Hardening Critical Infrastructure Networks Against Attacker Reconnaissance

Kartik Palani and David M. Nicol

Information Trust Institute
University of Illinois at Urbana-Champaign
{palani2,dmnicol}@illinois.edu

Abstract. The knowledge an attacker gathers about the critical infrastructure network they infiltrate allows them to customize the payload and remain undetected while causing maximum impact. This knowledge is a consequence of internal reconnaissance in the cyber network by lateral movement and is enabled by exploiting discovered vulnerabilities. This stage of the attack is also the longest, thereby giving a defender the biggest opportunity to detect and react to the attacker.
This paper helps a defender minimize the information an attacker might gain once in the network. This can be done by curbing lateral movement, misdirecting the attacker or inhibiting reachability to a critical device. We use a linear threshold models of attack propagation to analyze potential attack loss and use this to find actions that a defender might invest in while staying within their budgetary constraints. We show that while finding the best solution subject to these constraints is computationally intractable, the objective function is supermodular, allowing for a tractable technique with a known approximation bound.

1 Introduction

An attack on a critical infrastructure is most effective when the attacker understands the cyber network as well as the physical process under control. The attacker primarily gains this knowledge in the reconnaissance stage. The goal during this phase is to learn more about the environment by lateral movement which is enabled by exploiting vulnerabilities in devices and thereby increasing network penetration. This stage is also the longest phase of the attack, thereby giving the best opportunity to discover and possibly respond to the presence of an attacker.

This paper provides the defender with a tool to minimize the information an attacker might gain during the reconnaissance stage. The defender response can be some combination of curbing lateral movement, misdirecting the attacker or inhibiting reachability to a critical device. We model the propagation of attack through the network, and use this to find actions that a defender might invest in while staying within their budgetary constraints.

Attack graphs have been used to map possible paths an attacker might take within a network to reach their goal. Given the scale of the network and the possible states it can be in, the size of an attack graph can be in the order of thousands of nodes. And, while there has been work in the past that looks at generating such graphs and verifying them [24], [23], very little work has been done on using such a graph to harden the network [8]. We argue that at the scale of current attack graphs, it is very hard for an analyst to make useful decisions without exploiting properties of the graph and the metric in question.

We pose the question of network hardening in terms of modifying the attack graph (deleting edges) and show that despite the problem being computationally complex, there exist properties that can be exploited to get guarantees on the analysis.

Our Contributions

In this paper we formally define the network hardening problem against a model of attacker reconnaissance. The paper makes the following contributions:

1. We describe the attack propagation model under an attacker whose goal is to maximize knowledge of the network.
2. We show that the network hardening problem is NP-hard.
3. Using special properties of the defender metric, we show that a greedy algorithm performs with a provable approximation bound of $1 - \frac{1}{e}$.
4. We improve the complexity of the greedy algorithm from quadratic to linear.

We will first present some background on the cyber kill chain in critical infrastructures, diffusion processes and supermodularity as it relates to the network hardening problem (Section 2). We then proceed to formally define the network hardening under reconnaissance problem by describing models of attack propagation through attack graphs and of defender actions (Section 3). We then use the notion of live-edge graphs to inform our proofs of supermodularity and monotonicity of the objective function under defender actions (Section 4). Finally, we present the algorithm for efficiently solving the supermodular optimization problem we describe (Section 5) before presenting a discussion (Section 7) and concluding.

2 Background

2.1 Critical infrastructure attacks

Our analysis of recent attacks, including the power outage in Ukraine in 2015 and 2016 [16] as well as the 2017 attack on a Saudi Arabian oil and gas facility [15], shows that in order to remain inconspicuous, attackers avoid using malware and techniques that can be associated with adversarial behavior. This type of attack is known as *living off the land*. The success of these attacks comes from the ability of the attacker to move in the network by building a knowledge base in and about the network. It is important to note that while most defensive counter measures dwell on monitoring and protecting against the final impact of an intrusion, the biggest opportunity lies in the steps that lead up to the final step. To this end, the Industrial Control System (ICS) Cyber Kill Chain which was developed to help the defender characterize an attack, can be used as a tool to understand the stages of an intrusion leading up to an attack. A detailed description of the kill chain can be found at [16], but our focus will be on the internal reconnaissance stage.

Internal reconnaissance is the phase where the attacker attempts to find potentially interesting targets, as well as tries to acquire passwords or other credentials in order to attain an increased access to the system. Using these acquired credentials, attackers move laterally and repeat while maintaining stealthy presence. The goal is to be able to understand the network and the process being controlled well enough to be able to develop and deliver an attack that has the intended impact. An example of this can be observed in the second attack on Ukraine's power grid where the attacker understood the ability of the safety control system to thwart their intended impact and used this knowledge to develop a device specific DoS attack for the safety controls. Gaining this information and formulating usable knowledge from it takes attackers a long time, in certain cases up to a year (Ukraine 2015). Thus, while the delivery of the final attack might be instantaneous thereby requiring detection to be high precision and to act in real-time, a greater opportunity for detecting/preventing attack is to monitor/block the reconnaissance activity of an attacker.

Implementing a defense-in-depth strategy improves security by raising the cost of an intrusion for an adversary while simultaneously improving the probability of detection by the defender. The end goal is to reduce the opportunities for an adversary to take advantage of the ability to move laterally through a critical infrastructure network. The use of multiple layers not only helps prevent direct attacks against critical systems but also greatly increases the difficulty of reconnaissance activities on ICS networks. Our goal in this paper is to develop a systematic methodology for implementing such hardening techniques.

