Keeping Intruders at Large
*A Graph-theoretic Approach to Reducing the Probability of Successful Network Intrusions**

Paulo Shakarian[1], Damon Paulo[2], Massimiliano Albanese[3] and Sushil Jajodia[3,4]
[1]*Arizona State University, Tempe, AZ, U.S.A.*
[2]*U.S. Military Academy, West Point, NY, U.S.A.*
[3]*George Mason University, Fairfax, VA, U.S.A.*
[4]*The MITRE Corporation, McLean, VA, U.S.A.*

Keywords: Moving Target Defense, Adversarial Modeling, Graph Theory.

Abstract: It is well known that not all intrusions can be prevented and additional lines of defense are needed to deal with intruders. However, most current approaches use honeynets relying on the assumption that simply attracting intruders into honeypots would thwart the attack. In this paper, we propose a different and more realistic approach, which aims at delaying intrusions, so as to control the probability that an intruder will reach a certain goal within a specified amount of time. Our method relies on analyzing a graphical representation of the computer network's logical layout and an associated probabilistic model of the adversary's behavior. We then artificially modify this representation by adding "distraction clusters" – collections of interconnected virtual machines – at key points of the network in order to increase complexity for the intruders and delay the intrusion. We study this problem formally, showing it to be NP-hard and then provide an approximation algorithm that exhibits several useful properties. Finally, we present experimental results obtained on a prototypal implementation of the proposed framework.

1 INTRODUCTION

Despite significant progress in the area of intrusion prevention, it is well known that not all intrusions can be prevented, and additional lines of defense are needed in order to cope with attackers capable of circumventing existing intrusion prevention systems. However, most current approaches are based on the use of honeypots, honeynets, and honey tokens to lure the attacker into subsystems containing only fake data and bogus applications. Unfortunately, these approaches rely on the unrealistic assumption that simply attracting an intruder into a honeypot would thwart the attack. In this paper, we propose a totally different and more realistic approach, which aims at delaying an intrusion, rather than trying to stop it, so as to control the probability that an intruder will reach a certain goal within a specified amount of time and keep such probability below a given threshold.

Our approach is aligned with recent trends in cyber defense research, which has seen a growing interest in techniques aimed at continuously changing a system's attack surface in order to prevent or thwart attacks. This approach to cyber defense is generally referred to as Moving Target Defense (MTD) (Jajodia et al., 2013) and encompasses techniques designed to change one or more properties of a system in order to present attackers with a varying *attack surface*[2], so that, by the time the attacker gains enough information about the system for planning an attack, its attack surface will be different enough to disrupt it.

In order to achieve our goal, our method relies on analyzing a graphical representation of the computer network's logical layout and an associated probabilistic model of the adversary's behavior. In our model, an adversary can penetrate a system by sequentially gaining privileges on multiple system resources. We

*This work was partially supported by the Army Research Office under award number W911NF-13-1-0421. Paulo Shakarian and Damon Paulo were supported by the Army Research Office project 2GDATXR042. The work of Sushil Jajodia was also supported by the MITRE Sponsored Research Program.

[2]Generally, the attack surface refers to system resources that can be potentially used for an attack.

Shakarian P., Paulo D., Albanese M. and Jajodia S..
Keeping Intruders at Large - A Graph-theoretic Approach to Reducing the Probability of Successful Network Intrusions.
DOI: 10.5220/0005013800190030
In *Proceedings of the 11th International Conference on Security and Cryptography* (SECRYPT-2014), pages 19-30
ISBN: 978-989-758-045-1

model the adversary as having a particular target (e.g., an intellectual property repository) and show how to calculate the probability of him reaching the target in a certain amount of time (we also discuss how our framework can be easily generalized for multiple targets). We then modify our graphical representation by adding "distraction clusters" – collections of interconnected virtual machines – at key points of the network in order to reduce the probability of an intruder reaching the target. We study this problem formally, showing it to be NP-hard and then provide an approximation algorithm that possesses several useful properties. We also describe a prototypal implementation and present our experimental results.

Related Work. Moving Target Defense (MTD) (Jajodia et al., 2013; Jajodia et al., 2011; Evans et al., 2011) is motivated by the asymmetric costs borne by cyber defenders. Unlike prior efforts in cyber security, MTD does not attempt to build flawless systems. Instead, it defines mechanisms and strategies to increase complexity and costs for attackers. The recent trend of high-profile cyber-incidents resulting in significant intellectual property theft (Shakarian et al., 2013) indicates that current practical approaches may be insufficient.

MTD differs from current practical approaches which primarily rely on three aspects: (a) attempting to remove vulnerabilities from software at the source, (b) patching software as rapidly as possible, and (c) identifying attack code and infections. The first approach is necessary but insufficient because of the complexity of software. The second approach is standard practice in large enterprises, but has proven difficult to keep ahead of the threat, nor does it provide protection against zero-day attacks. The last approach is predicated on having a signature of malicious attacks, which is not always possible.

MTD approaches aiming at selectively altering a system's attack surface (Manadhata and Wing, 2011) are relatively new. In Chapter 8 of (Jajodia et al., 2011), Huang and Ghosh present an approach based on diverse virtual servers, each configured with a unique software mix, producing diversified attack surfaces. In Chapter 9, Al-Shaer investigates an approach to enables end-hosts and network devices to change their configuration (e.g., IP addresses). In Chapter 6, Rinard describes mechanisms to change a system's functionality in ways that eliminate security vulnerabilities while leaving the system able to provide acceptable functionality. A game-theoretic approach to increase complexity for the attacker is presented in (Sweeney and Cybenko, 2012).

The efforts that are more closely related to our work are those based on the use of honeypots. However, such approaches significantly differ from our work in that they aim at either capturing the attacker and stopping the attack (Abbasi et al., 2012) or collecting information about the attacker for forensic purposes (Chen et al., 2013). There is also a relatively new corpus of work on attacker-defender models for cyber-security using game-theoretic techniques: an overview is provided in (Alpcan and Baar, 2010). Work in this area related to this paper include (Williamson et al., 2012) and (Píbil et al., 2012). The work presented in (Williamson et al., 2012) is similar to our work in that it models the adversary as moving through a graphical structure. However, that work differs in that the defender is trying to learn about the attacker's actions for forensic analysis purposes. In this work, we do not assume a forensic environment and rather than trying to understand the adversary, we are looking to delay him from obtaining access to certain machines (e.g., intellectual property repositories). In (Píbil et al., 2012), the authors use game theoretic techniques to create honeypots that are more likely to deceive (and hence attract) an adversary. We view their approach as complementary to ours, specifically with regard to the creation of distraction clusters.

2 TECHNICAL PRELIMINARIES

In this section, we first introduce the notion of *intruder's penetration network*, and then provide a formal statement of the problem we address in the paper. Note that we model a complex system as a set $S = \{s_1, \ldots, s_n\}$ of computer systems. Each system in S is associated with a level of access obtained by the intruder denoted by a natural number in the range $\mathcal{L} = \{0, \ldots, \ell_{\max}\}$. The level of access to a given system changes over time, which is treated as discrete intervals in the range $0, \ldots, t_{\max}$.

For a given system s and level ℓ, we shall use a *system-level pair* (s, ℓ) to denote that the intruder currently has level of access ℓ on system s. We shall use $\mathbf{S}$ to denote the set of all system-level pairs.

