A Learning-Based Approach to Reactive Security

Adam Barth, Benjamin I.P. Rubinstein, Mukund Sundararajan, John C. Mitchell, Dawn Song, *Member*, *IEEE*, and Peter L. Bartlett, *Member*, *IEEE*

Abstract—Despite the conventional wisdom that proactive security is superior to reactive security, we show that reactive security can be competitive with proactive security as long as the reactive defender learns from past attacks instead of myopically overreacting to the last attack. Our game-theoretic model follows common practice in the security literature by making worst case assumptions about the attacker: we grant the attacker complete knowledge of the defender's strategy and do not require the attacker to act rationally. In this model, we bound the competitive ratio between a reactive defense algorithm (which is inspired by online learning theory) and the best fixed proactive defense. Additionally, we show that, unlike proactive defenses, this reactive strategy is robust to a lack of information about the attacker's incentives and knowledge.

Index Terms—Reactive security, risk management, attack graphs, online learning, adversarial learning, game theory.

1 Introduction

Many enterprises employ a Chief Information Security Officer (CISO) to manage the enterprise's information security risks. Typically, an enterprise has many more security vulnerabilities than it can realistically repair. Instead of declaring the enterprise "insecure" until every last vulnerability is plugged, CISOs typically perform a cost-benefit analysis to identify which risks to address. But what constitutes an effective CISO strategy? The conventional wisdom [1], [2] is that CISOs ought to adopt a "forward-looking" proactive approach to mitigating security risk by examining the enterprise for vulnerabilities that might be exploited in the future. Advocates of proactive security often equate reactive security with myopic bug-chasing and consider it ineffective. We establish sufficient conditions for when reacting *strategically* to attacks is as effective in discouraging attackers.

We study the efficacy of reactive strategies in an economic model of the CISO's security cost-benefit trade-offs. Unlike previously proposed economic models of security (see Section 9), we do not assume the attacker acts according to a fixed probability distribution. Instead, we consider a game-theoretic model with a strategic attacker who responds to the defender's strategy. As is standard in the security literature, we make worst case assumptions about the attacker. For example, we grant the attacker complete knowledge of the defender's strategy and do not require the attacker to act rationally. Further, we make conservative assumptions about the reactive defender's knowledge and do not assume the defender knows all the vulnerabilities in the system or the attacker's incentives. However, we do assume that the defender can observe the attacker's past actions, for example, via an intrusion detection system or user metrics [3].

Under our theoretical model, we find that three assumptions are sufficient for a reactive strategy to perform as well as the best proactive strategies:

Assumption 1. *No single attack is catastrophic.*

Assumption 2. *The defender's budget is liquid.*

Assumption 3. *The attacker's cost for mounting an attack is linear in the defensive allocations.*

Our first assumption requires that the defender can survive a number of attacks. This is consistent with situations where intrusions (that, say, steal credit card numbers) are regrettable but not business-ending. Our results do not directly apply to settings in which catastrophic attacks are possible, e.g., in nuclear warfare strategy.

That the defender's budget is *liquid*—our second assumption—means that the defender can reallocate resources without penalty. Of course, not all defense spending is liquid (e.g., cement walls are difficult to move or sell), which means our results do not apply to all situations. However, our results do apply in common situations, such as when allocating headcount, when the defenders budget is approximately liquid. For example, a CISO can reassign members of the security team from managing firewall rules to improving database access controls at relatively low-switching costs.

Because our model abstracts many vulnerabilities into a single edge in an attack graph, we view the act of defense as increasing the attacker's *cost* for mounting an attack instead of preventing the attack (e.g., by patching a single bug). By

• *A. Barth and M. Sundararajan are with Google, Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043. E-mail: adam@adambarth.com, mukunds@google.com.*
• *B.I.P. Rubinstein is with Microsoft Research, 1288 Pear Ave, Mountain View, CA 94043. E-mail: benjamin.i.p.rubinstein@gmail.com.*
• *J.C. Mitchell is with the Department of Computer Science, Stanford University, Stanford, CA 94305-9045. E-mail: mitchell@cs.stanford.edu.*
• *D. Song and P.L. Bartlett are with the Department of Electrical Engineering and Computer Sciences, UC Berkeley, Berkeley, CA 94720-1776. E-mail: {dawnsong, bartlett}@cs.berkeley.edu.*

Manuscript received 16 Nov. 2010; revised 6 Apr. 2011; accepted 10 June 2011; published online 22 Aug. 2011.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSCSI-2010-11-0214.
Digital Object Identifier no. 10.1109/2011.42.

taking this viewpoint, we choose not to study the tactical patch-by-patch interaction of the attacker and defender. Instead, we model enterprise security at a more abstract level appropriate for the CISO. For example, the CISO might allocate a portion of his or her budget to engage a consultancy, such as WhiteHat or iSEC Partners, to find and fix cross-site scripting in a particular web application or to require that employees use SecurID tokens during authentication. This leads to our third technical assumption that attacker costs are linearly dependent on defense investments locally. This assumption does not reflect patch-by-patch interaction, which would be better represented by a step function (with the step placed at the cost to deploy the patch). Instead, this assumption reflects the CISO's higher level point of view where the staircase of summed step functions fades into a slope.

We evaluate the defender's strategy by measuring the attacker's cumulative return-on-investment, the *return-on-attack* (ROA), which has been proposed previously [4]. By studying this metric, we focus on defenders who seek to "cut off the attacker's oxygen," that is to reduce the attacker's incentives for attacking the enterprise. We do not distinguish between "successful" and "unsuccessful" attacks. Instead, we compare the payoff the attacker receives from his or her nefarious deeds with the cost of performing said deeds. We imagine that sufficiently disincentivized attackers will seek alternatives, such as attacking a different organization or starting a legitimate business.

In our main result, we show sufficient conditions for a learning-based reactive strategy to be competitive with the best fixed proactive defense in the sense that the competitive ratio between the reactive ROA and the proactive ROA is at most $1+\epsilon$, for all $\epsilon > 0$, provided the game lasts sufficiently many rounds (at least $\Omega(1/\epsilon)$). To prove our theorems, we draw on techniques from the online learning literature. We extend these techniques to the case where the learner does not know all the game matrix rows a priori, letting us analyze situations where the defender does not know all of the vulnerabilities in advance. Although our main results are in a graph-based model with a single attacker, our results generalize to a model based on Horn clauses with multiple attackers (see Section 8.1). Our results are also robust to switching from ROA to attacker profit and to allowing the proactive defender to revise the defense allocation a fixed number of times.

Although myopic bug chasing is most likely an ineffective reactive strategy, we find that in some situations a *strategic* reactive strategy is as effective as the optimal fixed proactive defense. In fact, we find that the natural strategy of gradually reinforcing attacked edges by shifting budget from unattacked edges "learns" the attacker's incentives and constructs an effective defense. Such a strategic reactive strategy is both easier to implement than a proactive strategy—because it does not presume that the defender knows the attacker's intent and capabilities—and is less wasteful than a proactive strategy because the defender does not expend budget on attacks that do not actually occur. Based on our results, we encourage CISOs to question the assumption that proactive risk management is inherently superior to reactive risk management. When using reactive strategies, CISOs should employ monitoring tools to help focus on real attacks, lead a more agile security organization, and avoid overreacting to the most recent attacks and instead learn from past attacks while discounting their importance exponentially.

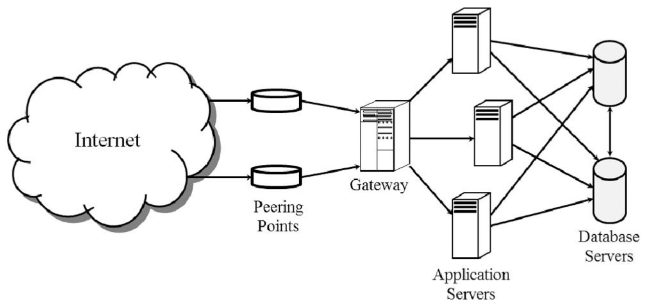


Fig. 1. An attack graph representing an enterprise data center.

1.1 Organization

Section 2 formalizes our model. Section 3 shows that perimeter defense and defense-in-depth arise naturally in our model. Section 4 describes the threat model and our new learning-based reactive strategy. Section 5 presents our main results bounding the competitive ratio of reactive versus proactive defense strategies. Section 6 provides a lower bound establishing the essential optimality of our learning-based defense among all reactive strategies. Section 7 outlines scenarios in which reactive security outperforms proactive security. Section 8 generalizes our results to Horn clauses and multiple attackers. Section 9 relates related work. Section 10 concludes. This paper adds proofs and a lower bound to an earlier conference version of this work [5].

