Anti-Malware Sandbox Games

Sujoy Sikdar
Binghamton University
Binghamton, NY, USA
ssikdar@binghamton.edu

Sikai Ruan
Rensselaer Polytechnic Institute
Troy, NY, USA
ruans2@rpi.edu

Qishen Han
Rensselaer Polytechnic Institute
Troy, NY, USA
hanq6@rpi.edu

Paween Pitimanaaree
SCB Securities Co. Ltd.
London, UK
paween.pit@gmail.com

Jeremy Blackthorne
Boston Cybernetics Institute
Cambridge, MA, USA
jeremy.blackthorne@gmail.com

Bulent Yener
Rensselaer Polytechnic Institute
Troy, NY, USA
yener@cs.rpi.edu

Lirong Xia
Rensselaer Polytechnic Institute
Troy, NY, USA
xial@cs.rpi.edu

ABSTRACT

We develop a game theoretic model of malware protection using the state-of-the-art sandbox method, to characterize and compute optimal defense strategies for anti-malware. We model the strategic interaction between developers of malware (M) and anti-malware (AM) as a two player game, where AM commits to a strategy of generating sandbox environments, and M responds by choosing to either attack or hide malicious activity based on the environment it senses. We characterize the condition for AM to protect all its machines, and identify conditions under which an optimal AM strategy can be computed efficiently. For other cases, we provide a quadratically constrained quadratic program (QCQP)-based optimization framework to compute the optimal AM strategy. In addition, we identify a natural and easy to compute strategy for AM, which as we show empirically, achieves AM utility that is close to the optimal AM utility, in equilibrium.

KEYWORDS

Anti-malware, Sandbox, Non-cooperative game theory

ACM Reference Format:
Sujoy Sikdar, Sikai Ruan, Qishen Han, Paween Pitimanaaree, Jeremy Blackthorne, Bulent Yener, and Lirong Xia. 2022. Anti-Malware Sandbox Games. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

Malicious programs, or malware (M) for short, are a threat to individuals, companies, and even military units and intelligence agencies. One approach to stopping malware is to scan suspected programs with an anti-malware (AM) program, which can check the suspected program against a list of known, bad programs. In response, malware resort to manipulating its own code dynamically during runtime to bypass static signature scanning. To counter this, AM programs execute suspected programs in a contained, simulated environment, called a *sandbox*, in an attempt to trick the malware into showing its true malicious activity. This allows the AM to check program behavior at run-time against a list of known bad behaviors. In response, malware now checks its environment to make an intelligent decision on whether to *attack* by unleashing its true malicious activity [7]. This raises the question, *what is the optimal way for AM to protect machines defended by it, from malware developers who will study its behavior?*

The tools of *game theory* are well suited to analyze the strategic interaction between malware developers (M) and anti-malware developers (AM) in our setting of sandboxing, where AM's strategies involve the use of sandbox environments to dynamically analyze malware. Despite the large literature and wide application of game theoretic analysis of problems in physical [20] and cyber [10, 16, 17] security, sandboxing has not been analyzed from a game-theoretic perspective.

We use computational game-theoretic methods to develop strategies and guidelines to improve the state of the art in sandbox analysis. Unfortunately, existing literature on game theoretic analysis of malware do not model the sandboxing problem, because the dynamics of the game and utility functions we study in this paper are quite different from previous work. See Section 2 for more details. We address the following key research question:

How can we model sandboxing as a game and characterize and compute optimal strategies for the anti-malaware?

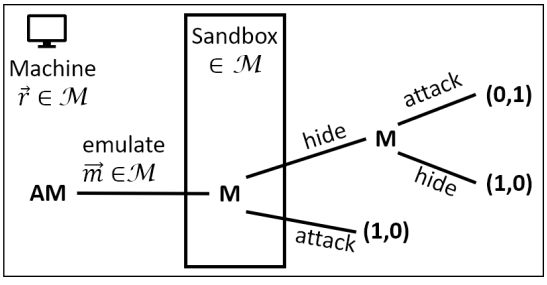


Figure 1: High level depiction of the timing of the game.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

1.1 Our Contributions

Our main conceptual contribution is the first game theoretic model of the strategic behavior between sandboxing AM and M. Figure 1 illustrates the timing and actions of AM and M on a real machine that AM defends.

Suppose an AM is installed on some (not necessarily all) machines. The sandbox game has three stages. On each machine that AM defends, the AM generates a sandbox according to a strategy π within which to analyze the potential malware. Given any machine environment $\vec{m}$ that M is executed in, M decides an action of attacking or hiding according to her chosen strategy ρ. The three stages are:

Stage 1: AM commits to a sandboxing strategy, when it is deployed on the real machines it defends.

Stage 2: M observes AM's strategy and responds with an attack strategy ρ.

Stage 3: If M bypasses the sandbox in stage 2, then it will be run on the real machine, and attack according to ρ.

We note that in stage 3, M uses the same attack strategy ρ as it does in stage 2. This is because M's decision is based only on the environment it perceives during execution, and M cannot distinguish between whether an environment corresponds to a real machine or one that is being emulated by AM in a sandbox.

M's goal is to successfully attack as many real machines as possible. AM's goal is to *protect* the maximum number of machines it *defends* as possible. A real machine is protected if either M attacks in the sandbox and is caught, or M bypasses the sandbox but chooses not to attack the real machine.

Our main technical contributions are analytical solutions of AM-optimal strategies in equilibrium for a wide range of cases, and practical algorithms to compute such solutions otherwise. We summarize our results as a set of guidelines for sandboxing AMs in Section 8. Our main results are:

- When the AM defends every machine, we identify natural and easy to compute AM-optimal solutions (Theorems 1 and 2). These results are summarized in Table 5.
- When AM defends at most half of all real machines, there is an equilibrium solution where AM protects all machines which it defends, as we show in Theorem 3. A potential application of this finding is that the deployment setting may be modified by creating undefended dummy virtual machines that emulate real machines, which superficially increase the number of machines where AM is not installed.
- We provide a quadratically constrained quadratic program (QCQP) framework in Algorithm 1 for computing an AM-optimal solution in equilibrium for real world settings where no analytical solution is known.
- We show through theoretical results or experiments that natural strategies of sandboxing are effective. For example, even on cases where no analytical solution is known, generating sandbox environments using a distribution identical to the the distribution of real machines in existence yields good AM utility in practice.

2 RELATED WORK

Anti-analysis. *Sandbox* is a general term referring to any simulated or contained environments, such as emulation, virtualization, or debuggers. Although these tools intend to mimic the original environment, they often do not do so perfectly. These imperfections allow programs that can detect them to distinguish between running within sandboxes from running within normal environments. This gives programs the capability of decision, specifically to act benignly while in a sandbox and act maliciously when running in the normal environment, in an attempt to distinguish the real environment from the artificial, sandbox environment. This results in a game of the malware inspecting the sandbox as the anti-malware attempts to inspect the program. Both parties are attempting inspection and classification.

Bulazel et al. [7] did a recent survey of automated malware analysis and evasion. There is extensive literature cataloging the various ways to detect emulators, virtual machines, and debuggers [6, 15]. There is also extensive work in engineering solutions that approximate transparent introspection of programs [1, 8].

