Case Studies of Network Defense with Attack Graph Games

Karel Durkota[1], Viliam Lisý[1,2], Christopher Kiekintveld[3], Branislav Bošanský[1], Michal Pěchouček[1]
[1]Artificial Intelligence Center, Dept. of Computer Science, FEE, Czech Technical University in Prague
{durkota,lisy,bosansky,pechoucek}@agents.fel.cvut.cz
[2]AICML, Dept. of Computing Science, University of Alberta
[3]Computer Science Department, University of Texas at El Paso
cdkiekintveld@utep.edu

***Abstract*—The increasing complexity of securing modern computer networks makes decision support systems an important tool for administrators. A challenge many existing tools fail to address is that attackers react strategically to new security measures, adapting their behavior in response. Game theory provides a methodology for making decisions that take into account these reactions, rather than assuming static attackers. We present an overview of how game theory can be used to inform one type of security decision: how to optimally place honeypots in a network. We demonstrate this approach on a realistic case study, and present initial validation results based on a study comparing our approach with human decision makers.**

***Index Terms*—Game theory, Network security, Attack graph**

I. Introduction

Computer network security is an example of asymmetric, strategic conflict between defenders and attackers. Attackers perform a wide range of intrusion actions such as scanning the network and exploiting vulnerabilities, while defenders counter with actions such as intrusion detection and filtering. Different defense mechanisms have varying costs and effectiveness against specific types of attacks. Network administrators face challenging decisions in how to assess the effectiveness of security measures, and optimize how these measures are selected and deployed. The initial effectiveness may be mitigated in the long term, as attackers adapt to the security measure, which makes the optimization even more challenging.

Game theory provides a methodology for developing new decision support tools that take into account the sophisticated responses of attackers to a defense strategy. The key idea is that rational attackers will respond to security, so we can model the network defense problem as a two-player, multi-stage game. Solving these models allows us to find an optimal defense plan to mitigate attacks, and can also provide a quantitative measure of the improvement in security. We demonstrate this methodology for one specific type of defensive strategy: deploying *honeypots*, which are fake hosts or services added to the network. Honeypots act as decoys to distract attackers from the real hosts, detect the presence of intruders, and gather detailed information about the attacker's activity.

Operating believable honeypots is expensive in terms of hardware, software and administrator time for managing the honeypot and analyzing data. In addition, there are many different types of honeypots that could be used in any given network. We describe a game-theoretic approach for optimizing honeypot deployments as a case study for how game theory can be used to make network security decisions. We present a case study on a realistic network to show the feasibility of this methodology, and then present experimental results from an initial validation study with human decision makers to evaluate the game-theoretic strategies.

II. Background

Honeypots are used by network administrators and security researchers to detect and analyze attackers' behavior and the tools they use [7]. For example, the Honeynet Project provides software and literature describing the knowledge acquired about attackers. Companies use honeypots as intrusion detection systems (IDS), e.g., a hardware product called *Canary* by Thinkst can emulate various operating systems. To effectively use honeypots, network administrators must decide how many to deploy (contingent on the cost), and where to place them to make them attractive targets.

A. Game Theory

Game theory models decision-making problems with multiple decision makers (*players*) in a common, often partially observable, environment. A game consists of (i) *players*, (ii) *strategies* (actions) available to each player, and (iii) *utilities* for each defender depending on the joint choices of all players. The players may have (iv) incomplete information about moves made by other players or the environment. The optimal strategy for a player generally depends on the behavior of the other players. Game theory provides a variety of solution concepts and algorithms for analyzing games with different characteristics.