2 Formal Model

In this section, we present a game-theoretic model of attack and defense. Unlike traditional bug-level attack graphs, our model is meant to capture a managerial perspective on enterprise security. The model is somewhat general in the sense that attack graphs can represent a number of concrete situations, including a network (see Fig. 1), components in a complex software system [6], or an Internet Fraud "Battlefield" [7].

2.1 System

We model a system using a directed graph (V, E), which defines the game between an attacker and a defender. Each vertex $v \in V$ in the graph represents a state of the system. Each edge $e \in E$ represents a state transition the attacker can induce. For example, a vertex might represent whether a particular machine in a network has been compromised by an attacker. An edge from one machine to another might represent that an attacker who has compromised the first machine might be able to compromise the second machine because the two are connected by a network. Alternatively, the vertices might represent different components in a software system. An edge might represent that an attacker sending input to the first component can send input to the second.

In attacking the system, the attacker selects a path a in the graph that begins with a designated *start vertex* s. Our results hold in more general models (e.g., based on Horn clauses), but we defer discussing such generalizations until Section 8. We think of the attack as driving the system through the series of state transitions indicated by the edges included in the path. In the networking example in Fig. 1, an attacker might first compromise a front-end server and then leverage the server's connectivity to the back-end database server to steal credit card numbers from the database.

2.2 Incentives and Rewards

Attackers respond to incentives. For example, attackers compromise machines and form botnets because they make money from spam [8] or rent the botnet to others [9]. Other attackers steal credit card numbers because credit card numbers have monetary value [10]. We model the attacker's incentives by attaching a nonnegative *reward* to each vertex. These rewards are the utility the attacker derives from driving the system into the state represented by the vertex. For example, compromising the database server might have a sizable reward because the database server contains easily monetizable credit card numbers. We assume the start vertex has zero reward, forcing the attacker to undertake some action before earning utility. Whenever the attacker mounts an attack, the attacker receives a *payoff* equal to the sum of the rewards of the vertices visited in the attack path $a : \mathrm{payoff}(a) = \sum_{v \in a} \mathrm{reward}(v)$. In the example from Fig. 1, if an attacker compromises both a front-end server and the database server, the attacker receives both rewards.

2.3 Attack Surface and Cost

The defender has a fixed *defense budget* $B > 0$, which the defender can divide among the edges in the graph according to a *defense allocation* d: for all $e \in E$, $d(e) \geq 0$ and $\sum_{e \in E} d(e) \leq B$.

The defender's allocation of budget to various edges corresponds to the decisions made by the Chief Information Security Officer about where to allocate the enterprise's security resources. For example, the CISO might allocate organizational headcount to fuzzing enterprise web applications for XSS vulnerabilities. These kinds of investments are continuous in the sense that the CISO can allocate $1/4$ of a full-time employee to worrying about XSS. We denote the set of feasible allocations of budget B on edge set E by $\mathcal{D}_{B,E}$.

By defending an edge, the defender makes it more difficult for the attacker to use that edge in an attack. Each unit of budget the defender allocates to an edge raises the cost that the attacker must pay to use that edge in an attack. Each edge has an *attack surface* [11] w that represents the difficulty in defending against that state transition. For example, a server that runs both Apache and Sendmail has a larger attack surface than one that runs only Apache because defending the first server is more difficult than the second. Formally, the attacker must pay the following *cost* to traverse the edges of an attack: $\mathrm{cost}(a, d) = \sum_{e \in a} d(e)/w(e)$. Allocating defense budget to an edge does not "reduce" an edge's attack surface. For example, consider defending a hallway with bricks. The wider the hallway (the larger the attack surface), the more bricks (budget allocation) required to build a wall of a certain height (the cost to the attacker).

In this formulation, the function mapping the defender's budget allocation to attacker cost is linear, preventing the defender from ever fully defending an edge. Our use of a linear function reflects a level of abstraction more appropriate to a CISO who can never fully defend assets, which we justify by observing that the rate of vulnerability discovery in a particular piece of software is roughly constant [12], which means an attacker with sufficient resources can often overcome even the best defenses. At a lower level of detail, we might replace this function with a step function, indicating that the defender can "patch" a vulnerability by allocating a threshold amount of budget.

2.4 Objective

To evaluate defense strategies, we measure the attacker's incentive for attacking using the *return-on-attack* [4], which we define as follows:

$$\mathrm{ROA}(a, d) = \frac{\mathrm{payoff}(a)}{\mathrm{cost}(a, d)}.$$

We use this metric for evaluating defense strategy because we believe that if the defender lowers the ROA sufficiently, the attacker will be discouraged from attacking the system and will find other uses for his or her capital or industry. For example, the attacker might decide to attack another system. Analogous results hold if we quantify the attacker's incentives in terms of profit (e.g., with $\mathrm{profit}(a, d) = \mathrm{payoff}(a) - \mathrm{cost}(a, d)$), but we focus our discussion on ROA for simplicity. Results for both ROA and additive profit are presented in Section 5.

A purely rational attacker will mount attacks that maximize ROA. However, a real attacker might not maximize ROA. For example, the attacker might not have complete knowledge of the system or its defense. We strengthen our results by considering all attacks, not just those that maximize ROA.

2.5 Proactive Security

We evaluate our learning-based reactive approach by comparing it against a *proactive* approach to risk management in which the defender carefully examines the system and constructs a defense in order to fend off future attacks. We strengthen this benchmark by providing the proactive defender complete knowledge about the system, but we require that the defender commit to a fixed strategy. To strengthen our results, we state our main result in terms of *all* such proactive defenders. In particular, this class of defenders includes the *rational proactive defender* who employs a defense allocation that minimizes the maximum ROA the attacker can extract from the system: $\mathrm{argmin}_d \mathrm{max}_a \mathrm{ROA}(a, d)$.

3 Case Studies

In this section, we describe instances of our model to build the reader's intuition. These examples illustrate that some familiar security concepts, including perimeter defense and defense in depth, arise naturally as optimal defenses in our model. These defenses can be constructed either by rational

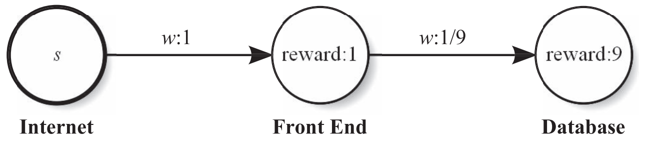


Fig. 2. Attack graph representing a simplified data center network.

proactive attackers or converged to by a learning-based reactive defense.

3.1 Perimeter Defense

Consider a system in which the attacker's reward is nonzero at exactly one vertex, t. For example, in a medical system, the attacker's reward for obtaining electronic medical records might well dominate the value of other attack targets such as employees' vacation calendars. In such a system, a rational attacker will select the minimum-cost path from the start vertex s to the valuable vertex t. The optimal defense limits the attacker's ROA by maximizing the cost of the minimum s-t path. The algorithm for constructing this defense is straightforward [13]:

1. Let C be the minimum weight s-t cut in (V, E, w).
2. Select the following defense:

$$d(e) = \begin{cases} Bw(e)/Z, & \text{if } e \in C, \\ 0, & \text{otherwise,} \end{cases}$$
$$\text{where } Z = \sum_{e \in C} w(e).$$

Notice that this algorithm constructs a *perimeter defense*: the defender allocates the entire defense budget to a single cut in the graph. Essentially, the defender spreads the defense budget over the attack surface of the cut. By choosing the minimum-weight cut, the defender is choosing to defend the smallest attack surface that separates the start vertex from the target vertex. Real defenders use similar perimeter defenses, for example, when they install a firewall at the boundary between their organization and the Internet because the network's perimeter is much smaller than its interior.

3.2 Defense in Depth

Many experts in security practice recommend that defenders employ defense in depth. Defense in depth arises naturally in our model as an optimal defense for some systems. Consider, for example, the system depicted in Fig. 2. This attack graph is a simplified version of the data center network depicted in Fig. 1. Although the attacker receives the largest reward for compromising the back-end database server, the attacker also receives some reward for compromising the front-end web server. Moreover, the front-end web server has a larger attack surface than the back-end database server because the front-end server exposes a more complex interface (an entire enterprise web application), whereas the database server exposes only a simple SQL interface. Allocating defense budget to the left-most edge represents trying to protect sensitive database information with a complex web application firewall instead of database access control lists (i.e., possible, but economically inefficient).

The optimal defense against a rational attacker is to allocate half of the defense budget to the left-most edge and half of the budget to the right-most edge, limiting the attacker to a ROA of unity. Shifting the entire budget to the right-most edge (i.e., defending only the database) is disastrous because the attacker will simply attack the front-end at zero cost, achieving an unbounded ROA. Shifting the entire budget to the left-most edge is also problematic because the attacker will attack the database (achieving an ROA of five).

