SAND2011-7545C

Computing Optimal Security Strategies in Networked Domains: A Cost-Benefit Approach

Joshua Letchford
Duke University
Durham, NC
jcl@cs.duke.edu

Yevgeniy Vorobeychik*
Sandia National Laboratories
Livermore, CA
yvorobe@sandia.gov

ABSTRACT

There has been much recent work on computing optimal security strategies in Stackelberg (leader-follower) models of security. Techniques to date, however, generally fail to explicitly account for interdependence between the targets to be secured, which is of vital importance in a variety of domains, including cyber, supply chain, and critical infrastructure security. We introduce a novel framework for computing optimal randomized security policies in networked domains which extends previous approaches in two ways. First, we extend previous linear programming techniques for Stackelberg security games to incorporate benefits and costs of arbitrary security configurations on individual assets. Second, we offer a principled model of failure cascades that allows us to capture both the direct and indirect value of assets. We use our framework to analyze four models, two based on random graph generation models, a simple model of interdependence between critical infrastructure and key resource sectors, and a model of the Fedwire interbank payment network.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed artificial intelligence—*Intelligent agents*

General Terms

Algorithms, Performance, Economics, Security

Keywords

Game theory, Security, Stackelberg Games, Networks, Cybersecurity

1. INTRODUCTION

Game theoretic approaches to security have received much attention in recent years. There have been numerous attempts to distill various aspects of the problem into a model that could then be solved in closed form, particularly accounting for interdependencies of security decisions [14, 7, 17, 20]. Numerous others, however, offer techniques based on mathematical programming to solve actual instances of security problems. Approaches to network interdiction [4, 23], for example, offer (usually) an integer programming formulation solving for an location of sensors that optimally interdict traffic (such as drug traffic) through a network.

Our point of departure is a different line of work that develops linear and integer programming methods for optimal randomized allocation of security resources among possible attack targets. In this work, the assumption is made that the defender is able to commit to a randomized policy which is subsequently observed by the attacker who optimally responds to it. Initial work on the subject offered an approach relying on multiple linear programs to compute such an optimal *commitment* strategy in general two-player games [3]. Follow-up work developed single linear and integer programming formulations for Bayesian and complete information settings [19, 2] and attempted to exploit the special structure of security scenarios to build faster, more scalable algorithms [12, 8, 13], with some finding use in actual security deployments [10]. Within this stream of work, there are also approaches for security in network settings, but the semantics of the networks is rather different from ours: the defender aims to thwart an attacker traversing a network towards a target [22, 9].

2. STACKELBERG SECURITY GAMES

A Stackelberg security game consists of two players, the leader (defender) and the follower (attacker), and a set of possible targets. The leader can decide upon a randomized policy of defending the targets, possibly with limited defense resources. The follower (attacker) is assumed to observe the randomized policy of the leader, but not the realized defense actions. Upon observing the leader's strategy, the follower chooses a target so as to maximize its expected utility.

In past work, Stackelberg security game formulations focused on defense policies that were costless, but resource bounded. Specifically, it had been assumed that the defender has K fixed resources available with which to cover (subsets of) targets. Additionally, security decisions amounted to covering a set of targets, or not. While in numerous settings to which such work has been applied (e.g., airport security, federal air marshall scheduling) this formulation is very

*Sandia National Labs, Livermore, CA. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

reasonable, in other settings one may choose among many *security configurations* for each valued asset, and, additionally, security resources are only available at some cost.[1] For example, in cybersecurity, protecting computing nodes could involve setting anti-virus and/or firewall configuration settings, with stronger settings carrying a benefit of better protection, but at a cost of added inconvenience, lost productivity, as well as possible licensing costs. Indeed, costs on resources may usefully take place of resource constraints, since such constraints are often not hard, but rather channel an implicit cost of adding further resources.

To formalize, suppose that the defender can choose from a finite set O of security configurations for each target $t \in T$, with $|T| = n$. A configuration $o \in O$ for target $t \in T$ incurs a cost $c_{o,t}$ to the defender. If the attacker happens to attack t while configuration o is in place, the expected value to the defender is denoted by $U_{o,t}$, while the attacker's value is $V_{o,t}$. A key assumption in Stackelberg security games is that the targets are completely independent: that is, a joint defender and attacker decision concerning one target has no impact on the values of others, and total defender and attacker utilities are additive over all targets. We revisit this assumption below when we turn to networked (interdependent) settings. We denote by $q_{o,t}$ the probability that the defender chooses o at target t, while a_t denotes the probability that the attacker attacks target t.

3. COMPUTING OPTIMAL RANDOMIZED SECURITY CONFIGURATIONS

Previous formulations of Stackelberg security games involved a fixed collection of defender resources, and in most cases a binary decision to be made for each target: to cover it, or not. To adapt these to our domains of interest, we first modify the well-known multiple linear program formulation to incorporate an arbitrary set of security configurations together with their corresponding costs of deployment. In the multiple-LP formulation, each linear program solves for an optimal randomized defense strategy *for a fixed attacker target* t, with the constraint that t is an optimal choice for the attacker. The defender then chooses the best solution from all feasible LPs as his optimal randomized defense configuration. At this point we make a crucial assumption that allows our approach to scale to tackle realistic problem instances: the utilities of both players only depend on which target was chosen by the attacker and the security configuration at that target. This assumption, which avoids an exponential blow-up of the defender pure strategy space, parallels the independence assumption made by Kiekintveld et al. [12]. The resulting formulation as n LPs is shown in Equations 1-4.