Definition 1 (Intruder's penetration network (IPN))**.** *Given a system $S = \{s_1, \ldots, s_n\}$, the* intruder's penetration network *for S is a directed graph $IPN = (S, R, \pi, f)$, where S is the set of nodes representing individual computer systems, $R \subseteq S \times S$ is a set of directed edges representing relationships among those systems. For a given $s_i \in S$, $\eta_i = \{(s_i, s') \in R\}$. We define the conditional success probability function $\pi : \mathbf{S} \times \mathbf{S} \rightarrow [0,1]$ as a function that, given two system-level pairs $(s, \ell), (s', \ell')$ returns the probability that an intruder with access level ℓ on s will gain access level ℓ' on s' in the next time step (provided that the attacker*

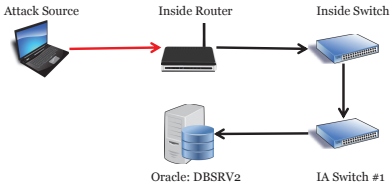


Figure 1: Sample network based on a real-world case: an attacker targeting the Oracle server penetrates the network exploiting a vulnerability in the Router.

selects s' as the next target). This function must have the following properties:

$$(\forall (s,s') \in R)(\forall \ell > 0)(\pi((s,0),(s',\ell)) = 0) \quad (1)$$

$$(\forall (s,s') \notin R)(\forall \ell,\ell' > 0)(\pi((s,\ell),(s',\ell')) = 0) \quad (2)$$

$$(\forall \ell_k \in \mathcal{L})(\ell_i \leq \ell_j \Rightarrow \pi((s,\ell_i),(s',\ell_k)) \leq \pi((s,\ell_j),(s',\ell_k))) \quad (3)$$

We define f: $\mathbf{S} \times \mathbf{S} \rightarrow \Re$ as a function that provides the "fitness" of a relationship. The intuition behind the fitness $f((s_i,\ell_i),(s_j,\ell_j))$ is that it is associated with the desirability for the attacker (who is currently on system s_i with level ℓ_i) to achieve level of access ℓ_j on system s_j. If $(s_i,s_j) \notin R$ then $f((s_i,\ell_i),(s_j,\ell_j)) = 0$.

We note that, there are mature pieces of software for generating the graphical structure of the IPN along with the success probability and fitness function such as Cauldron and Lincoln Labs' NetSpa. Additionally, there are vulnerability databases that can aide in the creation of an IPN as well. For instance, in NIST's NVD database[3], impact and attack difficulty can map to fitness and the inverse of the probability of success. While we are currently working with Cauldron to generate the IPN, we do not focus on the creation of the structure in this paper, but rather on reducing the overall probability of success of the intruder.

We assume that if a user has no access (i.e., level 0 access) to a system s then the probability of successfully infiltrating another system s' from that system is 0, which is why property 1 in Definition 1 above is valid and necessary. Similarly, property 2 articulates the fact that if no edge exists between s and s' then the likelihood of a successful attack on s' originating from s is also 0. Finally, property 3 defines the intuitive assumption that if an attacker can complete an attack with a certain probability of success then if he conducts the same attack with a higher level of permissions on the original system his probability of success must be at least the same as it was in the original attack.

Example 2.1. *Consider the simple network displayed in Fig. 1 which represents a subset of a real world network we examined. In this network a user can have one of two levels of access on each system; he can have guest privileges $(\ell_1 = 1)$ or root privileges $(\ell_2 = 2)$. The attacker begins with root privileges on his personal device $(s_1,2)$. The network is displayed in full in Fig. 2 where nodes represent system-level pairs and edges represent logical connections between them. All transitions between system-level pairs will either involve transitioning to a new system or to a higher level on the current system. Edges representing gaining higher access on the current system are red and bold in Fig. 2. Given two system-level pairs $((s_i,\ell_i),(s_j,\ell_j))$, the fitness of the relationship[4] between them $(f((s_i,\ell_i),(s_j,\ell_j)))$ is shown on the edge between them as f, and the conditional success probability function $(\pi((s_i,\ell_i),(s_j,\ell_j)))$ is shown as π. For example, in the sample network, when the attacker begins with root access on his personal computer $(s_1,2)$ he can gain guest privileges on the inside router $(s_2,1)$ with a fitness of 1 and a probability of success of 0.8. For ease of reading, for all system-level pairs where $f((s_i,\ell_i),(s_j,\ell_j)) = 0$ the edge is not displayed. The probability of a successful attack occurring is the product of the probability that the attack succeeds and the probability that the attack is selected by the intruder. In our sample network, then, when the intruder has guest privileges on the inside router $(s_2,1)$ his probability of successfully gaining root access on that router $(s_2,2)$ in the next time step is $\frac{1}{1+1} \times 0.6 = 0.3$.*

Penetration Sequence. A *penetration sequence* is simply a sequence of system-level pairs $\langle (s_0,\ell_0),\ldots,(s_n,\ell_n)\rangle$ such that for each $a = (s_i,\ell_i), b = (s_{i+1},\ell_{i+1})$ in the sequence we have $r = (s_i,s_{i+1}) \in R$, $\pi(a,b), f(a,b) > 0$. For a sequence σ we shall denote the number of system-level pairs with the notation $|\sigma|$. For a given sequence σ, let σ_m be the sub-sequence of σ consisting of the first $m-1$ system-level pairs in σ. For a sequence $\sigma = \langle (s_0,\ell_0),\ldots,(s_n,\ell_n)\rangle$, $curSys(\sigma) = s_n$, $curLvl(\sigma) = \ell_n$ and $cur(\sigma) = (s_n,\ell_n)$. We use the notation $next(\sigma)$ to denote the set of system-level pairs that could occur next in the sequence. Formally:

$$next(\sigma) = \{(s,\ell) \in \mathbf{S} \mid \ell > 0 \wedge (curSys(\sigma),s) \in R \wedge \nexists \ell' \geq \ell \text{ s.t. } (s,\ell') \in \sigma\} \quad (4)$$

[3] http://nvd.nist.gov/

[4] We note that in this particular example we have set the fitness values to all be one. Note that this is just one way to specify a fitness function - perhaps in the case with no information about desirability of a system (i.e. akin to using uniform priors). All of our results and algorithms are much more general and allow for arbitrary fitness values.

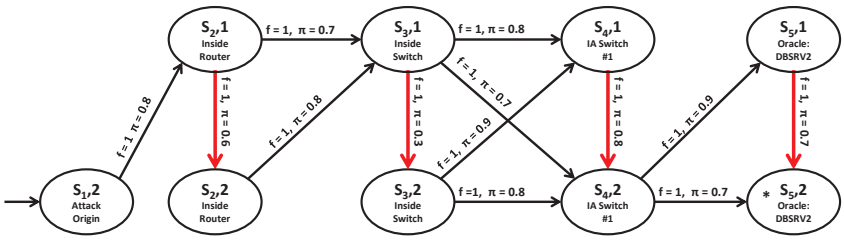


Figure 2: The sample network. Nodes are system-level pairs.

Example 2.2. *Fig. 3 depicts all five possible penetration sequences by which the attacker can gain root access to the Oracle: DBSRV2* $(s_5,2)$ *in our sample network in five time steps or fewer. The penetration sequences are labeled as* σ_1 *through* σ_5.