4 Reactive Security

To analyze reactive security, we model the attacker and defender as playing an iterative game of alternating moves. First, the defender selects a defense, and then the attacker selects an attack. We present a learning-based reactive defense strategy that is oblivious to vertex rewards and to edges that have not yet been used in attacks. We prove a theorem bounding the competitive ratio between this reactive strategy and the best proactive defense via a series of reductions to results from the online learning theory literature. Other applications of this literature include managing stock portfolios [14], playing zero-sum games [15], and boosting other machine learning heuristics [16]. Although we provide a few technical extensions, our main contribution comes from applying results from online learning to risk management.

4.1 Repeated Game

We formalize the repeated game between the defender and the attacker as follows: In each round t from 1 to T:

1. The defender chooses defense allocation $d_t(e)$ over the edges $e \in E$.
2. The attacker chooses an attack path a_t in G.
3. The path a_t and attack surfaces $\{w(e) : e \in a_t\}$ are revealed to the defender.
4. The attacker pays $\mathrm{cost}(a_t, d_t)$ and gains $\mathrm{payoff}(a_t)$.

In each round, we let the attacker choose the attack path after the defender commits to the defense allocation because the defender's budget allocation is not a secret (in the sense of a cryptographic key). Following the "no security through obscurity" principle, we make the conservative assumption that the attacker can accurately determine the defender's budget allocation.

4.2 Defender Knowledge

Unlike proactive defenders, reactive defenders do not know all of the vulnerabilities that exist in the system in advance. For example, browser vendors would routinely survive the Pwn2Own competition and conferences such as Black Hat Briefings would serve little purpose if defenders had complete knowledge of vulnerabilities. Instead, we reveal an edge (and its attack surface) to the defender after the attacker uses the edge in an attack. For example, the defender might monitor the system and learn how the attacker attacked the system by periodically analyzing logs for intrusions. Formally, we define a *reactive defense strategy* to be a function from attack sequences $\{a_i\}$ and the subsystem induced by the edges contained in $\bigcup_i a_i$ to defense allocations such that $d(e) = 0$ if edge $e \notin \bigcup_i a_i$. Notice that this requires the defender's strategy to be oblivious to the system beyond the edges used by the attacker.

4.3 Algorithm

Algorithm 1 is a reactive defense strategy based on the multiplicative update learning algorithm [15], [17]. The algorithm reinforces edges on the attack path multiplicatively, taking the attack surface into account by allocating more budget to easier-to-defend edges. When new edges are revealed, the algorithm reallocates budget uniformly from the already-revealed edges to the newly revealed edges. We state the algorithm in terms of a normalized defense allocation $P_t(e) = d_t(e)/B$. Notice that this algorithm is oblivious to unattacked edges and the attacker's reward for visiting each vertex. An appropriate setting for the algorithm parameters $\beta_t \in [0, 1)$ will be described below.

Algorithm 1. A reactive defense strategy for hidden edges.

```
• Initialize $E_0 = \emptyset$
• For each round $t \in \{2, \dots, T\}$
    - Let $E_{t-1} = E_{t-2} \cup E(a_{t-1})$
    - For each $e \in E_{t-1}$, let
        $S_{t-1}(e) = \begin{cases} S_{t-2}(e) + M(e, a_{t-1}) & \text{if } e \in E_{t-2} \\ M(e, a_{t-1}) & \text{otherwise.} \end{cases}$
        $\tilde{P}_t(e) = \beta_{t-1}^{S_{t-1}(e)}$
        $P_t(e) = \frac{\tilde{P}_t(e)}{\sum_{e' \in E_t} \tilde{P}_t(e')}$,
    where $M(e, a) = -\mathbf{1}[e \in a]/w(e)$ is a matrix with
    $|E|$ rows and a column for each attack.
```

The algorithm begins without any knowledge of the graph whatsoever, and so allocates no defense budget to the system. Upon the tth attack on the system, the algorithm updates E_t to be the set of edges revealed up to this point, and updates $S_t(e)$ to be a weighted count of the number of times e has been used in an attack thus far. For each edge that has ever been revealed, the defense allocation $P_{t+1}(e)$ is chosen to be $\beta_t^{S_t(e)}$ normalized to sum to unity over all edges $e \in E_t$. In this way, any edge attacked in round t will have its defense allocation reinforced.

The parameters β_t control how aggressively the defender reallocates defense budget to recently attacked edges. If β_t is infinitesimal, the defender will move the entire defense budget to the edge on the most recent attack path with the smallest attack surface. If $\beta_t \approx 1$, the defender will not be very agile and, instead, leave the defense budget in the initial allocation. For appropriate values of β_t, the algorithm will converge to the optimal defense strategy. For instance, the min-cut in the example from Section 3.1.

5 Main Results

We now describe a series of reductions establishing the main results that bound the difference between attacker profit under this reactive strategy and fixed proactive strategies—Theorem 3—and the ratio between the corresponding attacker ROAs—Theorem 7. First, we bound profits in the simpler setting where the defender knows the entire graph in Section 5.1. Second, in Section 5.2 we remove the hypothesis that the defender knows the edges in advance. Finally, in Section 5.3 we extend our results to ROA. To compare strategies, we use the notion of *regret* from online learning theory.

5.1 Attacker Profit (Known Edges Case)

Suppose that the reactive defender is granted full knowledge of the system[1] $(V, E, w, \text{reward}, s)$ from the outset, prior to the first round. Algorithm 2 is a reactive defense strategy that makes use of this additional knowledge.

Algorithm 2. Reactive defense strategy for known edges using the multiplicative update algorithm.

```
• For each $e \in E$, initialize $P_1(e) = 1/|E|$.
• For each round $t \in \{2, \dots, T\}$ and $e \in E$, let
        $P_t(e) = P_{t-1}(e) \cdot \beta^{M(e, a_{t-1})}/Z_t$
  where $Z_t = \sum_{e' \in E} P_{t-1}(e)\beta^{M(e', a_{t-1})}$
```

Lemma 1. *If defense allocations $\{d_t\}_{t=1}^T$ are output by Algorithm 2 with parameter $\beta = \left(1 + \sqrt{2 \log |E|/T}\right)^{-1}$ on any system $(V, E, w, \text{reward}, s)$ and attack sequence $\{a_t\}_{t=1}^T$, then*

$$\frac{1}{T}\sum_{t=1}^{T} \text{profit}(a_t, d_t) - \frac{1}{T}\sum_{t=1}^{T} \text{profit}(a_t, d^\star) \leq B\sqrt{\frac{\log |E|}{2T}} + \frac{B \log |E|}{T},$$

for all proactive defense strategies $d^\star \in \mathcal{D}_{B,E}$.

The lemma's proof is a reduction to the following regret bound from online learning [15, Corollary 4].

Theorem 2. *If the multiplicative update algorithm (Algorithm 2) is run with any game matrix M with elements in $[0, 1]$, and parameter $\beta = \left(1 + \sqrt{2 \log |E|/T}\right)^{-1}$, then*

$$\frac{1}{T}\sum_{t=1}^{T} M(P_t, a_t) - \min_{\substack{P^\star \geq 0: \\ \sum_{e \in E} P^\star(e) = 1}} \left\{ \frac{1}{T}\sum_{t=1}^{T} M(P^\star, a_t) \right\} \leq \sqrt{\frac{\log |E|}{2T}} + \frac{\log |E|}{T}.$$

In the original game-theoretic setting of Theorem 2, pure plays of the learner (adversary) correspond to rows (respectively columns) of a fixed game matrix, which records the learner's instantaneous loss. When either or both of the learner and adversary play mixed strategies according to some distribution(s) P and Q, then the expected loss (or *risk*) corresponding to the matrix product $P^T M Q$ is written as $M(P, Q)$ for convenience. Similarly for cases when one player plays a mixed strategy and the other plays a pure strategy we write $M(P, q)$ or $M(p, Q)$.

In our present setting we consider each edge as a possible pure play of the learner, and each attack path as a

1. The system consists of the weighted graph (V, E, w), rewards and start vertex s.

pure play of the attacker. In this way, the learner's mixed strategies correspond to allocations of a unit budget over subsets of edges within the graph.