2.2 Diffusion Networks

Diffusion networks are used to model the spreading behavior of disease, influence or information through large networks. In its basic form a diffusion network is a set of nodes with directed edges that indicate the potential transmission of information, disease etc. A transmission matrix is used to indicate pairwise transmission rates between the nodes. Various diffusion models [10], [7] proposed in the literature mostly differ in the how the transmission of infection is modeled.

Linear Threshold Model A commonly used diffusion process (and of special interest to us) that models influence spread is the linear threshold model [9]. At its core, it is a weighted directed graph $G = (V, E, w)$ called the influence graph, where V is the set of nodes, E is a set of directed edges and $w : V \times V \rightarrow [0, 1]$ is the weighting function on the edges. For any edge $(u, v) \notin E$, $w(u, v)$ is not defined. Further, for every node $v \in V$, it is required that $\sum_{u:(u,v)\in E} w(u, v) \leq 1$

i.e. the sum of weights from all incoming neighbors is at most 1. Given such an influence graph and a source node $S_0 = \{a\}$, the cascade diffusion process proceeds in discrete time steps $t = 0, 1, 2, ...$ as follows:

1. at the initial time step $t = 0$, every node $v \in V$ independently selects a threshold $\theta_v \in [0, 1]$ uniformly at random. This captures our uncertainty in nodes' true thresholds against influence;
2. in every subsequent step $(t + 1)$, an inactive node, v, becomes active if the sum of incoming influence exceeds the threshold
$$\sum_{u:u \in S_t,(u,v) \in E} w(u, v) \geq \theta_v$$
where S_t is the set of nodes activated until the previous timestep t;
3. the process terminates when no more nodes can be activated.

2.3 Supermodular Set Functions

In this work we will reduce the question of finding defensive interventions into an optimization problem which minimizes the knowledge gained by an adversary subject to budgetary constraints. We will see that while the objective function will turn out to be NP-hard to optimize, it possesses the special property of *supermodularity.*

A set function $f : 2^S \to \mathbb{R}$ defined over the power set 2^S of a set S is called supermodular iff $\forall A \subseteq B \subset S$, $\forall s \in S \setminus B$:

$$f(A \cup \{s\}) - f(A) \leq f(B \cup \{s\}) - f(B)$$

The property essentially states that for a non-decreasing supermodular set function f, the marginal utility obtained by adding a new element to a larger set is greater than the utility of adding the element to any subset of the larger set. This property is referred to as the *increasing differences* property, as opposed to the more commonly seen diminishing returns property of submodular functions [12].

Most optimization literature focuses on *submodular* objective functions, which we will use to inform our analysis of our supermodular objective. It has been shown that submodular maximization is NP-hard [4] as can be intuited by the combinatorial explosion of possible subsets. However, a proof by Nemhauser et al. [21] shows that a greedy algorithm for maximizing a monotone submodular function (or in our case minimizing a monotone supermodular function) while subject to cardinality constraints can provide a constant factor approximation of $1 - \frac{1}{e} \approx 63\%$.

3 Problem Statement

In order to find a set of actions that the defender can implement to harden the network, we need a model of the services in the network (knowledge about these services is what the attacker is after), a model for how the attacker gains this knowledge through internal reconnaissance, and a measure of how good the defender strategy is, thereby allowing us to quantitatively harden the system. In this section, we describe each of these requirements and formalize the network hardening problem. At a high level, we need to find a hardening policy π that minimizes knowledge an attacker gains $\mathcal{Q}(\pi)$ while making sure that the cost of the policy stays within a given security budget $\mathcal{B}$. We explore each of the elements of the following optimization later in the paper.

$$\underset{\pi}{\text{minimize}} \quad \mathcal{Q}(\pi) \quad \text{subject to} \quad cost(\pi) \leq \mathcal{B}$$

3.1 Attack Graph

An attack graph is a graphical model that represents the defenders' knowledge about network components, services, their vulnerabilities and their interactions, showing the different paths an attacker can follow to reach a given goal by exploiting a set of vulnerabilities. Along each attack path, vulnerabilities are exploited in sequence, so that each successful exploit gives the attacker more knowledge thereby leading to an increased foothold in the network.

Uncertain attack graphs [22], extend the notion of an attack graph by allowing for uncertainty in edge existence. Formally, an uncertain attack graph $\mathcal{A} = (V, E, p)$ is a directed graph, where the nodes in V represent states of an attack and a directed edge in E represents the transition between states. Each edge $(i, j) \in E$ is associated with a transition probability p_{ij} i.e. the likelihood that an attacker can compromise state j given the knowledge from the current compromised state i.

A state in the model represents an atomic unit of knowledge, gaining which allows the attacker to make better decisions about the subsequent stages of reconnaissance. Think of the attacker starting blind (or with partial visibility) and each node that they get a foothold on adds to their visibility into the system and its processes. One simple way of depicting a state is as a tuple of host identity, service and privilege level on the service. Thus the knowledge gained from a state is the knowledge available on that host service when attacker achieves the privilege level needed. A transition corresponds to an atomic attacker action that leads to an increased gain in knowledge. Examples of transition include password guessing to escalate privilege, vulnerability exploitation to gain foothold on previously uncompromised host and active scans to detect new services.