Formalization. Blackthorne et al. [3] define a formal framework for environmentally sensitive malware in which they show the conditions necessary for malware to distinguish sandboxes and remain undetected. Blackthorne et al. [2] formalize malware interacting with and distinguishing environments through the lens of cryptography with the label environmental authentication. They analyze the interaction in terms of correctness, soundness, and key sources, but they do not suggest any strategies for game players to use possible equilibria. Dinaburg et al. [9] present a formalization for transparent malware analysis. *Transparent* analysis means using an environment which is impossible to detect by the program being analyzed, i.e., the analysis environment is transparent. Kang et al. [12] also briefly formulate the problem of transparent malware analysis within emulators, but do not offer any further investigation than the beginnings of describing transparency.

Game Theory. To the best of our knowledge, our work is the first to formally model the interaction between M and AM via sandboxing through the lens of game theory. Game theory has been used to model various problems in cybersecurity. (See [17, 18] and [10, 16] for recent surveys). However, relatively little literature exists on game theoretic analysis of the interactions between the developers of malware and anti-malware.

Sinha et al. [19] discusses techniques developed to address the challenges of scale, uncertainty, and imperfect observation by the attacker in security games modeled as Stackelberg games, and applications to cybersecurity. Stackelberg games have also been used to model security games involving web applications [21], network security [11, 22], competition among multiple malwares [13], and audit games [4, 5].

Our model has a number of differences with previous work. First, the AM must commit to the same strategy on every machine of the same type. Unlike security games, when AM can commit to mixed strategies, there are an infinite number of pure strategies for even a naïve AM which uses the same strategy on every machine. Each pure strategy is represented by a vector with a component for each type of machine whose value correspond to the probability with which AM creates a sandbox of that type. This means that unfortunately, standard algorithms and techniques for solving security games do not apply to the sandbox game.

$$u_{\mathrm{M}}(\pi,\rho) = \sum_{\vec{r}\in\mathcal{M}} \underbrace{\left(1-\vec{d}_{\vec{r}}\right)\vec{\rho}_{\vec{r}}}_{\text{M attacks | No AM}} + \underbrace{\vec{d}_{\vec{r}}\left[1-\sum_{\vec{m}\in\mathcal{M}}\vec{\pi}^{\vec{r}}_{\vec{m}}\vec{\rho}_{\vec{m}}\right]\vec{\rho}_{\vec{r}}}_{\text{M evades sandbox and attacks real machine | AM installed}}$$

$$u_{\mathrm{AM}}(\pi,\rho) = \sum_{\vec{r}\in\mathcal{M}} \underbrace{\vec{d}_{\vec{r}}}_{\text{AM installed}} \left[\sum_{\vec{m}\in\mathcal{M}}\left[\underbrace{\vec{\pi}^{\vec{r}}_{\vec{m}}\vec{\rho}_{\vec{m}}}_{\text{M caught in sandbox}} + \underbrace{\vec{\pi}^{\vec{r}}_{\vec{m}}(1-\vec{\rho}_{\vec{m}})(1-\vec{\rho}_{\vec{r}})}_{\text{M evades sandbox, then hides}}\right] + \underbrace{\left(1-\sum_{\vec{m}\in\mathcal{M}}\vec{\pi}^{\vec{r}}_{\vec{m}}\right)(1-\vec{\rho}_{\vec{r}})}_{\text{No sandbox. M hides}}\right]$$

Figure 2: Utility functions of AM and M.

3 MODELING THE SANDBOX GAME

We are given a set $\mathcal{M}$ of all types of environments, where each $\vec{m} \in \mathcal{M}$ is represented by a k-vector of features (i.e., emulated clock time, fingerprint, network connection, etc.) that fully and uniquely describes machines of type $\vec{m}$. Every *real machine* presents some environment in $\mathcal{M}$ natively. Similarly, every sandbox generated by AM presents also presents an environment in $\mathcal{M}$. Table 1 summarizes the notation used throughout this paper.

Real World Settings are described by $|\mathcal{M}|$-vectors $\vec{e}$ and $\vec{d}$. The $\vec{r}$-th component of $\vec{e}$, denoted $\vec{e}_{\vec{r}}$ is the fraction of all machines in existence in the real world that are of type $\vec{r} \in \mathcal{M}$. The $\vec{r}$-th component of $\vec{d}$ (respectively $\vec{1}-\vec{d}$) is the fraction of all real machines that are of type $\vec{m}$ and are *defended* (respectively *not defended*) by AM. In addition, we will use D and $1-D$ to denote the proportion of all real machines that defended, and not defended by AM, respectively.

AM's Strategy Space is the set Π of all functions π that map each real machine type $\vec{r} \in \mathcal{M}$ to an $|\mathcal{M}|$-vector $\pi(\vec{r}) = \vec{\pi}^{\vec{r}}$ which describes a distribution over $\mathcal{M}$. On any given real machine of type $\vec{r} \in \mathcal{M}$, the $\vec{m}$-th component of $\vec{\pi}^{\vec{r}}$, denoted $\vec{\pi}^{\vec{r}}_{\vec{m}}$, gives the probability with which AM generates a sandbox of type $\vec{m} \in \mathcal{M}$ on a real machine of type $\vec{r}$ when AM uses strategy π. The probability that an AM using a strategy π does not generate a sandbox on a real machine $\vec{r}$ is $1-\sum_{\vec{m}\in\mathcal{M}}\vec{\pi}^{\vec{r}}_{\vec{m}}$.

Malware's Strategy Space is the set P of all vectors in $[0,1]^{|\mathcal{M}|}$. Each strategy $\rho \in \mathrm{P}$ is represented by an $|\mathcal{M}|$-vector $\vec{\rho}$. The $\vec{m}$-th component of $\vec{\rho}$, denoted $\vec{\rho}_{\vec{m}}$ is the probability with which M attacks when presented with an environment $\vec{m} \in \mathcal{M}$. Note that M cannot distinguish between a sandbox and a real machine that present the same environment. M's strategy only depends on the execution environment $\vec{m}$ presented to it, regardless of whether it is an environment presented by a real machine or by a sandbox.

Restrictions on strategy space. We say that AM is *naíve* if it generates sandboxes with the same probability distribution on every type of real machine, i.e., when AM is restricted to a strategy $\pi \in \Pi$, where for every pair of real machine types $\vec{r}, \vec{r}' \in \mathcal{M}$, it holds that $\vec{\pi}^{\vec{r}} = \vec{\pi}^{\vec{r}'}$. We say that AM is *deterministic* if for every machine $\vec{r} \in \mathcal{M}$, there is a sandbox environment $\vec{m}$ which is deterministically always generated by AM in $\vec{r}$. Otherwise, AM is *non-deterministic*. We say that AM is *sophisticated*, if AM can choose different distributions over $\mathcal{M}$ to create sandboxes on any real machine, i.e., its strategy space is all of Π.

Similarly, M is said to be naïve, if its probability of attack does not depend on the environment it perceives, i.e., for every pair of possible environment types $\vec{m}, \vec{m}' \in \mathcal{M}$, it holds that $\vec{\rho}_{\vec{m}} = \vec{\rho}_{\vec{m}'}$. M is *deterministic* if M is restricted to strategies ρ, where for each $\vec{m} \in \mathcal{M}$, $\vec{\rho}_{\vec{m}} \in \{0,1\}$, i.e., in each environment M can either only always attack or always hide. Otherwise, M is *non-deterministic*. M is sophisticated when its strategy space is all of P.

We note that in our formulation, for AM and M, deterministic strategies and non-deterministic strategies are pure strategies. In other words, there are infinite pure strategies, each of which assigns a number between 0 and 1 to each machine type, and deterministic strategies simply mean that each component is either 0 or 1 but not in-between.