Model of the Intruder's Actions. Now, we shall describe our model. Consider an attacker who has infiltrated through a sequence of systems specified by σ. If the penetration is to continue, the intruder must select a system-level pair from $next(\sigma)$. The intruder selects exactly one system-level pair (s,ℓ) with the following probability:

$$\frac{f(cur(\sigma),(s,\ell))}{\sum_{(s',\ell')\in next(\sigma)} f(cur(\sigma),(s',\ell'))} \tag{5}$$

Hence, the probability of selection is proportional to the relative fitness of (s,ℓ) compared to the other options for the attacker. This aligns with our idea of fitness: an intruder will attempt to gain access to systems that are more "fit" with respect to his expertise, available tools, desirability of the next system, etc. Note that this probability of selection is not tied to the intruder's probability of success. In fact, we consider the two as independent. Hence, the probability that an intruder selects *and* successfully reaches (s,ℓ) can be expressed as follows:

$$\frac{f(cur(\sigma),(s,\ell))\pi(cur(\sigma),(s,\ell))}{\sum_{(s',\ell')\in next(\sigma)} f(cur(\sigma),(s',\ell'))} \tag{6}$$

Hence, given that the attacker starts at a certain (s,ℓ), we can compute the **sequence probability** or probability of taking sequence σ (provided that σ starts at (s,ℓ) – this probability would be zero otherwise).

$$\prod_{i=0}^{|\sigma|-2} \frac{f(cur(\sigma_i),cur(\sigma_{i+1}))\pi(cur(\sigma_i),cur(\sigma_{i+1}))}{\sum_{(s,\ell)\in next(\sigma_i)} f(cur(\sigma_i),(s,\ell))} \tag{7}$$

Hence, for a given initial (s,ℓ) and ending (s',ℓ'), and length $t+1$, we can compute the probability of starting at (s,ℓ) and ending at (s',ℓ') in t time-steps or less by taking the sum of the sequence probabilities for all valid sequences that meet that criterion. Formally, we shall refer to this as the *penetration probability* and for a given $\mathsf{IPN}, t, (s,\ell), (s',\ell')$ we shall denote this probability as $\mathsf{Pen}^t_{\mathsf{IPN}}((s,\ell),(s',\ell'))$. Intuitively, $\mathsf{Pen}^t_{\mathsf{IPN}}((s,\ell),(s',\ell'))$ is the probability that an attacker at system s with level of access ℓ reaches system s' with level of access ℓ' or greater in t time steps or less.

Example 2.3. *The probability of the attacker successfully gaining access to* $(s_5,2)$ *in five time steps or fewer is the sum of the probabilities of each of the five possible penetration sequences depicted in Fig. 3. For each penetration sequence* σ_n *the probability of the attacker successfully gaining access to* $(s_5,2)$ *through that particular sequence is* p_n. *For the sample network:* $p_1 = 0.023$, $p_2 = 0.021$, $p_3 = 0.021$, $p_4 = 0.004$, $p_5 = 0.016$. *Thus, the total probability of a successful attack occurring on system 5 at level 2 in three time steps or fewer is* $\mathsf{Pen}^t_{IPN}(s_1,2) = 0.085$.

3 DISTRACTION CHAINS AND CLUSTERS

We now introduce the idea of a *distraction chain*. A distraction chain is simply a sequence of decoy systems that we wish to entice an adversary to explore to distract him from the real systems of the network. In order to entice the adversary to explore a distraction chain, we propose adding one-way *distraction clusters* to S. Hence, the adversary enters such a distraction cluster and is delayed from returning to the actual network for a number of time steps proportional to the size of the cluster. Ideally, a distraction cluster would be large enough to delay the attacker for a long time; however, larger distraction clusters will obviously require more resources to construct. Distraction clusters differ from honeypots because they do not prevent intruders from reaching other portions of the network, thus minimizing the risk of the intruder realizing that he is trapped. Again, the goal of a honeypot is to prevent an attacker from completing his attack by trapping him under the assumption

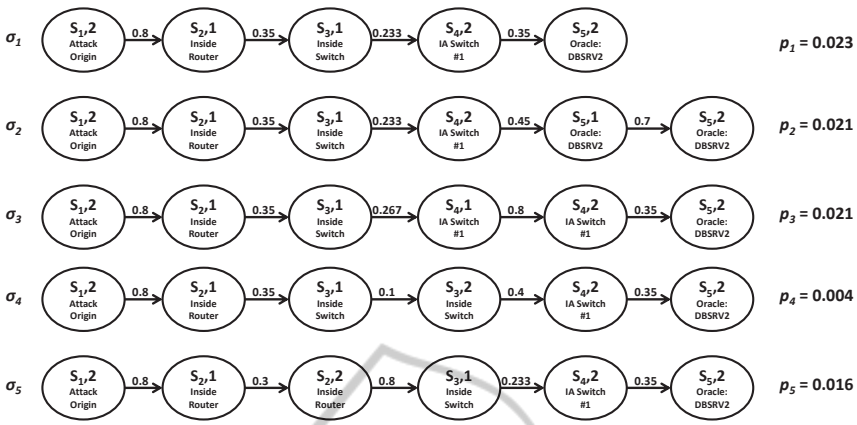


Figure 3: All possible penetration sequences with five time steps or fewer.

that once he is trapped he will not leave it, while the goal of a distraction cluster is – more realistically – to delay the attacker in order to reduce the probability that he will successfully complete his attack in a given amount of time.

Clearly, a valid distraction cluster would be connected to the rest of the system through a one-way connection and the cluster must be created in a manner where there is at least one (preferably many) distraction chain of the necessary length (based on an expected limit of time we expect the intruder to remain in the network before discovery).

For now we shall leave the creation of distraction clusters to future work (e.g., the work of (Píbil et al., 2012) may provide some initial insight into this problem) and instead focus on the problem of adding distraction clusters to a system. The configuration of the first system of a distraction cluster will be a key element in setting up a distraction chain. In particular, the open ports, patch level, installed software, operating system version, and other vulnerabilities present on that lead system, as well as any references of that system found elsewhere on the network will dictate how "fit" an attacker will determine such a system to be and the probability of success he will have in entering into the distraction chain. Note that the fitness of this first system cannot be arbitrarily high and should be considered based on a realistic assessment of why the attacker would select such a system. Further, the probability of an adversary obtaining privileges on such a system should be set in such a way where it is not overly simple for the intruder to gain access - or he might suspect it is a decoy. Additionally, the last system in the distraction cluster must be configured in a way to reconnect it to the actual network.

Throughout the paper, we will consider a set of configurations available to the defender denoted $\mathbf{CFG}$. For instance, $\mathbf{CFG}$ may consist of a predetermined set of virtual machine images available to the security team. In addition we will consider a set of potential distraction clusters denoted $\mathbf{CL}$. For each $\mathbf{cl} \in \mathbf{CL}$ there exists value $t_{\mathbf{cl}} \in \mathbb{N}$, a natural number equal to the minimum number of time steps elapsed before an attacker is able to leave the cluster and return to the network. For each $\mathbf{cfg} \in \mathbf{CFG}$, for the lead and last systems in the distraction cluster (resp. s_{dc1}, s_{dc2}) there are associated conditional probabilities (resp. $\pi_{\mathbf{cfg,cl}} : \mathbf{S} \times \{(s_{dc1}, \ell)\} \rightarrow [0,1]$, $\pi_{\mathbf{cfg,cl}} : \{(s_{dc2}, \ell)\} \times \mathbf{S} \rightarrow [0,1]$) and fitness function (resp. $f_{\mathbf{cfg,cl}} : \mathbf{S} \times \{(s_{dc1}, \ell)\} \rightarrow \Re$, $f_{\mathbf{cfg,cl}} : \{(s_{dc2}, \ell)\} \times \mathbf{S} \rightarrow \Re$). These functions are based on the software installed on and the vulnerabilities present in that particular configuration. Hence, once a distraction cluster is added it contains a lead system configured with configuration **cfg**. The resulting IPN formed with the addition of distraction cluster includes conditional probability and fitness functions that are the concatenation of $\pi, \pi_{\mathbf{cfg,cl}}$ and $f, f_{\mathbf{cfg,cl}}$ respectively. Additionally, for each $s \in S$ we add (s, s_{dc1}) to R where there exists s, ℓ where $\ell > 0$ s.t. $\pi_{\mathbf{cfg,cl}}((s,\ell),(s_{dc1},1)) > 0$ and $f_{\mathbf{cfg,cl}}((s,\ell),(s_{dc1},1)) > 0$ and we add (s_{dc2}, s) to R where there exists s, ℓ where $\ell > 0$ s.t. $\pi_{\mathbf{cfg,cl}}((s_{dc2},1),(s,\ell)) > 0$ and $f_{\mathbf{cfg,cl}}((s_{dc2},1),(s,\ell)) > 0$. In other words, a logical connection is formed from all systems in S for which, if connected to s_{dc1} or s_{dc2} there is a non-zero probability that the intruder can gain a level of access greater than $\ell = 0$. We can easily restrict which relationship are added by modifying the $f_{\mathbf{cfg,cl}}$ functions. For a given IPN, set of distraction clusters, and set of configuration-cluster pairs $\mathrm{PCP} \subseteq \mathbf{CFG} \times \mathbf{CL}$, we will use the notation $\mathrm{IPN} \cup \mathrm{PCP}$ to denote the concatenation of the intrusion penetration network and the set of configuration-cluster pairs.