A successful reconnaissance campaign is a sequence of transitions or paths in the graph that leads to knowledge states from where an attacker can affect the process being controlled in the critical infrastructure. One metric to measure success of attacker reconnaissance is to count the number of nodes that are infected at the end of the attack propagation phase, but we will dive deeper on these issues in the upcoming sections.

3.2 Attack Propagation Model

We define the attack propagation model as follows, based on the linear threshold diffusion model in [9].

1. The attacker starts with an initial knowledge S_0, where $S_0 \in V$ is a subset of nodes in the uncertain attack graph.
2. Each node v has a threshold $\theta_v \in [0, 1]$. This represents the weighted fraction of neighbors that must be compromised in order for v to be compromised. The value for the threshold is drawn at random. This threshold is the reason why the attack propagation process is stochastic.

Let us explore the threshold θ_v a little more. By definition, it can be interpreted as the likelihood that a node is compromised given that it's neighbors have been compromised. In the case of complete knowledge, about the services running on a node or attacker capabilities, the threshold is no longer a random variable. However, we more commonly encounter situations of uncertainty regarding θ_v. One scenario being the lack of detailed knowledge of services, their versions and the vulnerabilities in the network. Note that sometimes, despite perfect knowledge of the network, uncertainties arise from unknown unknowns such as zero-day vulnerabilities. Another scenario is limited understanding of attacker capabilities, for example an adversary may possess information that is gained external to the reconnaissance activity, thereby allowing them to compromise a node without having to compromise all its neighbors.

Given this uncertainty in the node threshold θ_v, the subsequent question is how to model it. Traditionally in the Linear Threshold model, it is assumed that θ_v is sampled from a standard uniform distribution ($U[0, 1]$). However, this does not capture any knowledge that the defender might possess. We suggest using a Beta distribution to sample the node threshold. The parameters $\alpha and \beta$ of a $Beta(\alpha, \beta)$ allow the defender to capture their prior knowledge. In the base case of $\alpha = \beta = 1$, we end up with a $U[0, 1]$ which captures complete lack of knowledge.

Thus, given the initial knowledge of the attacker (an initial compromised set) and the thresholds of all the uncompromised nodes, the attack propagation process unfolds in discrete time steps: at time t all nodes that were compromised in timestep $t-1$ remain compromised and any uncompromised node v is activated as:

$$\sum_{w:(v,w)\in E} p_{vw} \geq \theta_v$$

3.3 Modeling Prior knowledge of Attacker

Attackers can often gain auxiliary knowledge about the control system devices and processes from sources external to the network. They might know device manufacturer names and model numbers from reading public documents such as public presentations and requests for tender (similar to the attack on the

Kudankulam Nuclear Power Plant [1]). This has in the past, allowed attackers to successfully conduct watering hole attacks by adding malware to vendor websites [20]. Adversaries can also acquire attacks for known vulnerabilities on the dark web, thereby making the lateral movement process faster. They might acquire stolen passwords for system operators of one system from an attack that was not targeted for it, like in the case of the Ukraine attack where some passwords were gained as a consequence of the ransomware notPetya (that did not target ICS specifically) [16]. Thus, an attacker starts in the network with some prior knowledge. We model this by designating a subset of the nodes as compromised at the start of the attack propagation process. $S_0 \subseteq V$ is the knowledge an attacker possesses at time t_0. In order to simplify computation we add a dummy node s to the set of nodes V with edges to the nodes S_0 each with probability 1. This allows us to designate s as the source node for attacks while allowing us to maintain S_0 as the initial foothold of the attacker.

3.4 Defender Actions

Knowledge reduction is equivalent to reducing the coverage of nodes in the uncertain attack graph. We define two actions a defender can perform to harden the network.

1. Add a security rule. We consider scenarios where a firewall or intrusion prevention system is present in the network and the defender adds a fixed rule or signature that the security appliance must match against to decide if a flow is permitted or not. Adding a rule is equivalent to reducing the edge traversal probability of an edge in the attack graph. If the defender believes that a rule prevents the attacker from reaching the next knowledge state then the edge is deleted i.e. edge traversal probability goes to zero. In this work we only consider the latter scenario where an edge is completely removed from the attack graph. We discuss the limitation of this in Section 7.
2. Add a security appliance or apply a patch. When a new security appliance is added to the network with the correct configuration or when a known vulnerability is patched, multiple edge traversal probabilities are modified by a single action. However, such an action has a higher cost associated with it. In the case of a new security appliance this might be the capital cost of the technology and its deployment as well as the operational cost of hiring analysts to dig through false positives. In the case of patching the cost is mostly that of testing the patch in an identical environment before deeming it fit to be reproduced in the operational network.

The set of actions that a defender undertakes is known as the *policy* π. Each policy π is a set of deleted edges corresponding to defender actions. We denote the implementation of a policy as a modification to the uncertain attack graph $\mathcal{A}$:

$$\mathcal{A}_\pi = \mathcal{A}(V, E \setminus \pi, p)$$

A policy is *feasible* if the total cost of actions taken is no greater than a given budget limit $\mathcal{B}$. For a given uncertain attack graph $\mathcal{A}$ with an initial infection S_0, the network hardening problem is to select the optimal policy among all feasible policies, such that the information gained by the attacker is minimized.

