | 0 | 0 | 0 |
| | AM | $- C_h$ | $- C_m$ | $- C_l$ |

(a)Player *i* is Malware

| Strategy | | High | Medium | Low |
|---|---|---|---|---|
| Not-Active | N | 0 | 0 | 0 |
| | AM | $- C_h$ | $- C_m$ | $- C_l$ |

(b)Player *i* is Normal software

- $E_h$: is the expected probability of malware detection by the AM that runs at high level.
- $\bar{E}_h = 1 - E_h$: is the expected probability that the AM that runs at high level will not detect the malware.
- $E_m$: is the expected probability of malware detection by AM that runs at medium level.
- $\bar{E}_m = 1 - E_m$: is the expected probability that the AM runs at medium level without detecting the malware activity.
- $E_l$: is the expected probability of malware detection by AM that runs at low level.
- $\bar{E}_l = 1 - E_l$ : is the expected probability that the AM runs at low level without detecting malware.
- $V$: is a fixed value of the assets to protect. Different assets have different values within a system. For example, cookies, files that contain sensitive information used for authentication purposes, are more valuable than other user files.
- $C_a$ : is the cost to the attacker of launching the malware in the user device. It includes the cost of the development of the malware itself.
- $C_h$ : is the computational cost in terms of power consumption and CPU usage that the AM consumes when it is running at high level.
- $C_m$ *and* $C_l$ : are the computational costs of the AM when running at medium and low levels respectively.

Each entry in Table **1** represents payoff. The first part represents the payoffs of player *i* (program) and the second part represents the payoffs of player *j* (Anti-Malware). Moreover, each payoff consists of player's gains and losses respectively. For example, $\bar{E}_h V - C_a$ in the Table **1** (a) represents the payoff of the malware when it selects active strategy and AM runs at high level.

The gain of the malware is to comprise the assets of the user being attacked without being detected while it loses the cost of launching the malware. The $E_h V - C_h$ in Table **1** (a) represents the payoff of the AM. The gain of the AM is protecting the assets against a malware that has been launched by intruder and consequently preserving the value of the assets. AM loses the computational cost that is needed to run on the device at high level. Finally, these conditions $\bar{E}_h V \gg C_a$ *and* $E_h V \gg C_h$ must be satisfied for both players in order to be profitable.

After defining the game elements namely the players, their types, their strategies and their payoffs, we are going to solve this game to obtain the Nash equilibrium point. This point specifies the best strategy that leads to a payoff greater than or equal to any other strategy given that the opponent will always select her/his best strategy. In other words, Nash equilibrium is used to identify players' optimal strategy and to derive the threshold that will notify the AM to step from lower level to upper level and vice-versa. Actually, it is Bayesian Nash equilibrium [10] since player *i* actions depends on his type θ. By observing the behaviour of player *i* (Malware or Normal software) at time $t_k$, the AM can calculate the posterior belief evaluation function $\mu_{t_{k+1}}(\theta_i|a_i)$ using the following Bayes rule:

$$\mu_{t_{k+1}}(\theta_i|a_i) = \frac{\mu_{t_k}(\theta_i)\, P_{t_k}(a_i|\theta_i)}{\sum_{\theta i \in \Theta} \mu_{t_k}(\theta_i)\, P_{t_k}(a_i|\theta_i)} \qquad (1)$$

where $\mu_{t_k}(\theta_i) > 0$ and $P_{t_k}(a_i|\theta_i)$ is the probability that strategy $a_i$ is observed at this stage of the game given the type θ of player *i*. It is computed as follows:

$$P_{t_k}(Active|\theta_i = M) = E_m \times O + F_m(1 - O)$$
$$P_{t_k}(Active|\theta_i = N) = F_m$$

where

- $O$ is an observed event at certain time.
- $F_m$ is the probability of false alarm (false positive alarm) generated by the AM (AM notifies that an

attack (active malware) is taking place while such attack does not exists), and

- $E_m$ is the probability of detecting an active malware that runs at medium level (true positive alarm).

B. *Play the Game*

Let us first recall the definition of a dominant strategy. *Strategic dominance* occurs when one strategy is better than another strategy for one player, no matter how that player's opponents may play. In our game, we notice that low strategy will be dominated by medium strategy. This is due to the fact that the payoff of Medium is greater than low. Consequently, we only use two strategies to solve this game. The obtained game is simplified as represented in table II.