We focus on two-player games where the administrator (*defender*) chooses a honeypot allocation minimizing cost and the expected loss caused by an attacker compromising the network. The *attacker* chooses a strategy that maximizes the value of attacking the network while avoiding honeypots and minimizing costs. We use Stackelberg game models similar to those used for physical security domains [8]. The defender acts

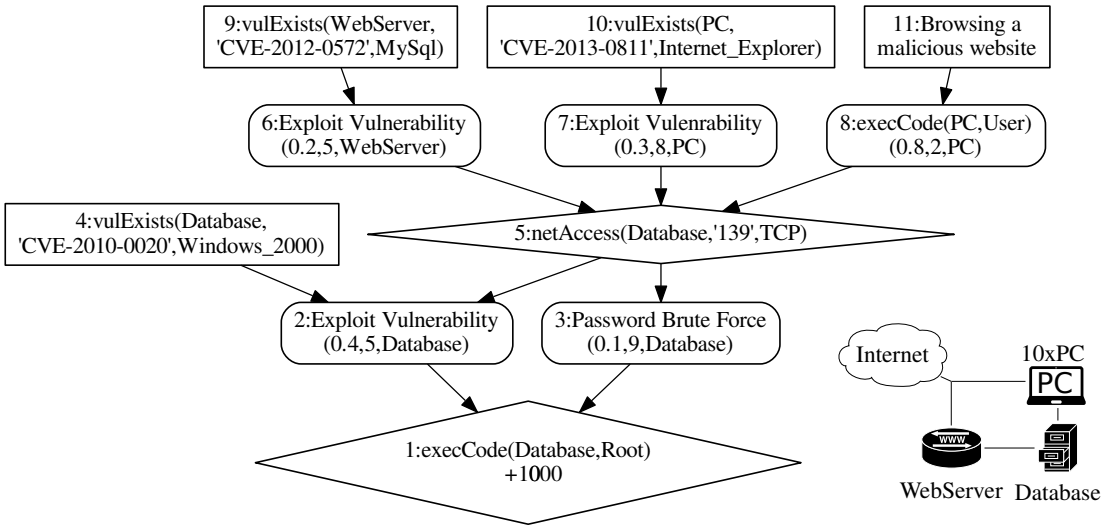

Fig. 1: A network and an attack graph representing ways to gain access to the Database.

first, taking actions to harden the network by adding honeypots. The attacker then chooses the optimal attack plan based on (limited) knowledge about the network and defender's strategy. Solving the game means computing a strategy for each player, which describes an optimal (stochastic) action choice in every possible situation.

B. Attack Graph

Attack graphs represent possible sequential attacker strategies for compromising a specific computer network. They are automatically generated based on known vulnerabilities [5] and are used to identify the minimal subset of vulnerabilities/sensors to be fixed/placed to prevent known attacks, to calculate security risk measures [9], or to find the shortest attack plan in penetration testing. We use AGs to compactly represent possible attack plans (attacker strategies) in our game models.

III. Model Overview

We model a network as a set of *host-types*, e.g., WebServer or PC in Figure 1. Two hosts are of the same host-type if they run the same services, have the same network connectivity, and the same value. For example, a collection of workstations that run the same OS image are modeled as the same type. Host-type PC in Figure 1 has ten equivalent hosts, all of which present the same attack opportunities. By representing each type only once in the attack graph we can scale with the number of unique types rather than individual hosts.

A. Defending

The defender places k honeypots into the network by duplicating the configurations of existing hosts (with obfuscated data) or creating new host-types. The defender pays a cost dependent on the host-type for each honeypot. Adding more honeypots of a specific host-type increases the likelihood that the attacker will interact with a honeypot instead of a real host. If the attacker interacts with a honeypot during an attack, an alert is sent to the defender who immediately stops the attack or applies other countermeasures.

B. Attacking

We consider *exploit* actions that target a specific vulnerability in a host. Successfully executing an exploit results in the attacker gaining *privileged* or *non-privileged access* to that host.

An attack graph (AG) compactly represents all known sequences of exploit actions for the host in the network. It captures the dependencies between the exploit *actions* and true/false *facts* that represent logical statement about the network state. Figure 1 depicts a possible AG for the network. Each action (rounded rectangular node) has preconditions and effects, represented by incoming and outgoing edges, respectively. All preconditions must be true to perform the action. If an exploit action succeeds, then all its effects become true and the attacker obtains rewards. Initially true facts are represented with rectangle nodes, and initially false facts with diamond nodes.