Utility Function. AM wants to protect machines it defends (but not all machines). AM's payoff (utility) is the expected fraction of real machines which it defends that are not attacked. M's payoff is the expected fraction of machines that are successfully attacked, regardless of whether the machine is defended by AM. Consequently, given mixed strategies π and ρ of AM and M respectively, we define M's utility $u_{\mathrm{M}}(\pi,\rho)$ and AM's utility $u_{\mathrm{AM}}(\pi,\rho)$ as shown in Figure 2. Notice that if M is caught in a sandbox, it is not allowed to execute on the real machine, and therefore cannot attack the real machine. Notice also that payoffs are bound between 0 and 1.

Solution Concept. We will focus on characterizing and computing the pure strategy subgame perfect equilibrium (SPNE) of the game. This will give us the optimal strategy for AM while M can perfectly respond to AM's strategies. Computing an AM optimal SPNE solution involves solving the following optimization problem:

$$\max_{\pi,\rho} u_{\mathrm{AM}}(\pi,\rho)$$
$$\text{such that } \rho \in \arg\max_{\hat{\rho}\in\mathrm{P}} u_{\mathrm{M}}(\pi,\hat{\rho})$$

Unfortunately, this involves solving multiple non-convex quadratically constrained quadratic programs which are generally NP-hard. To deal with this, we identify several tractable cases, and suggest practical algorithms for computing the solution otherwise.

Notation	Meaning
$\mathcal{M}$	The set of all possible types of environments.
$\vec{m} \in \mathcal{M}$	An environment in $\mathcal{M}$, which may be presented by either a real machine or a sandbox.
$\vec{r} \in \mathcal{M}$	An environment type in $\mathcal{M}$ presented by a real machine.
$\vec{e}$	$\lvert\mathcal{M}\rvert$-vector where $\vec{e}_{\vec{m}}$ denote the fraction of real machines of type $\vec{m} \in \mathcal{M}$.
$\vec{d}$	$\lvert\mathcal{M}\rvert$-vector where $\vec{d}_{\vec{m}}$ denotes the fraction of real machines of type $\vec{m} \in \mathcal{M}$ that are defended by AM.
D	Fraction of all real machines in existence that are defended AM.
$\vec{\pi}$	AM's strategy: maps each possible real machine environment $\vec{r}$ to a distribution over $\mathcal{M}$ of generating sandboxes.
$\vec{\pi}^{\vec{r}}_{\vec{m}}$	The probability that AM generates a sandbox of type $\vec{m}$ given a real machine of type $\vec{r}$.
$\vec{\rho}$	M's strategy: An $\lvert\mathcal{M}\rvert$-vector where $\vec{\rho}_{\vec{m}}$ denotes the probability that M attacks when presented an environment $\vec{m}$.
$u_{\text{AM}}, u_{\text{M}}$	Utility functions of AM and M.

Table 1: A summary of notation used frequently throughout this paper.

Natural Strategies. We will evaluate the AM-optimal strategies against some natural strategies as described in Table 2. For example, in the Existence strategy, AM generates a sandbox of type $\vec{m}$ with probability $\vec{\pi}_{\vec{m}} = \vec{e}_{\vec{m}}$, i.e., the fraction of type $\vec{m}$ machines.

Name	$\vec{\pi}_{\vec{m}}$	Distribution
Existence	$\vec{e}_{\vec{m}}$	Real machines
Defended	$\frac{\vec{d}_{\vec{m}}}{D}$	Defended machines
Undefended	$\frac{1-\vec{d}_{\vec{m}}}{1-D}$	Undefended machines
%Defended	$\frac{\vec{d}_{\vec{m}}}{\vec{e}_{\vec{m}}}$	$\propto$ % Defended
%Undefended	$\frac{1-\vec{d}_{\vec{m}}}{\vec{e}_{\vec{m}}}$	$\propto$ % Undefended
Majority	$\vec{\pi}_{\arg\max \vec{e}_{\vec{m}}} = 1$	Majority
Uniform	$\frac{1}{\lvert\mathcal{M}\rvert}$	Uniform

Table 2: Natural strategies for AM.

4 SPNE WHEN AM DEFENDS EVERY MACHINE

We first consider the case where AM defends all machines. In this case $\vec{d}_{\vec{m}} = \vec{e}_{\vec{m}}$ for all $\text{M} \in \mathcal{M}$. We will prove that Existence is the optimal strategy to generate sandboxes when AM defends all real machines when AM is naïve in Theorem 1. This involves randomly generating sandboxes according to the distribution $\vec{e}$, i.e., for each real machine $\vec{r} \in R$, and possible environment $\vec{m} \in \mathcal{M}$, $\vec{\pi}^{\vec{r}}_{\vec{m}} = \vec{e}_{\vec{m}}$. When AM can commit to a sophisticated strategy, the solution is even more simple and intuitive: *deterministically create a sandbox of the same type as the real machine being defended.* In equilibrium, AM is guaranteed a utility of 0.75, as we prove in Theorem 2.

Note that in the special case where M is naïve, AM's optimal strategy is any strategy that always creates a sandbox on any real machine. This is because a naïve M cannot distinguish between different types of machines and the game degenerates into a single-type model. The example below shows different cases where AM and a non-naïve M are restricted on different strategy spaces.

Example 1. The simple example of Table 3 illustrates the following natural message about the impact of modeling choices of the game on the payoff of AM. **In general, allowing AM more flexibility, or placing more restrictions on M leads to a higher payoff for AM in equilibrium.** At a high level, these roughly correspond to giving AM more resources and constraining the resources of M.

Table 3 illustrates an example of the various equilibrium strategies and payoffs for a setting with two machines A and B with $\vec{e}_{\vec{a}} = 0.4$ and $\vec{e}_{\vec{b}} = 0.6$. Specifically, the *Equilibrium Strategy* column shows the strategies, where its first column is $\vec{\pi}_{\vec{a}}$ for AM and $\vec{\rho}_{\vec{a}}$ for M; and its second column is $\vec{\pi}_{\vec{b}}$ for AM and $\vec{\rho}_{\vec{b}}$ for M. The exception is the fourth row where AM takes a sophisticated deterministic strategy. Here AM's strategy is to create a sandbox that emulates the real machine it is defending, by creating a sandbox of type A for all real machines A and a sandbox of type B for all real machines B.

When AM uses non-deterministic sandboxing strategies to fight against a deterministic-strategy-only M, AM's utility increases from 0.6 to 0.76 (first row vs. second row in Table 3). If M is also allowed to be non-deterministic, then AM's payoff in equilibrium reduces slightly to 0.75 (third row in Table 3). As another example, allowing AM to use sophisticated (but still deterministic) strategies to fight against M increases the utility from 0.6 to 0.75 (first row vs. fourth row in Table 3).

We consider the setting with two machines A and B represented by feature vectors $\vec{a}$ and $\vec{b}$ respectively. We start by restricting M's ability to be deterministic only.

Naïve Deterministic vs. Naïve Nondeterministic AM (first row vs. second row in Table 3). Allowing the flexibility of randomized sandbox generation increases AM's utility to 0.76 from 0.6 when AM is restricted to deterministic generation. Suppose AM assigns probability $\pi \in [0, 1]$ to emulate $\vec{a}$ and $1 - \pi$ to emulate $\vec{b}$. M's payoff becomes:

$$\max \underbrace{(0.4 \times (1-\pi)}_{\text{Always attack } \vec{a}}, \quad \underbrace{0.6 \times \pi)}_{\text{Always attack } \vec{b}}$$

Note that when M's strategy is to always attack $u_M = 0$ since AM always creates a sandbox, and that when M always does not attack $u_M = 0$ again. It is easy to see that u_M is minimized when $\pi = 0.4$, i.e., AM mimics the distribution of real machines. M is indifferent between always attacking $\vec{a}$ or always attacking $\vec{b}$, which results in a payoff of 0.76 to AM.