$$\forall_{\hat{t}} \ \max \quad \sum_o U_{o,\hat{t}} q^{\hat{t}}_{o,\hat{t}} - \sum_t \sum_o c_{o,t} q^{\hat{t}}_{o,t} \tag{1}$$

s.t.

$$\forall_{o,t} \quad q^{\hat{t}}_{o,t} \in [0,1] \tag{2}$$

$$\forall_t \quad \sum_o q^{\hat{t}}_{o,t} = 1 \tag{3}$$

$$\forall_t \quad \sum_o V_{o,t} q^{\hat{t}}_{o,t} \leq \sum_o V_{o,\hat{t}} q^{\hat{t}}_{o,\hat{t}} \tag{4}$$

The intuition behind this is that in an optimal solution, the attacker must (weakly) prefer to attack some target, and consequently, one of these LPs must correspond to an optimal defense policy. (We note that a MILP ERASER formulation [12] can be modified similarly to allow for multiple defense configurations and a benefits-costs optimization.[2])

3.1 Single Linear Program Formulation

Starting with the multiple-LP formulation above we construct a single LP that aggregates all of these (this formulation is different from a single-LP version by Conitzer and Korzhyk [2] in that it does not require any post processing to derive the optimal defense strategy). We cannot do so immediately, however, because some of the n LPs may actually be infeasible: some targets may not be optimal for the attacker for any defense policy. Consequently, we must prune out all such targets in order to ensure that the combined LP is feasible. Formally, it suffices to check, for each target $\hat{t}$ that

$$\max_o V_{o,\hat{t}} \geq \max_t \min_o V_{o,t}, \tag{5}$$

that is, that $\hat{t}$ is not strictly dominated for the attacker. Let $\hat{T} \subset T$ be the set of targets for which Equation 5 holds. The aggregate single-LP formulation is then shown in Equations 6-9.

$$\max \quad \sum_{\hat{t} \in \hat{T}} \left(\sum_o U_{o,\hat{t}} q^{\hat{t}}_{o,\hat{t}} - \sum_t \sum_o c_{o,t} q^{\hat{t}}_{o,t} \right) \tag{6}$$

s.t.

$$\forall_{\hat{t},o,t} \quad q^{\hat{t}}_{o,t} \in [0,1] \tag{7}$$

$$\forall_{\hat{t},t} \quad \sum_o q^{\hat{t}}_{o,t} = 1 \tag{8}$$

$$\forall_{\hat{t},t} \quad \sum_o V_{o,t} q^{\hat{t}}_{o,t} \leq \sum_o V_{o,\hat{t}} q^{\hat{t}}_{o,\hat{t}} \tag{9}$$

Notice that we can easily incorporate additional linear constraints in any of these formulations. For example, it is often useful to add a budget constraint of the form:

$$\forall_{\hat{t},t} \quad \sum_o c_{o,t} q^{\hat{t}}_{o,t} \leq B.$$

4. INCORPORATING NETWORK STRUCTURE

[1] Security configurations may be firewall or anti-virus settings, or could even serve the purpose of introducing non-linear costs of resources spent on security. For example, we can introduce several levels of defense intensity, with higher levels having a higher *marginal* cost (i.e., having a convex cost function).

[2] An important computational advantage of using a cost-benefits formulation here is that we do not need to concern ourselves with enforcing the constraint on a fixed number of available resources during actual sampling, a non-trivial problem in its own right.

Thus far, a key assumption has been that the utility of the defender and the attacker for each target depends only on the defense configuration for that target, as well as whether it is attacked or not. In many domains, such as cybersecurity and supply chain security, key assets are fundamentally interdependent, with an attack on one target having potential consequences for others. In this section, we show how to transform certain important classes of problems with interdependent assets into a formulation in which targets become effectively independent, for the purposes of our solution techniques.

Below we focus on the defender's utilities; attacker is treated identically. Let w_t be an *intrinsic worth* of a target to the defender, that is, how much loss the defender would suffer if this target were to be compromised with no other target affected (i.e., not accounting for indirect effects). In doing so, we assume that these worths are independent for different targets. Let $s = \{o_1, \ldots, o_n\}$ be the security configuration on all nodes. Assuming that the utility function is additive in target-specific worths and the attacker can only attack a single target, we can write it as

$$U_t(s) = E[\sum_{t'} w_{t'} 1(\text{t' affected} \mid \text{s, t})] = \sum_{t'} w_{t'} z_{s,t'}(t),$$

where $1(\cdot)$ is an indicator function and $z_{s,t'}(t)$ is the marginal probability that target t' is affected when the attacker attacks target t. From this expression, it is apparent that in general, $U_t(s)$ depends on defense configurations at all targets, creating an intractable large space of configurations over which the defender has to reason. We now make the crucial assumption that enables fast computation of defender policies by recovering inter-target independence.

ASSUMPTION 1. *For all t and t', $z_{s,t'}(t) = z_{o_t,t'}(t)$.*

In words, the probability that a target t' is affected when t is attacked only depends on the security configuration at the attacked target t. Below, we use a shorthand o instead of o_t where t is clear from context.

A way to interpret our assumption is that once some target is compromised, the fault may spread to other assets in spite of good protection policies. This assumption was operational in other work on interdependent security [14], where a justification is through a story about airline baggage screening: baggage that is transferred between airlines is rarely thoroughly screened, perhaps due to the expense. Thus, even while an airline may have very strong screening policies, it is poorly protected from luggage entering its planes via transfers. Cybersecurity has similar shortcomings: defense is often focused on external threats, with little attention paid to threats coming from computers internal to the network. Thus, once a computer on a network is compromised, the attacker may find it much easier to compromise others on the same network. The problem is exacerbated by the use of common operating environments, since once an exploit is found, it can often be reused to compromise other computing resources on a common network.

