Trading Resiliency for Security: Model and Algorithms

Tian Bu and Samphel Norden and Thomas Woo
Bell Laboratories
{*tbu, norden, woo*}*@bell-labs.com*

***Abstract*— An attack-resistant network is a purpose-built network to survive attacks; by construction, it should be both *resilient* and *secure*. Resiliency is the ability to provide alternative communication paths should one path become disrupted due to failures or attacks; while security is the ability to contain and limit the impact of compromises. Interestingly, these two can present conflicting demands. In this paper, we provide a *first* formulation of a new class of problems focusing on the engineering of attack-resistant networks. Our model considers both resiliency and security, and uses a notion of *blocking probability* as a rigorous measure for evaluating different network constructions. We propose several efficient approximation algorithms for computing blocking probability and provide bounds for their errors. Based on these algorithms, we introduce a family of heuristics to guide the construction of optimal attack-resistant networks with minimum blocking probabilities. We also present extensive results to evaluate and demonstrate the near-optimal performance of our heuristics and approximation algorithms.**

I. Introduction

Various security threats continue to plague the Internet. Two of the particularly disruptive attacks are host compromise and denial of service (DoS) attacks. In a host compromise attack, an attacker exploits security weaknesses of a host to gain control[1] over the host. In a DoS attack, an attacker tries to overwhelm a victim node by directing a large amount of bogus traffic to it, thus rendering the victim node unable to service legitimate traffic. These two techniques can be used together to effect a more malicious attack, namely, a *distributed DoS* (DDoS) attack. Specifically, an attacker attempts to compromise a large number of hosts, from which DoS attacks can be launched.

Traditionally, most approaches for countering DDoS attacks are *reactive* in nature. That is, they focus on detecting ongoing attacks and devising mitigating mechanisms (e.g., filtering [1]) to alleviate the attacks. More recently, measures that are more *proactive* in nature are being proposed. These include *constructive* approaches where networks are purpose-built using specialized nodes to explicitly resist and survive attacks [2], [3], [4]. These purpose-built networks provide an extended defensive zone around the end servers to deflect attacks, and are particularly suited to mission critical applications that must provide strong protection to their servers and maintain a minimum level of performance even in times of attacks.

Although there are different concrete approaches to constructing these networks, their high-level structures tend to be similar. Specifically, the structure typically consists of 2 main types of nodes, *access points* and *targets*, among others, and an *interconnection network* between them (see Figure 2 in Section III). The access points serve as the entry points to the network; all clients obtain services via the access points. The targets form a "ring" around the end servers, and are the only nodes that an end server will communicate with. This can be enforced either by physical connectivity or by installing packet filters around the end servers that allow only packets to and from the targets. While the access points are publicly known, the identity of the targets are typically secret. The interconnection network relays communication between the access points and the targets. A service request from a legitimate client is first posted to an access point. Upon proper authentication, it then traverses the interconnection network to reach a target, which presents it to the end servers; the service response takes the reverse path. The interconnection network is the network equivalence of a "minefield." Only the access points and targets know the paths to each other, and and are able to safely navigate through it. Illegitimate traffic without the knowledge of routing, will be spread over the network, thus blunting an otherwise disruptive attack. The interconnection network can be implemented as a regular physical network using the default routing protocols, or as an overlay network using some specialized routing protocols. The latter is more flexible because it can take into account application-level metrics.

Not surprisingly, there has not been much work in understanding the design and engineering of these attack-resistant networks. We believe it is important to be able to guide the construction of these networks and reason about their properties. We refer to this as *security engineering* of networks.[2] In this paper, we propose an abstract model for security engineering based on the aforementioned architecture, and study various algorithms to guide the construction of a survivable network.

To illustrate the key issues in engineering an attack-resistant network, we start with an example. Consider Figure 1 which shows a highly simplified setup with only 2 access points and 2 targets. Typically, there should be a lot more access points than targets, e.g., a 10 to 1 ratio. The clients and the end servers

[1]The exact consequences of a compromise can vary. In the worst case, an attacker can assume complete control of the host. In other cases, it may simply mean an instance of a *zombie* client on the host can be commanded by the attacker to perform specific actions.

[2]Indeed, engineering of many aspects of networks is well studied. An example is traffic engineering, where there are well-established metrics such as bandwidth, delay, and utilization and catalogs of algorithms to guide specific constructions.

Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP'04)
1092-1648/04 $ 20.00 IEEE

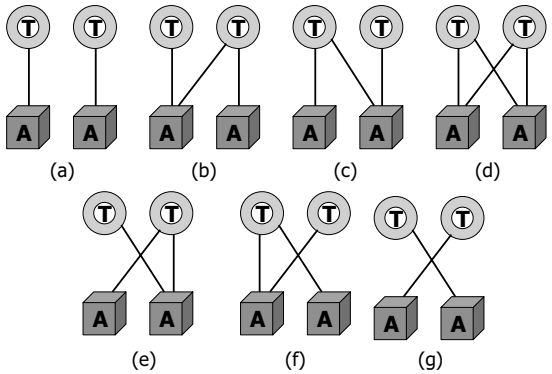

Fig. 1. Examples

are not shown because they are external users of the network, and are not considered in the engineering of the network itself. Obviously, there is the performance-related problem of placing the access points relative to the clients. This is well studied in the literature as the *server placement* problem and is orthogonal to our security interest. The lines connecting the access points and targets model the communication between them through the interconnection network. More precisely, if access point A is connected to target T, then A can forward client traffic via T to the end servers, and vice versa. We emphasize that the lines model actual communication, not the physical connectivity. It could very well be that an access point A can reach every target, but A may choose to use only a subset of them for forwarding to the end servers. In the following, we say a target T is *assigned* to an access point A if there is a line connecting A and T. Figure 1 shows 7 potential constructions for a network with 2 access points and 2 targets. There are actually more possible constructions. For brevity, we omit the ones that at least 1 access point or target does not have any connection. As you will see, in our model, no target should be left unassigned as a construction always improves by assigning an unused target.

The objective of an attack-resistant network is to sustain communication between clients and end servers at all times, especially during times of attacks. Our model of attack is as follows: given that only the access points know how to navigate to the targets, and that the access points are publicly known and accessible, an attacker will try to disrupt communication by first compromising an access point and then launch DoS attacks from there to disable the subset of targets used by the access point. An access point or target is *blocked* if it can no longer service any client requests. In our model, an access point is blocked because (1) it has been compromised by an attacker; (2) it is under a DoS attack; or (3) all targets assigned to it are blocked. A target is blocked because an access point that it is assigned to has been compromised.[3] A less stringent model that uses a conditional probability to reflect the impact of access point compromise on the target is possible. We do not do so for 3 reasons: (1) We believe that security engineering makes most sense using the worst case analysis, which equates to using a value of 1 as the conditional probability; (2) There does not exist a practical way of estimating such a conditional probability; and (3) It is an unnecessary complication for this paper as our formulation and algorithms will work with such a model as well.