Proof of Lemma 1. Due to the normalization by Z_t, the sequence of defense allocations $\{P_t\}_{t=1}^T$ output by Algorithm 2 is invariant to adding a constant to all elements of matrix M. Let M' be the matrix obtained by adding constant C to all entries of arbitrary game matrix M, and let sequences $\{P_t\}_{t=1}^T$ and $\{P'_t\}_{t=1}^T$ be obtained by running multiplicative update with matrix M and M', respectively. Then, for all $e \in E$ and $t \in [T-1]$,

$$\begin{aligned} P'_{t+1}(e) &= \frac{P_1(e)\beta^{\sum_{i=1}^t M'(e,a_i)}}{\sum_{e'\in E} P_1(e')\beta^{\sum_{i=1}^t M'(e',a_i)}} \\ &= \frac{P_1(e)\beta^{\left(\sum_{i=1}^t M(e,a_i)\right)+tC}}{\sum_{e'\in E} P_1(e')\beta^{\left(\sum_{i=1}^t M(e',a_i)\right)+tC}} \\ &= \frac{P_1(e)\beta^{\sum_{i=1}^t M(e,a_i)}}{\sum_{e'\in E} P_1(e')\beta^{\sum_{i=1}^t M(e',a_i)}} \\ &= P_{t+1}(e). \end{aligned}$$

In particular Algorithm 2 produces the same defense allocation sequence as if the game matrix elements are increased by one to

$$M'(e,a) = \begin{cases} 1 - 1/w(e) & \text{if } e \in a \\ 1 & \text{otherwise.} \end{cases}$$

Because this new matrix has entries in $[0,1]$ we can apply Theorem 2 to prove that, for the original matrix M

$$\begin{aligned} &\frac{1}{T}\sum_{t=1}^T M(P_t,a_t) - \min_{P^\star\in\mathcal{D}_{1,E}}\left\{\frac{1}{T}\sum_{t=1}^T M(P^\star,a_t)\right\} \\ &\quad \le \sqrt{\frac{\log|E|}{2T}} + \frac{\log|E|}{T}. \end{aligned} \tag{1}$$

Now, by definition of the original game matrix

$$\begin{aligned} M(P_t,a_t) &= \sum_{e\in E} -(P_t(e)/w(e))\cdot \mathbf{1}[e\in a_t] \\ &= -\sum_{e\in a_t} P_t(e)/w(e) \\ &= -B^{-1}\sum_{e\in a_t} d_t(e)/w(e) \\ &= -B^{-1}\text{cost}(a_t,d_t). \end{aligned}$$

Thus, Inequality (1) is equivalent to

$$\begin{aligned} &-\frac{1}{T}\sum_{t=1}^T B^{-1}\text{cost}(a_t,d_t) \\ &\quad - \min_{d^\star\in\mathcal{D}_{1,E}}\left\{-\frac{1}{T}\sum_{t=1}^T B^{-1}\text{cost}(a_t,d^\star)\right\} \\ &\quad \le \sqrt{\frac{\log|E|}{2T}} + \frac{\log|E|}{T}. \end{aligned}$$

Simple algebraic manipulation yields

$$\begin{aligned} &\frac{1}{T}\sum_{t=1}^T \text{profit}(a_t,d_t) - \min_{d^\star\in\mathcal{D}_{B,E}}\left\{\frac{1}{T}\sum_{t=1}^T \text{profit}(a_t,d^\star)\right\} \\ &= \frac{1}{T}\sum_{t=1}^T (\text{payoff}(a_t) - \text{cost}(a_t,d_t)) \\ &\quad - \min_{d^\star\in\mathcal{D}_{B,E}}\left\{\frac{1}{T}\sum_{t=1}^T (\text{payoff}(a_t) - \text{cost}(a_t,d^\star))\right\} \\ &= \frac{1}{T}\sum_{t=1}^T (-\text{cost}(a_t,d_t)) - \min_{d^\star\in\mathcal{D}_{B,E}}\left\{\frac{1}{T}\sum_{t=1}^T (-\text{cost}(a_t,d^\star))\right\} \\ &\le B\sqrt{\frac{\log|E|}{2T}} + B\frac{\log|E|}{T}, \end{aligned}$$

establishing the result. □

5.2 Attacker Profit (Hidden Edges Case)

The following is an additive regret bound relating the attacker's profit under reactive and proactive defense strategies, in the more restricted setting of a defender observing edges and weights *only* after they are first exploited by the attacker.

Theorem 3. *The average attacker profit against Algorithm 1 converges to the average attacker profit against the best proactive defense. Formally, if defense allocations $\{d_t\}_{t=1}^T$ are output by Algorithm 1 with parameter sequence $\beta_t = \left(1+\sqrt{2\log|E_t|/(t+1)}\right)^{-1}$ on any system $(V, E, w, \text{reward}, s)$ revealed online and any attack sequence $\{a_t\}_{t=1}^T$, then*

$$\begin{aligned} &\frac{1}{T}\sum_{t=1}^T \text{profit}(a_t,d_t) - \frac{1}{T}\sum_{t=1}^T \text{profit}(a_t,d^\star) \\ &\quad \le B\sqrt{\frac{\log|E|}{2T}} + \frac{B(\log|E| + \overline{w^{-1}})}{T}, \end{aligned}$$

for all proactive defense strategies $d^\star \in \mathcal{D}_{B,E}$ where $\overline{w^{-1}} = |E|^{-1}\sum_{e\in E} w(e)^{-1}$, the mean of the surface reciprocals.

Remark 4. We can interpret Theorem 3 as establishing sufficient conditions under which a reactive defense strategy is within an additive constant of the best proactive defense strategy. Instead of carefully analyzing the system to construct the best proactive defense, the defender need only react to attacks in a principled manner to achieve almost the same quality of defense in terms of attacker profit.

We now detail the proof of this first main theorem. The standard algorithms in online learning assume that the rows of the game matrix are known in advance. Here, the edges are not known in advance and so we must relax this assumption using a simulation argument, which is perhaps the least obvious part of the reduction. The defense allocation chosen by Algorithm 1 at time t is precisely the same as the defense allocation that would have been chosen by Algorithm 2 had the defender run Algorithm 2 on the currently visible subgraph. The following lemma formalizes this equivalence. Note that Algorithm 1's parameter is reactive: it corresponds to Algorithm 2's parameter, but for the subgraph induced by the edges revealed so far. That is, β_t depends only on edges visible to the defender in round t, letting the defender actually run the algorithm.

Lemma 5. *Consider arbitrary round $t \in [T]$. If Algorithms 1 and 2 are run with parameters $\beta_k = (1+\sqrt{2\log|E_k|/(k+1)})^{-1}$ for $k \in [t]$ and parameter $\beta = (1+\sqrt{2\log|E_t|/(t+1)})^{-1}$, respectively, with the latter run on the subgraph induced by E_t, then the defense allocations $P_{t+1}(e)$ output by the algorithms are identical for all $e \in E_t$.*

Proof. If $e \in E_t$ then $\tilde{P}_{t+1}(e) = \beta^{\sum_{i=1}^{t} M(e,a_i)}$ because $\beta_t = \beta$, and the round $t+1$ defense allocation of Algorithm 1 P_{t+1} is simply $\tilde{P}_{t+1}$ normalized to sum to unity over edge set E_t, which is exactly the defense allocation output by Algorithm 2. □

Armed with this correspondence, we show that Algorithm 1 is almost as effective as Algorithm 2. In other words, hiding unattacked edges from the defender does not cause much harm to the reactive defender's ability to disincentivize the attacker.

Lemma 6. *If defense allocations $\{d_{1,t}\}_{t=1}^{T}$ and $\{d_{2,t}\}_{t=1}^{T}$ are output by Algorithms 1 and 2 with parameters $\beta_t = (1+\sqrt{2\log|E_t|/(t+1)})^{-1}$ for $t \in [T-1]$ and $\beta = (1+\sqrt{2\log|E|/(T)})^{-1}$, respectively, on a system $(V, E, w, \mathrm{reward}, s)$ and attack sequence $\{a_t\}_{t=1}^{T}$, then*

$$\frac{1}{T}\sum_{t=1}^{T}\mathrm{profit}(a_t, d_{1,t}) - \frac{1}{T}\sum_{t=1}^{T}\mathrm{profit}(a_t, d_{2,t}) \le \frac{B}{T}\overline{w^{-1}}.$$

Proof. Consider attack a_t from a round $t \in [T]$ and consider an edge $e \in a_t$. If $e \in a_k$ for some $k < t$, then the defense budget allocated to e at time t by Algorithm 2 cannot be greater than the budget allocated by Algorithm 1. Thus, the instantaneous cost paid by the attacker on e when Algorithm 1 defends is at least the cost paid when Algorithm 2 defends: $d_{1,t}(e)/w(e) \ge d_{2,t}(e)/w(e)$. If $e \notin \bigcup_{k=1}^{t-1} a_k$ then for all $k \in [t]$, $d_{1,k}(e) = 0$, by definition. The sequence $\{d_{2,k}(e)\}_{k=1}^{t-1}$ is decreasing and positive. Thus $\max_{k<t}(d_{2,k}(e) - d_{1,k}(e))$ is optimized at $k = 1$ and is equal to $B/|E|$. Finally since each edge $e \in E$ is first revealed exactly once this leads to

$$\begin{aligned}\sum_{t=1}^{T}\mathrm{cost}(a_t, d_{2,t}) - \sum_{t=1}^{T}\mathrm{cost}(a_t, d_{1,t}) \\ = \sum_{t=1}^{T}\sum_{e\in a_t}\frac{d_{2,t}(e) - d_{1,t}(e)}{w(e)} \\ \le \sum_{e\in E}\frac{B}{|E|w(e)}.\end{aligned}$$

Combined with the fact that the attacker receives the same payout whether Algorithm 2 or Algorithm 1 defends completes the result. □

Proof of Theorem 3. The result follows immediately from Lemma 1 and Lemma 6. □

Finally, notice that Algorithm 1 enjoys the same time and space complexities as Algorithm 2, up to constants.