3.5 Objective Function

We define two metrics of interest in the network hardening problem: penetrability and expected risk.

Definition 1. ***Penetrability*** *of an uncertain attack graph $\mathcal{A}$ is defined as the expected number of compromised nodes at the end of an attack propagation process that starts at an initial foothold, S_0.*

$$\mathcal{P}(\mathcal{A}) = E[S_\infty] \tag{1}$$

where S_∞ is the number of nodes that are compromised at the end of attack propagation.

Penetrability is analogous to reachability in stochastic reliability graphs. In a deterministic attack graph, penetrability is defined as the number of nodes that can be reached from a source set S_0.

We recall that the uncertain attack graph $\mathcal{A}$ is a probabilistic graph. A probabilistic graph can be a generator for $2^{|E|}$-many deterministic graphs, based on the presence or absence of an edge. In the case of uncertain attack graph $\mathcal{A}$, each deterministic graph it generates is called an attack scenario. Each attack scenario is a possible set of nodes that an attacker has visited using the set of compromised edges. Note that not all of the $2^{|E|}$-many attack scenarios may be plausible, and thus can be excluded from the search space.

Definition 2. ***Risk*** *Consider an attack scenario $A \in \Omega_{\mathcal{A}}$, where $\Omega_{\mathcal{A}}$ is the space of attack scenarios that can be generated from uncertain attack graph $\mathcal{A}$. Also, a loss function, $L : A \rightarrow \mathbb{R}^+$, for attack scenario A, $L(A)$. The risk for a network modeled by the uncertain attack graph $\mathcal{A}$ is defined as the expected loss across all possible attack scenarios:*

$$\mathcal{R}(\mathcal{A}) = \sum_{A \in \Omega_{\mathcal{A}}} Pr[A]L(A) \tag{2}$$

The goal of the loss function is to quantify the direct (monetary loss due to equipment damage and repair) and indirect losses (loss of intellectual property) a critical infrastructure network faces under attack. The loss function is generally monotone non-decreasing: the more nodes an attacker compromises the greater the loss. While, there is more discussion on loss functions in Section 7, in this work we only look at monotone loss functions defined as follows:

Definition 3. *Given an attack scenario $A \in \Omega_{\mathcal{A}}$, the loss under such an attack can be computed as:*

$$L(A) = \sum_{i \in P(A)} c_i \tag{3}$$

where c_i is the dollar cost of losing the knowledge stored in node i to the attacker and $P(A)$ is the set of compromised nodes in A.

We understand that there is uncertainty associated with deriving the value c_i. One source of uncertainty is from attributing the loss as either a loss of confidentiality or one of availability. A loss in confidentiality is characterized by the loss of industrial secrets and proprietary technology present on a node. A loss in availability is due to denial of service which can range from monetary loss due to a blackout for few hours to equipment damage. Thus this loss is more of a distribution rather than a number. While understanding this, we defer this to future work and allow for a treatment of c_i as a number which the user can either define as worst case or average monetary loss based on their analysis.

In any network, some nodes (say an authentication server or the data historian) are more valuable than others. The risk metric intrinsically captures the case where nodes are valued differently. Also, note that the risk metric can be used to define *Penetrability.* If $L(v) = 1$ for every node $v \in A$ and $L(v) = 0$ for every other node, the expected loss is the expected number of nodes reached at the end of attack propagation. While the risk metric is a general solution, it requires more information from the defender, thereby making penetrability a viable alternative. Under this alternate definition, penetrability can be formalized as:

$$\mathcal{P}(\mathcal{A}) = \sum_{A \in \Omega_{\mathcal{A}}} Pr[A] r(A) \tag{4}$$

where $r(A)$ is the $\{0, 1\}$ loss function describing the number of nodes that can be reached in an attack scenario A.

3.6 Constraints

There is a cost associated with each defender action and the security budget must be split such that the defender actions that restrict attacker movement most should be preferred over the rest. In a simple case where all defender actions cost the same, a cardinality constraint can be used (total number of deleted edges is less than or equal to the budget). In most cases however, the cost is more of a knapsack constraint i.e. the sum of costs of each action must be below the budget.

Considering the objective function and the constraints we defined in this section, the network hardening problem can be translated into the following optimization problem:

$$\underset{\pi \subseteq E}{\operatorname{argmin}} \quad \mathcal{Q}(\mathcal{A} \setminus \pi) \quad \text{subject to} \quad \sum_{i \in \pi} k_i \leq \mathcal{B} \tag{5}$$

Where, the function $\mathcal{Q}(\mathcal{A})$ can be substituted for either $\mathcal{P}(\mathcal{A})$ or $\mathcal{R}(\mathcal{A})$ depending on the analysis.

4 Analyzing the Objective function

In this section we analyze some interesting properties of the objective function. Specifically, we show that the objective function is monotonically decreasing and supermodular in the policy π.

4.1 Live Edge Paths

Influence maximization literature shows that an alternate method to compute the influence function under the linear threshold process is using *live-edge* paths in the influence graph. The claim (as proved in [9]) is that given an initial set of nodes, S, the distribution over active nodes obtained by running the linear threshold process to completion is the same as the distribution of nodes reachable from S via live-edge paths, where the live-edge paths are selected as described below:

Each node $v \in V$ picks at most one of its incoming edges at random with probability p_{uv} and selects no edge with probability $1 - \sum_{u:(u,v)\in E} p_{uv}$.