In Figure 1, action "2:Exploit Vulnerability" has preconditions: (Fact #4) vulnerability exists and (Fact #5) the attacker can access the Database on port 139. If he successfully performs the action, he will obtain root-privileges to the Database (Fact #1) and reward +1,000. The *success probability* is the likelihood that an exploit will succeed *given* that all preconditions are met. The *cost* represents the attacker's monetary cost and effort for attempt to perform an action and the consequences of possibly disclosing the exploit. Action #2 in Figure 1 has success probability 0.4 and cost 5.

We use MulVAL [5] to automatically construct AGs from information collected by network scanning tools (i.e., Nessus or OpenVAS). The action costs and success probabilities can be estimated using the Common Vulnerability Scoring System [4] or other data sources.

In our game models, the attacker chooses the optimal attack policy that fully characterizes the attacker's behavior at every point during the attack. It specifies the *order* of the actions to

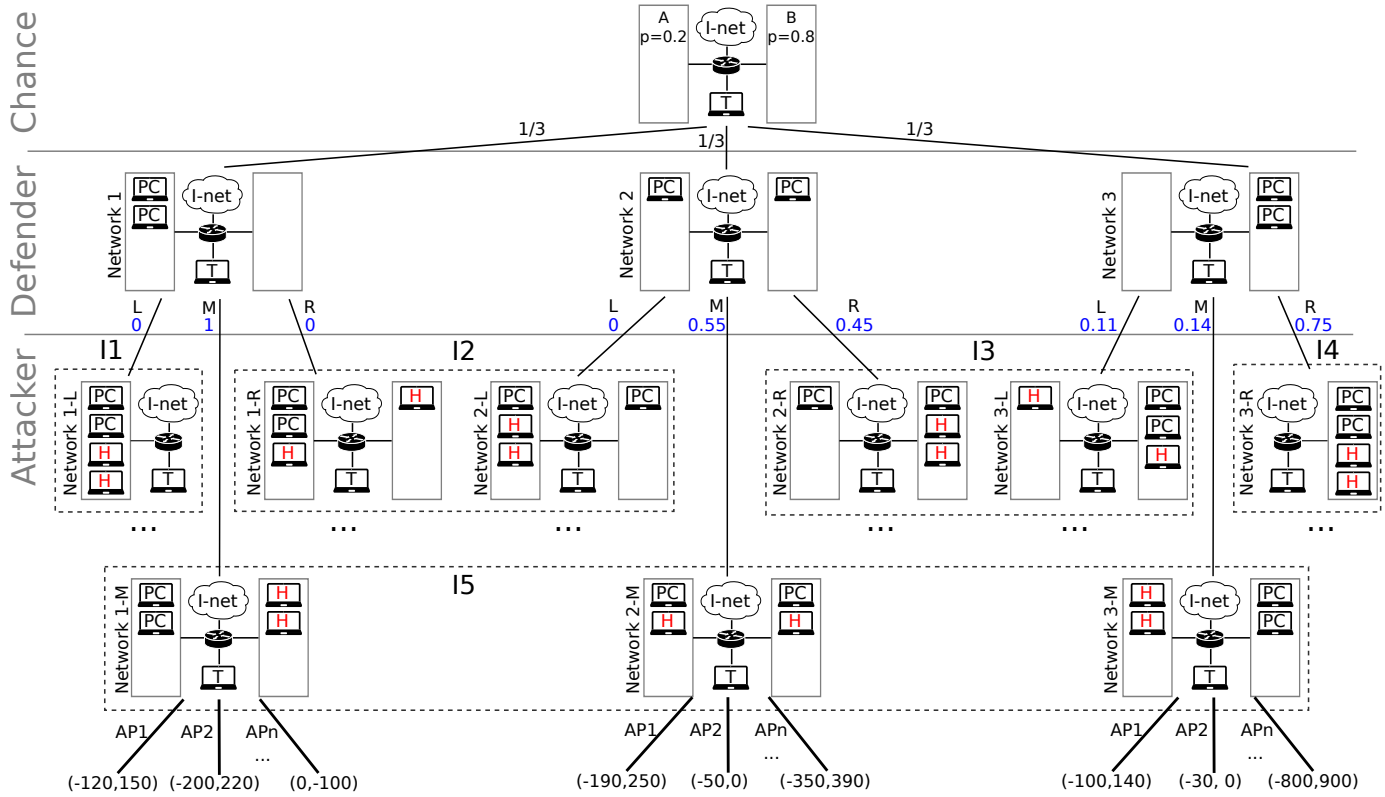

Fig. 2: Game tree for a simple network with host-types A and B, with their attack success probabilities 0.2 and 0.8, respectively. Chance plays uniformly into three possible networks, each contains two hosts. The defender chooses possible combinations of allocating two honeypots in each possible network; the blue probability values is a possible defense strategy (OPT_d discussed in Section VI). The attacker selects attack policy AP_1 through AP_n according to the networks' attack graphs.

be executed by a rational attacker. The problem of computing the optimal attack policy can be represented as a finite horizon Markov Decision Process (MDP). We use domain-specific MDP algorithms to find the optimal strategies [2].

IV. Game Model

Honeypots in network security are a form of a deception. We can model deception in games where one player has more information about the game state than the other (e.g., network structure, honeypot locations). Predicting the players' behavior is more difficult in these games because it depends on their *beliefs* about the likelihood of possible states. We model the honeypot allocation problem as an extensive-form game that captures the attacker's limited information about the network, allowing for deception.

We assume that the attacker has prior beliefs about the structure of the network. For example, he knows that in a bank network there will certainly be some office computers, a client database, and an internet banking backend, but he does not know how many offices there are or if the bank uses a VPN server. We refer to the part of the network known to the attacker as the *core* of a network.