5.3 Attacker ROA (Hidden Edges Case)

Reactive defense strategies can also be competitive with proactive defense strategies when we consider an attacker motivated by return on attack. The ROA formulation is appealing because (unlike with profit) the objective function does not require measuring attacker cost and defender budget in the same units. Our second main result considers the competitive ratio between the ROA for a reactive defense strategy and the ROA for the best proactive defense strategy.

Theorem 7. *The ROA against Algorithm 1 converges to the ROA against best proactive defense. Formally, consider the cumulative ROA*

$$\mathrm{ROA}(\{a_t\}_{t=1}^{T}, \{d_t\}_{t=1}^{T}) = \frac{\sum_{t=1}^{T}\mathrm{payoff}(a_t)}{\sum_{t=1}^{T}\mathrm{cost}(a_t, d_t)}.$$

(We abuse notation slightly and use singleton arguments to represent the corresponding constant sequence.) If defense allocations $\{d_t\}_{t=1}^{T}$ are output by Algorithm 1 with parameters $\beta_t = (1+\sqrt{2\log|E_t|/(t+1)})^{-1}$ on any system $(V, E, w, \mathrm{reward}, s)$ revealed online, such that $|E| > 1$, and any attack sequence $\{a_t\}_{t=1}^{T}$, then for all $\alpha > 0$ and proactive defense strategies $d^\star \in \mathcal{D}_{B,E}$

$$\frac{\mathrm{ROA}(\{a_t\}_{t=1}^{T}, \{d_t\}_{t=1}^{T})}{\mathrm{ROA}(\{a_t\}_{t=1}^{T}, d^\star)} \le 1 + \alpha,$$

provided T is sufficiently large.[2]

Remark 8. Notice that the reactive defender can use *the same algorithm* regardless of whether the attacker is motivated by profit or by ROA. As discussed in Section 7.2, the optimal proactive defense is not similarly robust.

We now translate our bounds on profit into bounds on ROA by observing that the ratio of two quantities is small if the quantities are large and their difference is small. We consider the competitive ratio between a reactive defense strategy and the best proactive defense strategy after the following technical lemma, which asserts that the quantities are large.

Lemma 9. *For all attack sequences $\{a_t\}_{t=1}^{T}$, $\max_{d^\star\in\mathcal{D}_{B,E}}\sum_{t=1}^{T}\mathrm{cost}(a_t, d^\star) \ge VT$ where game value V is*

$$\max_{d\in\mathcal{D}_{B,E}}\min_{a}\mathrm{cost}(a, d) = \frac{B}{\sum_{e\in\mathrm{inc}(s)} w(e)} > 0,$$

where $\mathrm{inc}(v) \subseteq E$ denotes the edges incident to vertex v.

Proof. Let $d^\star = \mathrm{argmax}_{d\in\mathcal{D}_{B,E}}\min_a \mathrm{cost}(a, d)$ witness the game's value V, then $\max_{d\in\mathcal{D}_{B,E}}\sum_{t=1}^{T}\mathrm{cost}(a_t, d) \ge \sum_{t=1}^{T}\mathrm{cost}(a_t, d^\star) \ge TV$. Consider the defensive allocation for each $e \in E$. If $e \in \mathrm{inc}(s)$, let $\tilde{d}(e) = Bw(e)/\sum_{e\in\mathrm{inc}(s)} w(e) > 0$, and otherwise $\tilde{d}(e) = 0$. This allocation is feasible because

$$\sum_{e\in E}\tilde{d}(e) = \frac{B\sum_{e\in\mathrm{inc}(s)} w(e)}{\sum_{e\in\mathrm{inc}(s)} w(e)} = B.$$

By definition $\tilde{d}(e)/w(e) = B/\sum_{e\in\mathrm{inc}(s)} w(e)$ for each edge e incident to s. Therefore, $\mathrm{cost}(a, \tilde{d}) \ge B/\sum_{e\in\mathrm{inc}(s)} w(e)$ for any nontrivial attack a, which necessarily includes at least one s-incident edge. Finally, $V \ge \min_a \mathrm{cost}(a, \tilde{d})$ proves

2. To wit: $T \ge (\frac{13}{\sqrt{2}}(1+\alpha^{-1})(\sum_{e\in\mathrm{inc}(s)} w(e)))^2 \log|E|$.

$$V \geq \frac{B}{\sum_{e\in \operatorname{inc}(s)} w(e)}.$$

Now, consider a defense allocation d and fix an attack a that minimizes the total attacker cost under d. At most one edge $e \in a$ can have $d(e) > 0$, for otherwise the cost under d can be reduced by removing an edge from a. Moreover any attack $a \in \operatorname{argmin}_{e\in \operatorname{inc}(s)} d(e)/w(e)$ minimizes attacker cost under d. Thus, the maximin V is witnessed by defense allocations that maximize $\min_{e\in \operatorname{inc}(s)} d(e)/w(e)$. This maximization is achieved by allocation $\tilde{d}$ and so Inequality (2) is an equality. □

We are now ready to prove the main ROA theorem:

Proof of Theorem 7. First, observe that for all $B > 0$ and all $A, C \in \mathbb{R}$

$$\frac{A}{B} \leq C \iff A - B \leq (C-1)B. \tag{3}$$

We will use this equivalence to convert the regret bound on profit to the desired bound on ROA. Together Theorem 3 and Lemma 9 imply

$$\begin{aligned} &\alpha \sum_{t=1}^{T} \operatorname{cost}(a_t, d_t) \\ &\quad \geq \alpha \max_{d^\star \in \mathcal{D}_{B,E}} \sum_{t=1}^{T} \operatorname{cost}(a_t, d^\star) \qquad (4)\\ &\qquad - \alpha \frac{B}{2}\sqrt{T \log|E|} - \alpha B(\log|E| + \overline{w^{-1}}) \\ &\geq \alpha V T - \alpha \frac{B}{2}\sqrt{T\log|E|} - \alpha B(\log|E| + \overline{w^{-1}}), \qquad (5)\end{aligned}$$

where $V = \max_{d\in\mathcal{D}_{B,E}} \min_a \operatorname{cost}(a, d) > 0$. If

$$\sqrt{T} \geq \frac{13}{\sqrt{2}}\left(1+\alpha^{-1}\right)\sqrt{\log|E|} \sum_{e\in\operatorname{inc}(s)} w(e),$$

we can use inequalities $V = B/\sum_{e\in\operatorname{inc}(s)} w(e)$, $\overline{w^{-1}} \leq 2\log|E|$ (since $|E| > 1$), and $\left(\sum_{e\in\operatorname{inc}(s)} w(e)\right)^{-1} \leq 1$ to show

$$\sqrt{T} \geq \frac{(1+\alpha)B + \sqrt{[(1+\alpha)B + 24\alpha V](1+\alpha)B}}{2\sqrt{2}\alpha V}\sqrt{\log|E|},$$

which combines with Theorem 3 and Inequality (5) to imply

$$\begin{aligned} &\alpha \sum_{t=1}^{T} \operatorname{cost}(a_t, d_t) \\ &\geq \alpha V T - \alpha\frac{B}{2}\sqrt{T\log|E|} - \alpha B(\log|E| + \overline{w^{-1}}) \\ &\geq \frac{B}{2}\sqrt{T\log|E|} + B(\log|E| + \overline{w^{-1}}) \\ &\geq \sum_{t=1}^{T}\operatorname{profit}(a_t, d_t) - \min_{d^\star\in\mathcal{D}_{B,E}} \sum_{t=1}^{T}\operatorname{profit}(a_t, d^\star) \\ &= \sum_{t=1}^{T}(-\operatorname{cost}(a_t, d_t)) - \min_{d^\star\in\mathcal{D}_{B,E}} \sum_{t=1}^{T}(-\operatorname{cost}(a_t, d^\star)) \\ &= \max_{d^\star\in\mathcal{D}_{B,E}} \sum_{t=1}^{T}\operatorname{cost}(a_t, d^\star) - \sum_{t=1}^{T}\operatorname{cost}(a_t, d_t).\end{aligned}$$

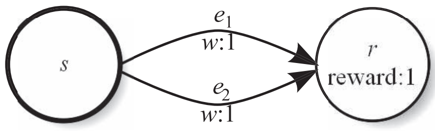


Fig. 3. The two node attack graph used to establish a lower bound for ROA in Section 6.