Note that the set of nodes in the generated live-edge graph X is equal to V and the set of *live* edges, E_X is a subset of E, i.e., $E_X \subseteq E$. Also note that these sampled edges are unweighted and each vertex has at most one incoming edge.

We will use this live-edge graph to inform our proofs of monotonicity and supermodularity. Before we get to the proofs, we make some useful observations. The probability of a node $v \in V$ having the edge configuration as seen in attack scenario A is given by:

$$p(v, A, \mathcal{A}) = \begin{cases} p_{uv} & \text{if } \exists u : (u,v) \in E_A \\ 1 - \sum_{u:(u,v)\in E} p_{uv} & \text{otherwise} \end{cases} \tag{6}$$

where E_A is the set of edges in A. Using this, we can get the probability of a scenario graph A as:

$$Pr[A] = \prod_{v \in V} p(v, A, \mathcal{A}) \tag{7}$$

Note that the loss $(L(A))$ term is deterministic given an instance of the scenario graph A. The likelihood term is computed as shown in Equation 7.

4.2 Monotonicity

In this section, we prove that the objective function is monotonically decreasing.

Lemma 1. *The reachability function $r(A \setminus \pi)$ is a monotonically decreasing function of the policy π.*

Proof. Given a scenario graph $A \in \Omega_{\mathcal{A}}$, we need to show that for any policy $\pi \subseteq E$ and an edge $e = (u, v) \in E \setminus \pi$

$$r(A \setminus \pi) - r(A \setminus \{\pi \cup e\}) \geq 0$$

Since, A is a live-edge graph such that each vertex $v \in V$ has at most one incoming edge it can be seen that removing an edge $e = (u, v)$ to v will make it unreachable from the source s. In fact, if v is not a leaf node, then all its children are rendered unreachable by the removal of edge e. Thus, the number of nodes reachable in $A \setminus \pi$ is higher than that in $A \setminus \{\pi \cup e\}$ ■

Lemma 2. *The loss function $L(A \setminus \pi)$ is a monotonically decreasing function of the policy π.*

Proof. Similar to Lemma 1 we must show that

$$L(A \setminus \pi) - L(A \setminus \{\pi \cup e\}) \geq 0$$

By definition, the loss function is a linear combination of non-negative costs based on the reachability of nodes. Thus, since the reachability in $A \setminus \pi$ is higher than that in $A \setminus \{\pi \cup e\}$ by at least 1 (due to v), the loss is also higher by at least c_v, which is non-negative ■

Theorem 1. *$\mathcal{R}(\mathcal{A} \setminus \pi)$ is a monotonically decreasing function of the policy π.*

Proof. We defer the proof to the appendix for better readability.

Theorem 2. *$\mathcal{P}(\mathcal{A} \setminus \pi)$ is a monotonically decreasing function of the policy π.*

Proof. This proof follows the same structure as above and uses Lemma 1 in the final step ■

4.3 Supermodularity

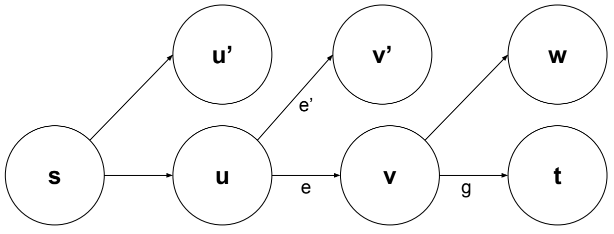

Fig. 1. Proof of supermodularity.

Lemma 3. *The reachability function $r(A \setminus \pi)$ is a supermodular function of the policy π.*

Proof. In order to prove supermodularity, we must show for an attack graph $\mathcal{A} = (V, E, p)$, a subset of edges to be deleted $\pi \subset E$ and two edges $e, g \in E \setminus \pi)$ that

$$r(A \setminus \pi) - r(A \setminus \pi \cup \{e\}) \geq r(A \setminus \pi \cup \{g\}) - r(A \setminus \pi \cup \{e, g\})$$

Figure 1 illustrates our proof. Note that in the live edge graph model, each node has at most one incoming edge as shown in the figure. Consider the scenario shown in the figure to be the $A \setminus \pi$. The difference we are computing is the reduction in reachability when an edge is deleted from this graph. We consider two cases: (1) deleting an edge e' that does not fall in the same path as edge g and (2) deleting an edge e in the same path as edge g. In case 1, deleting edge e' from $A \setminus \pi$ leads to the same reduction as in $A \setminus \pi \cup \{g\}$. This is due to the fact that the deletion causes the same set of nodes to become unreachable. Thus, case 1 leads to equality. In case 2, deletion of the edge e from $A \setminus \pi$ disconnects additional nodes which had edge g in their path from the source. Thus in case 2, the reduction in reachable nodes is greater when e is deleted from $A \setminus \pi$ instead of $A \setminus \pi \cup \{g\}$ ■

Lemma 4. *The loss function $L(A \setminus \pi)$ is a supermodular function of the policy π.*

Proof. Similar to Lemma 3, we must show

$$L(A \setminus \pi) - L(A \setminus \pi \cup \{e\}) \geq L(A \setminus \pi \cup \{g\}) - L(A \setminus \pi \cup \{e, g\})$$

Note that in the reachability function, the marginal value after the difference represents the number of nodes that are rendered unreachable after the edge e is deleted. Since the loss function is a linear function of reachability it can be seen that the marginal loss is in fact the sum of costs of nodes that are rendered unreachable by deleting e. Thus, it can be seen that for non-negative costs, the marginal loss is greater when e is deleted from $A \setminus \pi$ instead of $A \setminus \pi \cup \{g\}$ ■

Lemma 5. *If $f(S)$ is a supermodular function then its expectation $E[f(S)]$ is a supermodular function of S.*

Proof. This is a simple extension of the fact that supermodular functions are closed under linear combinations and the expectation function is essentially a non-negative weighted sum of the supermodular function.