There are 2 key aspects to the engineering of an attack-resistant network, namely *resiliency* and *security*. By resiliency, we mean the ability of a network to provide alternative communication paths should one path become disrupted due to failures or attacks. The more such paths exist, the more likely a client can sustain communication with the end servers. In our model, the more targets assigned to an access point, the more resilient the construction is. That is, Figure 1(d) is the most resilient among all constructions in Figure 1.

By security, we mean the ability of a network to contain and limit the impact of compromises. In our model, the compromise of an access point leads to the attack of all targets assigned to it. Therefore, to enhance security, an access point should not have too many targets assigned to it. That is, Figures 1(a),(g) are the most secure among all constructions in Figure 1.

To summarize, resiliency calls for access points to have more cross connections to targets, while security calls for the opposite. This tension between resiliency and security is indeed the heart of the security engineering problem. The overall attack-resistance of a network is modeled using a notion of *blocking probability*[4], which given a particular assignment of targets to access points, provides a measure of the percentage of client requests that can not be relayed to the end servers. The mutual coupling at the access points (one access point having multiple targets assigned to it) and targets (one target being assigned to multiple access points) introduce complex correlation in the calculation of this blocking probability, making the determination of the optimal assignment a non-trivial task. For example, the problem of determining which of the 7 constructions shown in Figure 1 is the "best" overall (smallest blocking probability) is intimately dependent on the values of the different input probabilities. A slight change in these values could easily tip the balance between resiliency and security, and result in a different choice of the optimal assignment.

Main Contributions. The formulation and algorithms in this paper attempt to provide answers to the security engineering problem. Our main contributions are:

- We provide the *first* formulation for a new class of problems targeting the engineering of attack-resistant network. It takes into account both resiliency and security, and lays a rigorous foundation for balancing their opposing tension.
- We introduce a family of heuristics to guide the construction of optimal attack-resistant networks.
- We propose several efficient approximation algorithms for computing the probability that a request is blocked and provide bounds for their errors.
- We evaluate the performance of our heuristics and approximation algorithms using extensive simulation, and demonstrate their efficacy.

[3]To avoid further complexity, we omit consideration of an independent failure probability in this paper. This can be easily accommodated if needed.

[4]The objective of security engineering a network is to allow a client to communicate with any target site, irrespective of the current network state including whether the nodes are compromised or under DoS. In this regard, the blocking probability effectively captures the loss of communication between the client and its desired target during attacks.

The balance of this paper is organized as follows. In Section II, we present related work in this area. As far as we know, the problem we introduce in this paper is a new one, we have not found much prior work addressing similar issues. Instead, we will focus on describing complementary work. In Section III, we first describe at a high-level, the general structure of an attack-resistant network and the attack model we assume, and then provide a rigorous formulation of the optimal assignment problem. Section IV presents several exact and approximation algorithms for computing the overall blocking probabilities and a family of heuristics for solving the optimal assignment problem. The former is a key step for the latter. In Section V, we present results of experimental evaluation of the approximation algorithms and heuristics. Section VI provides discussion on how to obtain the input data for the assignment problem. We conclude in Section VII.

II. Related Work

There has been a lot of recent work, spanning a diverse number of approaches, to counter DoS attacks. Most approaches, however, focus more on specific architectural approaches, mechanisms, and algorithms. Our work in this paper is fundamentally different. We take an abstract view of a network and attempt to provide a holistic approach to the understanding of two critical properties — resiliency and security — in network construction for countering attacks. Our interest is primarily on the *security engineering* of attack-resistant networks, and providing algorithms to guide their constructions. As such, our model and analysis are new, and do not directly compare to any existing work. Instead, we will use the rest of this section to highlight how the different elements in our model relate to work done by others.

Our network model can be viewed as an abstract representation of various concrete network constructions proposed for network resiliency and DoS mitigation. For example, Secure Overlay Services (SOS) [2] was proposed for the goal of protecting a fixed end server from DoS. SOS uses various classes of specialized network nodes to construct its overlay. Its access points are identical to ours, and its *servlets* function essentially like our targets. The SOS architecture creates an infrastructure that proactively protect the end server by surrounding it with multiple fences. SOS is mostly an architectural approach, it does not address the issue of how best to engineer such an network, and in particular, it does not consider the impact of compromised overlay nodes on the overall construction.

Resilient Overlay Networks (RON) [3] proposes the use of overlay networks for recovering from link outages and router failures. Though RON does not directly address DoS attacks, link outages and router failures can be considered as their concrete manifestations. RON attempts to provide resiliency by exploiting the rich connectivity available in an overlay network. Again, RON does not consider the impact of compromised nodes. Indeed, its almost complete connectivity means the compromise of one node can easily propagate to the whole network.

The most relevant work is the analysis by Wang and Chien [5] that proposes an analytical model to determine the impact of node compromises as well as DoS attacks on the resource availability and the secrecy of server locations. The model assumes that attackers have an upper bound on the number of simultaneous nodes that can be compromised as well as a degree constraint that limits the connectivity of an overlay node. They model the occurrence of a compromised node or a node under DoS as Poisson distributed random variables. The primary goal of this work is to analyze the impact of attacks on resource availability; it does not address the issue of constructing attack-resistant networks. This work also does not consider correlated attacks.

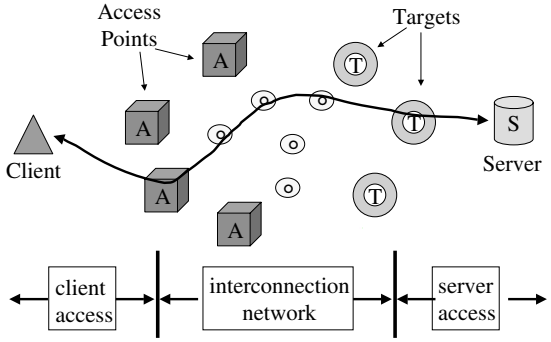

Fig. 2. Elements of an Attack-Resistant Network Architecture

As we can see, most of the prior work focusses on different concerns. Most of them consider only the merit of resiliency (i.e., rich connectivity) for performance and/or reliability gain; none consider the downside of resiliency in terms of correlated blocking. We believe our work is one of the first attempts to investigate this tradeoff of resiliency and security, and making explicit the impact of node compromises.

III. Architecture, Model, and Problem Formulation

In this section, we first present an overview of the different elements in a network architecture representative of an attack-resistant network. We then describe a probabilistic model that attempts to capture the host compromise and DoS attacks that are possible. Based on this model, we formulate the problem of constructing optimal attack-resistant networks.

A. Attack-Resistant Network Architecture

Figure 2 shows the three key elements in an abstract architecture for an attack-resistant network: *Access Points* (APs), *Routers*, and *Targets*. The clients and the servers are technically outside the architecture. The abstract path for communication between a client and a server is as follows:

Client: For a client to obtain a service protected by the architecture, it needs to have been pre-configured with the proper credentials to access the servers. Specifically, it needs to have knowledge of the location of at least one access point, and the required credentials to mutually authenticate itself to the access point. The authentication process results in a *security association* between the client and the access point. This means only legitimate clients will be able to access the servers and they can initiate services only through APs. In general, a client has knowledge of multiple APs, and can forward service

requests to any of them based on some criteria (performance, routing policy, load balancing, etc.).

