A Learning-Based Approach to Reactive Security

Adam Barth[1], Benjamin I. P. Rubinstein[1], Mukund Sundararajan[3], John C. Mitchell[4], Dawn Song[1], and Peter L. Bartlett[1,2]

[1]Computer Science Division [2]Department of Statistics, UC Berkeley
[3] Google Inc., Mountain View, CA
[4]Department of Computer Science, Stanford University

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.

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 [28, 21] 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 7), 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 [4].

In our model, we find that two properties are sufficient for a reactive strategy to perform as well as the best proactive strategies. First, no single attack is catastrophic, meaning 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. Second, the defender's budget is *liquid*, meaning the defender can re-allocate resources without penalty. 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 graph edge, 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 making this assumption, 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. We make the 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 viewpoint 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 [8]. 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 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. Our results

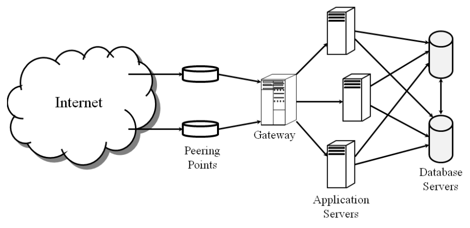

Fig. 1.1. An attack graph representing an enterprise data center.

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.

Organization. Section 2 formalizes our model. Section 3 shows that perimeter defense and defense-in-depth arise naturally in our model. Section 4 presents our main results bounding the competitive ratio of reactive versus proactive defense strategies. Section 5 outlines scenarios in which reactive security out-performs proactive security. Section 6 generalizes our results to Horn clauses and multiple attackers. Section 7 relates related work. Section 8 concludes.

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 Figure 1.1), components in a complex software system [9], or an Internet Fraud “Battlefield” [13].

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 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 6. 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 Figure 1.1, 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.

Incentives and Rewards. Attackers respond to incentives. For example, attackers compromise machines and form botnets because they make money from spam [20] or rent the botnet to others [32]. Other attackers steal credit card numbers because credit card numbers have monetary value [10]. We model the attacker's incentives by attaching a non-negative *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: $\text{payoff}(a) = \sum_{v \in a} \text{reward}(a)$. In the example from Figure 1.1, if an attacker compromises both a front-end server and the database server, the attacker receives both rewards.

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 (CISO) 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* [19] 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 edge: $\text{cost}(a, d) = \sum_{e \in a} d(e)/w(e)$.

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 [29]. 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.

Objective. To evaluate defense strategies, we measure the attacker's incentive for attacking using the *return-on-attack* (ROA) [8], 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 on ROA for simplicity.

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.

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: $\mathrm{argmin}_d \max_a \mathrm{ROA}(a, d)$.

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

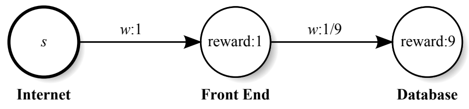

Fig. 3.1. Attack graph representing a simplified data center network.

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 [7]:

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

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

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.

Defense in Depth. Many experts in security practice recommend that defenders employ defense in depth. Defense in depth rises naturally in our model as an optimal defense for some systems. Consider, for example, the system depicted in Figure 3.1. This attack graph is a simplified version of the data center network depicted in Figure 1.1. 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 5).

4 Reactive Security

To analyze reactive security, we model the attacker and defender as playing an iterative game, 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 [26], playing zero-sum games [12], and boosting other machine learning heuristics [11]. Although we provide a few technical extensions, our main contribution comes from applying results from online learning to risk management.

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 $\text{cost}(a_t, d_t)$ and gains $\text{payoff}(a_t)$.

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.

Defender Knowledge. Unlike proactive defenders, reactive defenders do not know all of the vulnerabilities that exist in the system in advance. (If defenders had complete knowledge of vulnerabilities, conferences such as Black Hat Briefings would serve little purpose.) 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 doing a post-mortem analysis of intrusion logs. 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.

Algorithm 1 A reactive defense strategy for hidden edges.

```
– Initialize $E_0 = \emptyset$
– For each round $t \in \{2, ..., T\}$
    • 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.
```

Algorithm. Algorithm 1 is a reactive defense strategy based on the multiplicative update learning algorithm [6, 12]. 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 re-allocates 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.

The algorithm begins without any knowledge of the graph whatsoever, and so allocates no defense budget to the system. Upon the t^{th} 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 weight 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 parameter β controls how aggressively the defender reallocates defense budget to recently attacked edges. If β 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 β is enormous, the defender will not be very agile and, instead, leave the defense budget in the initial allocation. For an appropriate value of β, the algorithm will converge to the optimal defense strategy. For instance, the min cut in the example from Section 3.

Theorems. To compare this reactive defense strategy to all proactive defense strategies, we use the notion of *regret* from online learning theory. The following is an additive regret bound relating the attacker's profit under reactive and proactive defense strategies.