Remark: When AM is Naïve and deterministic and M is sophisticated, the best strategy for AM is Majority . Note that in this case AM's strategy is to pick a single type $\vec{m}$ and create a sandbox of type $\vec{m}$ on all real machines. Then, M's best response is attacking

Strategy space		Players	Equilibrium Strategy		Utility
AM	M		Machine A ($\vec{e}_{\vec{a}} = 0.4$)	Machine B ($\vec{e}_{\vec{b}} = 0.6$)	
Naïve deterministic	deterministic	AM	0	1	0.6
		M	1	0	0.4
Naïve	deterministic	AM	0.4	0.6	0.76
		M	1	0	0.24
Naïve	Sophisticated	AM	0.4	0.6	0.75
		M	0.5	0.5	0.25
Sophisticated deterministic	Sophisticated	AM	Emulate A	Emulate B	0.75
		M	0.5	0.5	0.25

Table 3: Example of the equilibrium strategies and payoffs for AM and M for two machines A and B with $\vec{e}_{\vec{a}} = 0.4$ and $\vec{e}_{\vec{b}} = 0.6$.

$\vec{m}$ with probability 0.5, and always attacking other types. Therefore, $u_M = 1 - 0.75\vec{e}_{\vec{m}}$ and $u_{AM} = 0.75\vec{e}_{\vec{m}}$. Therefore, AM's utility is maximized when it picks the type with the highest proportion to emulate as a sandbox environment, which is exactly the strategy Majority .

Sophisticated and non-deterministic M (third row vs. second row in Table 3). Suppose AM assigns probability $\pi \in [0, 1]$ to emulate $\vec{a}$ and $1 - \pi$ to emulate $\vec{b}$, and suppose M chooses to attack with probability ρ_A (respectively, ρ_B) in environment $\vec{a}$ (respectively, $\vec{b}$). M's payoff becomes

$$\underbrace{0.4 \times (\pi(1-\rho_A) + (1-x)(1-\rho_B)) \times \rho_A}_{\text{M succeeds on machine A}} + \underbrace{0.6 \times (\pi(1-\rho_A) + (1-\pi)(1-\rho_B)) \times \rho_B}_{\text{M succeeds on machine B}}. \quad (1)$$

While this formula seems hard to solve analytically, observe that no matter what π is, M's payoff is always 0.25 when it chooses $\rho_A = \rho_B = 0.5$. Also notice that when $\pi = 0.4$, Equation (1) becomes $(0.4\rho_A + 0.6\rho_B)(0.4(1-\rho_A) + 0.6(1-\rho_B))$, which is maximized at $\rho_A = \rho_B = 0.5$ giving the equilibrium shown in Table 3.

Sophisticated deterministic AM vs. Nondeterministic M. This is the last row in Table 3. Similarly to the naíve mixed vs. mixed case, M can choose $\rho_A = \rho_B = 0.5$ to guarantee a payoff of 0.25. AM's optimal strategy now becomes always emulating the machine being defended. ∎

Theorems 1 and 2 follow from the observation that (1) when AM defends all machines, we are faced with a zero-sum game, and (2) identifying an equilibrium under which AM achieves the highest possible utility under any equilibrium.

Theorem 1. *When AM is installed on every real machine, and AM is naïve, and can commit to mixed strategies, the natural strategy* Existence *, is optimal in equilibrium.*

Proof. When AM is naïve, it takes the same distribution for all types of machine $\vec{r} \in \mathcal{M}$. Therefore, we use $\vec{\pi}$ to denote AM's strategy. Then we can rewrite the utility of M $u_M(\pi, \rho)$ as the following formula: $u_M(\pi, \rho) = (\sum_{\vec{r}\in\mathcal{M}} \vec{e}_{\vec{r}}\vec{\rho}_{\vec{r}})(1 - \sum_{\vec{m}\in\mathcal{M}} \vec{\pi}_{\vec{m}}\vec{\rho}_{\vec{m}})$. Note that since all machines are defended, we have $\vec{d}_{\vec{r}} = \vec{e}_{\vec{r}}$ for all $\vec{r} \in \mathcal{M}$. We also have $u_{AM}(\pi, \rho) = 1 - u_M(\pi, \rho)$.

We begin by noting that for *any* AM strategy π, M can guarantee a utility of at least 0.25 when M uses a naïve strategy of attacking with probability 0.5, because $\sum_{\vec{r}\in\mathcal{M}} \vec{e}_{\vec{r}} = 1$, $\sum_{\vec{m}\in\mathcal{M}} \vec{\pi}_{\vec{m}} \le 1$ and M's utility is

$$u_M(\pi, \rho) = (0.5 \sum_{\vec{r}\in\mathcal{M}} \vec{e}_{\vec{r}})(1 - 0.5 \sum_{\vec{m}\in\mathcal{M}} \vec{\pi}_{\vec{m}}) \ge 0.5 \times (1 - 0.5) = 0.25.$$

Therefore, for any choice of π for AM, if M selects a best response, $u_{AM}(\pi, \rho) \le 0.75$.

Now, consider the case where AM adopts the strategy Existence π^* where $\vec{\pi}^*_{\vec{m}} = \vec{e}_{\vec{m}}$ for every $\vec{r} \in \mathcal{M}$. Then $u_M(\pi, \rho) = \big(\sum_{\vec{r}\in\mathcal{M}} \vec{e}_{\vec{r}}\vec{\rho}_{\vec{r}}\big)\big(1 - \sum_{\vec{m}\in\mathcal{M}} \vec{e}_{\vec{m}}\vec{\rho}_{\vec{m}}\big)$. In order to maximize u_M, we consider its derivative:

$$\frac{\partial u_M}{\partial \vec{\rho}_{\vec{r}}} = \vec{e}_{\vec{r}}\big(1 - 2 \sum_{\vec{m}\in\mathcal{M}} \vec{e}_{\vec{m}}\vec{\rho}_{\vec{m}}\big)$$

Note that when $\vec{\rho}_{\vec{m}} = 0.5$ for all $\vec{m} \in \mathcal{M}$ (M attacks every type with probability 0.5), $\frac{\partial u_M}{\partial \vec{\rho}_{\vec{r}}} = 0$ for every $\vec{m} \in \mathcal{M}$. and $u_M(\vec{\pi}, \vec{\rho})$ is maximized. Then we have $u_M = 0.25$ and $u_{AM} = 0.75$ which is the highest utility that AM can obtain in equilibrium. Therefore, Existence is an AM-optimal strategy in equilibrium. □

Theorem 2. *When AM is installed on every real machine, and AM is sophisticated, an AM optimal strategy in equilibrium is to emulate the current machine, i.e., $\pi(\vec{r}, \vec{r}) = 1$ for every real machine $\vec{r}$.*

Proof. When AM is sophisticated, $u_M(\pi, \rho) = \sum_{\vec{r}\in\mathcal{M}} \vec{e}_{\vec{r}}\vec{\rho}_{\vec{r}}\big(1 - \sum_{\vec{m}\in\mathcal{M}} \vec{\pi}^{\vec{r}}_{\vec{m}}\vec{\rho}_{\vec{m}}\big)$. First, we note again that if M uses a naïve strategy of attacking with probability 0.5, we have

$$u_M(\pi, \rho) = 0.5 \sum_{\vec{r}\in\mathcal{M}} \vec{e}_{\vec{r}}\big(1 - 0.5 \sum_{\vec{m}\in\mathcal{M}} \vec{\pi}^{\vec{r}}_{\vec{m}}\big) \ge 0.25 \sum_{\vec{r}\in\mathcal{M}} \vec{e}_{\vec{r}} = 0.25.$$

Therefore, M's utility is at least 0.25 for any AM strategy, and $u_{AM}(\pi, \rho) \le 0.75$ for any π whenever M takes a best response.