Access Points (AP): APs are the entry points to the service, they form the "outer" perimeter of the attack-resistant network. Once a client is properly authenticated and granted access to the servers, the APs forward traffic from clients to the servers and vice versa. Ideally, some scheme should be used to distribute clients among the access points for better load distribution. Many such schemes (client-based, server-based, or hybrid) exist in the literature, and we do not discuss them further here. Generally, APs operate relatively independently of each other. If so desired, they provide ideal locations for collaborative network intrusion detection at the expense of extra communication.

Routers: The interconnection network is populated by routers. They are responsible for forwarding traffic between APs and targets according to its routing table. The communication between APs and targets is assumed to be secure. This can be achieved by setting up secure tunnels (e.g., using IPsec) between them. Thus a router cannot distinguish traffic between an AP and a server from background traffic. A router may be a regular network-layer router or an overlay node. A regular network-layer router forwards traffic along the default Internet path as computed by some standard Internet routing protocol. On the other hand, an overlay node may not follow the default Internet path and can provide many alternative paths between an access point and a target.

Targets: Traffic from the APs are handed off to the actual end servers by the targets. In other words, they represent the "inner" perimeter of the attack-resistant network. Typically, packet filters are set around servers such that only traffic from targets are let through. In our architecture, an AP can make use of multiple targets to forward traffic to end servers, and a target can serve multiple APs.

Server: The concept of a server is abstract here. It can be a single entity, which occurs in cases where the service cannot be easily replicated at different nodes (e.g., due to security concerns, dynamically changing content etc.); or it can be a replicated collection of servers distributed geographically. The internal structure of the server network can be complex. In fact, in the extreme case, the servers can themselves be the APs to a next "layer" of attack-resistant network, forming a stack of back-to-back protection perimeters in front of the ultimate end servers.

In the following, we denote the set of APs and the set of targets by $\mathcal{A}$ and $\mathcal{T}$ respectively, and we specify the assignment relationship between the APs and the targets using an $|\mathcal{A}|\times|\mathcal{T}|$ matrix, called an *assignment* matrix, $\mathcal{M}$. Specifically, $\mathcal{M}(i,j)=1$ if AP i is configured to forward requests to end servers using target j.

B. Attack Model

The primary goal of an attacker is to prevent communication between the clients and the servers. Attackers have the ability to launch two types of attacks: *Compromise* and *DoS* attacks. In the first type of attack, nodes can be compromised such that an attacker has complete access to all information stored at the node. Let C_i and $\overline{C_i}$ denote respectively the events that node i is compromised or not compromised, and $P(C_i)$ the probability that node i is compromised.

The access points are usually deployed in different Internet domains in order to improve the resilience by reducing the probability of correlated failure. Different Internet domains may enforce different levels of defenses. For instance, some access points may be behind firewalls whereas others may be completely exposed to the Internet. Moreover, access points do not know the identities of each other. Therefore, the compromise of one access point does not necessarily affect the probabilities of compromising other access points. In order to simplify the model, we therefore assume that the compromise probabilities of different APs are independent. There are several approaches to obtain the compromising probabilities and the DoS attack probabilities that are used as inputs to our model. For instance, one can derive them from historical activities or perform security analysis on softwares being used. We will provide more details in Section VI.

Once a node is compromised, an attacker can launch DoS attacks to any other node in the network using the compromised node. Let D_i and $\overline{D_i}$ denote respectively the events that node i is under DoS attacks or not under DoS attacks, and $P(D_i)$ the probability that node i is under DoS attacks. Using the same argument as we described above for the compromising probabilities, we assume that the D_i's of different APs are independent, and furthermore, that D_i's and C_i's are independent. We believe this is a reasonable assumption since the compromising of nodes and the launching of DoS attacks are usually performed using different sets of techniques.

If AP i is compromised, an attacker has a non-zero probability $P(D_j|C_i)$ to obtain the identities of all targets j where $\mathcal{M}(i,j)=1$ and launch DoS to these targets. In this paper, we consider only the worst case attack scenario (for obvious security reasons) where $P(D_j|C_i)=1$, $\forall j\in\mathcal{T}$ and $\mathcal{M}(i,j)=1$. We note, however, that our approaches can be easily generalized to handle any values of $P(D_j|C_i)$.

The best adversarial strategy to bring down a network is to focus attacks on the servers, targets, and APs used by the service. The set of APs is publicly known whereas the set of targets/servers are only known within the attack-resistant network. Therefore, the probability that an AP is attacked is substantially higher than the probability that a target independently comes under attacks, not due to AP compromises. Thus, it is reasonable to assume $P(C_j)=0$, $\forall j\in\mathcal{T}$ and $P(D_j|\overline{C_{i_1}},\overline{C_{i_2}},\ldots,\overline{C_{i_k}},\ldots,\overline{C_{i_n}})=0$, $\forall i_k\in\mathcal{A}$ and $\mathcal{M}(i_k,j)=1$. In other words, an attacker is unable to directly compromise a target node. Also, an attacker will not be able to launch a DoS attack on a target, if it does not compromise at least one AP that connects to that target.

A request arriving at an AP is blocked if either the access point is under attack or all targets that it is configured to use to reach servers are under DoS due to the compromising of other APs. Let A_i denote the event that a request arrives at AP i and $P(A_i)$ is the probability of A_i. We now formulate the optimal assignment problem.

C. Optimal Assignment

Given an assignment matrix $\mathcal{M}$, the APs compromise probabilities $P(C_i)$, $\forall i \in \mathcal{A}$, and the probabilities $P(D_i)$, $\forall i \in \mathcal{A}$, that APs are under DoS attack, the probability that a request is blocked can be computed as follows. Let R denote the event that a request is blocked and $P_{\mathcal{M}}(R)$ the blocking probability under assignment $\mathcal{M}$. Let B_i denote the event that a request arriving at AP i is blocked and $P_{\mathcal{M}}(B_i)$ the blocking probability under assignment $\mathcal{M}$. We have

$$P_{\mathcal{M}}(R) = \sum_{i \in \mathcal{A}} (P_{\mathcal{M}}(B_i) \times P(A_i)). \quad (1)$$

Recall that a request arriving at AP i can be blocked due to the following three reasons: (a) i is compromised; (b) i is under DoS attack; or (c) All targets assigned to i are under DoS attack. Thus,

$$\begin{aligned} P_{\mathcal{M}}(B_i) &= P(D_i) + (1 - P(D_i)) \times (P(C_i) + (1 - P(C_i)) \\ &\quad \times P(D_{j_1}, \ldots, D_{j_k}, \ldots, D_{j_n} | \bar{C}_i)) \\ &\quad \forall j_k \in \mathcal{T}, \text{ such that } \mathcal{M}(i, j_k) = 1 \end{aligned} \quad (2)$$

Our goal is to come up with an assignment matrix such that the blocking probability of a request is minimized in spite of the existence of compromised APs. There is an inherent tradeoff between $P_{\mathcal{M}}(R)$ and $P_{\mathcal{M}}(B_i)$, $\forall i \in \mathcal{A}$. Specifically, the more targets assigned to an AP i, the lower the value of $P_{\mathcal{M}}(B_i)$, since a request to an AP is only blocked when every target assigned to that AP is under attack or the AP itself is under attack. However, from a global perspective, compromising an AP potentially brings down all targets assigned to it, thus negating the benefit of a high fan-out AP.