The attacker's beliefs about possible networks can be represented by a *Chance* player who makes an initial random move with probability corresponding to the attacker's belief of likelihood of each network, as shown in Figure 2. The attacker is uncertain which of the possible *extended networks* (networks after the Chance move) is the defender's real network *before* the defender added honeypots. We assume that the attacker's belief is a common knowledge (if not, the defender can approximate it by analyzing the frequency of the networks in the real-world). In our example, the core network (at the Chance layer) consists of a gateway router and a target host T. We model the attacker who knows that there are two additional hosts, each either of host-type A or B, with equal probability. Therefore, the chance player extends the core network by adding possible combinations of host-types A and B, each with probability of 1/3 (known to both players), which leads to Networks 1-3.

The defender decides what honeypot host-types to add to each extended network. Although the defender knows the actual network, the strategy specifies the honeypots to add to every possible network. We assume that the attacker knows that the defender can add up to k honeypots, therefore, he must reason about what the defender would do in each possible situation. This creates an interesting information structure. If the defender adds two honeypots to A in Network 1, which results in Network 1-L, the attacker can be certain that two out of four hosts in A are honeypots. However if the defender adds one honeypot to A and one to B in Network 1 and two honeypots to A in Network 2, the resulting Networks 1-R and 2-L look exactly the same to the attacker. When the attacker

observes this network, he cannot be certain if the host in B is a honeypot or not. An *information-set* is the set of networks that look the same to the attacker, in Figure 2 denoted by dashed rectangles labeled as I1 through I5. The attacker must play the same strategy for all networks in an information-set, since they are indistinguishable. But in each information-set the attacker may have different attack plans to choose from (e.g., in Networks 1-L only host-type A can be attacked, while in Network 3-R only host-type B). The defender controls the attacker's observations about the network to a large extent, leading to the potential for deception and a complex decision problem for the attacker.

We assume that the honeypot duplicate is indistinguishable from the original host to the attacker. If the attacker performs an attack on host-type A in Network 1-L, he has a 50% chance of attacking the honeypot. Therefore, the attacker has to weigh the probability of being detected against the expected benefit of the successful attack and the cost of alternative attacks. Each attack policy results in a potentially different pair of utility values specified in the leaves of the game tree (AP1 through APn in I5 in Figure 2). The first value is the defender's utility, which contains honeypot costs and expected losses from attacks. The second value is the attacker's utility, which contains the expected reward, cost, and penalty for being detected.