Theorem 3. *Penetrability is supermodular in the policy π.*

Proof. We note that by the live edge definition in Equation 4, penetrability is essentially the expected reachability in an attack graph. Hence, by Lemma 3 and Lemma 5 it can be seen that $\mathcal{P}(\mathcal{A} \setminus \pi)$ is supermodular in the policy π ■

Theorem 4. *Risk is supermodular in the policy π.*

Proof. Risk is by definition (Equation 2) expected loss. Hence, by Lemma 4 and Lemma 5 it can be seen that $\mathcal{R}(\mathcal{A} \setminus \pi)$ is supermodular in the policy π ■

5 Algorithm

In the previous section we showed our hardening objective functions to be monotone and supermodular under the linear threshold model of attack propagation. Our optimization problem is one of minimizing a monotone supermodular function under a knapsack constraint. In the case where every defender action has the same cost and we can perform k defender actions, the total possible policies are $\binom{|E|}{k}$. It is combinatorially hard to search this large space for an optimal policy. In fact, the problem of maximizing a monotone submodular function (and in our case a monotone supermodular function) under a cardinality constraint has been shown to be NP-hard [12]. The proof is a reduction of the vertex cover problem [9].

5.1 The Greedy Algorithm

A naive approach is to find approximate solutions using the greedy algorithm. The algorithm starts with an empty policy $\pi = \emptyset$ and proceeds in an iterative fashion. There are a total of k iterations (cardinality constraint: $|\pi| \leq k$). In the j-th iteration we pick an edge e such that $e = \text{argmax}_{e \in E \setminus \pi} \Delta(e)$, where $\Delta(e)$ is the marginal loss defined as $\mathcal{R}(\mathcal{A} \setminus \pi) - \mathcal{R}(\mathcal{A} \setminus (\pi \cup \{e\}))$. Note that similar steps apply to the Penetrability objective.

A fundamental result by Nemhauser et al. [21] shows that under the cardinality constraint, this greedy algorithm produces a near optimal solution. The greedy algorithm will choose a policy π such that $\mathcal{R}(\mathcal{A} \setminus \pi) \geq 1 - \frac{1}{e}\ \mathcal{R}(\mathcal{A} \setminus \pi^*)$, where π^* is the optimal policy for hardening the network.

Drawbacks of Naive Greedy While the greedy algorithm gives a near optimal solution, it needs some tweaks in the practical setting. Note that the $\mathcal{R}(\mathcal{A} \setminus \pi)$ function is evaluated every time we need to find the edge that maximizes $\Delta(e)$. Also, computing this function accurately is #-P hard as shown in [3]. Thus, we need a way to approximate the expected risk in sub-linear time.

5.2 Monte Carlo Estimation

In order to find an approximation of risk, we use Monte Carlo estimation. To estimate the risk, the Monte Carlo estimator first draws N possible attack scenarios denoted by $A_0, A_1, ..., A_N$ from the attack graph $\mathcal{A}$ using the live-edge sampling rule stated in Equation 6. Then, for each possible attack scenario A_i, the algorithm evaluates the loss function $L(A_i)$. Finally, the risk estimate (denoted by $\hat{\mathcal{R}}$) is obtained by:

$$\hat{\mathcal{R}}(\mathcal{A}) = \frac{1}{N} \sum_{i \in N} L(A_i) \tag{8}$$

Note that due to the linear combination property of supermodular functions, this estimated expected risk is also monotone and supermodular in the policy π.

Given this estimate, the marginal loss $\Delta(e)$ is given by:

$$\frac{1}{N}\sum_{i\in N} L(A_i \setminus \pi) - L(A_i \setminus \pi \cup \{e\}) \tag{9}$$

5.3 Knapsack Budget Constraint

When the defender actions have different costs, the simple greedy algorithm can perform arbitrarily badly. However, an algorithm that defines a loss-cost ratio $\Delta(e)/k_e$ and uses partial enumeration along with the greedy step of picking the edge $e \in E$ that maximizes the loss-cost ratio at every iteration has been shown to achieve an approximation guarantee of $1 - \frac{1}{e}$ [26] [13].

5.4 Complexity of the Greedy Algorithm

Note, that in a given iteration we need to compute the marginal loss for every edge $e \in E \setminus \pi$ and every scenario A_i. This involves a breadth first search $(O(|V| + |E|))$ to compute reachability which is then used to compute loss. Note that in the live edge graphs, we have at most $|V|$ edges since each node can have only one incoming edge. Thus, the complexity of each iteration is $O(|V|^2 + |V||E|)$. However, we can use the following lemma to improve the time complexity from quadratic to linear.