Under the above assumption, we can write the defender utility when t is attacked under security configuration o as,

$$U_{o,t} = z_{o,t}(t) w_t + \sum_{t' \neq t} z_{o,t'}(t) w_{t'}.$$

By a similar argument and an analogous assumption for the attacker's utility, we thereby recover target independence required by the linear programming formulations above.

4.1 Cascading Failures Model

In general, one may use an arbitrary model to compute or estimate $U_{o,t}$ above. Indeed, often simulation tools are available to perform the analysis of global consequences of attacks on particular pieces of the infrastructure [6]. Nevertheless, we offer a specific model of interdependence between targets that is simple, natural, and applies across a wide variety of settings.

Suppose that dependencies between targets are represented by a graph (T, E), with T the set of targets (nodes) as above, and E the set of edges (t, t'), where an edge from t to t' (or an undirected edge between them) means that target t' depends on target t (and, thus, a successful attack on t may have impact on t'). Each target has associated with it a worth, w_t as above, although in this context this worth is incurred only if t is affected (compromised, affected by a flaw that spreads from one of its dependencies, etc). The security configuration determines the probability $z_{o,t}(t)$ that target t is compromised (affected) if the attacker attacks it *directly* and the defense configuration is o. We model the interdependencies between the nodes as independent cascade contagion, which has previously been used primarily to model diffusion of product adoption and infectious disease [11, 5].[3] The contagion proceeds starting at an attacked node t, affecting its network neighbors t' each with probability $p_{t,t'}$. The contagion can only occur once along any network edge, and once a node is affected, it stays affected through the diffusion process. The simple way to conceive of this is to start with the network (T, E) and then remove each edge (t, t') with probability $(1 - p_{t,t'})$. The entire connected component of an attacked node is then deemed affected.

4.2 Computing Expected Utilities

Given the independent cascade model of interdependencies between targets, we must compute expected utilities, $U_{o,t}$ and $V_{o,t}$, of the defender and the attacker respectively. In general, we can do so by simulating cascades starting at every node t, with expected utilities estimated as sample average utilities over K simulated cascades (expectation in this case is with respect to random realizations of attack success for specific targets as well as edges that become a part of the failure contagion. In the special case when the network is a tree, however, we can compute these exactly. A naive algorithm can do it in linear time for a given target t, yielding quadratic time in total (since we must do this for all targets). In fact, we can do this in linear time *for all targets*, as the following theorem asserts.

THEOREM 1. *If (T, E) is an undirected tree we can calculate the expected utilities at all targets in $O(n)$ time.*

The proof of this result is in the appendix.

5. EXAMPLE: A SIMPLE SUPPLY CHAIN

In this section we illustrate the tools introduced above with a simple example. Consider a seven-node supply chain (directed acyclic graph) shown in Figure 1. We suppose that the entire supply chain (or at least the relevant security decisions) is controlled by a single firm which is primarily

[3] A similar model was also used by Mounzer et al. [15] in the context of risk modeling and management in organizations.

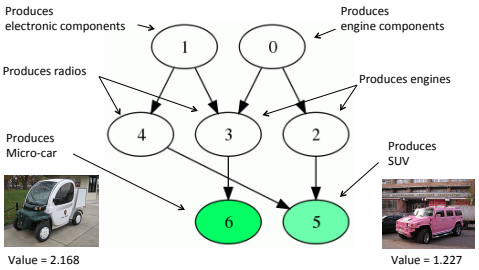

Figure 1: A simple automotive supply chain example.

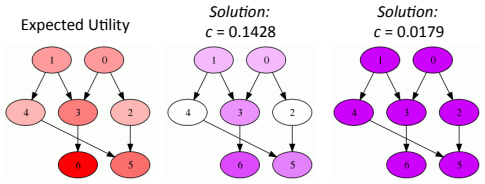

Figure 2: Randomized defense configurations for the simple supply chain example under two defense cost scenarios.

concerned with manufacturing two types of cars, one more profitable than the other. The actual components that ultimately comprise the cars are not intrinsically valuable to the manufacturer (or are valued so low relative to the final product as to make them effectively unimportant in this decision). All parts of the supply chain may be inspected at some cost c, or not (in which case no cost is incurred).

The first step in our framework is to compute (or estimate) the expected utility for each node in the supply chain. To do this, we first specify the probability that an attacked node is affected (in this case, becomes faulty), $z_{o,t}(t)$. We let $z_{o,t}(t) = 1$ when node t is not inspected and $z_{o,t}(t) = 0$ when it is. Next, we must specify the contagion probabilities for each edge. We use $p_{t,t'} = 0.5$ for all edges here. The results are color coded in Figure 2 (left): the darker colors correspond to more valuable nodes. Note that while intrinsic worth is only ascribed to the final products, all components carry some value, due to their indirect impact on the final product (for example, a faulty part will, with some probability, make the component which uses it faulty as well). Supposing now that the game is zero-sum, the expected utilities of the attacker are completely determined by the defender's utilities, and we can use these as an input into the linear programs above. We show the results for two different inspection costs, $c_{high} = 0.1428$ and $c_{low} = 0.0179$ in Figure 2. The higher cost setting (Figure 2, middle) yields a security configuration in which five of the seven nodes incur some probability of inspection, with the heavier colors corresponding to high inspection probability. The low-cost setting (Figure 2, right) yields a solution in which every node is defended with probability 1.