Adding Distraction Clusters. We now have the pieces we need to introduce the formal definition of our problem.

Definition 2 (Cluster Addition Problem)**.** *Given IPN (S,R,π,f), systems $s,s' \in S$, access levels $\ell_s,\ell_{s'} \in \mathcal{L}$, set of potential distraction clusters* **CL***, set of configurations* **CFG***, natural number k, real number $x \in [0,1]$, and time-limit t, find $PCP \subseteq$ **CFG** $\times$ **CL** s.t. $|PCP| \leq k$ and $Pen^t_{IPN}((s,\ell),(s',\ell')) - Pen^t_{IPN \cup PCP}((s,\ell),(s',\ell')) > x$.*

Example 3.1. *Following along with our sample network, the set of configurations,* **CFG***, displayed in Fig. 4, and with $k = 2$, $t = 5$, $x = 0.05$, and* **CL** $= \{$**cl**$\}$ *(with $t_{\textbf{cl}} = 6$) we find that $PCP = \{(\textbf{cfg}_1, \textbf{cl}), (\textbf{cfg}_3, \textbf{cl})\}$ is a solution to the Chain Addition Problem because $Pen^t_{IPN}(s_1,2) - Pen^t_{IPN \cup PCP}(s_1,2) = 0.063 > 0.05 = x$ The modified IPN is displayed in Fig. 5.*

Unfortunately, this problem is difficult to solve exactly by the following result (the proof is provided in the appendix).

Theorem 1. *The Cluster Addition Problem is NP-hard and the associated decision problem is NP-Complete when the number of sequences from (s,ℓ) to (s',ℓ') is a polynomial in the number of nodes in the intruder penetration network.*

For a given instance of the Cluster Addition Problem, for a given $PCP \subseteq$ **CFG** $\times$ **CL** let $orc(PCP) = Pen^t_{IPN}((s,\ell),(s',\ell')) - Pen^t_{IPN \cup PCP}((s,\ell),(s',\ell'))$. In the optimization version of this problem, this is the quantity we attempt to optimize. Unfortunately, as a by-product of Theorem 1 and the results of (Feige, 1998) (Theorem 5.3), there are limits to the approximation we can be guaranteed to find in polynomial time.

Theorem 2. *With a cardinality constraint, finding set PCP s.t. $orc(PCP)$ cannot be approximated in PTIME within a ratio of $\frac{e-1}{e} + \varepsilon$ for some $\varepsilon > 0$ (where e is the base of the natural log) unless P=NP.*

However, orc does have some useful properties.

Lemma 1 (Monotonicity)**.** *For $PCP' \subseteq PCP \subseteq$ **CFG** $\times$ **CL**, $orc(PCP') \leq orc(PCP)$.*

Lemma 2 (Submodularity)**.** *For $PCP' \subseteq PCP \subseteq$ **CFG** $\times$ **CL** and $pc = (\textbf{cfg}, \textbf{cl}) \notin PCP$ we have:*

$$orc(PCP \cup \{pc\}) - orc(PCP) \leq orc(PCP' \cup \{pc\}) - orc(PCP')$$

4 ALGORITHMS

Greedy Approach. Now we introduce our greedy heuristic for the Chain Addition Problem.

Algorithm 1: GREEDY_CLUSTER.

Require: Systems s,s', access levels $\ell_s,\ell_{s'}$, set of distraction chains **CL**, set of protocols **CFG**, natural number k and time limit t

Ensure: Subset $PCP \subseteq$ **CFG** $\times$ **CL**

```
1:  PCP = ∅
2:  while |PCP| ≤ k do
3:      curBest = null, curBestScore = 0
4:      for (cfg, cl) ∈ (CFG × CL) − PCP do
5:          curScore = orc(PCP ∪ {(cfg, cl)}) − orc(PCP)
6:          if curScore ≥ curBestScore then
7:              curBest = (cfg, cl)
8:              curBestScore = curScore
9:          end if
10:     end for
11:     PCP = PCP ∪ {curBest}
12: end while
13: return PCP.
```

Example 4.1. *When run on our sample network, GREEDY_CLUSTER selects $(\textbf{cfg}_1, \textbf{cl})$ in the first iteration of the loop at line 4, lowering $Pen^t_{IPN \cup PCP}(s_1,2)$ from 0.085 to 0.037. In the next iteration GREEDY_CLUSTER selects $(\textbf{cfg}_3, \textbf{cl})$, lowering $Pen^t_{IPN \cup PCP}(s_1,2)$ from 0.037 to 0.022. At this point, since $|PCP| = 2 = k$, GREEDY_CLUSTER returns $PCP = \{(\textbf{cfg}_1, \textbf{cl}), (\textbf{cfg}_3, \textbf{cl})\}$, a solution to the Chain Addition Problem under our given constraints.*

Though simple, GREEDY_CLUSTER, can provide the best approximation guarantee unless P=NP under the condition that orc can be solved in PTIME. Consider the following theorem.

Theorem 3. *GREEDY_CLUSTER provides the best PTIME approximation of orc unless P=NP if orc can be solved in PTIME.*

However, the condition on solving orc in PTIME may be difficult to obtain in the case of a very general IPN. Though we leave the exact computation of this function as an open problem, we note that the straight-forward approach for computation would imply the need to enumerate all sequences from (s,ℓ) to (s',ℓ') - which can equal the number of t-sized (or smaller) permutations of the elements of **S** - a quantity that is not polynomial in the size of IPN. Hence, a method to approximate orc is required in practice. Our intuition is that the expensive computation of the penetration probability can be approximated by summing up the sequence probabilities of the most probable sequences from (s,ℓ) to (s',ℓ'). The intuition of approximating the probability of a path between two nodes in a network (given a diffusion model) based on high-probability path computation was introduced in (Chen et al., 2010) where it was applied to the maximum influence problem. However, our approach differs in that (Chen et al., 2010) only considered the