Then we only need to show that M attacking with probability 0.5 is the best response when AM set $\pi^{\vec{r}}_{\vec{r}} = 1$ for every real machine $\vec{r}$. Note that in this case, M's utility is $u_M(\pi, \rho) = \sum_{\vec{r}\in\mathcal{M}} \vec{e}_{\vec{r}}\vec{\rho}_{\vec{r}}\big(1 - \vec{\rho}_{\vec{r}}\big)$. We can easily find that this is maximized at $\vec{\rho}_{\vec{r}} = 0.5$ for every $\vec{r} \in \mathcal{M}$, and $u_M = 0.25$. Therefore, $u_{AM} = 0.75$ when AM set $\vec{\pi}^{\vec{r}}_{\vec{r}} = 1$, which is the maximum AM can achieve. Therefore, we conclude that $\vec{\pi}^{\vec{r}}_{\vec{r}} = 1$ is an optimal strategy for AM in equilibrium. □

5 COMPUTING SPNE WHEN THERE ARE UNDEFENDED MACHINES

In this section, we consider the case where AM defends at most half the real machines, there is an AM-optimal strategy where

every machine that AM defends is protected in equilibrium, and Undefended is one such strategy.

Theorem 3. *When at most half of all machines are defended ($D \le \frac{1}{2}$), there exists an equilibrium solution where AM protects every defended machine (AM utility is D), and M always attacks. Specifically,* Undefended *is an AM-optimal strategy in equilibrium, for which M's best response is to always attack.*

Proof. We start by deriving the utility M receives if it always attacks. We start by noting that given AM's naïve strategy π, M evades $D(1-\sum_{\vec{m}\in\mathcal{M}}\vec{\pi}_{\vec{m}}\vec{\rho}_{\vec{m}})$ sandboxes in expectation. However, M does not derive utility from machines on which it bypasses the sandbox but does not attack on the real machine which occurs with expectation $(1-\sum_{\vec{m}\in\mathcal{M}}\vec{\pi}_{\vec{m}}\vec{\rho}_{\vec{m}})\sum_{\vec{r}\in\mathcal{M}}\vec{d}_{\vec{r}}(1-\vec{\rho}_{\vec{r}})$. Thus, we rewrite M's utility as: $u_{\mathrm{M}} = \sum_{\vec{m}\in\mathcal{M}}\left(1-\vec{d}_{\vec{m}}\right)\vec{\rho}_{\vec{m}} + D(1-\sum_{\vec{m}\in\mathcal{M}}\vec{\pi}_{\vec{m}}\vec{\rho}_{\vec{m}}) - (1-\sum_{\vec{m}\in\mathcal{M}}\vec{\pi}_{\vec{m}}\vec{\rho}_{\vec{m}})\sum_{\vec{r}\in\mathcal{M}}\vec{d}_{\vec{r}}(1-\vec{\rho}_{\vec{r}}) = \sum_{\vec{m}\in\mathcal{M}}\vec{\rho}_{\vec{m}}(1-\vec{d}_{\vec{m}} - D\vec{\pi}_{\vec{m}}) + D - (1-\sum_{\vec{m}\in\mathcal{M}}\vec{\pi}_{\vec{m}}\vec{\rho}_{\vec{m}})\sum_{\vec{r}\in\mathcal{M}}\vec{d}_{\vec{r}}(1-\vec{\rho}_{\vec{r}})$.

Note that whenever $D\vec{\pi}_{\vec{m}} \le \left(1-\vec{d}_{\vec{m}}\right)$ for every $\vec{m}\in\mathcal{M}$, we have that u_M is maximized when $\vec{\rho}_{\vec{m}} = 1$ for every $\vec{m}\in\mathcal{M}$. This is because $\vec{\rho}_{\vec{m}}\in[0,1]$ for every $\vec{m}\in\mathcal{M}$, and $D\vec{\pi}_{\vec{m}} - (1-\vec{d}_{\vec{m}}) \le 0$ for every $\vec{m}\in\mathcal{M}$. Notice that when M always attacks, AM gets utility D when it always creates a sandbox, which is the maximum possible utility AM can get, and therefore optimal.

It is easy to see that when AM uses the natural strategy Undefended , i.e., setting $\vec{\pi}_{\vec{m}} = \frac{1-\vec{d}_{\vec{m}}}{1-D}$, the condition $D\vec{\pi}_{\vec{m}} \le \left(1-\vec{d}_{\vec{m}}\right)$ is satisfied for every $\vec{m}$, and that AM creates a sandbox on every defended real machine, giving AM a utility of D in equilibrium. Therefore, Undefended is an optimal strategy. □

Example 2. We now consider a world with three types of machines A, B, and C, with features $\vec{a}, \vec{b}$, and $\vec{c}$, respectively. 10% of real machines are of type $\vec{a}$, 20% of type $\vec{b}$, and the rest are of type $\vec{c}$. AM is installed on 70% of machines of type $\vec{a}$ and type $\vec{b}$, and 30% machines of type $\vec{c}$.

When AM's strategy is Undefended , M's best response is to always attack, allowing AM to protect 100% of machines it defends. In comparison, AM's utility decreases to protecting only 92.2% of machines it defends, if it uses the Existence strategy. This is in sharp contrast to our findings for AM optimal solutions when AM defends all machines that we observed in Section 4.

Here, we give a detailed analysis of M's utility when AM uses the strategy Existence . We list the relevant values in Table 4.

Type $\vec{r}$	$\vec{e}_{\vec{r}} = \vec{\pi}_{\vec{r}}$	$\vec{d}_{\vec{r}}$	$1-\vec{d}_{\vec{r}}$	$D\vec{\pi}_{\vec{r}}$
A	0.1	0.07	0.03	0.42
B	0.2	0.14	0.06	0.84
C	0.7	0.49	0.49	0.294
Total	1	$D = 0.42$	$1-D = 0.58$	

Table 4: Values for computing u_M in Example 2 when AM's strategy is Existence **.**

When AM uses Existence , $\vec{\pi}_{\vec{r}} = \vec{e}_{\vec{r}}$ for every $\vec{r}\in\mathcal{M}$. Note that for A and B, we have $1-\vec{d}_{\vec{a}} < D\vec{\pi}_{\vec{a}}$ and $1-\vec{d}_{\vec{b}} < D\vec{\pi}_{\vec{b}}$. On the other hand, we have $1-\vec{d}_{\vec{c}} > D\vec{\pi}_{\vec{c}}$. As we show below, in the best response strategy $\vec{\rho}$ for M, $\vec{\rho}_{\vec{a}}$ and $\vec{\rho}_{\vec{b}}$ do not equal to 1, while $\vec{\rho}_{\vec{c}} = 1$.