6. EXPERIMENTS IN INTERDEPENDENT SECURITY ANALYSIS

In this section we apply our framework to several networked domains. First, we consider networks generated from two major generative random graph models: Erdos-Renyi (ER) and Preferential Attachment (PA) [16]. In the ER model every directed link is made with a specified and fixed probability p. The PA which adds nodes in a fixed sequence, starting from an arbitrary seed graph with at least two vertices. Each node i is attached to m others stochastically (unless $i \leq m$, in which case it is connected to all preceding nodes), with probability of connecting to a node j proportional to the degree of j, d_j. In a generalized version of this model that we consider below, connection probabilities are $(d_j)^\mu$, such that when $\mu = 0$ the degree distribution is relatively homogeneous, just as in ER, $\mu = 1$ recovers the "standard" PA model, and large values of μ correspond to highly inhomogeneous degree distributions. Throughout, we use $\mu = 1$ unless otherwise specified.

In addition, we explore three networks derived from real security settings: one with 18 nodes that models dependencies between critical infrastructure and key resource sectors (CIKR), as inferred from the DHS and FEMA websites, the second with 66 nodes that captures payments between banks in the core of the Fedwire network [21], and the third a snapshot of the autonomous systems (AS) network using Oregon routeviews [18] containing 6474 targets and 13233 edges.

For the randomly generated networks, all data presented is averaged over 100 graph samples. Since we generate graphs that may include undirected cycles, we obtain expected utilities for all nodes using 10,000 simulated cascades. (We later revisit this issue and show that this is more than a sufficient number for problems of this scale). Intrinsic worths w_t are generated uniformly randomly. Cascade probabilities $p_{t,t'}$ were set to 0.5 unless otherwise specified. For the CIKR network, each node was somewhat arbitrarily assigned a low, medium, and high worth of 0.2, 0.5, and 1, respectively, based on perceived importance (for example, the energy sector was assigned a high worth, while the national monuments and icons sector a low worth). Each edge was categorized based on the importance of the dependency (as gleaned from the DHS and FEMA websites) as "highly" and "moderately" significant, with cascade probabilities of 0.5 and 0.1 respectively. For the Fedwire network, all nodes were assigned an equal worth of 0.5, and cascade probabilities were discretely chosen between 0.05 and 0.5 in 0.05 increments depending on the weight of the corresponding edges shown in [21]. Finally, for the AS network, we generated values for each node uniformly randomly on a unit interval, and our experiments give an average of 10 such assignments. Cascade probabilities $p_{t,t'}$ were fixed at 0.5 for all edges.

We restrict the defender the defender to two security configurations at every target, one with a cost of 0 which stops attacks 0% of the time and one with a cost of c which prevents attacks 100% of the time, and, additionally, that we have a zero-sum game between the defender and attacker.

6.1 Evaluation of Heuristics on the Autonomous Systems Network

Besides offering a framework for computing optimal randomized security strategies on networks, we also proposed several heuristics for identifying nodes to secure. In this section, our purpose is to evaluate the quality of these heuristics, both relative to each other, and relative to the optimal solution. The hope is that if a particular heuristic is highly effective, it may allow our general approach to scale to net-

works much larger than those we can currently handle in a reasonable amount of time. A secondary purpose of this section is to illustrate that our framework already scales to a graph with thousands of nodes. We therefore perform our evaluation on a real instance (snapshot) of an autonomous systems graph (AS), which is a much larger graph than those used in the remainder of the experiments section. Indeed, it took us under 1 hour to solve optimally on this graph for a fixed setting of node values; the results we report are averages over 10 such instances. To facilitate a fair comparison, we first obtained an optimal solution for a given instance and fixed the corresponding total cost C. We then applied the proposed greedy heuristics iteratively until we exhausted the cost budget C.

As we can see fro Figure 3, most of the heuristics do rather poorly.[4] For example, the widely used heuristic which ranks nodes in order of their degrees is much worse than optimal when defense costs are sufficiently high, even in its fractional instantiation. The reason is simply that security decision is driven both by connectivity and intrinsic valuations, and the latter is not in any way captured such a heuristic. Interestingly, a non-fractional greedy heuristic which allocates security to nodes in order of their expected utility does just as badly as the degree-based heuristic. In contrast, a *fractional* greedy heuristic is remarkably efficacious: it is only about 4% worse than optimal when defense costs are high.

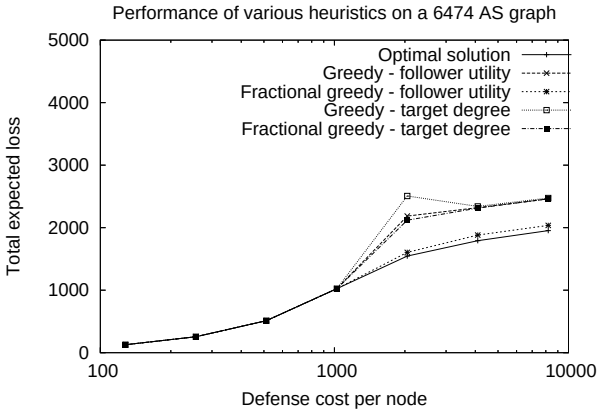

Figure 3: Expected total loss under various heuristics.