The defender's strategy OPT_d is shown in Figure 2 with probabilities in blue color. The strategy prescribes the probability of playing each action in that network. The defender can influence the attacker's probabilities of observing the networks. Typically, the best strategies make the attacker indifferent between which host-type to attack first, which leads to least effective attacks. For example, if the defender plays OPT_d strategy, than the attacker who observes network in I3, with probability $\frac{0.45}{0.45+0.11} \approx 0.8$ faces Network 2-R and with 0.2 Network 3-L. Since attacks are four times more likely to succeed at host-type B than at A, the defender prefers adding honeypot of B four times more than A.

Computing the defender's strategy is difficult because: i) the number of possible defender actions grows combinatorially with the number of honeypots and host-types, ii) computing the attacker's optimal attack plan is NP-hard problem, and iii) computing the defender's strategy in Stackelberg equilibrium is an NP-hard problem for imperfect-information games. We describe the game model in more detail along with several approximation algorithms in [1].

V. Case Study I – TV Company

We now present a case study demonstrating how we can use this framework to model a real network, and feasibly compute optimal honeypot allocations for this network. The optimal strategies for this network incorporate deception, with the defender exploiting the attacker's uncertainty about the network.

A. Domain Description

The case study is based on a network used during cyber security exercises by Swedish Defence Research Agency in 2012 [6], where networks were deliberately left vulnerable to attacks. We use TV company network depicted in Figure 3. For demonstration, we simplify the original network by representing complete subnetworks as single host-types and reducing the number of vulnerabilities to three of PCs and routers. The resulting attack graph has 61 actions and 102 facts.

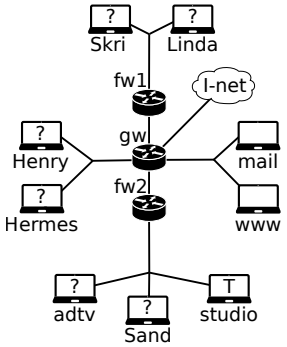

Fig. 3: Network for TV company.

The attacker can initiate an attack on any PC (e.g., via a malicious website visited by a user of that host) except the target *studio* host (T in Figure 3). The routers cannot be attacked remotely, only locally. The attacker aims to gain root privileges to the *studio* host, which has value 1,000 for both players. Exploit actions have costs between 5 and 10 (7.5 on average) and success probabilities are estimated by MulVAL (0.7 on average). The attacker has a penalty of 200 if he is detected. The defender's cost for creating a complex honeypot mimicking *studio* is 130, while any other PC costs 30.

B. Game Analysis

The attacker knows the core network (nodes without a question mark in Figure 3). While the attacker is uncertain about the hosts with question marks, he knows that the attacked network contains two of the hosts with a question mark, but not which ones. The defender knows his network, which consists of the core network and PC host-types *adtv* and *Linda*. We refer to this as the *real network*.

Solving the game means finding the defender's optimal strategy specifying a honeypot allocation in *every* possible network *instance* that can occur (after the chance move). With two honeypots the defender adds honeypots of *studio* host-type to secure the target host, despite their high cost. A more interesting strategy occurs with three honeypots, where the attacker reasons about $\binom{7}{3} = 35$ possible networks and computes attack plans. Since *adtv* is appealing for the attacker, the defender's strategy recommends adding *adtv* as a honeypot in the networks where *adtv* does not exist (i.e., in hypothetical networks other than the real network). This makes the attacker cautious about attacking *adtv*. The defender of the real network (with *Linda* and *adtv*) allocates honeypots as follows:

1) with probability 0.75: $\pi_1 = \{fw2, adtv, studio\}$, and
2) with probability 0.25: $\pi_2 = \{fw2, gw, Skri\}$,

with expected utility -412. This leads to a counterintuitive strategy for the rational attacker. He attacks *adtv only* when

two *adtv* hosts are present, one *real* and one honeypot (after action π_1). Although there is probability 0.5 that he will interact with a honeypot, it is worth the risk. Counterintuitively, when the defender plays π_2, in which case *adtv* could be attacked without interacting with a honeypot, the attacker reasons that it is "too good to be true" and instead attacks *www*. When the attacker observes a single *adtv* host, the host is more likely to be a honeypot (with probability 0.65) than a real host, because in the alternative networks *adtv* is often added as a honeypot.