Theorem 1 *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* $\beta_s = \left(1 + \sqrt{2 \log |E_s| / (s+1)}\right)^{-1}$ *on any system* $(V, E, w, \mathrm{reward}, s)$ *revealed online and any attack sequence* $\{a_t\}_{t=1}^T$*, then*

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

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 2 *We can interpret Theorem 1 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.*

Reactive defense strategies can also be competitive with proactive defense strategies when we consider an attacker motivated by return on attack (ROA). 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. The next result considers the competitive ratio between the ROA for a reactive defense strategy and the ROA for the best proactive defense strategy.

Theorem 3 *The ROA against Algorithm 1 converges to the ROA against best proactive defense. Formally, consider the cumulative ROA:*

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

(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* $\beta_s = \left(1 + \sqrt{2 \log |E_s| / (s+1)}\right)^{-1}$ *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* $d^\star \in \mathcal{D}_{B,E}$

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

provided T *is sufficiently large.*[1]

Remark 4 *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 5 the optimal proactive defense is not similarly robust.*

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

We present proofs of these theorems in the full version [3]. We first prove the theorems in the simpler setting where the defender knows the entire graph. Second, we remove the hypothesis that the defender knows the edges in advance.

Lower Bounds. In the full version [3], we use a two-vertex, two-edge graph to establish a lower bound on the competitive ratio of the 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 algorithm. The proof gives an example where the best proactive defense (slightly) out-performs every reactive strategy, suggesting the benchmark is not unreasonably weak.

5 Advantages of Reactivity

In this section, we examine some situations in which a reactive defender out-performs 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.

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 Figure 5.1. 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.

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 Figure 5.2, and assume the defender has a budget of 9. 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).

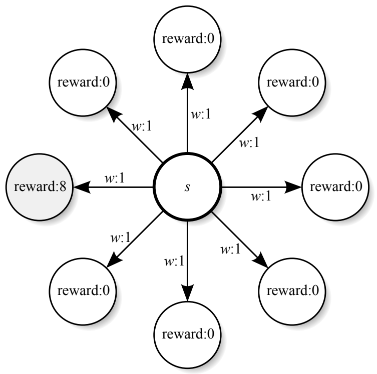

Fig. 5.1. Star-shaped attack graph with rewards concentrated in an unknown vertex.

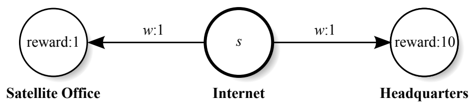

Fig. 5.2. An attack graph that separates the minimax strategies optimizing ROA and attacker profit.

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 out-performing 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.

6 Generalizations

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 [27]. 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 pre-conditions 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,

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

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 non-obvious substitution is inc(s), which becomes the set of clauses that lack antecedents.

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.

Adaptive Proactive Defenders. A simple application of an online learning result [18], omitted due to space constraints, modifies our regret bounds for a proactive defender who re-allocates budget a fixed number of times. In this model, our results remain qualitatively the same.

7 Related Work

Anderson [1] and Varian [31] informally discuss (via anecdotes) how the design of information security must take incentives into account. August and Tunca [2] 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 [15] and Hausken [17] analyze the costs and benefits of security in an economic model (with non-strategic 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 [30] studies various (single-shot) security games and identifies how much agents invest in security at equilibrium. Grossklags [16] extends this model by letting agents self-insure.

Miura et al. [24] study externalities that appear due to users having the same password across various websites and discuss pareto-improving security investments. Miura and Bambos [25] 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. [22] investigate optimal security architectures for a multi-division enterprise, taking into account losses due to lack of availability and confidentiality. None of the above papers explicitly model a truly adversarial attacker.

Fultz [14] generalizes [16] by modeling attackers explicitly. Cavusoglu et al. [5] 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 [23] 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.

8 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 out-perform 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 out-perform 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 might be appropriate. For the parts of an enterprise where the defender's budget is liquid and there are no catastrophic losses, a carefully constructed reactive strategy can be as effective as the best proactive defense in the worst case and significantly better in the best case.