Let $\Psi(\mathcal{A}, \mathcal{T})$ denote the set of all possible assignment of targets to APs. It is obvious that $|\Psi(\mathcal{A}, \mathcal{T})| = 2^{|\mathcal{A}| \times |\mathcal{T}|}$. *The problem of optimal assignment is to find a matrix $\mathcal{M}$ such that $P_{\mathcal{M}}(R)$ is minimized.* In other words, our objective function can be expressed as:

$$\min_{\forall \mathcal{M} \in \Psi(\mathcal{A}, \mathcal{T})} P_{\mathcal{M}}(R) \quad (3)$$

IV. Algorithm

In this section, we present algorithms for the problems formulated in Section III, namely, computing the blocking probability as defined in (2), and determining the optimal assignment of targets to APs as specified in the objective function in (3).

For the former problem, we first describe two algorithms that can compute the exact blocking probabilities, followed by two approximation algorithms that are significantly more efficient, but can only provide approximate values. For the latter problem, we first present a brute-force optimal assignment algorithm and four efficient heuristics that yield close to optimal results.

A. Computing the Blocking Probability

The input to the blocking probability computation is a tuple $(P(C_i)_{\forall i \in \mathcal{A}}, P(D_i)_{\forall i \in \mathcal{A}}, \mathcal{M})$, i.e., the probabilities that each AP is compromised, the probabilities that each AP is under DoS attack, and the assignment matrix.

Enumerate ($\mathcal{M}$)
  $bp = 0$; $V = \{S_N, S_C, S_D, S_A\}^{|\mathcal{A}|}$;
  forall $v \in V$ **do**
    compute the probability of occurrence of v, $P(v)$;
    forall $i \in \mathcal{A}$ **do**
      if (i is blocked) $bp = bp + P(v) \times P(A_i)$;
  return bp;

Fig. 3. Enumeration Algorithm

We first present two algorithms that compute the exact value of blocking probability. These are computationally expensive, and are useful mostly for reference purpose. We then describe two approximation algorithms that run efficiently and yield near optimal results. Moreover, we show that the algorithms are bounded and prove their convergence.

A.1 Exact Algorithms

Enumeration: As mentioned above, there are a set of four possible states for an AP — normal, compromised, under DoS attacks, compromised and under DoS attack. We denote these states by S_N, S_C, S_D, and S_A respectively and let $S = \{S_N, S_C, S_D, S_A\}$. The enumeration algorithm does not try to compute the individual blocking probability of each AP separately. Instead, it enumerates all possible combinations of each AP being in one of the four states, and sums the product of the probability of the occurrence of each combination and the probability a call arrives at a blocked AP.

Given a combination $v = (S_1, \ldots, S_{|\mathcal{A}|})$. The probability of the occurrence of v, $P(v)$, can be computed as $\prod_i P(\text{AP } i \text{ in state } S_i)$. The inner term simply expands as follows, for an AP i: $P(i \text{ in state } S_N) = P(\overline{C_i}, \overline{D_i})$, $P(i \text{ in state } S_C) = P(C_i, \overline{D_i})$, $P(i \text{ in state } S_D) = P(\overline{C_i}, D_i)$, and $P(i \text{ in state } S_A) = P(C_i, D_i)$.

The details of the algorithm is presented in Figure 3. The V as defined represents the set of all combinations of the states of all AP. Since an AP could be in one of the four states, $|V| = 4^{|\mathcal{A}|}$. Thus the complexity of the algorithm is $O(4^{|\mathcal{A}|})$.

Expansion: Instead of computing the blocking probability directly, the expansion algorithm computes its inverse, or the *success probability* of an AP. Essentially, this is the probability that a request arriving at an AP can be forwarded to a target. This requires that an AP not be compromised, and that among all targets assigned to it, at least one can communicate with the AP. We can express this as:

$$\begin{aligned} P(\overline{B_i}) &= P(\overline{D_i}) \times P(\overline{C_i}) \\ &\quad \times P(\overline{D_{j_1}} \cup \ldots \overline{D_{j_k}} \cup \ldots \cup \overline{D_{j_n}} | \overline{C_i}) \\ &\quad \text{where } \mathcal{M}(i, j_k) = 1 \end{aligned} \quad (4)$$

Let us denote $P(\overline{D_{j_1}} \cup \ldots \cup \overline{D_{j_k}} \cup \ldots \cup \overline{D_{j_n}} | \overline{C_i})$ by $P_i(\overline{D_{j_1}} \cup \ldots \cup \overline{D_{j_k}} \cup \ldots \cup \overline{D_{j_n}})$, which can then be expanded

```
MonteCarlo (M)
  n = 0; m = 0;
  while m/n not converged do
    forall i ∈ A do
      randomly assign a state to AP i according to
      the probability of each state;
    randomly pick an access point i according to P(A_i);
    if (AP i is blocked) m = m + 1;
    n = n + 1;
  return m/n;
```

Fig. 4. Monte Carlo Algorithm

as,

$$\begin{aligned} & P_i(\overline{D_{j_1}} \cup \ldots \cup \overline{D_{j_k}} \cup \ldots \cup \overline{D_{j_n}}) \\ = & \sum_{\ell \in J} P_i(\overline{D_\ell}) \\ + & (-1)^1 \sum_{\ell_1 \neq \ell_2 \in J} P_i(\overline{D_{\ell_1}}, \overline{D_{\ell_2}}) + \ \ldots\ldots \\ + & (-1)^{k-1} \sum_{\ell_1 \neq \ell_2 \ldots \neq \ell_k \in J} P_i(\overline{D_{\ell_1}}, \ldots, \overline{D_{\ell_k}}) + \ \ldots\ldots \\ + & (-1)^{n-1} \sum_{\ell_1 \neq \ell_2 \ldots \neq \ell_n \in J} P_i(\overline{D_{\ell_1}}, \ldots, \overline{D_{\ell_n}}) \end{aligned} \tag{5}$$

where $J = \{j_1, j_2, \ldots, j_k, \ldots, j_n\}$.