With three honeypots the administrator can exploit the attacker's uncertainty by playing mixed strategies and reduce their total costs. With more honeypots the optimal strategies become difficult to comprehend in full detail, let alone calculated by hand. Therefore, we argue that a decision support system of this kind is valuable for the administrators.

VI. Case Study II - Human Strategies

We conducted a survey to compare the performance of the game-theoretic solutions to human opponents. The 45 respondents were participants in a four-day-long forensic malware seminar, members of multi-agent technology group at CTU, and employees of two computer security companies. The survey was presented as a competition among the respondents to motivate the respondents to create effective strategies.

To avoid overwhelming participants with too much information, we used the simple networks in Figure 2 and set all honeypot costs and attack action costs to zero. We kept the reward of 1,000 for target host T and a penalty of 200 for the attacker if detected, and the exploit success probabilities of 0.2 and 0.8 for host-types A and B, respectively. Attacking routers and host-type T is always successful, but the attacker can initiate attack only from host-types A or B.

A. Respondent Behaviors

Analysis of 45 collected responses revealed five main clusters in the respondents' defense strategies. We compared this clustering to clustering of 500 uniformly random data sets using five standard quality metrics. The quality of our clustering was better than the 95th percentile indicating that the clusters are not likely to appear by chance. A strategy belongs to a cluster if its L1 distance from a hand-selected centroid strategy is less than 1/3.

For each cluster we present the percentage of the strategies in that cluster and defender's average expected utility (u_d) of the strategies against a worst-case attacker. The clusters can be described as follows:

1) **Optimal Strategy ($\mathbf{OPT}_d$)** $(15\%, u_d = -236 \pm 23)$ is a cluster around the game theoretic solution.
2) **Perfect Information ($\mathbf{PI}_d$)** $(26\%, u_d = -267 \pm 21)$ defends each network in isolation by playing into I1, I4 and I5, not exploiting the attacker's uncertainty.
3) **Single Information Set ($\mathbf{SIS}_d$)** $(11\%, u_d = -343 \pm 16)$ in contrast to PI_d plays always to I5.
4) **Biased Uniform ($\mathbf{BU}_d$)** $(26\%, u_d = -292 \pm 22)$ mostly allocates one honeypot to each side, and with probabilities 0.1 and 0.2 both honeypots to A and B, respectively.
5) **Both to B ($\mathbf{BB}_d$)** $(15\%, u_d = -255 \pm 24)$ protects more vulnerable host-type B always with two honeypots.
6) **Outliers ($\mathbf{OL}_d$)** $(13\%, u_d = -334 \pm 111)$ are defense strategies with larger distance than 1/3 to any centroid strategy.

We visualize the individual defense strategies in Figure 4. Each triangle represents one possible network and the points represent the normalized probability distributions over pure strategies (each corner represents a pure strategy).

B. Survey Analysis

In Table I is a summary of the strategy analysis. Each entry contains the defender's mean utility μ, standard deviation σ and 10th (resp. 90th) percentile from 10^6 simulations for a pair of a defense strategy (row) against an attack strategy (column).

The OPT_d strategy maximizes the defender's utility against worst-case attacker. Column BR_a is best-response (worst-case) attack strategy against each defense strategy, which shows how *exploitable* a defense strategy is. The R_d represents the respondents' defense strategies in our dataset. We generated a single "mean" respondent strategy by averaging the individual strategies, labeled MR_d. By comparing these strategies to the baseline uniform strategy, we see that respondents played better than the random baseline.