Lemma 6. *The marginal reduction in reachability when an edge $e = (u, v)$ is added to the policy is the same as the size of the sub-tree which is induced by the node v.*

Proof. We recall that the marginal reduction in reachability, denoted by $r(A \setminus \pi) - r(A \setminus \pi \cup \{e\})$ is the same as the number of nodes rendered unreachable when edge e is deleted from the graph $A \setminus \pi$. Referring to Figure 1 we see that this is essentially the size of the sub-tree rooted at v.

Thus, we can create a tree data structure, which stores at each node the size of the sub-tree it would generate if the incoming edge to it is deleted. Given the data structure, the marginal loss can be computed in a single breadth first search traversal. Also since the number of edges is at most V, this takes $O(|V|)$. The total time of the modified greedy algorithm is $O(kN|V|)$ which is linear in the size of the attack graph.

6 Related Work

A closely related area is that of influence diffusion through social networks. The question of influence maximization is one of finding the initial set of nodes to infect in order to maximize spread of information over the network [9]. This has been applied to study influence of users on their followers in a social network [2], to examine cascade of news through the web [18] and the propagation of recommendations [17]. There has also been work on inferring the structures of such networks [5] [6]. Our work differs from this line of research since our focus is not on empirical analysis of diffusion but rather on modifying a network to minimize the diffusion through it. A similar problem has been addressed in [11] in a different application context, with a slightly different objective function and constraints.

A closer line of inquiry is that of placing sensors to detect changes in the environment. [14], [19] use submodular maximization to maximize the information gained by a sensor placement. There are also actuator placement problems [25] that look to choose a set of actuators to maximize controllability of a control network. Our work differs from this line of inquiry since our study of attack graphs requires analysis of graph modification. Also, the metrics of interest are very different.

7 Conclusion and Limitations

Critical infrastructures have complex networks which control unique processes. In order to perform a successful attack on such a network, the adversary must understand the network components they can leverage and the processes they can affect. This requires a long and detailed reconnaissance phase. Thus, the best opportunity for a defender is to harden against such recon activities. This paper provides the defender with a systematic approach to making decisions while limiting the cost of defense. We show that despite imperfect knowledge of the attacker and their actions the defender can still improve their risk posture. While the hardening problem is NP-hard we provide a linear time algorithm with a provable approximation bound of $1 - \frac{1}{e}$.

While this paper provides analysts with an algorithm for hardening their network, there are a few limitations we wish to discuss. First, the loss distribution is often not available to the analysts and while a metric like penetrability can give some insights into the hardening process, in the future we hope to explore other metrics that can capture the loss. Another assumption is the loss function being monotone, however, with the growth of reactive security techniques like honeypots, we will have to look into non-monotone loss in the future. Second, we note that all properties studied in the paper hold only for the linear threshold model and while this is a strong model for reconnaissance we might have to find other models to capture other stages of the attack kill-chain.

A Proof of Theorem 1

Given an attack graph $\mathcal{A} = (V, E, p)$ we need to show that for any set $\pi \subseteq E$ and an edge $e = (u, v) \in E \setminus \pi$

$$\mathcal{R}(\mathcal{A} \setminus \pi) - \mathcal{R}(\mathcal{A} \setminus (\pi \cup \{e\})) \geq 0$$

This proof is very similar to the proof in [11] and only differs in our function of interest (attack loss function).

The space of attack scenarios $\Omega_{\mathcal{A}\setminus\pi}$, can be divided into three disjoint partitions based on the edge selected for node v. $\Omega^{e}_{\mathcal{A}\setminus\pi}$ (edge $e = (u, v)$ is chosen), $\Omega^{\overline{e}}_{\mathcal{A}\setminus\pi}$ (a different edge $\overline{e} = (u^{'}, v)$ is chosen) and $\Omega^{\emptyset}_{\mathcal{A}\setminus\pi}$ (no incoming edge is selected).

Now for, the space $\Omega_{\mathcal{A}\setminus(\pi\cup\{e\})}$ we note that the space is a subset of $\Omega_{\mathcal{A}\setminus\pi}$ since any scenario graph in the former can be generated in the latter. Also, the only scenarios not present in the former are ones where the edge e is involved. Thus, $\Omega_{\mathcal{A}\setminus(\pi\cup\{e\})}$ can be defined based on two partitions as: $\Omega^{\overline{e}}_{\mathcal{A}\setminus\pi} \cup \Omega^{\emptyset}_{\mathcal{A}\setminus\pi}$.

Using these disjoint partitions, we can write the difference as:

$$\begin{aligned}
&\mathcal{R}(\mathcal{A}\setminus\pi) - \mathcal{R}(\mathcal{A}\setminus(\pi\cup\{e\})) \\
&\quad= \sum_{A\in\Omega^{e}_{\mathcal{A}\setminus\pi}} Pr[A|\mathcal{A}\setminus\pi]L(A) \\
&\quad+ \sum_{A\in\Omega^{\overline{e}}_{\mathcal{A}\setminus\pi}} (Pr[A|\mathcal{A}\setminus\pi] - Pr[A|\mathcal{A}\setminus(\pi\cup e)])L(A) \\
&\quad+ \sum_{A\in\Omega^{\emptyset}_{\mathcal{A}\setminus\pi}} (Pr[A|\mathcal{A}\setminus\pi] - Pr[A|\mathcal{A}\setminus(\pi\cup e)])L(A)
\end{aligned}$$

For the space $\Omega^{\overline{e}}_{\mathcal{A}\setminus\pi}$ we have $Pr[A|\mathcal{A}\setminus\pi] - Pr[A|\mathcal{A}\setminus(\pi\cup e)] = 0$ since from Equation 6 we have, $p(v, A, \mathcal{A}\setminus\pi) = p(v, A, \mathcal{A}\setminus(\pi\cup\{e\})) = p(\overline{e})$. This is due to the fact that in this space, under both cases, edge $\overline{e}$ is chosen for node v.