$$\mathbf{cfg}_1:\begin{cases}\mathbf{f}(x,(s_{dc},1))=\begin{cases}1, & x=(s_1,2)\\ 1, & x=(s_2,2)\\ 0, & otherwise\end{cases}\\ \pi(x,(s_{dc},1))=\begin{cases}0.9, & x=(s_1,2)\\ 0.8, & x=(s_2,2)\\ 0, & otherwise\end{cases}\end{cases}\quad \mathbf{cfg}_2:\begin{cases}\mathbf{f}(x,(s_{dc},1))=\begin{cases}1, & x=(s_2,2)\\ 1, & x=(s_3,2)\\ 0, & otherwise\end{cases}\\ \pi(x,(s_{dc},1))=\begin{cases}0.7, & x=(s_2,2)\\ 0.9, & x=(s_3,2)\\ 0, & otherwise\end{cases}\end{cases}\quad \mathbf{cfg}_3:\begin{cases}\mathbf{f}(x,(s_{dc},1))=\begin{cases}1, & x=(s_4,1)\\ 1, & x=(s_4,2)\\ 0, & otherwise\end{cases}\\ \pi(x,(s_{dc},1))=\begin{cases}0.8, & x=(s_4,1)\\ 0.9, & x=(s_4,2)\\ 0, & otherwise\end{cases}\end{cases}$$

$$\mathbf{cfg}_4:\begin{cases}\mathbf{f}(x,(s_{dc},1))=\begin{cases}1, & x=(s_3,1)\\ 1, & x=(s_3,2)\\ 0, & otherwise\end{cases}\\ \pi(x,(s_{dc},1))=\begin{cases}0.7, & x=(s_3,1)\\ 0.8, & x=(s_3,2)\\ 0, & otherwise\end{cases}\end{cases}\quad \mathbf{cfg}_5:\begin{cases}\mathbf{f}(x,(s_{dc},1))=\begin{cases}1, & x=(s_3,2)\\ 1, & x=(s_4,2)\\ 0, & otherwise\end{cases}\\ \pi(x,(s_{dc},1))=\begin{cases}0.8, & x=(s_3,2)\\ 0.7, & x=(s_4,2)\\ 0, & otherwise\end{cases}\end{cases}$$

Figure 4: The set **CFG** of possible configurations available to the defender.

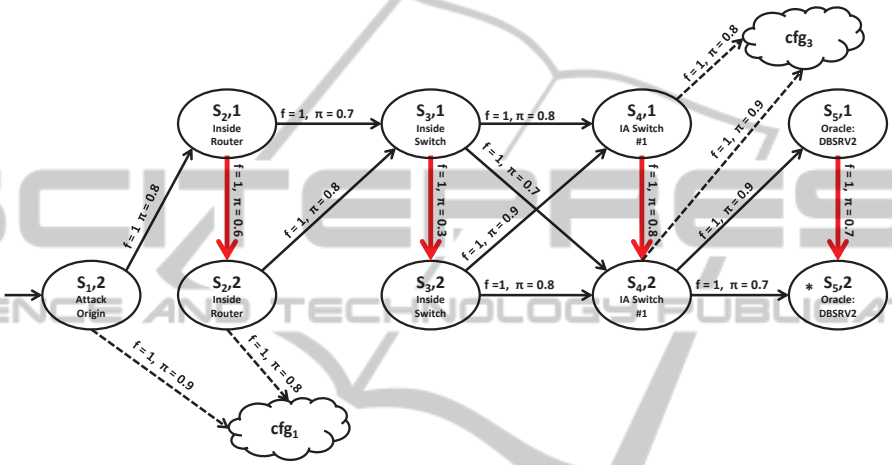


Figure 5: The updated network with PCP added.

most probable path, as we consider a set of most probable paths. Our intuition in doing so stems from the fact that alternate paths can contribute significantly to the probability specified by $\mathsf{Pen}^t_{\mathsf{IPN}}((s,\ell),(s',\ell'))$.

Computing orc. Next, we introduce a simple sampling-based method that randomly generates sequences from (s,ℓ) to (s',ℓ') of the required length. These sequences are generated with a probability proportional to their sequence probability, hence the computation of orc based on these samples is biased toward the set of most-probable sequences from (s,ℓ) to (s',ℓ'), which we believe will provide a close approximation to orc. Our pre-processing method, ORC_SAM below generates a set of sequences that the attacker could potentially take. Using the sequences from SEQ, we can then calculate the penetration probability ($\mathsf{Pen}^t_{\mathsf{IPN}}((s,\ell),(s',\ell'))$) by summing over the probabilities over the sequences in SEQ as opposed to summing over all sequences (a potentially exponential number).

Extensions. Our approach is easily generalizable for studying other problems related to Cluster-Addition. For instance, suppose an intruder initiates his infiltration from one of a set of systems chosen based on a probability distribution. We can encode this problem in Chain-Addition by adding a dummy system to the penetration network and establishing relationships to each of the potential initial systems in the set. The fitness and conditional probability functions can then be set up in a manner to reflect the probability distribution over the set of potential initial systems. Note

Algorithm 2: ORC_SAM.

Require: Systems s,s', access levels $\ell_s,\ell_{s'}$, natural numbers $t, maxIters$
Ensure: Set of sequences SEQ

$curIters = 0, SEQ = \emptyset$
while $curIters < maxIters$ **do**
  $(s_i,\ell_i) = (s,\ell_s)$, $curSeq = \langle(s,\ell_s)\rangle$, $curLth = 0$
  while $curLth < t$ and $(s_i,\ell_i) \neq (s',\ell_{s'})$ **do**
    Select (s',ℓ') from the set $(\eta_i \times \mathcal{L}) - curSeq$ with a probability of $\frac{f((s_i,\ell_i),(s',\ell'))\pi((s_i,\ell_i),(s',\ell'))}{\sum_{(s_j,\ell_j)\in(\eta_i\times\mathcal{L})-curSeq} f((s_i,\ell_i),(s_j,\ell_j))}$
    $curSeq = curSeq \cup \{(s',\ell')\}$
  end while
  If $(s',\ell') = (s',\ell_{s'})$ then $SEQ = SEQ \cup \{curSeq\}$
  $curIters += 1$
end while
Return SEQ

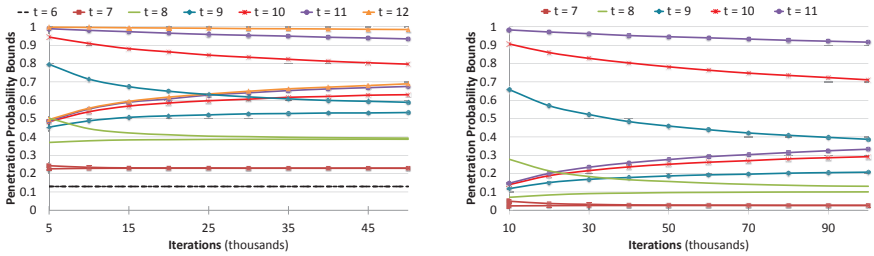
that this problem would still maintain the same mathematical properties and guarantees of our already described method to solve the Cluster-Addition problem. Another related problem that can be solved using our methods (again with only minor modification) is to minimize the expected number of compromised systems in a set of potential targets. As the expected number of systems would simply be the sum of the penetration probability for each of those machines, a simple modification to Line 5 of GREEDY_CLUSTER would allow us to represent this problem, in this case instead of examining orc we would examine the sum of the value for orc returned for paths going to each of the potential targets. Note that by the fact that positive linear combinations of submodular functions are also submodular, we are able to retain our theoretical guarantees here as well.