Finally, combining this equation with Equivalence (3) yields

$$\begin{aligned} &\frac{\operatorname{ROA}\left(\{a_t\}_{t=1}^{T}, \{d_t\}_{t=1}^{T}\right)}{\min_{d^\star\in\mathcal{D}_{B,E}} \operatorname{ROA}\left(\{a_t\}_{t=1}^{T}, d^\star\right)} \\ &\quad = \frac{\sum_{t=1}^{T}\operatorname{payoff}(a_t, d_t)}{\sum_{t=1}^{T}\operatorname{cost}(a_t, d_t)} \cdot \max_{d^\star\in\mathcal{D}_{B,E}} \frac{\sum_{t=1}^{T}\operatorname{cost}(a_t, d^\star)}{\sum_{t=1}^{T}\operatorname{payoff}(a_t, d^\star)} \\ &\quad = \frac{\max_{d^\star\in\mathcal{D}_{B,E}} \sum_{t=1}^{T}\operatorname{cost}(a_t, d^\star)}{\sum_{t=1}^{T}\operatorname{cost}(a_t, d_t)} \leq 1 + \alpha,\end{aligned}$$

proving the main result. □

6 Lower Bounds

In this section we use a two-vertex, two-edge graph to establish a lower bound on the competitive ratio of the attacker ROA for all reactive strategies. The lower bound shows that the analysis of Algorithm 1 is tight and that Algorithm 1 is optimal given the information available to the defender. The proof gives an example where the best proactive defense (slightly) out-performs every reactive strategy, suggesting the benchmark is not unreasonably weak.

To construct the lower bound, we first show in the following lemma that Algorithm 1 has optimal convergence time for small enough α, up to constants. (For very large α, Algorithm 1 converges in constant time, and therefore is optimal up to constants, vacuously.) The argument considers an attacker who randomly selects an attack path, rendering knowledge of past attacks useless, in the two-vertex graph of Fig. 3: let start vertex s be connected to a vertex r (with reward one) by two parallel edges e_1 and e_2, each with an attack surface of one. Further suppose that the defense budget $B = 1$.

Lemma 10. *For all reactive algorithms A, the competitive ratio C is at least $(x + \Omega(\sqrt{T}))/x$, i.e., at least $(T + \Omega(\sqrt{T}))/T$ because $x \leq T$.*

Proof. Consider the following random attack sequence: For each round, select an attack path uniform IID from the set $\{e_1, e_2\}$. A reactive strategy must commit to a defense in every round without knowledge of the attack, and therefore every strategy that expends the entire budget of one inflicts an expected cost of $1/2$ in every round. Thus, every reactive strategy inflicts a total expected cost of (at most) $T/2$, where the expectation is over the coin-tosses of the random attack process.

Given an attack sequence, however, there exists a proactive defense allocation with better performance. We can think of the proactive defender being prescient as to which edge (e_1 or e_2) will be attacked most frequently and allocating the entire defense budget to that edge. It is

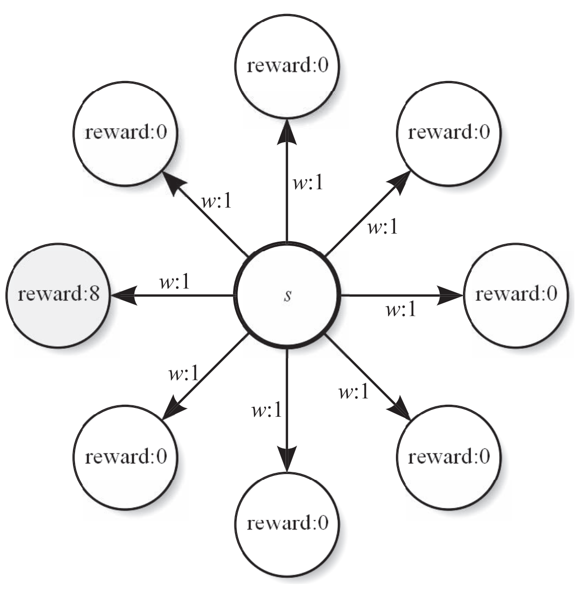


Fig. 4. Star-shaped attack graph with rewards concentrated in an unknown vertex.

well known (for instance via an analysis of a one-dimensional random walk) that in such a random process, one of the edges will occur $\Omega(\sqrt{T})$ more often than the other, in expectation.

By the probabilistic method, a property that is true in expectation must hold existentially, and, therefore, for every reactive strategy A, there *exists* an attack sequence such that A has a cost x, whereas the best proactive strategy (in retrospect) has a cost $x + \Omega(\sqrt{T})$. Because the payoff of each attack is 1, the total reward in either case is T. The prescient proactive defender, therefore, has an ROA of $T/(x + \Omega(\sqrt{T}))$, but the reactive algorithm has an ROA of T/x, establishing the lemma. □

Given this lemma, we show that Algorithm 1 is optimal given the information available. In this case, $n = 2$ and, ignoring constants from Theorem 7, we are trying to match a convergence time T is at most $(1 + \alpha^{-1})^2$, which is approximately α^{-2} for small α. For large enough T, there exists a constant c such that $C \geq (T + c\sqrt{T})/T$. By easy algebra, $(T + c\sqrt{T})/T \geq 1 + \alpha$ whenever $T \leq c^2/\alpha^2$, concluding the argument.

We can generalize the above argument of optimality to $n > 2$ using the combinatorial Lemma 3.2.1 from [18]. Specifically, we can show that for every n, there is an n edge graph for which Algorithm 1 is optimal up to constants for small enough α.

7 Advantages of Reactivity

In this section, we examine some situations in which a reactive defender outperforms a proactive defender. Proactive defenses hinge on the defender's model of the attacker's incentives. If the defender's model is inaccurate, the defender will construct a proactive defense that is far from optimal. By contrast, a reactive defender need not reason about the attacker's incentives directly. Instead, the reactive defender learns these incentives by observing the attacker in action.

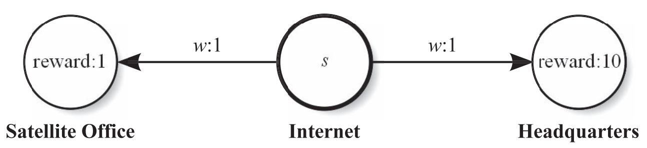


Fig. 5. An attack graph that separates the minimax strategies optimizing ROA and attacker profit.

7.1 Learning Rewards

One way to model inaccuracies in the defender's estimates of the attacker's incentives is to hide the attacker's rewards from the defender. Without knowledge of the payoffs, a proactive defender has difficulty limiting the attacker's ROA. Consider, for example, the star system whose edges have equal attack surfaces, as depicted in Fig. 4. Without knowledge of the attacker's rewards, a proactive defender has little choice but to allocate the defense budget equally to each edge (because the edges are indistinguishable). However, if the attacker's reward is concentrated at a single vertex, the competitive ratio for attacker's ROA (compared to the rational proactive defense) is the number of leaf vertices. (We can, of course, make the ratio worse by adding more vertices.) By contrast, the reactive algorithm we analyze in Section 4 is competitive with the rational proactive defense because the reactive algorithm effectively learns the rewards by observing which attacks the attacker chooses.

7.2 Robustness to Objective

Another way to model inaccuracies in the defender's estimates of the attacker's incentives is to assume the defender mistakes which of profit and ROA actually matter to the attacker. The defense constructed by a rational proactive defender depends crucially on whether the attacker's actual incentives are based on profit or based on ROA; whereas, the reactive algorithm we analyze in Section 4 is robust to this variation. In particular, consider the system depicted in Fig. 5, and assume the defender has a budget of nine. If the defender believes the attacker is motivated by profit, the rational proactive defense is to allocate the entire defense budget to the right-most edge (making the profit 1 on both edges). However, this defense is disastrous when viewed in terms of ROA because the ROA for the left edge is infinite (as opposed to near unity when the proactive defender optimizes for ROA).