6.2 The Impact of Marginal Defense Cost

Our first analysis endeavor is to study the impact of marginal defense cost c on defender expected losses, his total costs, and the sum of these (i.e., negative expected utility). The results for ER and BA (both with 100 nodes and average degree of 2), as well as CIKR and Fedwire networks are shown in Figure 4. All the plots feature a clear pattern: expected loss and (negative) utility are monotonically increasing, as expected, while total costs start at zero, initially rise, and ultimately fall (back to zero in 3 of the 4 cases). It may at first be surprising that total costs eventually fall even as marginal costs continue to decrease, but this clearly must be the case: when c is high enough, the defender will not wish to invest in security at all, and total costs will be zero. What is much more suprising is the presence of a dual-peak in Preferential Attachment and Fedwire networks. Both these networks share the property that there is a non-negligible fraction of nodes with a very high connectivity [16, 21]. When the initial peak is reached, the network is fully defended, and as marginal costs rise further, the defender begins to reduce the defense resources expected on less important targets. At a certain point, only the most connected targets are protected, and since these are so vital to protect, total costs begin increasing again. After the second peak is reached, c is finally large enough to discourage the defender from fully protecting even the most important targets, and the subsequent fall of total costs is no longer reversed.

[4]All approaches are nearly identical if defense costs are low, since then defense budget suffices to defend all important nodes. Interesting cases arise when there is a meaningful tradeoff to be made in optimizing defense configurations.

6.3 Resilience to Targeted Attacks: Impact of Network Structure

One of the important streams in the broad network science literature is the question of relative resilience of different network topologies to failures, random or targeted. A key results, replicated in a number of contexts, is that network topology is a vital factor in determining resilience [1, 16]. Of particular interest to us is the observation that scale-free networks such as PA exhibit poor tolerance to targeted attacks as compared to ER, which is precisely the context that we consider.

In Figure 5 we show the defender's utility for three different network topologies, PA, ER, and Fedwire as a function of cost c. Remarkably, there is essentially no difference between PA and ER (and not much between these and Fedwire) until c is quite high, at which point they begin to diverge. This seems to contradict essentially all the previous findings in that network topology seems to play little role in resilience in our case! A superficial difference here is that we consider a cascading failure model, where most of the previous work on the subject focused on diminished connectivity due to attacks. We contend that the most important distinction, however, is that previous work on the subject did not account for a simple observation that most important targets are also most heavily defended; indeed, there was no notion of endogenous defense at all. In scale-free graphs, there are well connected nodes failure of which has global consequences. These are the nodes which are most important, and are heavily defended in optimal decisions prescribed by our framework. Once the defense decision becomes endogenous, differences in network topology disappear. Naturally, once c is high enough, defense of important targets weakens, and eventually we recover the standard result: for high c, Preferential Attachment is considerably more vulnerable than Erdos-Renyi.

To investigate the impact of network topology on resilience further, we consider the generalized PA model in which we systematically vary the homogeneity of degree distribution by way of the parameter μ. The results are shown in Figure 6. In this graph, we do observe clear variation in resilience as a function of network topology, but the operational factor in this variation is homogeneity in the distribution of expected utilities, rather than degrees: increasing *homogeneity* of the utility distribution *lowers network resilience*. This seems precisely the opposite of the standard results in network resilience, but the two are in fact closely related, as we now demonstrate. Superficially, the trend in

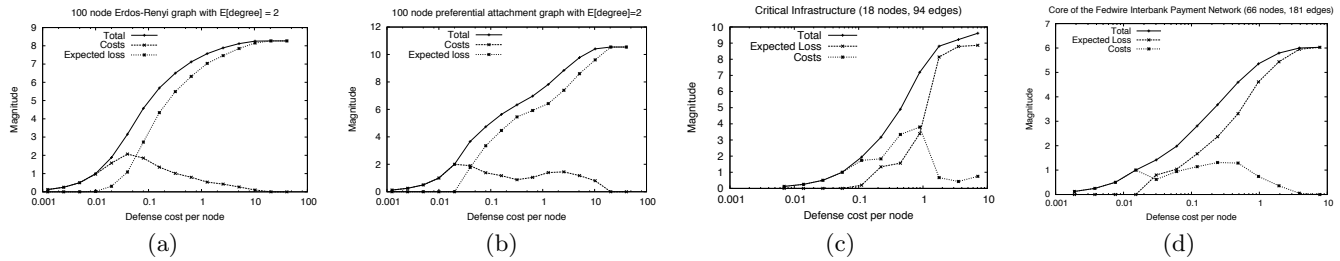

Figure 4: Expected loss, cost, and their sum in 100-node Erdos-Renyi (a), 100-node PA (b), 18-node critical infrastructure (c), and 66-node core of the fedwire (d) networks as defense cost increases. The results for ER and PA are averages over 100 stochastic realizations of these networks. ER have edge probability of 0.02.

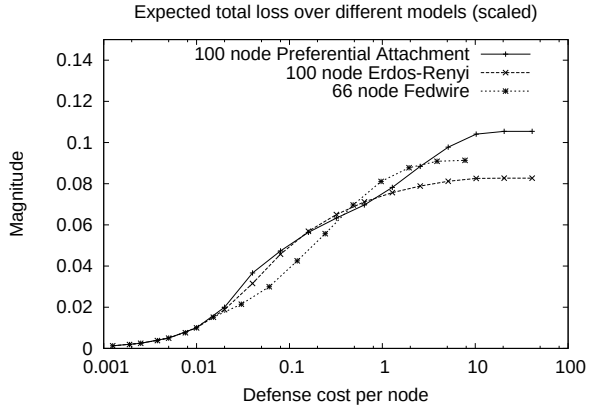

Figure 5: Expected total loss: comparison across different network structures.