5 EXPERIMENTAL RESULTS

We conducted several experiments in order to test the effectiveness of our algorithm under several circumstances. As explained in the preceding running example we tested our algorithm on a small network derived from a real network. As shown, by adding two distraction clusters to that network we were able to reduce the probability of a successful attack from 8.5% to 2.2%, an almost 75% reduction. While this result shows relevance to a real-world network and displays the algorithm on an easily understood scale, we also wanted to test the scalability of our algorithm by experimenting on larger networks. Due to the difficulty of obtaining large datasets for conducting research in cyber security we generated random graphs that resembled the network topology of the types of penetration networks we wish to examine, allowing us to test the scalability of our algorithm. To do this we generated networks which were divided into layers, meaning that each system is only connected to other systems in its own layer or to systems in the next layer. This is important because it means that in a network with n layers, the shortest path between the source of the attack and the target is $n-1$. We did this because it mimics the topology of the real-world networks we wished to replicate in our experiments. It provides structure to our networks so that the simulated attackers will have to gain access to systems across a series of layers before gaining access to a system from which they can access their target. For all of our experiments we assumed that $\pi = 1$. We did this because it makes it easier to see and understand the relationship between the lower and upper bounds of the penetration probability. This does not compromise the value of our results because adding distraction clusters to a network only effects the relative fitness of an attack, so the probability of success for any one attack does not influence our results.

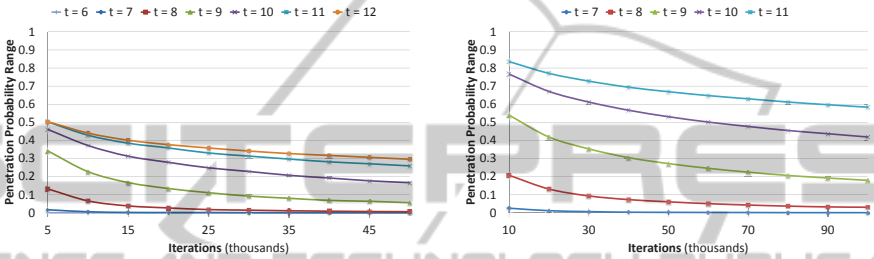
The first test we ran sought to measure the effectiveness of the ORC_SAM sampling algorithm by testing against the value of t—the maximum length of a penetration sequence—and the number of iterations conducted in the sampling algorithm (ORC_SAM). We found—as would be expected—that the number of iterations required for the most accurate possible result increases as t increases and as the number of system-level pairs (nodes) in the network increases. We ran our tests on randomly generated networks with 50 and 100 systems each with 2 levels per system. Figs. 6, 7, and 8 display the results of this test. Fig. 6 shows that, as t increases, the number of iterations of ORC_SAM required to get an accurate prediction of the penetration probability increases. For high values of t, with a relatively small number of iterations, the difference between the upper and lower bounds of the test is very large. This effect is displayed more clearly in Fig. 7 in which this difference is graphed for each value of t against the number of iterations. Finally, Fig. 8 displays the time in seconds that each individual test took to run.

One important factor when implementing distraction clusters is determining the location in the network in which to place them. The initial instinct may be that the goal should be to distract the attacker early in his pursuit and thus the clusters should be placed close to the source of the attack; however, this has little effect on the resulting decrease in penetration probability. We conducted experiments on the same two networks from the previous test in which we randomly generated a cluster and connected it to 10 systems in a given layer for each layer in the system (other than the first and last layers which only have one system each—the source and the target, respectively). We then measured the changes in penetration probability that occurred as a result. For each network we conducted 10 trials. Our results showed little evidence that the proximity to the source of the attack matters and instead suggest that the size of the layer is the more influential variable. Clusters with configurations that connected them to systems in a layer with a relatively small number of systems showed a larger decrease in the penetration probability. In addition to being suggested by our evidence, this also makes intuitive sense because providing the attacker a distraction in a layer with fewer systems means that more of his options will lead into the distraction cluster rather than allowing him to progress forward with his attack.



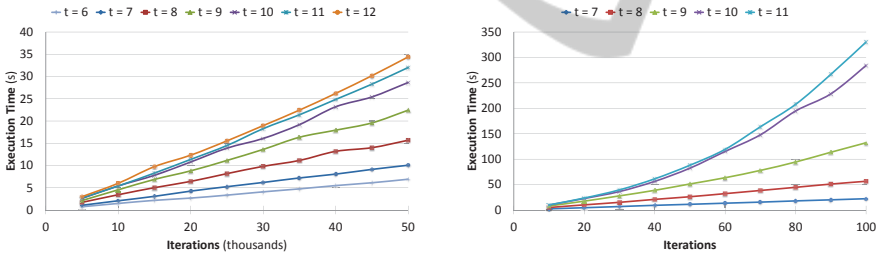
(a) Network with 50 systems. (b) Network with 100 systems.

Figure 6: Upper and lower bounds of the predicted penetration probability vs. number of iterations of ORC_SAM, for different values of t.



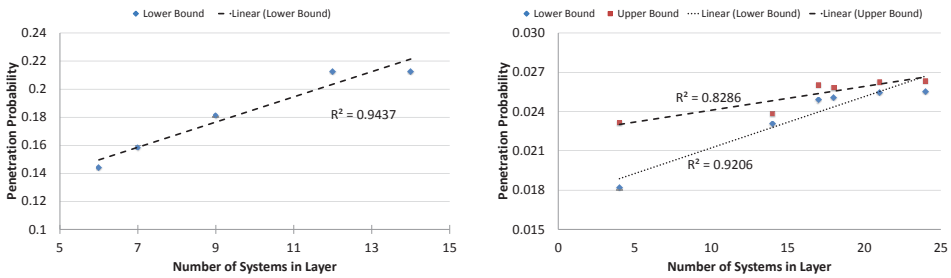
(a) Network with 50 systems. (b) Network with 100 systems.

Figure 7: Difference between upper and lower bounds of the predicted penetration probability vs. number of iterations of ORC_SAM, for different values of t.



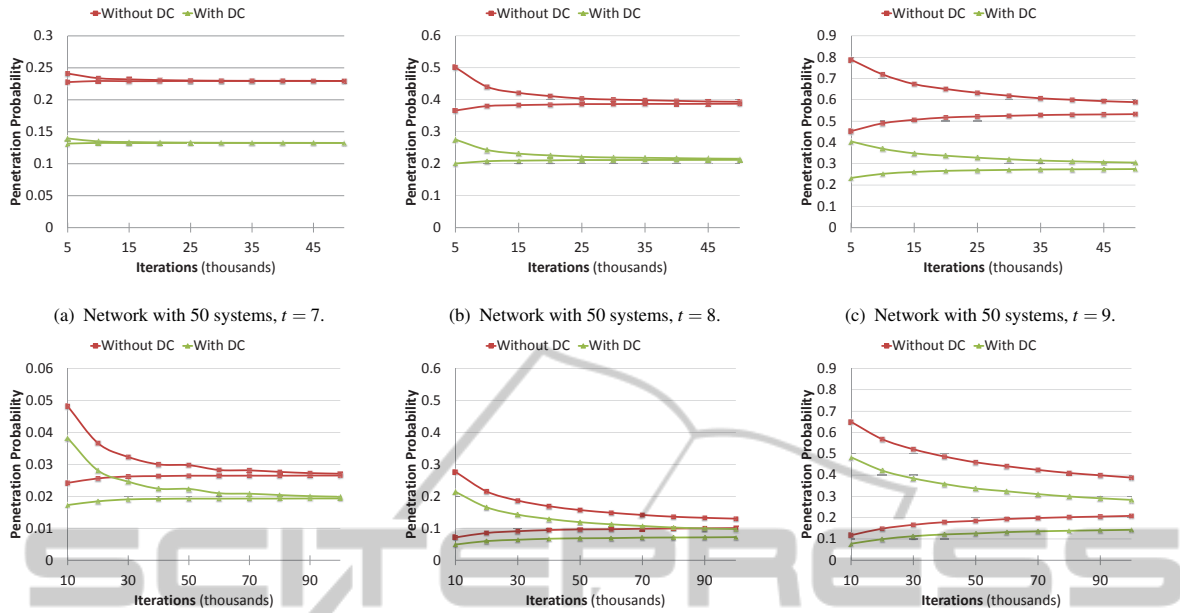
(a) Network with 50 systems. (b) Network with 100 systems.