**TABLE II: Medium to high Bayesian game**

| Strategy | High | Medium |
|---|---|---|
| Active | $\bar{E}_h V - C_a$ | $\bar{E}_m V - C_a$ |
| | $E_h V - C_h$ | $E_m V - C_m$ |
| Not-Active | 0 | 0 |
| | $-C_h$ | $-C_m$ |

(a)Player *i* is Malware

| Strategy | High | Medium |
|---|---|---|
| Not- Active | 0 | 0 |
| | $-C_h$ | $-C_m$ |

(b)Player *i* is Normal Software

Applying the same pure strategy again will lead to no Nash equilibrium which is the ultimate goal for both players. Therefore, we choose to play another strategy namely mixed strategy to find the Bayesian Nash equilibrium where q is the probability to run the AM in high level and p is the probability of malware to be active (attack). Obviously, $1-q$ is the probability to run the AM at medium level and $1-p$ is the probability of active malware if he chooses not-active strategy. Utility function for each player is defined in terms of his payoffs with corresponding probabilities.

The utility function of the AM is:

$$U_{AM} = [qp(E_h V - C_h) + p(1-q)(E_m V - C_m) - q(1-p)C_h - (1-q)(1-p)C_m]\mu(\theta = M) - [qC_h + (1-q)C_m](1-\mu(\theta = M))$$

While the utility function of the malware is:

$$U_a = qp(\bar{E}_h V - C_a) + p(1-q)(\bar{E}_m V - C_a)$$

The main goal of each player is to maximize his utility function which is achieved by finding his optimal strategy that defeats the other player optimal strategy leading to the player's equilibrium. Therefore, it is necessary for each player to determine the other player optimal strategy. From mathematical point of view, such optimal strategy values are the highest point for a function to find through calculation of the first derivative of that function [5]. Therefore, after defining each utility function, their derivative will be computed to determine the value of the optimal strategy of the other player where:

- $p^*$ the value of the optimal strategy of malware to attack by being active and
- $q^*$ is value of the optimal strategy of the AM to protect the asset against such attack.

In order to obtain $p^*$, the AM will compute the first derivative of his utility function $U_{AM}$ with respect to $q^*$:

$$\frac{dU_{AM}}{dq} = p(E_h - E_m)V\mu - C_h + C_m$$

$p^*$ can be discovered directly after setting the derivative to zero:

$$p^* = \frac{C_h - C_m}{(E_h - E_m)V\mu}$$

In the malware side, the same procedure is performed to find $q^*$. First he will calculate the first derivative of $U_a$ with respect to $p^*$:

$$\frac{dU_a}{dp} = q\big((\bar{E}_h V - C_a) - (\bar{E}_m V - C_a)\big) + (\bar{E}_m V - C_a)$$

Then, he will set the derivative to zero to find the optimal value of the AM

$$q^* = \frac{C_a - \bar{E}_m V}{(\bar{E}_h - \bar{E}_m)V}$$

$p^*$ and $q^*$ represent the Bayesian Nash equilibrium for both players.

C. *Decision Making*

After we identify how to drive AM threshold $p^*$, let us investigate how the value of $p^*$ effects the AM decision in activating appropriate detection level.

According to our model, the AM will run in medium level initially as well as monitoring a system interacting with different programs. First, the AM calculates the posterior belief function using Equation 1 for each program (Player) with the help of prior observed actions of those programs on the system. Then for each program, AM computes the threshold $p^*$. If an observed action of a program $O$ exceeds the value of the corresponding computed threshold $p^*$, then AM should step into high level to protect the asset against any compromising attack. Likewise, the jump from high level to medium level will take place whenever threshold $p^*$ exceeds the observation of the program $O$ to reduce the additional unnecessarily computational cost of running high level.

To show the use of our game, let us consider a program A interacting with a system. The AM records the current belief of A($\mu = 0.6$) and its observed activity ($O = 0.32$). Assume $C_h - C_m = 10$, $E_h - E_m = 0.83$ and $V = 100$. AM computes the threshold $p^* = \frac{10}{0.83*100*0.6} = 0.2$.

Since the value of $O$ is greater than $p^*$ then the high level of AM will be tuned to provide the most accurate detection and suitable monitoring mechanism. As you can notice, the result of the game provides reduction of the computation cost in terms of CPU usage and battery consumption without scarifying the security of the protected system as well as assuring accurate detection against active malware.

## III. Simulation Results

To demonstrate the benefits our model offers, we simulate two different scenarios: the scenario where a malware is identified by detecting an abnormal behaviour, and the scenario of false positive (an abnormal behaviour detected but normal software is running). In each of these scenarios, we consider the case of high value assets and low value assets,

To simulate these two scenarios, we also consider a host running an anti-malware with different detection levels and with the following parameters:

- $E_h - E_m = 0.25$,
- $C_h - C_m = 10$,
- $\mu_0 = 0.5$, and
- $F_m = 0.1$ .