7.3 Catachresis

The defense constructed by the rational proactive defender is optimized for a rational attacker. If the attacker is not perfectly rational, there is room for outperforming the rational proactive defense. There are a number of situations in which the attacker might not mount "optimal" attacks:

- The attacker might not have complete knowledge of the attack graph. Consider, for example, a software vendor who discovers five equally severe vulnerabilities in one of their products via fuzzing. According to proactive security, the defender ought to dedicate equal resources to repairing these five vulnerabilities. However, a reactive defender might dedicate more resources to fixing a vulnerability actually exploited by attackers in the wild. We can model these situations by making the attacker oblivious to some edges.

- The attacker might not have complete knowledge of the defense allocation. For example, an attacker attempting to invade a corporate network might target computers in human resources without realizing that attacking the customer relationship management database in sales has a higher return-on-attack because the database is lightly defended.

By observing attacks, the reactive strategy learns a defense tuned for the *actual* attacker, causing the attacker to receive a lower ROA.

8 Generalizations

In this section, we consider several extensions to our basic model for which results analogous to Theorems 3 and 7 can be shown.

8.1 Horn Clauses

Thus far, we have presented our results using a graph-based system model. Our results extend, however, to a more general system model based on Horn clauses. Datalog programs, which are based on Horn clauses, have been used in previous work to represent vulnerability-level attack graphs [19]. A Horn clause is a statement in propositional logic of the form $p_1 \wedge p_2 \wedge \cdots \wedge p_n \rightarrow q$. The propositions $p_1, p_2, \ldots, p_n$ are called the *antecedents*, and q is called the *consequent*. The set of antecedents might be empty, in which case the clause simply asserts the consequent. Notice that Horn clauses are negation free. In some sense, a Horn clause represents an edge in a hypergraph where multiple preconditions are required before taking a certain state transition.

In the Horn model, a system consists of a set of Horn clauses, an attack surface for each clause, and a reward for each proposition. The defender allocates defense budget among the Horn clauses. To mount an attack, the attacker selects a *valid proof*: an ordered list of rules such that each antecedent appears as a consequent of a rule earlier in the list. For a given proof Π,

$$\text{cost}(\Pi, d) = \sum_{c \in \Pi} d(c)/w(e) \quad \text{payoff}(\Pi) = \sum_{p \in [\![\Pi]\!]} \text{reward}(p),$$

where $[\![\Pi]\!]$ is the set of propositions proved by Π (i.e., those propositions that appear as consequents in Π). Profit and ROA are computed as before.

Our results generalize to this model directly. Essentially, we need only replace each instance of the word "edge" with "Horn clause" and "path" with "valid proof." For example, the rows of the matrix M used throughout the proof become the Horn clauses, and the columns become the valid proofs (which are numerous, but no matter). The entries of the matrix become $M(c, \Pi) = 1/w(c)$, analogous to the graph case. The one nonobvious substitution is $\text{inc}(s)$, which becomes the set of clauses that lack antecedents.

8.2 Multiple Attackers

We have focused on a security game between a *single* attacker and a defender. In practice, a security system might be attacked by several uncoordinated attackers, each with different information and different objectives. Fortunately, we can show that a model with multiple attackers is mathematically equivalent to a model with a single attacker with a randomized strategy: Use the set of attacks, one per attacker, to define a distribution over edges where the probability of an edge is linearly proportional to the number of attacks which use the edge. This precludes the interpretation of an attack as an s-rooted path, but our proofs do not rely upon this interpretation and our results hold in such a model with appropriate modifications.

8.3 Adaptive Proactive Defenders

A simple application of an online learning result [20] modifies our regret bounds for a proactive defender who reallocates budget a fixed number of times. In this model, our results remain qualitatively the same.

9 Related Work

Anderson [21] and Varian [22] informally discuss (via anecdotes) how the design of information security must take incentives into account. August and Tunca [23] compare various ways to incentivize users to patch their systems in a setting where the users are more susceptible to attacks if their neighbors do not patch.

Gordon and Loeb [24] and Hausken [25] analyze the costs and benefits of security in an economic model (with nonstrategic attackers) where the probability of a successful exploit is a function of the defense investment. They use this model to compute the optimal level of investment. Varian [26] studies various (single-shot) security games and identifies how much agents invest in security at equilibrium. Grossklags et al. [27] extend this model by letting agents self-insure.

Miura-Ko et al. [28] study externalities that appear due to users having the same password across various websites and discuss pareto-improving security investments. Miura-Ko and Bambos [29] rank vulnerabilities according to a random-attacker model. Skybox and RedSeal offer practical systems that help enterprises prioritize vulnerabilities based on a random-attacker model. Kumar et al. [30] investigate optimal security architectures for a multidivision enterprise, taking into account losses due to lack of availability and confidentiality. None of the above papers explicitly model a truly adversarial attacker.

Fultz and Grossklags [31] generalize [27] by modeling attackers explicitly. Cavusoglu et al. [32] highlight the importance of using a game-theoretic model over a decision theoretic model due to the presence of adversarial attackers. However, these models look at idealized settings that are not generically applicable. Lye and Wing [33] study the Nash equilibrium of a single-shot game between an attacker and a defender that models a particular enterprise security scenario. Arguably this model is most similar to ours in terms of abstraction level. However, calculating the Nash equilibrium requires detailed knowledge of the adversary's incentives, which as discussed in the introduction, might not be readily available to the defender. Moreover, their game contains multiple equilibria, weakening their prescriptions.

10 Conclusions

Many security experts equate reactive security with myopic bug-chasing and ignore principled reactive strategies when they recommend adopting a proactive approach to risk management. In this paper, we establish sufficient conditions for a learning-based reactive strategy to be competitive with the best fixed proactive defense. Additionally, we show that reactive defenders can outperform proactive defenders when the proactive defender defends against attacks that never actually occur. Although our model is an abstraction of the complex interplay between attackers and defenders, our results support the following practical advice for CISOs making security investments:

- Employ monitoring tools that let you detect and analyze attacks against your enterprise. These tools help focus your efforts on thwarting real attacks.
- Make your security organization more agile. For example, build a rigorous testing lab that lets you roll out security patches quickly once you detect that attackers are exploiting these vulnerabilities.
- When determining how to expend your security budget, avoid overreacting to the most recent attack. Instead, consider all previous attacks, but discount the importance of past attacks exponentially.

In some situations, proactive security can outperform reactive security. For example, reactive approaches are ill suited for defending against catastrophic attacks because there is no "next round" in which the defender can use information learned from the attack. We hope our results will lead to a productive discussion of the limitations of our model and the validity of our conclusions.

Instead of assuming that proactive security is always superior to reactive security, we invite the reader to consider when a reactive approach, or a blend of reactive and proactive elements, might be appropriate. For the parts of an enterprise where the defender's budget is liquid and there are no catastrophic losses, employing reactive elements is likely to improve security.

Acknowledgments

The authors would like to thank Elie Bursztein, Eu-Jin Goh, and Matt Finifter for their thoughtful comments and helpful feedback. We gratefully acknowledge the support of the US National Science Foundations (NSF) through the TRUST Science and Technology Center and grants DMS-0707060, CCF-0424422, 0311808, 0448452, and 0627511, and the support of the AFOSR through the MURI Program, and the support of the Siebel Scholars Foundation.

References

[1] J.P. Pironti, "Key Elements of an Information Security Program," *Information Systems Control J.*, vol. 1, 2005.

[2] K. Kark, J. Penn, and A. Dill, "2008 CISO Priorities: The Right Objectives but the Wrong Focus," *Le Magazine de la Sécurité Informatique*, Apr. 2009.

[3] C. Beard, "Introducing Test Pilot," http://labs.mozilla.com/2008/03/introducing-test-pilot/, Mar. 2008.

[4] M. Cremonini, "Evaluating Information Security Investments from Attackers Perspective: The Return-On-Attack (ROA)," *Proc. Fourth Workshop the Economics of Information Security*, 2005.

[5] A. Barth, B.I.P. Rubinstein, M. Sundararajan, J.C. Mitchell, D. Song, and P.L. Bartlett, "A Learning-Based Approach to Reactive Security," *Proc. 14th Int'l Conf. Financial Cryptography and Data Security (FC '10)*, pp 192-206, 2010.

[6] D. Fisher, "Multi-Process Architecture," http://dev.chromium.org/developers/design-documents/multi-process-architecture, July 2008.

[7] J. Friedberg, "Internet Fraud Battlefield," http://www.ftc.gov/bcp/workshops/proofpositive/Battlefield_Overview.pdf, Apr. 2007.

[8] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G.M. Voelker, V. Paxson, and S. Savage, "Spamalytics: An Empirical Analysis of Spam Marketing Conversion," *Proc. ACM Conf. Computer and Comm. Security*, pp. 3-14, 2008.

[9] B. Warner, "Home PCs Rented Out in Sabotage-for-Hire Racket," *Reuters*, July 2004.