The respondents' mean attack strategy MR_a reveals some interesting observations. MR_d against MR_a has lower mean utility than the optimal strategy OPT_d against MR_a. It means that against the human attackers, it might be better to defend with human defense strategy MR_d rather than OPT_d. However, MR_d may not be a good strategy in the long-term perspective. Attackers will likely learn from experience and move towards the best-response BR_a by increasing the frequency of successful attacks and reducing failed ones [3]. Instead of playing OPT_d, it might be better to begin with MR_d strategy, but switch to OPT_d as the attackers start adapting.

However, we can develop even better defense strategy than OPT_d or MR_d, which exploits the human behavior. BR_d is the defender's best-response defense strategy against the human MR_d attack strategy. This results in highest utility for the defender; however, there is added risk because it is vulnerable against BR_a attackers who specifically exploit this strategy.

Our strategies have large σ due to two factors. First, networks have different levels of security (Network 3 is more vulnerable than Network 1) and more vulnerable networks will naturally have lower utility. For example, with strategy OPT_d the administrator of Network 1, 2, and 3 has expected utility -23, -247, and -350, respectively. Second, randomized strategies are necessary to minimize the strategy's predictability, but they also result in variability in outcomes. The OPT_d strategy has the lowest standard deviation against both types of attackers, which can be a desirable property.

VII. Conclusion

Developing effective decision support tools is critical to improving security as networks and attacks become more complex. A particular concern is finding policies that are robust as attackers learn and respond to the security strategy.

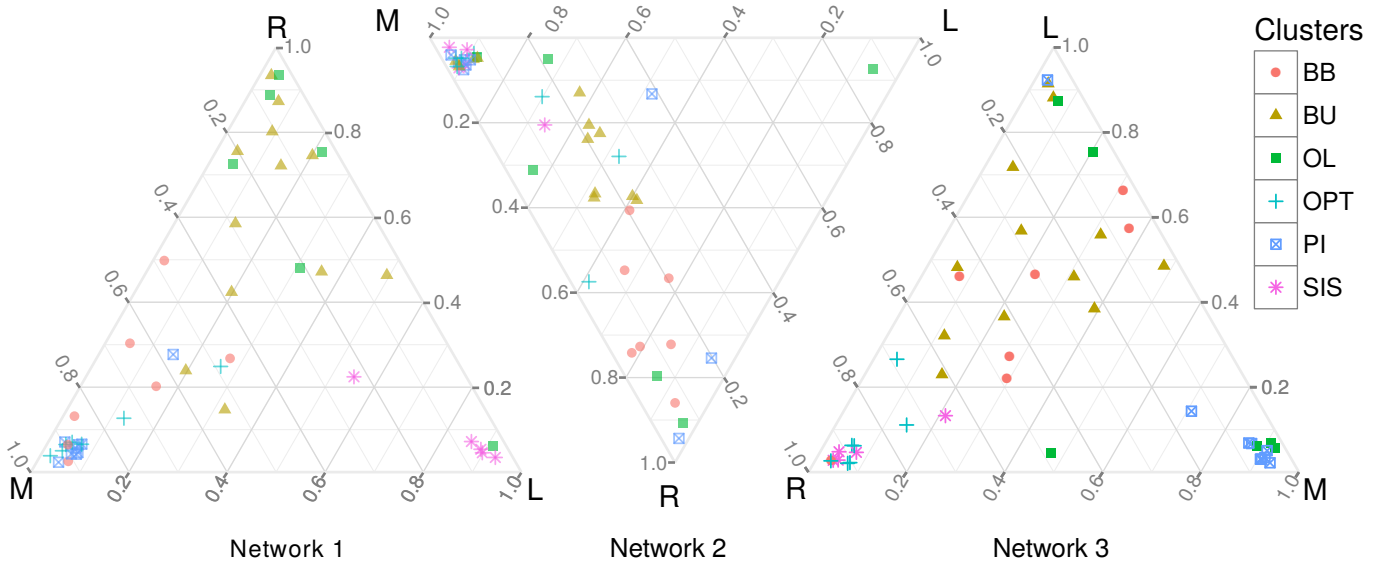

Fig. 4: Respondent's defense strategies and a clustering of the strategies. Each triangle represents the space of possible defender strategies for the network in Figure 2 (each corner is a pure strategy).