We first compute the derivative of u_M based on the formula used in the proof of Theorem 3.

$$\frac{\partial u_M}{\partial\vec{\rho}_{\vec{m}}} = \left(1-\vec{d}_{\vec{m}} - D\vec{\pi}_{\vec{m}}\right) + \vec{d}_{\vec{m}}(1-\sum_{\vec{r}\in\mathcal{M}}\vec{\pi}_{\vec{r}}\vec{\rho}_{\vec{r}}) + \vec{\pi}_{\vec{m}}\sum_{\vec{r}\in\mathcal{M}}\vec{d}_{\vec{r}}(1-\vec{\rho}_{\vec{r}})$$
$$= \vec{e}_{\vec{m}} - \vec{d}_{\vec{m}}\sum_{\vec{r}\in\mathcal{M}}\vec{\pi}_{\vec{r}}\vec{\rho}_{\vec{r}} - \vec{\pi}_{\vec{m}}\sum_{\vec{r}\in\mathcal{M}}\vec{d}_{\vec{r}}\vec{\rho}_{\vec{r}}.$$

Substituting the values in Table 4, we get:

$$\frac{\partial u_M}{\partial\vec{\rho}_{\vec{a}}} = 0.1 - 0.014\vec{\rho}_{\vec{a}} - 0.028\vec{\rho}_{\vec{b}} - 0.07\vec{\rho}_{\vec{c}}$$
$$\frac{\partial u_M}{\partial\vec{\rho}_{\vec{b}}} = 0.2 - 0.028\vec{\rho}_{\vec{a}} - 0.056\vec{\rho}_{\vec{b}} - 0.14\vec{\rho}_{\vec{c}}$$
$$\frac{\partial u_M}{\partial\vec{\rho}_{\vec{c}}} = 0.7 - 0.07\vec{\rho}_{\vec{a}} - 0.14\vec{\rho}_{\vec{b}} - 0.294\vec{\rho}_{\vec{c}}$$

Note that $\frac{\partial u_M}{\partial\vec{\rho}_{\vec{c}}}$ is always positive. Therefore $\vec{\rho}_{\vec{c}} = 1$, and M always attacks on $\vec{c}$. On the other hand, $\frac{\partial u_M}{\partial\vec{\rho}_{\vec{a}}}$ and $\frac{\partial u_M}{\partial\vec{\rho}_{\vec{b}}}$ equals to zero simultaneously when $0.014\vec{\rho}_{\vec{a}} + 0.028\vec{\rho}_{\vec{b}} = 0.03$. (Note that $\frac{\partial u_M}{\partial\vec{\rho}_{\vec{b}}} = 2\frac{\partial u_M}{\partial\vec{\rho}_{\vec{a}}}$). Therefore, $\vec{\rho}_{\vec{a}} = \vec{\rho}_{\vec{b}} = \frac{5}{7}$, and $(\frac{5}{7}, \frac{5}{7}, 1)$ is an optimal strategy for M.

Next, we consider the fraction of defended machines AM protects. Note that due to M's optimal strategy, M can only bypass the sandbox when AM generates either a type A or B sandbox. This means M can bypass the sandbox with probability $\frac{2}{7}$. Then, the probability that a defended machine is attacked by M is:

$$0.3\times\frac{2}{7}\times(0.3\times\frac{5}{7}+0.7\times1)\approx0.078$$

Therefore, when AM adopts the Existence strategy, it successfully protects only 92.2% of machines it defends. ■

Impact. Theorem 3 raises interesting possibilities. One such possibility is to cheaply create sufficiently many virtual or dummy machines to manipulate the setting so that $D \le \frac{1}{2}$, allowing us to trade off some computational resources for complete protection. At a high level, the idea is similar to honeypots in other cybersecurity problems.

6 COMPUTING M'S BEST RESPONSE

In this paragraph we provide a method to compute M's best response given a fixed AM strategy. This will be the basis of our QCQP framework. Before we proceed further, we rewrite the utility functions of AM and M in vector notation for convenience:

$$u_{\mathrm{M}}(\vec{\pi},\vec{\rho}) = \underbrace{(\vec{1}-\vec{d})\cdot\vec{\rho}}_{\text{M attacks | No AM}} + \underbrace{\vec{d}\cdot\vec{\rho} - \vec{d}\cdot((\Pi\cdot\vec{\rho})\odot\vec{\rho})}_{\text{M not caught, attacks | AM installed}}$$
$$= \vec{e}\cdot\vec{\rho} - \vec{d}\cdot((\Pi\cdot\vec{\rho})\odot\vec{\rho})$$
$$u_{\mathrm{AM}}(\vec{\pi},\vec{\rho}) = \vec{d}\cdot(1-\vec{\rho}) + \vec{d}\cdot((\Pi\cdot\vec{\rho})\odot\vec{\rho})$$

where $\odot$ is the component-wise multiplication operator between two vectors, and Π is the matrix of AM's strategies, whose $\vec{r}$-th row-vector is $\vec{\pi}^{\vec{r}}$. For a fixed AM strategy π, we can solve for M's best response ρ^* using the Lagrange multipliers for the optimization

AM		M		AM-optimal strategy
Randomized	Sophisticated	Randomized	Sophisticated	
*	*	*	No	Always creating a sandbox
No	No	*	Yes	Majority
*	Yes	*	Yes	Emulate machine being defended
Yes	No	*	Yes	Existence

Table 5: Guidelines when AM is installed on every real machine. $*$ indicates that added flexibility has no affect.

problem $\max_{\vec{\rho}} u_M(\vec{\pi}, \vec{\rho})$ which are: $\forall \vec{m} \in \mathcal{M}, L_{\vec{m}} = \frac{\partial u_{\mathrm{M}}(\vec{\pi},\vec{\rho})}{\partial \vec{\rho}_{\vec{m}}} = \vec{e}_{\vec{m}} - 2\vec{d}_{\vec{m}}\vec{\pi}_{\vec{m}}\vec{\rho}_{\vec{m}} - \sum_{\vec{l}\in\mathcal{M}\setminus\{\vec{m}\}}(\vec{d}_{\vec{m}}\vec{\pi}_{\vec{l}} + \vec{d}_{\vec{l}}\vec{\pi}_{\vec{m}})\vec{\rho}_{\vec{l}}$.

Computing $\vec{\rho}^*$ involves computing the solutions to the following $3^{|\mathcal{M}|}-1$ systems of linear equations, and picking the solution that is feasible and maximizes M's utility. Each system of linear equations is indexed by an $|\mathcal{M}|$-vector $\vec{c} \in \{0, 1, b\}^{|\mathcal{M}|}$ (except $\vec{c} = \vec{0}$), where b means "between 0 and 1", and is constructed by adding the $|\mathcal{M}|$ equations as follows: (i) $\vec{\rho}_{\vec{m}} = 0$ if $\vec{c}_{\vec{m}} = 0$, (ii) $\vec{\rho}_{\vec{m}} = 1$, if $\vec{c}_{\vec{m}} = 1$, or (iii) $L_{\vec{m}} = 0$ if $\vec{c}_{\vec{m}} = [0, 1]$.

7 QCQP FRAMEWORK FOR COMPUTING AM-OPTIMAL SPNE

We now shift our focus to settings where $D > \frac{1}{2}$. Algorithm 1 provides a general quadratically constrained quadratic program (QCQP) formulation of the problem of computing AM-optimal SPNEs. We simultaneously solve for AM's strategy π^* and M's strategy ρ^* by setting u_{AM} to be the objective, under the constraint that ρ^* is the best response to the output strategy for AM π^*.