Figure 8: Execution time vs. number of iterations of ORC_SAM, for different values of t.



(a) Network with 50 systems. (b) Network with 100 systems.

Figure 9: Correlation between the number of systems in a layer and the penetration probability when the distraction cluster was configured to systems in that layers (note that for the network with 50 systems, the bounds nearly matched and only the lower is displayed).



(a) Network with 50 systems, $t = 7$.
(b) Network with 50 systems, $t = 8$.
(c) Network with 50 systems, $t = 9$.
(d) Network with 100 systems, $t = 7$.
(e) Network with 100 systems, $t = 8$.
(f) Network with 100 systems, $t = 9$.

Figure 10: Penetration probabilities before and after adding the cluster chosen by GREEDY_CLUSTER.

Fig. 9 shows the results from the tests. The y-axis represents the average penetration probability with the configuration added to the layer while the x-axis represents the number of systems in the layer. The dotted line shows the result of the linear regression which shows a linear relationship between the two variables. While we show a linear regression the relationship is not exactly linear as the effect is lessened as the layers become larger. Note that the average percent decrease in penetration probability when clusters were added to the network with 50 systems was 20.7% while the average when they were added to the larger network was only 6.4%. We hope to do further testing in the future to identify other variables that affect the outcome. In addition to investigating the ideal location for placing a cluster we also examined the effects of the number of iterations of ORC_SAM that are run on the results of the test. These results are shown in Fig. 10. As explained earlier, ORC_SAM is a heuristic that generates a random sample of possible paths between the source and target. To test this we ran GREEDY_CLUSTER with varying numbers of iterations run on ORC_SAM. We generated ten possible configurations for GREEDY_CLUSTER to choose from and assigned it to choose the optimal configuration. We ran the test on both networks with iterations of ORC_SAM ranging from 5,000 to 50,000 in increments of 5,000 on the smaller network and from 10,000 to 100,000 in increments of 10,000 on the larger one. We ran this test with the values of t—the maximum length of a penetration sequence—set to 7, 8, and 9 for three different tests on both networks. We found that regardless of the number of iterations run or the value of t, GREEDY_CLUSTER selected the same configuration and there was a small difference between the percent decrease in penetration probabilities after the clusters were added to the network. This suggests that accurate predictions about the effects of a cluster on a network can be made without sampling all possible paths from the source to the target. Additionally, as stated earlier, it suggests that seeking to reduce the probability of the attack being reached in the shortest number of steps will also help reduce the probability of paths of greater lengths.

6 CONCLUSIONS

Despite significant progress in the area of intrusion prevention, it is well known that not all intrusions can be prevented, and additional lines of defense are needed in order to cope with attackers capable of circumventing existing intrusion prevention systems.

In this paper, we proposed a novel approach to *intrusion prevention* that aims at delaying an intrusion, rather than trying to stop it, so as to control the probability that an intruder will reach a certain goal within a specified amount of time. In the future, we plan to do more experiments to better understand the factors that influence the ideal location of distraction clus-

ters and to determine the relationship between penetration probability, iterations of ORC_SAM, and the selection of configurations for adding distraction clusters as well as investigate the impact of the length of distraction clusters on the network and the location of the point at which the distraction clusters reconnect to the network.

REFERENCES

Abbasi, F., Harris, R., Moretti, G., Haider, A., and Anwar, N. (2012). Classification of malicious network streams using honeynets. In *Global Communications Conference (GLOBECOM)*, pages 891–897.

Alpcan, T. and Baar, T. (2010). *Network Security: A Decision and Game-Theoretic Approach*. Cambridge University Press, New York, NY, USA, 1st edition.

Chen, C.-M., Cheng, S.-T., and Zeng, R.-Y. (2013). A proactive approach to intrusion detection and malware collection. *Security and Communication Networks*, 6(7):844–853.

Chen, W., Wang, C., and Wang, Y. (2010). 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.

Evans, D., Nguyen-Tuong, A., and Knight, J. C. (2011). *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*, chapter Effectiveness of Moving Target Defenses, pages 29–48. Springer.

Feige, U. (1998). A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652.

Jajodia, S., Ghosh, A. K., Subrahmanian, V. S., Swarup, V., Wang, C., and Wang, X. S., editors (2013). *Moving Target Defense II: Application of Game Theory and Adversarial Modeling*, volume 100 of *Advances in Information Security*. Springer, 1st edition.

Jajodia, S., Ghosh, A. K., Swarup, V., Wang, C., and Wang, X. S., editors (2011). *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*, volume 54 of *Advances in Information Security*. Springer.

Manadhata, P. K. and Wing, J. M. (2011). An attack surface metric. *IEEE Transactions on Software Engineering*, 37(3):371–386.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. (1978). An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294.

Píbil, R., Lisý, V., Kiekintveld, C., Bosanský, B., and Pechoucek, M. (2012). Game theoretic model of strategic honeypot selection in computer networks. In *GameSec*, pages 201–220.

Shakarian, P., Shakarian, J., and Ruef, A. (2013). *Introduction to Cyber-Warfare: A Multidisciplinary Approach*. Syngress.

Sweeney, P. and Cybenko, G. (2012). An analytic approach to cyber adversarial dynamics. In *SPIE Defense, Security, and Sensing*, pages 835906–835906. International Society for Optics and Photonics.

Williamson, S. A., Varakantham, P., Hui, O. C., and Gao, D. (2012). Active malware analysis using stochastic games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 29–36, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

APPENDIX

Proof of Theorem 1. *The Cluster Addition Problem is NP-hard and the associated decision problem is NP-Complete when the number of sequences from (s,ℓ) to (s',ℓ') is a polynomial in the number of nodes in the intruder penetration network.*

Proof. Membership in NP (if the number of sequences from (s,ℓ) to (s',ℓ') is polynomial can be shown when the certificate is the set of configuration-cluster pairs.

For NP-hardness, consider the set cover problem (Feige, 1998) where the input consists of as set of elements S, a family of subsets of S denoted as H, and natural numbers K, X. The output of this problem is a subset of H of size K or less such that their union covers X or more elements of S. This problem is NP-hard and can be embedded into an instance of the cluster addition problem as follows: $\mathcal{L} = \{0,1\}$, $S = \{s,t\} \cup \{v_w | w \in S\}$, $R = \{(s,v_w),(v_w,t) | w \in S\}$, $\forall w \in S$, set $\pi((s,1),(v_w,1)) = 1$ and $\pi((v_w,1),(t,1)) = 1$ (otherwise set π by definition), $\forall w \in S$, set $f((s,1),(v_w,1)) = 1$ and $f((v_w,1),(t,1)) = 1$ (otherwise set f by definition), $\ell_s, \ell_{s'} = 1$, $\mathbf{CL} = \{\mathbf{cl}\}$, $\mathbf{CFG} = \{\mathbf{cfg}_h | h \in H\}$, for each $\mathbf{cfg}_h \in \mathbf{CFG}$ and $\mathbf{cl}$ set $\pi_{\mathbf{cfg}_h,\mathbf{cl}}(v_w) = 1$ if $s \in h$ and 0 otherwise, for each $\mathbf{cfg}_h \in \mathbf{CFG}$ and $\mathbf{cl}$ set $f_{\mathbf{cfg}_h,\mathbf{cl}}(v_w) = |S|$ if $s \in h$ and 0 otherwise, $x = 1 - \frac{|S|-X}{|S|} - \frac{X}{|S|(|S|+1)}$, $k = K$, and $t = 2$. Clearly this construction can be completed in polynomial time.