Acknowledgments We 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 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. Anderson, R.: Why information security is hard—An economic perspective. 17th Annual Computer Security Applications Conference pp. 358–365 (2001)
2. August, T., Tunca, T.I.: Network software security and user incentives. Management Science 52(11), 1703–1720 (2006)
3. Barth, A., Rubinstein, B.I.P., Sundararajan, M., Mitchell, J.C., Song, D., Bartlett, P.L.: A learning-based approach to reactive security (2009), `http://arxiv.org/abs/0912.1155`
4. Beard, C.: Introducing Test Pilot (March 2008), `http://labs.mozilla.com/2008/03/introducing-test-pilot/`
5. Cavusoglu, H., Raghunathan, S., Yue, W.: Decision-theoretic and game-theoretic approaches to IT security investment. Journal of Management Information Systems 25(2), 281–304 (2008)
6. Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D.P., Schapire, R.E., Warmuth, M.K.: How to use expert advice. Journal of the Association for Computing Machinery 44(3), 427–485 (May 1997)
7. Chakrabarty, D., Mehta, A., Vazirani, V.V.: Design is as easy as optimization. In: 33rd International Colloquium on Automata, Languages and Programming (ICALP). LNCS 4051, vol. Part I, pp. 477–488 (2006)
8. Cremonini, M.: Evaluating information security investments from attackers perspective: the return-on-attack (ROA). In: Fourth Workshop on the Economics of Information Security (2005)
9. Fisher, D.: Multi-process architecture (July 2008), `http://dev.chromium.org/developers/design-documents/multi-process-architecture`
10. Franklin, J., Paxson, V., Perrig, A., Savage, S.: An inquiry into the nature and causes of the wealth of internet miscreants. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security. pp. 375–388. ACM, New York, NY, USA (2007)

11. Freund, Y., Schapire, R.: A short introduction to boosting. Journal of the Japanese Society for Artificial Intelligence 14(5), 771–780 (1999)
12. Freund, Y., Schapire, R.E.: Adaptive game playing using multiplicative weights. Games and Economic Behavior 29, 79–103 (1999)
13. Friedberg, J.: Internet fraud battlefield (April 2007), `http://www.ftc.gov/bcp/workshops/proofpositive/Battlefield_Overview.pdf`
14. Fultz, N., Grossklags, J. (eds.): Blue versus Red: Towards a model of distributed security attacks. Proceedings of the Thirteenth International Conference Financial Cryptography and Data Security (February 2009)
15. Gordon, L.A., Loeb, M.P.: The economics of information security investment. ACM Transactions on Information and System Security 5(4), 438–457 (2002)
16. Grossklags, J., Christin, N., Chuang, J.: Secure or insure?: A game-theoretic analysis of information security games. In: Proceeding of the 17th international conference on World Wide Web. pp. 209–218. ACM, New York, NY, USA (2008)
17. Hausken, K.: Returns to information security investment: The effect of alternative information security breach functions on optimal investment and sensitivity to vulnerability. Information Systems Frontiers 8(5), 338–349 (2006)
18. Herbster, M., Warmuth, M.K.: Tracking the best expert. Machine Learning 32(2), 151–178 (1998)
19. Howard, M.: Attack surface: Mitigate security risks by minimizing the code you expose to untrusted users. MSDN Magazine (November 2004), `http://msdn.microsoft.com/en-us/magazine/cc163882.aspx`
20. Kanich, C., Kreibich, C., Levchenko, K., Enright, B., Voelker, G.M., Paxson, V., Savage, S.: Spamalytics: An empirical analysis of spam marketing conversion. In: Proceedings of the 2008 ACM Conference on Computer and Communications Security. pp. 3–14. ACM, New York, NY, USA (2008)
21. Kark, K., Penn, J., Dill, A.: 2008 CISO priorities: The right objectives but the wrong focus. Le Magazine de la Sécurité Informatique (April 2009)
22. Kumar, V., Telang, R., Mukhopadhyay, T.: Optimal information security architecture for the enterprise, `http://ssrn.com/abstract=1086690`
23. Lye, K.w., Wing, J.M.: Game strategies in network security. In: Proceedings of the Foundations of Computer Security Workshop. pp. 13–22 (2002)
24. Miura-Ko, R.A., Yolken, B., Mitchell, J., Bambos, N.: Security decision-making among interdependent organizations. In: Proceedings of the 21st IEEE Computer Security Foundations Symposium. pp. 66–80. IEEE Computer Society, Washington, DC, USA (2008)
25. Miura-Ko, R., Bambos, N.: SecureRank: A risk-based vulnerability management scheme for computing infrastructures. In: Proceedings of IEEE International Conference on Communications. pp. 1455–1460 (June 2007)
26. Ordentlich, E., Cover, T.M.: The cost of achieving the best portfolio in hindsight. Mathematics of Operations Research 23(4), 960–982 (1998)
27. Ou, X., Boyer, W.F., McQueen, M.A.: A scalable approach to attack graph generation. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. pp. 336–345 (2006)
28. Pironti, J.P.: Key elements of an information security program. Information Systems Control Journal 1 (2005)
29. Rescorla, E.: Is finding security holes a good idea? IEEE Security and Privacy 3(1), 14–19 (2005)
30. Varian, H.: System reliability and free riding (2001)
31. Varian, H.R.: Managing online security risks. New York Times. Jun 1, 2000
32. Warner, B.: Home PCs rented out in sabotage-for-hire racket. Reuters (July 2004)