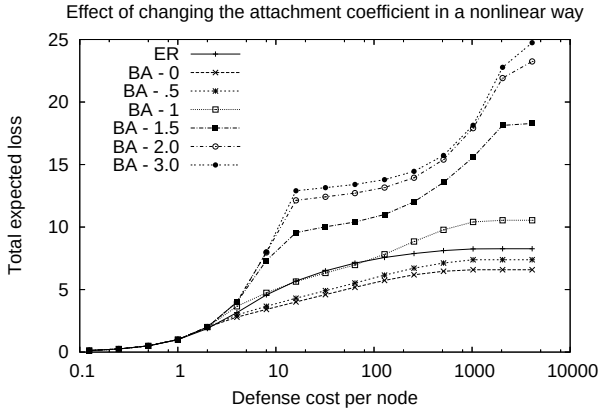

Figure 6: Expected defender disutility in the generalized PA model as we vary μ (keeping average degree fixed at 2). ER is also shown for comparison.

the figure seems to follow the common intuition in the resilience literature: as the degree distribution becomes more inhomogeneous (more star-like), it becomes more difficult to defend. Observe, however, that ER is actually more difficult to defend than PA with $\mu = 0$. The lone difference of the latter from ER is the fact that nodes that enter earlier are more connected and, therefore, the PA variant should actually be more *inhomogeneous* than ER! The answer is that random connectivity combined with inhomogeneity actually makes the distribution of *utilities* less homogeneous, and, as a result, fewer nodes on which defense can focus. On the other hand, as the graph becomes more star-like, the utilities of all nodes become quite similar; in the limiting case, all nodes are only two hops apart, and attacking any one of them yields a loss of many as a result of cascades.

There is another aspect of network topology that has an important impact on resilience: network density. Figure 7 shows a plot of an Erdos-Renyi network with the probability of an edge varying between 0.0025 to 0.08 (average degree between .25 and 8) and cost c fixed at 0.04. Clearly, expected utility and loss of the defender are increasing in density, but it is rather surprising to observe how sharply they jump once average degree exceeds 1 (the ER network threshold for a large connected component); in any case, network density has an unmistakable impact. The reason is intuitive: increased density means more paths between targets, and, consequently, greater likelihood of large cascades in the event any target is compromised. Total cost initially increases in response to increased density, in part to compensate for the increased vulnerability to attacks, but eventually falls, since it is too expensive to protect everything, and anything short of that seems largely ineffective.

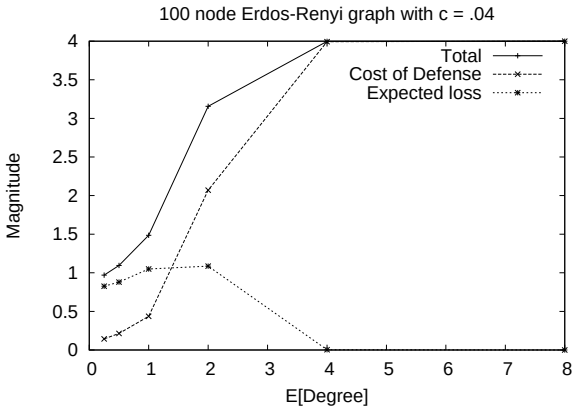

Figure 7: Expected loss, cost, and their sum in 100-node Erdos-Renyi networks as a function of network density (equivalently, expected degree).

6.4 Sampling Efficiency

Another question that we raised earlier is how many simulation samples are sufficient to have confidence in the results? In our experiments we used 10000, which seems large, but whether it's enough for the networks we consider is unclear. Figure 8 offers strong evidence that 10000 is more than enough samples; indeed, it seems that 100 would suffice in networks of comparable size.

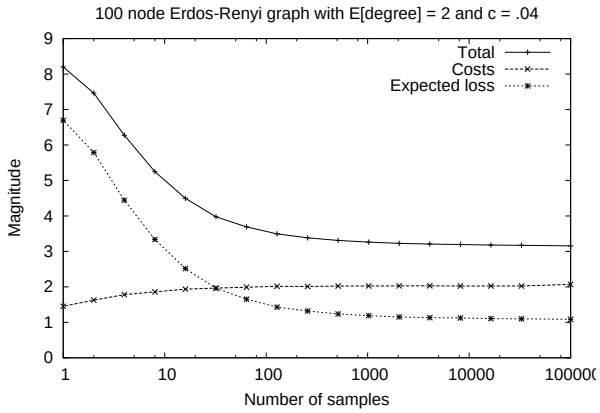

Figure 8: Expected loss, cost, and the sum under incomplete information about network structure.

7. EXTENSIONS

In this section we contemplate several extensions of our basic framework. The first introduces a possibility that with some probability, known to the defender, there is no attacker and failure happens "exogenously", according to some distribution over all targets (also known to the defender). The second extends our framework to allow an attacker to have multiple capabilities among which one is chosen for an attack. Our third extension allows us to capture, in a limited way, the possibility of an attacker simultaneously compromising more than one target as a part of the initial attack. We present our extensions in the single-LP formulation, noting that they are just as easy to apply to the other formulations.