The joint probability that a set of k targets are not under DoS attack given that AP i is under attack is simply the joint probability that all APs associated with the k targets except AP i are not compromised. This can be expressed as the product of the probabilities of not being compromised for all APs excluding i that is assigned to any of the k targets since the compromise probabilities are independent of each other. That is,

$$P_i(\overline{D_{\ell_1}}, \ldots, \overline{D_{\ell_k}}) \quad = \quad \prod_{\substack{m \in \mathcal{A} \ \wedge \ m \neq i \ \wedge \\ \exists x : \mathcal{M}(m, \ell_x) = 1}} P(\overline{C_m})$$

The number of terms in (5) is $2^{|\mathcal{T}|}$, and the worst case complexity of computing each term is $O(|\mathcal{A}| \times |\mathcal{T}|)$. This algorithm is substantially more efficient than the *Enumeration* algorithm which has a complexity of $O(4^{|\mathcal{A}|})$ since typically, $|\mathcal{A}| >> |\mathcal{T}|$.

A.2 Approximation Algorithms

Monte Carlo: Instead of explicitly computing the blocking probability, the *Monte Carlo* [6] method attempts to determine the probability by "sampling" the actual system. The algorithm is shown in Figure 4. The two variables, n and m, count respectively the total number of iterations and the number of iterations that a request would have been blocked because of a blocked AP. The sampling is done as follows: at each iteration, each AP is first randomly assigned to one of the four states according to the probability of the respective states. Then an AP is randomly picked among all APs according to probability $P(A_i)$, $\forall i \in \mathcal{A}$; essentially "simulating" the arrival of a client request at an AP. If the selected AP is blocked, the client request will also be blocked, thus we increment m by one. The algorithm iterates until the estimate of blocking probability, m/n converges. The following two theorems state that the procedure not only always converges to the real blocking probability value, but also has a bounded error for a given number of iterations.

Theorem 4.1: $\forall \epsilon > 0$,

$$\lim_{n \to \infty} P(|\frac{m}{n} - P_{\mathcal{M}}(R)| < \epsilon) = 1$$

This theorem suggests that the estimate approaches the real value of blocking probability with probability 1 as the number of iterations goes to infinity. The theorem can be proved by applying Bernoulli's theorem [7]. Please refer to [8] for detailed proof.

Theorem 4.2: $\forall \epsilon > 0$,

$$P(|\frac{m}{n} - P_{\mathcal{M}}(R)| \leq \epsilon) > 1 - \frac{1}{4n\epsilon^2}$$

For a pre-determined error, the upper bound on the number of iterations can be estimated using Theorem 4.2. It is important to observe that the number of iterations is independent of $|\mathcal{A}|$ and $|\mathcal{T}|$. This means the Monte Carlo approach is scalable even for large problem size. This upper bound is derived from Chebyshev's inequality. Interested readers may refer to [8] for details.

Truncated Expansion: In the aforementioned *Expansion* algorithm, the number of terms is $2^{|\mathcal{T}|}$ which is exponential. These terms, however, do not contribute equally to the final result. Specifically, as the number of targets in a term increases, the joint probability becomes smaller and smaller, and thus contributes less and less to the overall result.

Thus, a reasonable approximation approach is to truncate the expansion process to include only the initial significant terms. According to Boole-Bonferroni inequality [9], we have the upper bound,

$$\begin{aligned} & P_i(\overline{D_{j_1}} \cup \ldots \cup \overline{D_{j_k}} \cup \ldots \cup \overline{D_{j_n}}) \\ \leq & \sum_{m=1}^{k} (-1)^{m+1} \sum_{\ell_1 \neq \ell_2 \ldots \neq \ell_m \in J} P_i(\overline{D_{\ell_1}}, \ldots, \overline{D_{\ell_m}}) \\ & \text{where } k \text{ is odd and } \quad J = \{j_1, j_2, \ldots, j_k, \ldots, j_n\} \end{aligned} \tag{6}$$

and the lower bound of the expansion,

$$\begin{aligned} & P_i(\overline{D_{j_1}} \cup \ldots \cup \overline{D_{j_k}} \cup \ldots \cup \overline{D_{j_n}}) \\ \geq & \sum_{m=1}^{k} (-1)^{m+1} \sum_{\ell_1 \neq \ell_2 \ldots \neq \ell_m \in J} P_i(\overline{D_{\ell_1}}, \ldots, \overline{D_{\ell_m}}) \\ & \text{where } k \text{ is even and } \quad J = \{j_1, j_2, \ldots, j_k, \ldots, j_n\} \end{aligned} \tag{7}$$

Therefore, we can expand the joint probabilities to terms involving i targets where either the sum of terms involving i targets is less than a pre-determined error bound or i reaches some pre-determined bound k. We define this approach as *k-expansion*. By bounding k to a constant number, this provides a polynomial time approximation algorithm.

```
Find-Link-to-Add ()
  min_bp = 1;
  forall i ∈ A, j ∈ T do
    if (M(i, j) ≠ 1)
      M(i, j) = 1; bp = compute_blocking_probability (M);
      if (bp < min_bp) min_bp = bp; min_ap = i; min_tgt = j;
      M(i, j) = 0;
  return ((min_ap, min_tgt), min_bp); }

Grow ()
  forall i ∈ A, j ∈ T: M(i, j) = 0;
  min_bp = 1;
  while (true) do
    ((i, j), bp) = Find-Link-to-Add ();
    if (bp < min_bp) M(i, j) = 1; min_bp = bp;
    else break;
```

Fig. 5. Greedy Grow Algorithm

```
Find-Link-to-Delete ()
  min_bp = 1;
  forall i ∈ A, j ∈ T do
    if (M(i, j) ≠ 0)
      M(i, j) = 0; bp = compute_blocking_probability (M);
      if (bp < min_bp) min_bp = bp; min_ap = i; min_tgt = j;
      M(i, j) = 1;
  return ((min_ap, min_tgt), min_bp);

Shrink ()
  forall i ∈ A, j ∈ T: M(i, j) = 1;
  min_bp = 1;
  while (true) do
    ((i, j), bp) = Find-Link-to-Delete ();
    if (bp < min_bp) M(i, j) = 0; min_bp = bp;
    else break;
```

Fig. 6. Greedy Shrink Algorithm

B. Assignment Algorithms

The brute-force approach to finding the optimal assignment is by enumerating all possible assignments and picking the one with the minimum blocking probability. The complexity of this algorithm is, not unexpectedly, $O(2^{|\mathcal{A}|\times|\mathcal{T}|})$. Thus instead, we study four greedy heuristics that are significantly more efficient. As we will see, all four heuristics attempt to lower the blocking probability by making a small change (addition or removal of assignments) to the current assignment matrix. The algorithms continue as long as the blocking probability reduces; they are called *greedy* because they prefer changes that give the best instantaneous reduction in each iteration.

Greedy Grow (Figure 5): This algorithm begins with no targets assigned to any AP. In each iteration, the algorithm attempts to add a single new assignment. It does so by trying all possible candidates for this new assignment, i.e., all APs and targets that are not "connected" yet, and selects the pair that gives the lowest overall blocking probability. The algorithm continues until there is no further reduction in blocking probability by making an additional assignment. The running complexity of each iteration is $O(|\mathcal{A}| \times |\mathcal{T}|)$.