[10] J. Franklin, V. Paxson, A. Perrig, and S. Savage, "An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants," *Proc. ACM Conf. Computer and Comm. Security*, pp. 375-388, 2007.

[11] M. Howard, "Attack Surface: Mitigate Security Risks by Minimizing the Code You Expose to Untrusted Users," *MSDN Magazine*, http://msdn.microsoft.com/en-us/magazine/cc163882.aspx, Nov. 2004.

[12] E. Rescorla, "Is Finding Security Holes a Good Idea?," *IEEE Security and Privacy*, vol. 3, no. 1, pp. 14-19, Jan./Feb. 2005.

[13] D. Chakrabarty, A. Mehta, and V.V. Vazirani, "Design is As Easy As Optimization," *Proc. 33rd Int'l Colloquium Automata, Languages and Programming (ICALP)*, pp. 477-488, 2006.

[14] E. Ordentlich and T.M. Cover, "The Cost of Achieving the Best Portfolio in Hindsight," *Math. of Operations Research*, vol. 23, no. 4, pp. 960-982, 1998.

[15] Y. Freund and R.E. Schapire, "Adaptive Game Playing Using Multiplicative Weights," *Games and Economic Behavior*, vol. 29, pp. 79-103, 1999.

[16] Y. Freund and R. Schapire, "A Short Introduction to Boosting," *J. Japanese Soc. for Artificial Intelligence*, vol. 14, no. 5, pp. 771-780, 1999.

[17] N. Cesa-Bianchi, Y. Freund, D. Haussler, D.P. Helmbold, R.E. Schapire, and M.K. Warmuth, "How to Use Expert Advice," *J. Assoc. for Computing Machinery*, vol. 44, no. 3, pp. 427-485, May 1997.

[18] N. Cesa-Bianchi, Y. Freund, D.P. Helmbold, D. Haussler, R.E. Schapire, and M.K. Warmuth, "How to Use Expert Advice," *Proc. 25th Ann. ACM Symp. Theory of Computing*, pp. 382-391, 1993.

[19] X. Ou, W.F. Boyer, and M.A. McQueen, "A Scalable Approach to Attack Graph Generation," *Proc. 13th ACM Conf. Computer and Comm. Security*, pp. 336-345, 2006.

[20] M. Herbster and M.K. Warmuth, "Tracking the Best Expert," *Machine Learning*, vol. 32, no. 2, pp. 151-178, 1998.

[21] R. Anderson, "Why Information Security Is Hard—An Economic Perspective," *Proc. 17th Ann. Computer Security Applications Conf.*, pp. 358-365, 2001.

[22] H.R. Varian, "Managing Online Security Risks," New York Times, June 1 2000.

[23] T. August and T.I. Tunca, "Network Software Security and User Incentives," *Management Science*, vol. 52, no. 11, pp. 1703-1720, 2006.

[24] L.A. Gordon and M.P. Loeb, "The Economics of Information Security Investment," *ACM Trans. Information and System Security*, vol. 5, no. 4, pp. 438-457, 2002.

[25] K. Hausken, "Returns to Information Security Investment: The Effect of Alternative Information Security Breach Functions on Optimal Investment and Sensitivity to Vulnerability," *Information Systems Frontiers*, vol. 8, no. 5, pp. 338-349, 2006.

[26] H. Varian, "System Reliability and Free Riding," *Economics of Information Security*, vol. 12, pp. 1-16, 2001.

[27] J. Grossklags, N. Christin, and J. Chuang, "Secure or Insure?: A Game-Theoretic Analysis of Information Security Games," *Proc. 17th Int'l Conf. World Wide Web*, pp. 209-218, 2008.

[28] R.A. Miura-Ko, B. Yolken, J. Mitchell, and N. Bambos, "Security Decision-Making among Interdependent Organizations," *Proc. 21st IEEE Computer Security Foundations Symp.*, pp. 66-80, 2008.

[29] R. Miura-Ko and N. Bambos, "SecureRank: A Risk-Based Vulnerability Management Scheme for Computing Infrastructures," *Proc. IEEE Int'l Conf. Comm.*, pp. 1455-1460, June 2007.

[30] V. Kumar, R. Telang, and T. Mukhopadhyay, "Optimal Information Security Architecture for the Enterprise," http://ssrn.com/abstract=1086690, 2011.

[31] N. Fultz and J. Grossklags, "Blue Versus Red: Towards a Model of Distributed Security Attacks," *Proc. 13th Int'l Conf. Financial Cryptography and Data Security,* 2009.

[32] H. Cavusoglu, S. Raghunathan, and W. Yue, "Decision-Theoretic and Game-Theoretic Approaches to IT Security Investment," *J. Management Information Systems,* vol. 25, no. 2, pp. 281-304, 2008.

[33] K.-w. Lye and J.M. Wing, "Game Strategies in Network Security," *Proc. Foundations of Computer Security Workshop,* pp. 13-22, 2002.

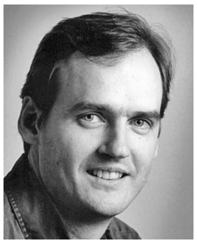
Adam Barth received the PhD degree in computer science from Stanford University. He is a software engineer at Google, Inc. Prior to joining Google, he was a postdoctoral fellow at UC Berkeley. His research focuses on web security, privacy, and risk management.

Benjamin I.P. Rubinstein Prior to joining MSR in 2010, he received the PhD degree at UC Berkeley under Peter Bartlett. He is a researcher in the Interaction & Intent Group at Microsoft Research Silicon Valley. His dissertation research focused on machine learning in adversarial environments, and applications of learning in computer security. During this time he was awarded the Yahoo! Key Scientific Challenges Award in Adversarial Machine Learning, Best Poster Award at RAID08, and a Siebel Scholars fellowship. His current interests extend to applications of machine learning in privacy & security, search, measurement, social network analysis, and basic research questions in learning & statistics.

Mukund Sundararajan received the PhD degree in computer science from Stanford University. He is a research scientist at Google, Inc. His research interests include foundations of market design, market analysis, electronic commerce and privacy.

John C. Mitchell is the Mary and Gordon Crary Family professor in the Stanford Computer Science Department. His research in computer security focuses on trust management, privacy, security analysis of network protocols, and web security. He has also worked on programming language analysis and design, formal methods, and other applications of mathematical logic to computer science. He is currently involved in the multiuniversity PORTIA research project to study privacy concerns in databases and information processing systems, and the US National Science Foundations (NSF) TRUST Center.

Dawn Song is an associate professor of computer science at UC Berkeley. Prior to joining UC Berkeley, she was an assistant professor at Carnegie Mellon University from 2002 to 2007. Her research interest lies in security and privacy issues in computer systems and networks, including areas ranging from software security, networking security, database security, distributed systems security, to applied cryptography. She is the recipient of various awards including the MacArthur Fellowship, the Guggenheim Fellowship, the US National Science Foundation (NSF) CAREER Award, the Alfred P. Sloan Research Fellowship, the MIT Technology Review TR-35 Award, the IBM Faculty Award, the George Tallman Ladd Research Award, the Okawa Foundation Research Award, and the Li Ka Shing Foundation Women in Science Distinguished Lecture Series Award. She is also the author of multiple award papers in top security conferences, including the best paper award at the USENIX Security Symposium and the highest ranked paper at the IEEE Symposium on Security and Privacy. She is a member of the IEEE.



Peter L. Bartlett is a professor in the Division of Computer Science and Department of Statistics at the University of California at Berkeley. He is the coauthor, with Martin Anthony, of the book *Learning in Neural Networks: Theoretical Foundations*, has edited three other books, and has coauthored many papers in the areas of machine learning and statistical learning theory. He has served as an associate editor of the journals *Machine Learning*, *Mathematics of Control Signals and Systems*, *the Journal of Machine Learning Research*, *the Journal of Artificial Intelligence Research*, and *the IEEE Transactions on Information Theory,* as a member of the editorial boards of Machine Learning, *the Journal of Artificial Intelligence Research*, and Foundations and Trends in Machine Learning, and as a member of the steering committees of the Conference on Computational Learning Theory and the Algorithmic Learning Theory Workshop. He has consulted to a number of organizations, including General Electric, Telstra, and SAC Capital Advisors. In 2001, he was awarded the Malcolm McIntosh Prize for Physical Scientist of the Year in Australia, for his work in statistical learning theory. He was a Miller Institute visiting research professor in statistics and computer science at UC Berkeley in Fall 2001, and a fellow, senior fellow, and professor in the Research School of Information Sciences and Engineering at the Australian National University's Institute for Advanced Studies (1993-2003), and an honorary professor in the School of Information Technology and Electrical Engineering at the University of Queensland. His research interests include machine learning, statistical learning theory, and adaptive control. He is a member of the IEEE.