7.1 Attacker vs. Nature

Introducing nature, or exogenous failures, into the picture is the simplest extension to make. Let r be the probability that target failures happen as a result of an intelligent attack and $1-r$ be the probability that they happen according to a known distribution g_t over targets. We need only to change the objective of the linear program to capture this to be

$$\max \sum_{\hat{t}} \left[r \left(\sum_o U_{o,\hat{t}} q_{o,\hat{t}}^{\hat{t}} \right) + (1-r) \left(\sum_{t,o} g_t U_{o,t} q_{o,t}^{\hat{t}} \right) - \sum_t \sum_o c_{o,t} q_{o,t}^{\hat{t}} \right]$$

7.2 Multiple Attacker Capabilities

Suppose that the attacker has a set of capabilities K_t that can be used to attack a target t. Incorporating this into our linear programming formulation is straightforward, if somewhat messy. In essence, we extend the set of targets to be a cross product between targets and capabilities, but must, of course, take care not to assign defense resources to capabilities, but to targets only. To keep things simpler, we only alter the original formulation of the problem, keeping in mind that it is direct to combine all our extensions. The modified objective becomes

$$\max \sum_{\hat{t},\hat{k}} \left(\sum_o U_{o,\hat{t},\hat{k}} q_{o,\hat{t}}^{\hat{t},\hat{k}} - \sum_t \sum_o c_{o,t} q_{o,t}^{\hat{t},\hat{k}} \right).$$

The constraints on attacker's expected utility are also modified:

$$\forall_{(\hat{t},\hat{k}),(t,k)} \sum_o V_{o,t,k} q_{o,t}^{\hat{t},\hat{k}} \leq \sum_o V_{o,\hat{t},\hat{k}} q_{o,\hat{t}}^{\hat{t},\hat{k}}$$

7.3 Attacking Multiple Targets

Allowing attackers to attack multiple targets is known to be NP-Hard in general—indeed, the attacker's decision alone is NP-Hard in such a setting [11]. However, there is a limited extent to which we can incorporate this possibility into our framework. Suppose that there are two kinds of nodes, a set A of attacks and a set T of targets, and suppose that an attacker can only activate a node in A, corresponding to a particular attack method he can use (e.g., a denial-of-service attack). The semantics of defense strategies in this setting that preserve the crucial assumption of independence we have made is that we can only defend attack nodes, but not targets. However, this redefined model allows an attacker to direct attack multiple targets, since there can be multiple edges between a node in A and set T. This graphical structure is of great value if the attacker has a limited number of possible subsets of all targets he can aspire to attack directly, and nothing is gained if the attacker can attack any possible subset of targets.

8. CONCLUSION

We present a framework for computing optimal randomized security policies in network domains, extending previous linear programming approaches to Stackelberg security games in two ways. First, we construct a single linear programming formulation which incorporates costs of arbitrary security configurations and may be extended to enforce budget constraints. Second, we demonstrate how to transform a general setting with interdependent assets into a security game with independent targets, allowing us to leverage the compact linear programming formulation for security games. We apply our framework to study four models of interdependent security. Two are based on standard generative models of random graphs, and two others use real networks representing interdependent assets. We show that there is a surprising bi-modal behavior of expected utility in preferential attachment networks as defense costs increase, that such networks lead to greater expected losses than Erdos-Renyi networks when defense costs are high. We also demonstrate that increased network density has substantial deleterious effect on expected losses of targeted attacks and that, as a result, highly interdependent networks such as that representing critical infrastructure sector dependence are extremely difficult to defend. Finally, we show that having some information, even very limited, about the true network structure can be of substantial value in guiding high quality defense decisions.

9. REFERENCES

[1] Reka Albert, Hawoong Jeong, and Albert-Laszlo Barabasi. Error and attack tolerance of complex networks. *Nature*, 406:378–382, 2000.

[2] Vincent Conitzer and Dmytro Korzhyk. Commitment to correlated strategies. In *Twenty-Fifth National Conference on Artificial Intelligence*, pages 632–637, 2011.

[3] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, EC '06, pages 82–90, New York, NY, USA, 2006. ACM.

[4] Kelly J. Cormican, David P. Morton, and R. Kevin Wood. Stochastic network interdiction. *Operations Research*, 46(2):184–197, 1998.

[5] Peter S. Dodds and Duncan J. Watts. A generalized model of social and biological contagion. *Journal of Theoretical Biology*, 232:587–604, 2005.

[6] Donald D. Dudenhoeffer, May R. Permann, and Milos Manic. CIMS: A framework for infrastructure interdependency modeling and analysis. In *Winter Simulation Conference*, pages 478–485, 2006.

[7] Jens Grossklags, Nicolas Christin, and John Chuang. Secure or insure? A game-theoretic analysis of information security games. In *Seventeenth International World Wide Web Conference*, pages 209–218, 2008.

[8] Manish Jain, Erim Kardes, Christopher Kiekintveld, Milind Tambe, and Fernando Ordonez. Security games with arbitrary schedules: A branch and price approach. In *Twenty-Fourth National Conference on Artificial Intelligence*, 2010.

[9] Manish Jain, Dmytro Korzhyk, Ondrej Vanek, Vincent Conitzer, Michal Pechoucek, and Milind Tambe. A double oracle algorithm for zero-sum security games on graphs. In *Tenth International Conference on Autonomous Agents and Multiagent Systems*, 2011.

[10] Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordóñez. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces*, 40:267–290, July 2010.

[11] David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence in a social network. In *Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.

[12] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *In AAMAS-09*, 2009.

[13] Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, 41:297–327, 2011.

[14] Howard Kunreuther and Geoffrey Heal. Interdependent security. *Journal of Risk and Uncertainty*, 26(2-3):231–249, 2003.