Greedy Shrink (Figure 6): This algorithm is the exact dual of the previous *Grow* algorithm. It starts with a full assignment where every target is assigned to all APs. In each iteration, a single assignment is deleted by going through all current assignments. The assignment that provides the most reduction in blocking probability is selected (and deleted) and the procedure

```
Find-Links-to-Rewire ()
  min_bp = 1;
  forall i ∈ A, j ∈ T do
    forall u ∈ A, v ∈ T do
      if (M(i, j) = 1 ∧ M(u, v) = 0) // rewire a link
          M(i, j) = 0; M(u, v) = 1;
          bp = compute_blocking_probability (M);
          if (bp < min_bp)
              min_bp = bp; min_ap_0 = i; min_tgt_0 = j;
              min_ap_1 = u; min_tgt_1 = v;
          M(i, j) = 1; M(u, v) = 0;
  return ((min_ap_0, min_tgt_0), (min_ap_1, min_tgt_1), min_bp);

Rewire ()
  forall i ∈ A, j ∈ T: M(i, j) = 0;
  min_bp = 1;
  while (true) do
    ((i_a, j_a), bp_a)      = Find-Link-to-Add ();
    ((i_r, j_r), bp_r)      = Find-Link-to-Delete ();
    ((i, j), (u, v), bp_w)  = Find-Links-to-Rewire ();
    bp = min (bp_a, bp_r, bp_w);
    if (bp < min_bp)
      if (bp = bp_a) M(i_a, j_a) = 1;
      elseif (bp = bp_r) M(i_r, j_r) = 0;
      else M(i, j) = 0; M(u, v) = 1;
      min_bp = bp;
    else break;
```

Fig. 7. Greedy Rewire Algorithm

```
Random (p, q)
  forall i ∈ A, j ∈ T: M(i, j) = 0;
  min_bp = 1; change = 0; retries = |A| × |T|;
  while (true) do
    r = random (0, 1);
    if (r < p)
      ((i_a, j_a), bp_a) = Find-Link-to-Add ();
      if (bp_a < min_bp) M(i_a, j_a) = 1; change = 1;
    elseif (r < p + q)
      ((i_r, j_r), bp_r) = Find-Link-to-Delete ();
      if (bp_r < min_bp) M(i_r, j_r) = 0; change = 1;
    else
      ((i, j), (u, v), bp_w) = Find-Links-to-Rewire ();
      if (bp_r < min_bp) M(i, j) = 0; M(u, v) = 1; change = 1;
    if (change = 0) ∧ (retries = 0) return;
    elseif (unchange = 0) retries = retries − 1 ;
    else retries = |A| × |T|;
```

Fig. 8. Randomized Greedy Algorithm

is repeated as long as there exists such an assignment.

Greedy Rewire (Figure 7): Besides adding (as in *Grow*) or removing (as in *Shrink*) assignments, the blocking probability may also be lowered by switching assignment without changing the total number of assignments. **Find-Links-to-Rewire** attempts to "rewire" an existing assignment by simultaneously removing it from an AP-target pair and assigning another AP-target pair. Since "rewiring" work only with some populated set of assignments, we pair it up with *Grow* and *Shrink* to produce a feasible set of assignments. Specifically, in each iteration of *Rewire*, the heuristic attempts three possible operations — adding; removing; and rewiring a link, and commits the one that provides the best reduction in the blocking probability. The complexity of each *Rewire* iteration is $O(|\mathcal{A}|^2 \times |\mathcal{T}|^2)$.

Randomized Greedy (Figure 8): Because *Rewire* tries all three possible operations in each iteration, it is computationally expensive. To retain the benefits of *Rewire* while avoiding its cost, we introduce *Randomized Greedy*, which is a randomized

version of *Rewire*. In *Randomized Greedy*, instead of deterministically performing the add, delete, and rewire operations in every iteration, the heuristic randomly selects one of the three operations to perform. As we will see in Section V, *Randomized Greedy* performs as well as the other algorithms.

A problem with all the above heuristics is that there is a possibility of early termination if the algorithms hit a local optima by chance. This problem is a direct result of the greedy selection scheme in each iteration. We mitigate this problem by maintaining a sorted buffer of k best assignments, and return a particular assignment from this buffer with the probability proportional to the quality of that assignment. In other words, we do not always return the best assignment. It should be noted that the number of terms in this buffer is independent of the problem size (number of APs and targets), hence maintaining this set does not raise any issues of scalability.

V. Evaluation

In this section, we describe the results of a comprehensive evaluation of the different assignment heuristics, as well as the accuracy of the different approximation mechanisms to compute the blocking probability. We evaluate the algorithms using systems with high and low vulnerabilities. By vulnerability, we refer to the probability that an AP is attacked (either compromised or under DoS attacks). For each AP, we assign the compromise probability and the probability under DoS attack according to uniform distributions with two different means. We use mean 0.04 to model a highly vulnerable system and mean 0.01 for a system with low vulnerability. We vary the size of problems by using different number of APs and targets. For a particular set of APs and vulnerability, We randomly generate 30 problems using different seeds. It should be noted that in the subsequent results, the absolute values of the parameters are not as important as the relative performance of the approximation algorithms to compute the blocking probability and the assignment heuristics. Also, our results are independent of any network topology, since the primary input parameters to our model are the number of APs and targets. However, the output assignment from our model will dictate the connectivity of a network that desires security engineering.

Recall that there are four different greedy heuristics that assign targets to APs: Greedy *Grow*, Greedy *Shrink*, Greedy *Rewire*, and *R-Greedy* (Randomized Greedy). For *R-Greedy*, we use 0.7, 0.1, 0.2 as the probability of adding, deleting and rewiring a link in the simulation. In addition, we have the optimal assignment algorithm (through exhaustive enumeration). *Blocking probability* as defined in Equation (1) is the key metric to determine the quality of the assignments made by the heuristics.

We first evaluate the heuristics for a set of small assignment problems where the number of targets is 3 and the number of APs is 10. This allows us to use the optimal assignment[5] as a baseline for comparison to the different assignment heuristics. We then evaluate the assignment heuristics for larger size problems of 10 targets and 100 APs where the different heuristics are compared among themselves. Moreover, for each heuristic, we evaluate how different approximation methods for computing the blocking probability may have an impact on the quality of the heuristic. Specifically, each of these assignment heuristics may use any one of following methods to compute the blocking probability: (a) The optimal exhaustive *Enumeration*, (b) optimal full *Expansion*; (c) *k-expansion* approximation; and (d) *Monte Carlo* approximation. Since the the complexity of exhaustive *Enumeration* is substantially higher than full *Expansion*, we use the latter as the baseline method to compute the exact blocking probability for the larger size problems.