Algorithm 1 QCQP to compute AM-optimal SPNE.

1: **Input:** A real world setting $\vec{e}, \vec{d}$.
2: **Output:** SPNE π^*, ρ^*.
3: An empty set of SPNE solutions S.
4: **for** each $\vec{c} \in \{0, 1, b\}^{|\mathcal{M}|}$ **do**
5: Create a QCQP problem P with variables $\vec{\rho}, \vec{\pi}$.
6: Set the objective as $\max_{\vec{\pi},\vec{\rho}} u_{\mathrm{AM}}(\vec{\pi}, \vec{\rho})$.
7: **for** each $\vec{m}$, $\vec{c}_{\vec{m}}$ add the binding constraints on $\vec{\rho}_{\vec{m}}$ **do**
8: if $\vec{c}_{\vec{m}} = 0$, add the constraint $\vec{\rho}_{\vec{m}} = 0$.
9: if $\vec{c}_{\vec{m}} = 1$, add the constraint $\vec{\rho}_{\vec{m}} = 1$.
10: if $\vec{c}_{\vec{m}} = b$, add $\vec{\rho}_{\vec{m}} \in [0, 1]$, and $L_{\vec{m}} = 0$.
11: Feasibility constraints $\vec{\pi}_{\vec{m}} \in [0, 1], \forall \vec{m}$, and $\vec{\pi}.\vec{1} \leq 1$.
12: Compute the solution $\vec{\pi}, \vec{\rho}$.
13: Test feasibility and constraint violations.
14: Fix $\vec{\pi}$, and compute M's best response $\vec{\rho}'$ to $\vec{\pi}$.
15: **if** $u_{\mathrm{M}}(\vec{\pi}, \vec{\rho}) \geq u_{\mathrm{M}}(\vec{\pi}, \vec{\rho}')$ **then**
16: Add $\vec{\pi}, \vec{\rho}$ as an SPNE to S.
return Return the SPNE from S with the highest AM utility.

Algorithm 1 involves (1) enumerating M's possible responses represented by the possible values that $\vec{c}$ (line 7-10) can take on. For each $\vec{c}$, the algorithm solves the QCQP problem to compute an optimal AM strategy $\vec{\pi}$ in equilibrium and the corresponding best response strategy $\vec{\rho}$ of M strategy (line 12), under the constraints $\vec{c}$. Notice that here, $\vec{\pi}$ is an optimal AM strategy in equilibrium when M is constrained by $\vec{c}$. In line 13, the algorithm tests constraint violations and discards the solution if some constraints are violated. This test is necessary because of limitations of current QCQP solvers. In lines 14-16, we fix the AM strategy to be $\vec{\pi}$, and compute $\vec{\rho}'$ which is M's best response to $\vec{\pi}$ without any constraints on M, using the technique described in Section 6, and test if $u_{\mathrm{M}}(\vec{\pi}, \vec{\rho}) \geq u_{\mathrm{M}}(\vec{\pi}, \vec{\rho}')$. If the inequality holds, the constrained best response $\vec{\rho}$ is also M's global best response, and $(\vec{\pi}, \vec{\rho})$ is an equilibrium, and added to S. If the inequality does not hold, the algorithm discards the solution. Finally, the algorithm chooses the equilibrium with the highest AM utility from S.

When AM is installed only on machines of a single type, we can solve for AM's optimal strategy efficiently as follows. The algorithm follows from the observation that u_{AM} becomes a linear function in $\vec{\pi}$, with no critical points in the interval $[0, 1]$. Thus we pick the solution which maximizes AM's utility from solving the $3^{|\mathcal{M}|} - 1$ sets of equations for every combination of setting $\vec{\pi}_{\vec{m}}$'s to either 0 or 1 and the corresponding Lagrangian first order conditions on M's strategy.

8 GUIDELINES FOR AM

Based on Theorems 1, 2 and 3, we refer to a setting as *easy*, if either AM defends all machines, or it defends at most half of all machines. In these cases, we have analytical solutions to AM optimal SPNE. We refer to other settings as *hard*, and solve them using our QCQP-based Algorithm 1.

Here, we summarize our findings and provide guidelines for sandboxing AMs. Specifically, we answer the following questions through experiments: *Are there natural and easy to compute strategies for AM, assuming M best responds, without compromising utility when compared to the optimal strategy?* We consider some natural strategies which are summarized in Table 2.

Guidelines for Easy Settings.

- When AM defends every real machine, Table 5 summarizes the AM-optimal strategies under various combinations of restrictions on strategy spaces of AM and M.
- When AM defends less than half of all real machines, AM should use the Undefended strategy.
- When AM defends a single type of real machine, AM should use the algorithm in Section 7 to compute an optimal strategy in equilibrium.

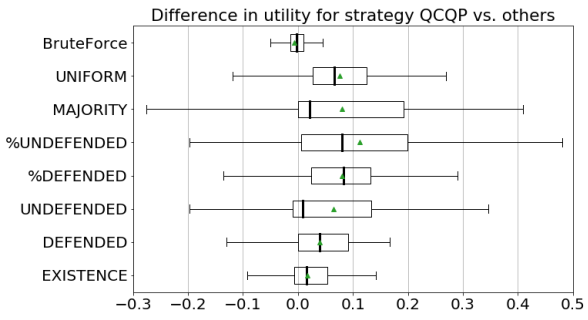
For hard settings, the QCQP SPNE computed using Algorithm 1 and the Existence strategy yield AM utility close to that of the BruteForce SPNE on average. Defended also performs consistently well in practice, yielding AM utility close to both QCQP and Existence strategies on average.

8.1 Experimental Setup

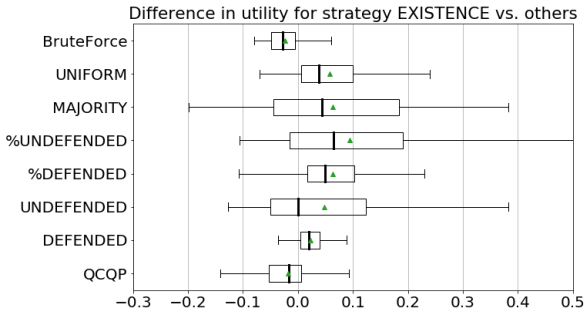
To evaluate various sandboxing strategies for hard settings, we create a dataset of 1000 settings by generating settings involving machines of two types A, B i.i.d. and retaining only the hard settings. For each hard setting, we compute AM's utility in the QCQP SPNE solution, a solution computed using brute force search which we call BruteForce, as well as AM's utility when AM plays each of the natural strategies in Table 2 and M best responds.

We use the QCQP[1] [14] extension of the CVXPY package using the suggest and improve method. For each subproblem, we compute 10 solutions with random initial suggestions and pick the solution with the highest AM utility. Solutions are computed using the alternating direction method of multipliers (ADMM). We discard solutions with large constraint violations (Line 13 in Algorithm 1). Then we fix AM's strategy and compute M's best response using the Lagrangian first order conditions as discussed in Section 6. We only retain solutions for which M's utility using the QCQP solution is within 0.01 of M's utility using the best response (Line 14-16 in Algorithm 1).

8.2 Experimental Results



(a) AM utility of QCQP solution against natural AM strategies.



(b) AM utility with real strategy against other AM strategies.

Figure 3: Difference in AM utility using QCQP and Existence vs other natural strategies, and benchmarked against BruteForce.