In the First scenario, we investigate how the behaviour of a malware will be treated according to our model in both medium and high level of detection. The second scenario simulates the behaviour of normal software and how our model smartly reduces false positive alarms. As mentioned previously, there is no need to run the AM in high level every time $O \geq 0.5$ and consume resources. Instead we suggest running AM in medium level initially until $O > p^*$. The $p^*$ suggests the appropriate time to jump from medium to high level to provide the protection needed.

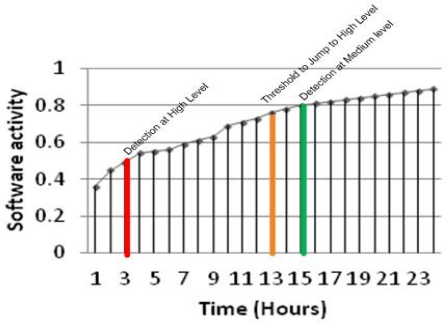


Figure 1: Malware Detection in Low Value Assets. Without our model the AM runs on High level at time 3, and Medium level at time 15 and in our model it runs on high at time 13.

Figure 1 shows the behavior of the AM in the case of low assets. The **red** line (respectively **green** line) shows the detection of the malware if the AM was running at high (respectively medium) level. With our model, the anti-malware jumped from medium to high at *time=13* (**orange** line), detecting as such the malware two hours earlier than running at medium level purely but almost ten hours later than running at high level purely.

Figure 2 shows the behavior of the AM in the case of high assets. The red line (respectively green line) shows the detection of the malware if the AM was running at high (respectively medium) level. With our model, the anti-malware jumped from medium to high at *time = 7* (orange line), detecting as such the malware eight hours earlier than running at medium level purely but 4 hours later than running at high level purely.

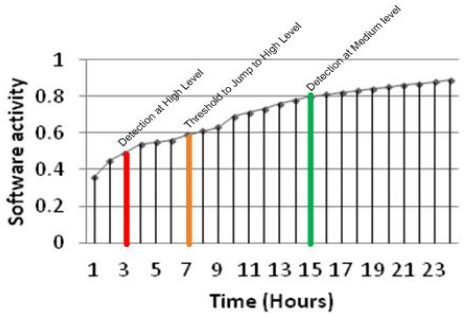


Figure 2: Malware Detection in High Value Asset. Without our smart model, the AM runs on High level at time 3, and Medium level at time 15 and in our model it runs on high at time 7.

## IV. Conclusion

This paper proposes a promising game theoretical model that analyses the interaction between malwares and malware detectors (Anti-malware). We first modelled the AM detection as nonzero–sum non-cooperative Bayesian game. Then, by solving the game we derived appropriate thresholds to switch between levels of detection. Switching between the levels of detection allows the preservation of the battery and increasing the lifetime of the battery-powered systems, such as smart phone, without endangering their security on one hand and helps to minimize the false alarms that a high level produces on the other hand. The driven thresholds make malware detectors smart and intelligent enough to decide when to jump into higher level to provide accurate detection.

## References


[1] B. Abhijit, *et al.*, "Behavioral detection of malware on mobile handsets," presented at the Proceeding of the 6th international conference on Mobile systems, applications, and services, Breckenridge, CO, USA, 2008.

[2] C. P.fleeger and S. L. fleege, *Security in computaing*, Third Edition ed.: prentice Hall, USA, 2003.

[3] V. Deepak and H. Guoning, "Efficient signature based malware detection on mobile devices," *Mob. Inf. Syst.*, vol. 4, pp. 33-49, 2008.

[4] D. Dagon, *et al.*, "Mobile phones as computing devices: the viruses are coming!," *Pervasive Computing, IEEE*, vol. 3, pp. 11-15, 2004.

[5] PCtools. (2011, October 2011 ). *ThreatFire online help*. Available: http://www.threatfire.com/help/

[6] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, pp. 3448-3470, 2007.

[7] P. Garcıa-Teodoroa, *et al.*, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *comp u t e r s & s e c u r i t y, Elsevier* vol. 2 8, pp. 18 - 2 8, 2009.

[8] S. Aubrey-Derrick, *et al.*, "Monitoring smartphones for anomaly detection," presented at the Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications, Innsbruck, Austria, 2007.

[9] D. Shuaifu, *et al.*, "Behavior-Based Malware Detection on Mobile Phone," in *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on*, pp. 1-4.

[10] M. Fredrikson, *et al.*, "Synthesizing Near-Optimal Malware Specifications from Suspicious Behaviors," in *Security and Privacy (SP), 2010 IEEE Symposium on*, pp. 45-60.

[11] H. Otrok, *et al.*, "Game theoretical model for detecting network intrusions," *The computer communications journal, Elsevier*, 2009.

[12] Y. Liu, *et al.*, "A Bayesian game approach in wireless ad hoc network," in *In Proceeding of the GameNets06*, 2006.