[15] Jeffrey Mounzer, Tansu Alpcan, and Nicholas Bambos. Integrated security risk management for it-intensive organizations. In *Sixth International Conference on Information Assurance and Security*, pages 329–334, 2010.

[16] Mark Newman. *Networks: An Introduction*. Oxford University Press, 2010.

[17] Kien C. Nguyen, Tansu Alpcan, and Tamer Basar. Stochastic games for security in networks with interdependent nodes. In *International Conference on Game Theory for Networks*, 2009.

[18] U. of Oregon Route Views Project. Online data and reports.

[19] Praveen Paruchuri. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In *In AAMAS*, 2008.

[20] Sankardas Roy, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, Vivek Shandilya, and Qishi Wu. A survey of game theory as applied to network security. In *Forty-Third Hawaii International Conference on System Sciences*, pages 1–10, 2010.

[21] Kimmo Soramaki, Morten L. Bech, Jeffrey Arnold, Robert J. Glass, and Walter Beyeler. The topology of interbank payment flows. *Physica A*, 379:317–333, 2007.

[22] Jason Tsai, Zhengyu Yin, Jun young Kwak, David Kempe, Christopher Kiekintveld, and Milind Tambe. Urban security: Game-theoretic resource allocation in networked physical domains. In *Twenty-Fourth National Conference on Artificial Intelligence*, 2010.

[23] David L. Woodruff, editor. *Network Interdiction and Stochastic Integer Programming*. Kluwer Academic Publishers, 2003.

APPENDIX

A. PROOFS

A.1 Proof of Theorem 1

In this proof we use a shorthand $z_{o,t}$ for $z_{o,t}(t)$. Let us define the neighbors of a target t as N_t. By definition, the expected utility of a given node t ($U_{o,t}$) is the direct utility at that node ($z_{o,t}w_t$) plus the expected utility due to the cascading failure. The expected utility due to cascading failure is

$$z_{o,t} \sum_{t' \neq t} w_{t'} p(failure(t')|t)$$

where $p(failure(t')|t)$ is the probability a node t' fails if node t fails. Since this is a tree, there is only one path between any pair of nodes, which means we can express $p(failure(t')|t)$ as the product of probabilities of the edges on the path between t and t'. Next, let us consider the set of paths generated by each pair of nodes in the tree. If we organize these paths by the edges they contain (and use linearity of expectation), we can express the expected utility of the contagion spreading across an edge (t, t'), $E[U_{(t,t')}]$, as:

$$E[U_{(t,t')}] = p_{t,t'} \left(w_{t'} + \sum_{t'' \in N_{t'}, t'' \neq t} E[U_{(t',t'')}] \right). \quad (10)$$

Thus, we can reason that for each node t:

$$U_{o,t} = z_{o,t} U_t,$$

where

$$U_t = w_t + \sum_{t' \in N_t} E[U_{(t,t')}] \quad (11)$$

Now let us describe a two-pass algorithm for calculating U_t for all t. First, choose an arbitrary node to be the root of the tree. In the first pass, we calculate the expected loss due to each edge from parent to child from the bottom of the tree upward. In the second pass, we calculate the expected loss on each edge from child to parent from the top of the tree downward. We can model this as a message passing algorithm, where calculating $E[U_{(t,t')}]$ is done by passing a message from t' to t. We can see by Equation 10 that the necessary inputs to calculate $E[U_{(t,t')}]$ are the messages from $N_{t'} \setminus t$ to t'. We will now show that at the time that each of these messages is generated, all of the necessary inputs will be available.

Consider the edges between a given node t and its neighbors. Unless t is the root, one of these edges will be between t and its parent P_t, and the rest (possibly 0 in the case where t is a leaf node) will be between t and its children. Since in the first pass we are passing messages from child to parent and a node has only one parent, we will have received messages from $N_t \setminus P_t$ when we generate the message from t to P_t.

For the second pass, when we pass information to a child of t, C_t, we will have received messages from N_t, thus we again have the necessary information to generate the message from t to C_t.

Finally, once a node has received messages from all of its neighbors we can easily calculate the expected loss at each node by Equation 11. However, to achieve a runtime of $O(n)$, we need to be slightly more clever in how we store these values. By combining Equations 10 and 11 we can reason that $E[U_{(t,t')}] = p_{t,t'}(U_{t'} - E[U_{(t',t)}])$. This allows us to give an equivalent definition of U_t:

$$U_t = w_t + \sum_{t \in N_t} p_{t,t'}(U_{t'} - E[U_{(t',t)}]). \tag{12}$$

Now, consider the same two-pass algorithm as before, but rather than storing the expected loss for every edge, we merely store a running total of the expected loss at each node. We argue that by the same reasoning as before that the necessary calculations will have been performed before we need them as inputs. However, we still need to show that we can recover the correct value out of the values stored at the two nodes. When we calculate $E[U_{(P,C)}]$ in the first pass, the value stored at C will be $(U_C - E[U_{(C,P)}])$, since we have not yet updated C with $E[U_{(C,P)}]$. However, when we reach this edge on the downward pass to to calculate $E[U_{(C,P)}]$, P will have U_P stored. Since the value stored at C is still $(U_C - E[U_{(C,P)}])$, we can easily calculate $E[U_{(C,P)}] = p_{C,P}(U_P - E[U_{(P,C)}]) = p_{C,P}(U_P - p_{P,C}(U_C - E[U_{(C,P)}]))$ and update C.

Since we visit each edge twice, and perform a constant amount of work each time, we can bound the runtime by $O(|E|)$. Since in a tree $|T| - 1 = |E|$, we can also bound the runtime by $O(n)$.