Small Assignment Problems: The goal here is to evaluate each of the assignment heuristics and compare them to the optimal assignment. We show the results using a *Boxplot*. For each randomly generated problem, we first normalize the blocking probability obtained from each heuristic by the blocking probability of optimal assignment. The boxplot produces a box and whisker plot for the normalized blocking probability of each heuristic. Recall that there are 30 randomly generated problems with different seeds for each heuristic. The box has lines at the lower quartile, median, and upper quartile of 30 values. The whiskers are lines extending from each end of the box to show the extent of the rest of the data. Outliers are data with values beyond the ends of the whiskers. A heuristic is considered to be good if (a) the mean of the normalized blocking is close to a value of 1, corresponding to the optimal value; and (b) the lower quartile and upper quartile are close. The heuristic satisfying the first condition has a good average performance, while the heuristic satisfying the second condition has low variance.

We observe from Figures 9-10 that all the heuristics perform near optimal for both low and high vulnerabilities, with *rewire* and *R-Greedy* performing slightly better than the others. Specifically, the three heuristics (*Grow, Rewire, R-Greedy*) except *Shrink* have little error for low vulnerabilities with a mean blocking within 2-3% of optimal, and exhibit small variance from the mean. *Shrink* performs slightly worse (within 6% of optimal) with larger variance. Since the overall blocking probability is under 4% for this vulnerability, the error is not really significant. As the vulnerability increases, the variance of the error decreases significantly for all heuristics. In addition, all the heuristics including *Shrink* perform within 3% of the optimal method.

It should be noted that the performance of the heuristics is closer to optimal, for high vulnerability, compared to the results for low vulnerability. This is because the blocking probability is higher such that a difference of a few assignments does not have as much impact as under low vulnerability which deals with smaller blocking probabilities. Thus, slight inaccuracies when using the heuristics do not severely impact the performance.

Large Assignment Problems: Our next set of experiments is performed on the problem set with 10 targets and 100 APs. Unfortunately, the optimal assignment algorithm cannot handle a problem set of this size. Instead, we normalize the blocking

[5]Exhaustive enumeration is computationally intractable for larger size problems due to its exponential complexity.

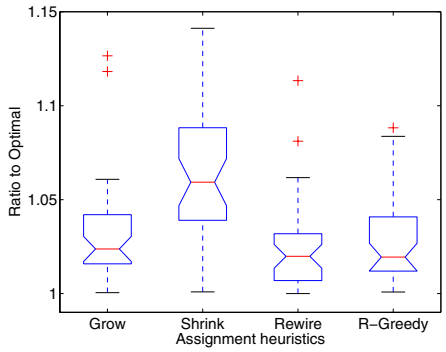

Fig. 9. Small Problem Set, Low Vulnerability

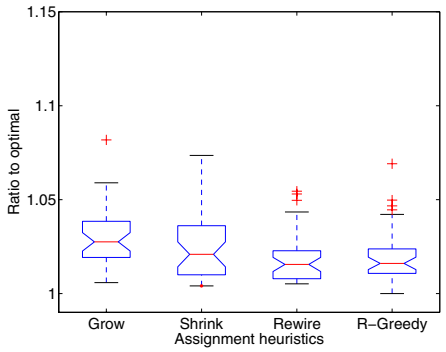

Fig. 10. Small Problem Set, High Vulnerability

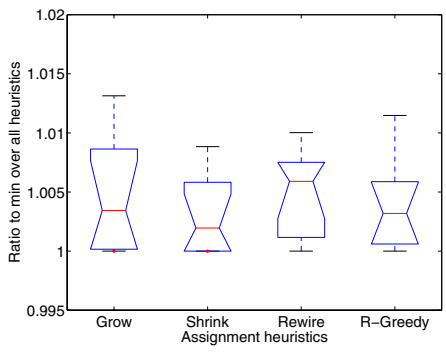

Fig. 11. Large Problem Set, Low Vulnerability

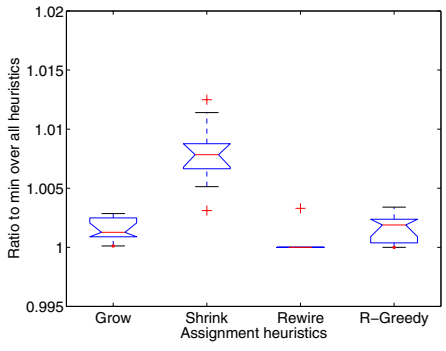

Fig. 12. Large Problem Set, High Vulnerability

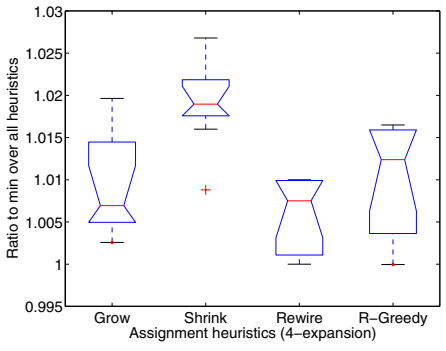

Fig. 13. 4-expansion, Low Vulnerability

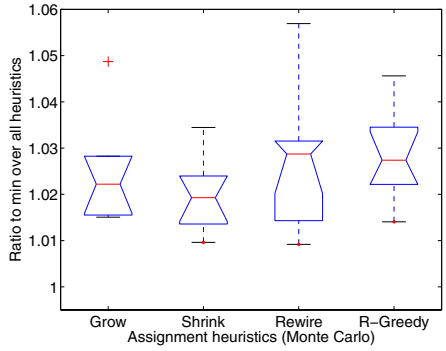

Fig. 14. Monte Carlo, Low Vulnerability

probability yielded by each heuristic to the minimum blocking probability over all the heuristics for each randomly generated problem. Note that minimum blocking for different problems may be achieved by different heuristics. Besides comparing the performance of different heuristics, we also run heuristics using different approximation algorithms to compute the blocking probability including the *truncated expansion* and the *Monte Carlo* methods. We present the results using only boxplots, since these provide a more detailed view as we have seen earlier.

The boxplot in Figure 11 shows the blocking probabilities yielded by all four heuristics for low vulnerability where the full *Expansion* is used to compute the blocking probability. As before, the difference between different algorithms is relatively small (less than 1%).

Figure 12 shows the boxplot for high vulnerabilities. All three heuristics perform very close except *shrink* which yields blocking probability about half percent higher than other heuristics. Again, all four heuristics differ less than 1%.

In order to evaluate how the *k-expansion* approximation impacts the performance of heuristics, we show the performance for *4-Expansion* in Figure 13 for low vulnerability. For each problem, we normalized the blocking probabilities obtained when using *4-Expansion* by the minimum blocking probability among all the heuristics using full *Expansion*. The normalized blocking probability is only slightly higher (less than 0.5%) than that in Figures 11 for all heuristics except *shrink*, which is about 1.5% higher. This is due to the fact that *shrink* starts with the full mesh connectivity. There is a strong correlation between different targets because all targets are assigned to all APs to begin with. The use of a *4-Expansion* results in inaccurate computation of the blocking probability.