Next, we show that a solution to set cover will provide a solution to the constructed cluster-addition problem. If H' is a solution to set cover, select the set $\{\mathbf{cfg}_h | h \in H'\}$. Clearly this meets the cardinality constraint. Note that in the construction, all sequences from $(s,1)$ to $(t,1)$ are of the form $\langle (s,1),(v_w,1),(t,1)\rangle$ where $w \in S$. Note that as every system $v_w \in S - \{s,t\}$ is now connected to a cluster. Hence, each cluster now has had its probability reduced from $1/|S|$ to at most $1/(|S|(|S|+1))$. Hence, $\mathsf{Pen}^t_{\mathsf{IPN}\cup\mathsf{PCP}}((s,\ell),(s',\ell')) < \frac{|S|-X}{|S|} + \frac{X}{|S|(|S|+1)}$ which completes this claim of the proof.

Going the other way, we show that a solution to the constructed cluster-addition problem will provide a solution to set cover. Given cluster-addition solution PCP, consider $H' = \{h | (\mathbf{cfg}_h, \mathbf{cl}) \in \mathsf{PCP}\}$. Note that, by the construction, all elements of PCP are of the form $(\mathbf{cfg}_h, \mathbf{cl})$ where $h \in H'$. Clearly, the cardinality constraint is met by the construction. Suppose, BWOC, H' is not a valid solution to set cover. We note that this must imply that there are some v_s that are not attached to a distraction cluster. Let us assume there are δ number of these systems. Hence, $\mathsf{Pen}^t_{\mathsf{IPN}\cup\mathsf{PCP}}((s,\ell),(s',\ell')) > \frac{|S|-X+\delta}{|S|} + \frac{X-\delta}{|S|(k|S|+1)}$. Let us now assume, by way of contradiction, that this quantity is less than or equal to $\frac{|S|-X}{|S|} + \frac{X}{|S|(|S|+1)}$, which is the upper bound on $\mathsf{Pen}^t_{\mathsf{IPN}\cup\mathsf{PCP}}((s,\ell),(s',\ell'))$ is at least X of the v_s systems have

a distraction cluster:

$$\frac{|S|-X+\delta}{|S|}+\frac{X-\delta}{|S|(k|S|+1)} \leq \frac{|S|-X}{|S|}+\frac{X}{|S|(|S|+1)} \quad (8)$$

$$\frac{\delta(k|S|+1)+X-\delta}{|S|(k|S|+1)} \leq \frac{X}{|S|(|S|+1)} \quad (9)$$

$$(\delta k|S|+X)(|S|+1) \leq X(k|S|+1) \quad (10)$$

However, as $\delta k|S|+X > k|S|+1$ and as $|S|+1 > X$, this give us a contradiction, completing the proof. ■

Proof of Theorem 2. *With a cardinality constraint, finding set PCP s.t. orc(PCP) cannot be approximated in PTIME within a ratio of $\frac{e-1}{e}+\varepsilon$ for some $\varepsilon > 0$ (where e is the base of the natural log) unless P=NP.*

Proof. Follows directly from the construction of Theorem 1 and Theorem 5.3 of (Feige, 1998). — Slightly longer version below: Note that Theorem 1 shows that we can find an exact solution to set-cover in polynomial time using an instance of the Cluster Addition problem. A optimization variant of this problem, the MAX-K-COVER problem (Feige, 1998) takes the same input and returns a subset of H size K whose union covers the maximum number of elements in S. This variant of the problem cannot be approximated within a ratio of $\alpha = \frac{e-1}{e}+\varepsilon$ for some $\varepsilon > 0$ by Theorem 5.3 of (Feige, 1998). ■

Proof of Lemma 1. *For PCP′ ⊆ PCP ⊆ CFG × CL, orc(PCP′) ≤ orc(PCP).*

Proof. Given an instance of the cluster addition problem, consider a single sequence from s to s'. Clearly the probability associated with that sequence will either remain the same with the addition of any configuration-cluster pair to the penetration network as the denominator of the probability of transitioning between system-level pairs will only increase (due to the additive nature of fitness in the denominator). Likewise, subsequent configuration-cluster pair additions will lead to further decrease. Hence, the penetration probability monotonically decreases with the addition of configuration-cluster pairs as it is simply the sum of the sequence probabilities form s to s' of length t which makes orc monotonically increasing, as per the statement. ■

Proof of Lemma 2. *For PCP′ ⊆ PCP ⊆ CFG × CL and $pc = (\mathbf{cfg}, \mathbf{cl}) \notin$ PCP we have:*

$$orc(PCP \cup \{pc\}) - orc(PCP) \leq orc(PCP' \cup \{pc\}) - orc(PCP')$$

Proof. It is well known that a positive, linear combination of submodular functions is also submodular. Hence, without loss of generality, we can prove the statement by only considering a single sequence. Let lth be the length (number of transitions) of the sequence. For the ith (s,ℓ) in the sequence, let $F_{i+1} = \sum_{(s',\ell')\in\eta_i\times\mathcal{L}} f((s,\ell),(s',\ell'))$, $B_{i+1} = \sum_{(s',\ell')\in\mathsf{PCP}'\times\mathcal{L}} f((s,\ell),(s',\ell'))$, $D_{i+1} = \sum_{(s',\ell')\in\mathsf{PCP}\times\mathcal{L}} f((s,\ell),(s',\ell')) - B_{i+1}$, $E_{i+1} = f((s,\ell),pc)$. Using this notation, we can apply the definition of orc and some easy algebra, we obtain that $\prod_i \frac{1}{F_i+B_i+D_i} - \prod_i \frac{1}{F_i+B_i+D_i+E_i}$ is less than or equal to $\prod_i \frac{1}{F_i+B_i} - \prod_i \frac{1}{F_i+B_i+E_i}$. Suppose, BWOC, the statement is false, this implies that $\prod_i \frac{F_i+B_i}{F_i+B_i+E_i}\left(1-\prod_i \frac{F_i+B_i+E_i}{F_i+B_i+D_i+E_i}\right)$ is greater than $1-\prod_i \frac{F_i+B_i}{F_i+B_i+D_i}$. Here, the term $\prod_i \frac{F_i+B_i}{F_i+B_i+E_i}$ decreases as each of the E_i's increase and that this value is no more than 1. Hence, the following must be true (under the assumption that the original statement is false).

$$\prod_i \frac{F_i+B_i+E_i}{F_i+B_i+D_i+E_i} < \prod_i \frac{F_i+B_i}{F_i+B_i+D_i}$$

For the above statement to be true, there must exist at least one i such that $(F_i+B_i+E_i)(F_i+B_i+D_i) < (F_i+B_i)(F_i+B_i+D_i+E_i)$ which implies $D_i, E_i \geq 0$, thus leading us to a contradiction and completing the proof. ■

Proof of Theorem 3. *GREEDY_CLUSTER provides the best PTIME approximation of orc unless P=NP if orc can be solved in PTIME.*

Proof. If orc can be solved in PTIME, it is easy to then show that GREEDY_CLUSTER runs in PTIME. The results of (Nemhauser et al., 1978) show a greedy algorithm provides a $\frac{e}{e-1}$ approximation for the maximization of a non-decreasing submodular function that returns zero on the empty set. Clearly, $orc(\emptyset) = 0$ and we showed that it is non-decreasing and submodular in Lemmas 1 and 2 respectively - which means that GREEDY_CLUSTER provides a $\frac{e}{e-1}$ approximation to the maximization of orc. This matches the theoretical bound proved in Theorem 2 which holds unless P=NP. ■