In Figure 3, we report the differences between AM's utilities using the QCQP SPNE solution QCQP SPNE solution (Figure 3 (a)), and when AM plays the Existence strategy and M best responds (Figure 3 (b)), respectively against the AM utilities obtained when AM uses one of the other natural strategies, as well as AM utility in the SPNE computed using BruteForce as a benchmark. BruteForce was obtained by discretizing AM and M's strategy spaces in 0.01 intervals.

[1]https://github.com/cvxgrp/qcqp

Guidelines for Hard Settings.

- Existence yields AM utility close to BruteForce strategy when M best responds.
- On average AM utility with both Existence and QCQP strategies are close to AM utility in the BruteForce SPNE. Among all natural strategies, Existence consistently outperforms or matches other natural strategies, and even beats QCQP strategies on 31.023% of simulated settings.
- On average, both QCQP and Existence strategies are close to each other in terms of AM utility with some exceptions where the solver failed to find a feasible solution that did not violate the first order conditions. The two methods complement each other, as shown in Figure 3, where we observe that they outperform each other depending on the setting.
- If AM does not know about the distribution $\vec{e}$ of real machines, sandboxing according to Defended is a viable alternative and works well on average.

9 CONCLUSIONS AND FUTURE WORK

We provided the first game theoretic analysis of the sandbox game and the first theoretical results and practical algorithms for sandboxing. Specifically, our results provide concrete guidelines for deploying sandboxing AMs, allowing an AM developer to compute AM-optimal strategies under several natural restrictions on the strategy space of AM and M which correspond to different practical considerations in the deployment of AM and M. When AM either defends every machine or defends fewer than half of all real machines, we identify natural and easy to compute strategies that are optimal for AM in equilibrium. The problem of computing an optimal AM strategy becomes harder when AM defends more than half of the real machines but not all of them. Our QCQP algorithm compute an optimal AM strategy but is computationally expensive. However, as we show empirically, the natural and easy to compute Existence strategy achieves AM utility that is close to optimal in practice.

There are several exciting avenues for future work: selecting AM strategies that are robust to M's selection of strategies in response to AM, and modeling the resource constraints as AM's ability to generate sandboxes, or M's inability to perfectly observe AM. Indeed, AM and M may often have different levels of information about a given deployment or constraints on the computational resources at their disposal. For example, commercially distributed AM may not be aware of the exact distribution of real machines where it may be deployed, and therefore be forced to commit to a naïve strategy. While our results already identify some natural and easy to compute AM-optimal solutions, fully exploring the impact of different constraints of information and computational resources on AM and M, and how to compute effective strategies for AM are an interesting question for future work.

REFERENCES

[1] Davide Balzarotti, Marco Cova, Christoph Karlberger, Christopher Kruegel, Engin Kirda, and Giovanni Vigna. 2010. Efficient Detection of Split Personalities in Malware. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. San Diego, CA.

[2] Jeremy Blackthorne, Benjamin Kaiser, Benjamin Fuller, and Bülent Yener. 2017. Environmental Authentication in Malware. In *Proceedings of the Fifth International Conference on Cryptology and Information Security in Latin America (Latincrypt)*.

[3] Jeremy Blackthorne, Benjamin Kaiser, and Bülent Yener. 2016. A formal framework for environmentally sensitive malware. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 9854 LNCS. 211–229.

[4] Jeremiah Blocki, Nicolas Christin, Anupam Datta, Ariel D Procaccia, and Arunesh Sinha. 2013. Audit Games. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

[5] Jeremiah Blocki, Nicolas Christin, Anupam Datta, Ariel D Procaccia, and Arunesh Sinha. 2015. Audit Games with Multiple Defender Resources. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[6] Rodrigo Rubira Branco, Gabriel Negreira Barbosa, and Pedro Drimel Neto. 2012. Scientific but Not Academical Overview of Malware Anti-Debugging, Anti-Disassembly and Anti-VM Technologies. In *Black Hat 2012*.

[7] Alexei Bulazel and Bulent Yener. 2017. A Survey On Automated Dynamic Malware Analysis Evasion and Counter-Evasion. In *In Proceedings of Reversing and Offensive-oriented Trends Symposium*. ACM.

[8] Zhui Deng, Xiangyu Zhang, and Dongyan Xu. 2013. SPIDER: Stealthy Binary Program Instrumentation and Debugging via Hardware Virtualization. In *Proceedings of the 29th Annual Computer Security Applications Conference (ACSAC '13)*. ACM, New York, NY, USA, 289–298.

[9] Artem Dinaburg, Paul Royal, Monirul Sharif, and Wenke Lee. 2008. Ether: malware analysis via hardware virtualization extensions. In *CCS '08 Proceedings of the 15th ACM conference on Computer and communications security*. 51–62.

[10] Cuong T Do, Nguyen H Tran, Choongseon Hong, Charles A Kamhoua, Kevin A Kwiat, Erik Blasch, Shaolei Ren, Niki Pissinou, and Sundaraja Sitharama Iyengar. 2017. Game Theory for Cyber Security and Privacy. *ACM Computing Surveys (CSUR)* 50, 2 (2017), 30.

[11] Karel Durkota, Viliam Lisy, Christofer Kiekintveld, and Branislav Bosansky. 2015. Game-theoretic algorithms for optimal network security hardening using attack graphs. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1773–1774.

[12] Min Gyung Kang, Heng Yin, Steve Hanna, Stephen McCamant, and Dawn Song. 2009. Emulating Emulation-Resistant Malware. In *VMSec '09 Proceedings of the 1st ACM workshop on Virtual machine security*. 11–22.

[13] Phillip Lee, Andrew Clark, Basel Alomair, Linda Bushnell, and Radha Poovendran. 2015. A host takeover game model for competing malware. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*. IEEE, 4523–4530.

[14] Jaehyun Park and Stephen Boyd. 2017. General heuristics for nonconvex quadratically constrained quadratic programming. *arXiv preprint arXiv:1703.07870* (2017).

[15] Thomas Raffetseder, Christopher Kruegel, and Engin Kirda. 2007. Detecting System Emulators. In *ISC'07 Proceedings of the 10th International Conference on Information Security*. 1–18.

[16] Sankardas Roy, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, Vivek Shandilya, and Qishi Wu. 2010. A survey of game theory as applied to network security. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*. IEEE, 1–10.

[17] Anshuman Singh, Arun Lakhotia, and Andrew Walenstein. 2010. Malware antimalware games. In *International Conference on Cyber Warfare and Security*. Academic Conferences International Limited, 319.

[18] Anshuman Singh, Bin Mai, Arun Lakhotia, and Andrew Walenstein. 2009. On optimal AV system strategies against obfuscated malware. In *4th Annual Symposium on Information Assurance (ASIA'09)*. 30.

[19] Arunesh Sinha, Thanh H Nguyen, Debarun Kar, Matthew Brown, Milind Tambe, and Albert Xin Jiang. 2015. From physical security to cybersecurity. *Journal of Cybersecurity* 1, 1 (2015), 19–35.

[20] Milind Tambe. 2011. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press.

[21] Satya Gautam Vadlamudi, Sailik Sengupta, Marthony Taguinod, Ziming Zhao, Adam Doupé, Gail-Joon Ahn, and Subbarao Kambhampati. 2016. Moving target defense for web applications using bayesian stackelberg games. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1377–1378.

[22] Ondřej Vaněk, Zhengyu Yin, Manish Jain, Branislav Bošanskỳ, Milind Tambe, and Michal Pěchouček. 2012. Game-theoretic resource allocation for malicious packet detection in computer networks. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 905–912.