For the space $\Omega^{\emptyset}_{\mathcal{A}\setminus\pi}$ we have:

$$Pr[A|\mathcal{A}\setminus\pi] - Pr[A|\mathcal{A}\setminus(\pi\cup e)] = -p_e \prod_{v'\neq v} p(v', A, \mathcal{A}\setminus\pi)$$

This stems from the fact that we can rewrite the above difference in terms of the node v and all other nodes $v' \neq v$ as $Pr[A|\mathcal{A}\setminus\pi] - Pr[A|\mathcal{A}\setminus(\pi\cup e)] = \prod_{v'\neq v} p(v', A, \mathcal{A}\setminus\pi) \times [p(v, A, \mathcal{A}\setminus\pi) - p(v, A, \mathcal{A}\setminus(\pi\cup\{e\}))]$.

As for the difference in probabilities when node v has no incoming edge we see that it goes to $-p_e$ due to the fact that $p(v, A, \mathcal{A}\setminus\pi) = 1 - \sum_{x\in E\setminus\pi} p_x = 1 - \sum_{x\in E\setminus(\pi\cup\{e\})} p_x - p_e = p(v, A, \mathcal{A}\setminus(\pi\cup\{e\})) - p_e$.

Now consider the following two facts:

- Every graph $A' \in \Omega^{e}_{\mathcal{A}\setminus\pi}$ has a corresponding graph $A \in \Omega^{\emptyset}_{\mathcal{A}\setminus\pi}$ and vice versa where $A' = A \cup \{e\}$ i.e. they differ only in the edge e.
- A graph $A' \in \Omega^{e}_{\mathcal{A}\setminus\pi}$ has probability $Pr[A'|\mathcal{A}\setminus\pi] = p_e \prod_{v'\neq v} p(v', A', \mathcal{A}\setminus\pi)$. Note that this is essentially Equation 7 rewritten in terms of e.

Hence:

$$\mathcal{R}(\mathcal{A}\setminus\pi) - \mathcal{R}(\mathcal{A}\setminus(\pi\cup\{e\})) = \sum_{A\in\Omega^{\emptyset}_{\mathcal{A}\setminus\pi}} Pr[A'|\mathcal{A}\setminus\pi][L(A') - L(A)]$$

Since this is a non-negative sum and by Lemma 2 we know that $L(A') - L(A) \geq 0$ we can see that the risk function is monotone decreasing in the policy π ■

References

1. P. Anantharaman and K. Palani. What happened when the Kudankulam nuclear plant was hacked – and what real danger did it pose? *Scroll.in*, Nov, 20, 2019.
2. E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts. Everyone's an influencer: quantifying influence on twitter. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 65–74, 2011.
3. W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1029–1038, 2010.
4. U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.
5. M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(4):1–37, 2012.
6. M. Gomez Rodriguez, J. Leskovec, and B. Schölkopf. Structure and dynamics of information pathways in online media. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 23–32, 2013.
7. M. Granovetter and R. Soong. Threshold models of diffusion and collective behavior. *Journal of Mathematical sociology*, 9(3):165–179, 1983.
8. S. Jha, O. Sheyner, and J. Wing. Two formal analyses of attack graphs. In *Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15*, pages 49–63. IEEE, 2002.
9. D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.
10. W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.
11. E. Khalil, B. Dilkina, and L. Song. Cuttingedge: influence minimization in networks. In *Proceedings of Workshop on Frontiers of Network Analysis: Methods, Models, and Applications at NIPS*, 2013.
12. A. Krause and D. Golovin. Submodular function maximization., 2014.
13. A. Krause and C. Guestrin. *A note on the budgeted maximization of submodular functions.* Carnegie Mellon University. Center for Automated Learning and Discovery, 2005.
14. A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526, 2008.
15. R. Lee. Trisis malware: Analysis of safety system targeted malware. dragos inc, 2017.
16. R. Lee, M. Assante, and T. Conway. Analysis of the cyber attack on the ukrainian power grid. Technical report, EISAC. Tech. rep, 2016.
17. J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5–es, 2007.
18. J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 497–506, 2009.

19. J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429, 2007.
20. E. Nakashima. Russian military was behind NotPetya cyberattack in Ukraine, CIA concludes. *The Washington Post*, 12, 2018.
21. G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.
22. H. H. Nguyen, K. Palani, and D. M. Nicol. An approach to incorporating uncertainty in network security analysis. In *Proceedings of the Hot Topics in Science of Security: Symposium and Bootcamp*, pages 74–84, 2017.
23. X. Ou, W. F. Boyer, and M. A. McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 336–345, 2006.
24. O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated generation and analysis of attack graphs. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 273–284. IEEE, 2002.
25. T. H. Summers, F. L. Cortesi, and J. Lygeros. On submodularity and controllability in complex dynamical networks. *IEEE Transactions on Control of Network Systems*, 3(1):91–101, 2015.
26. M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.