TABLE I: Defender's utility, standard deviation, 10th and 90th percentile for the defender's strategies (rows) and attacker's strategies (columns). BR_a - attacker's best-response strategy to defense strategy; MR_d,MR_a - mean respondent defense (resp. attack) strategy.

Defense / Attack	Best Response BR_a			Mean Respondent MR_a		
	μ	σ	10th, 90th percentile	μ	σ	10th, 90th percentile
Optimal (OPT_d)	-207	131	-350, -50	-189	181	-350, 100
Respondent (R_d)	-285	236	-500, 100	-167	208	-360, 100
Mean Respondent (MR_d)	-262	266	-500, 100	-164	207	-360, 100
Baseline uniform	-317	322	-800, 100	-162	240	-500, 100
Best Response (BR_d)	-302	187	-500, -50	-111	200	-500, 100

Game theory is a promising tools for developing these decision support systems because it explicitly models the reactions of the opponent. It can also model deception and information manipulation by predicting and manipulating opponents' beliefs.

We show how game theory can be used to construct and analyze models of defensive measures for realistic networks. Our models use automatically constructed attack graphs from publicly available data to represent possible attack plans. We conducted an initial validation study with human decision makers to directly compare the game-theoretic solutions with humans. The results show strengths and weaknesses of both the theoretical and human solutions: humans were effective at defending against human attackers, but fared poorly against worst-case opponents. The game-theoretic strategies are robust, but there are opportunities to further exploit weaknesses in human opponents. In addition, we had to simplify the game considerably so the human players could understand it. In complex scenarios humans may not be able to compute any plausible strategy with a reasonable effort. Further empirical studies are clearly needed to investigate the effectiveness of game models for network security, but we view this as a promising first step.

Our work has potential for a further research in several directions. The attacker's interaction with the honeypots can be modeled in more detail, including the attacker's attempts to detect the honeypots, etc. It would capture more realistically the attacker's penalties and results in more robust defense strategies. Another direction is to include into the model the network changes, which can be partially solved by deploying dynamic honeypots that can be reconfigured according to the new strategies for the modified networks. This model could consider set of possible network changes and find strategies that additionally minimize cost for honeypot reconfigurations.

Acknowledgment

This work was supported by the Grant Agency of the CTU in Prague (SGS16/235/OHK3/3T/13), Czech Science Foundation (15-23235S) and Cisco Systems.

References

[1] Karel Durkota, Viliam Lisý, Branislav Bošanský, and Christopher Kiekintveld. Approximate solutions for attack graph games with imperfect information. In *Decision and Game Theory for Security*, pages 228–249. Springer, 2015.

[2] Karel Durkota, Viliam Lisý, Branislav Bošanský, and Christopher Kiekintveld. Optimal network security hardening using attack graph games. In *Proceedings of IJCAI*, pages 7–14, 2015.

[3] Ehud Kalai and Ehud Lehrer. Rational learning leads to nash equilibrium. *Econometrica: Journal of the Econometric Society*, pages 1019–1045, 1993.

[4] Peter Mell, Karen Scarfone, and Sasha Romanosky. Common vulnerability scoring system. *Security & Privacy*, pages 85–89, 2006.

[5] Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *CCS*, pages 336–345, 2006.

[6] Teodor Sommestad and Fredrik Sandström. An empirical test of the accuracy of an attack graph analysis tool. *Information & Computer Security*, 23(5):516–531, 2015.

[7] Lance Spitzner. Honeypots: Catching the insider threat. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 170–179. IEEE, 2003.

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

[9] Lingyu Wang, Tania Islam, Tao Long, Anoop Singhal, and Sushil Jajodia. An attack graph-based probabilistic security metric. In *Data and Applications Security XXII*, volume 5094, pages 283–296. Springer Berlin Heidelberg, 2008.