Figure 14 shows the performance of the assignment heuristics when using the *Monte Carlo* approximation. As compared to Figure 11, The normalized blocking in Figure 14 is about 2.5% to 3% higher than that in Figure 11. Although the *Monte Carlo* approach is slightly worse than *k-expansion*, it is more scalable since the number of iterations does not depends on the size of the problem. We repeat the same set of experiments for high vulnerability and the observations are similar.

The Structure of Assignments

Recall that the primary goal of this paper is to find the best possible assignment between access points and targets for a given set of compromise and DoS attack probabilities. In this regard, we examined the earlier experiments for an insight into the *structure* of the assignment that is output from the algorithms. In particular, the impact of the probability parameters on the *degrees* of the targets and APs. We observed certain consistent patterns in the structure of assignments from our experiments. We summarize our observations as follows.

1) When all APs have the same compromise probability, our experiments suggest that all APs are connected to an equal number of targets and all targets are likewise connected to an equal number of APs, forming a complete bipartite graph. This is easy to see due to the symmetric nature of the input.
2) When APs have different compromise probabilities, we find that the degrees of APs with high compromise probability are always less than or equal to the degrees

of APs with lower compromise probabilities, i.e. the more vulnerable access points are always assigned to fewer targets. This is intuitive since compromising an AP results in jeopardizing all the targets that are assigned to it. Thus, compromising a high degree AP can severely affect the network.

3) We also observe that by removing APs with significantly higher compromise probabilities (compared to the other APs) from the network, the blocking probability is reduced substantially. An AP that is relatively easy to compromise can lead to attacks on any target(s) that are assigned to it. This would increase the blocking probabilities at other APs that are assigned to these targets.

Summary: We presented a comprehensive evaluation of different assignment heuristics along with multiple methods to approximate the computation of the blocking probability. The results suggest that the differences between the proposed assignment heuristics are small for a wide range of problems with differing sizes and vulnerabilities. Moreover, all of the heuristics perform close to optimal with *Rewire* and *R-Greedy* performing the best. We also see that heuristics using approximation algorithms such as *k-Expansion* and *Monte Carlo* for computing blocking probability perform very close to the heuristics using the exact algorithm. Interestingly, we also observed that a small value of k suffices to provide performance close to the exact algorithm. We also characterized the structure of the assignment algorithms and found several consistent patterns that provide an insight into the assignment process.

VI. Determining the Input Probabilities

Our model uses as input the compromise probabilities and the DoS attack probabilities. A straightforward way to determine these is to derive them from historical activities. In particular, analysis of event or alarm traces from the past can provide a rough estimate of these probabilities. For example, *"honeypots"* [10] can be installed in the network where the access points located in order to provide the first-order approximation of the actual probabilities. A honeypot is a resource that has no production value and there is no legitimate reason for anyone to interact with a honeypot. Thus, any attempt to communicate with the system is most likely a probe, scan, or attack. Analyzing these attempts can yield an estimate of the attacking probabilities.

A more scientific approach to determining these is via explicit analysis. Specifically, both probabilities are highly dependent on the defense mechanisms that have been put in place in the APs. For compromises, the ability of an attacker to penetrate a host is mostly determined by the number of vulnerabilities existing on the host, which in turn depends on the quality of the software and how up-to-date the maintenance (e.g., applying patches) has been. There is a body of literature on software vulnerability modeling [11], etc. that can be applied.

For DoS attacks, the success of an attack is determined by a function of the network defense mechanisms that are in place. For example, "stateful" firewall and intrusion detection system can reduce the probability of a successful DoS attack. The effectiveness of firewalls are determined by its vulnerabilities to attacks and an analysis of the vulnerabilities is presented in [12], [13]. This manner of vulnerability analysis can be used to derive attacking probabilities.

Finally, it should also be noted that our model and algorithms are useful for what-if analyses, or so called scenario planning exercises. In these applications, the absolute value of the input probabilities are not as important. Rather, the key is to study how different distributions of input will impact the final results.

VII. Conclusions

Similar to traffic engineering that aims to improve network utilization and performance, security engineering aims to improve the attack resistance of a network. In this paper, we presented the first formulation of a new class of problems focusing on the engineering of attack-resistant networks that takes into account both resiliency and security, and lays a rigorous foundation for balancing their opposing tension. We then introduced a family of heuristics to guide the construction of optimal attack-resistant network that minimizes the blocking probability. These heuristics make use of several efficient approximation algorithms for computing blocking probability with provable error bounds. We evaluated the performance of our heuristics and approximation algorithms using extensive simulation, and demonstrated their efficacy. Finally, we provided a detailed insight into the *structure* of the assignment algorithms.

For future work, we are working on two issues. First, we are investigating the theoretical bound for the optimal assignment. Second, we are enhancing the model to account for the case where the membership of the access points are dynamic.

References

[1] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which exploy IP source address spoofing," *RFC 2827*, 2000.

[2] A. Keromytis, V. Misra, and D. Rubenstein, "SOS: Secure Overlay Services," *Proc. of ACM SIGCOMM*, August 2002.

[3] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient Overlay Networks," *Proc. of 18th ACM SOSP*, October 2001.

[4] R. Stone, "CenterTrack: An IP overlay network for tracking DoS floods," *Proc. of USENIX Security Symposium*, 2000.

[5] J. Wang and A. A. Chien, "Using overlay networks to resist denial-of-service attacks," *Submitted to ACM Conf. on Computer and Comm. Security*, October 2003.

[6] I. M. Sobol, A. Sobol, and I. M. Sobol, *A Primer for the Monte Carlo Method*. CRC Press, 1994.

[7] W. Feller, *An Introduction to Probability Theory and Its Applications*. Wiley, New York, 1971, vol. 2.

[8] T. Bu, S. Norden, and T. Woo, "Trading Resiliency for Security: Model and Algorithms ," *Bell labs technical memo*.

[9] J. Galambos and I. Simonelli, *Bonferroni-Type Inequalities with Applications*. Springer-Verlag, New York, 1996.

[10] N. Weiler, "Honeypots for Distributed Denial of Service attacks," *Proc. of 11th IEEE WETICE*, June 2002.

[11] W. Du and A. Mathur, "Testing for Software Vulnerability Using Environment Perturbation ," *Proceedings of the International Conference on Dependable Systems and Networks*, 2000.

[12] M. Frantzen, F. Kerschbaum, E. Schultz, and S. Fahmy, "A framework for understanding vulnerabilities in firewalls using a dataflow model of firewall internals," *Journal of Computers and Security*, vol. 20, no. 3, pp. 263–270, May 2001.

[13] S. Kamara, S. Fahmy, E. Schultz, F. Kerschbaum, and M. Frantzen, "Analysis of Vulnerabilities in Internet Firewalls," *Journal of Computers and Security*, vol. 22, no. 3, pp. 214–232, April 2003.