Effectiveness of IP Address Randomization in Decoy-Based Moving Target Defense

Andrew Clark, Kun Sun, and Radha Poovendran

***Abstract*— In a decoy-based moving target defense (MTD), a computer network introduces a large number of virtual decoy nodes in order to prevent the adversary from locating and targeting real nodes. Since the decoys can eventually be identified and their Internet Protocol (IP) addresses blacklisted by the adversary, current MTD approaches suggest that the IP addresses of the real and decoy nodes should be randomly refreshed and reassigned over time. Refreshing and reassigning the IP addresses, however, disrupts services such as TCP/IP that rely on the IP address. We introduce an analytical approach to MTD and choosing the optimal randomization policy in order to minimize disruptions to system performance. Our approach consists of two components. First, we model the interaction between the adversary and a virtual node as a sequential detection process, in which the adversary attempts to determine whether the node is real or a decoy in the minimum possible time. We compute the optimal strategy for the adversary to decide whether the node is real or a decoy, and derive closed-form expressions for the expected time to identify the real node using this strategy. Second, we formulate the problem of deciding when to randomize the IP addresses, based on a trade-off between reducing the probability of detecting the real node and minimizing the disruption to network services, as an optimal stopping problem. We derive the optimal randomization policy for the network and analyze the detection probability, expected number of connections lost due to IP randomization, and expected time between randomizations under the proposed policy. Our results are illustrated via a simulation study using real-world data from NMAP, a software tool used to identify decoy nodes. Our simulation study indicates that our IP randomization policy reduces the probability of detection while minimizing the number of connections that are disrupted by the randomization.**

I. Introduction

Network security threats are typically preceded by a reconnaissance phase, in which one or more adversaries monitor and probe the network over a period of time. Through monitoring, the adversaries gather information on a set of network features, which include the Internet Protocol (IP) addresses and ports used by the nodes, the versions of protocols (such as DHCP and FTP) that are implemented [1], the operating systems used by nodes [2], the types of user input accepted [3], and the access privileges required to execute commands or receive services on each node [4]. Taken together, these features comprise the *attack surface* [4].

This work was supported by ARO grant W911NF-12-1-0448

A. Clark and R. Poovendran are with the Department of Electrical Engineering, University of Washington, Seattle, WA, 98195, USA, Email: {awclark, rp3}@uw.edu

K. Sun is with the Department of Computer Science, George Mason University, Fairfax, VA, 22030, USA, Email: ksun3@gmu.edu

When an attack surface is static, an adversary can monitor the network, identify vulnerabilities and entry points, and devise efficient, targeted attacks. One approach to prevent such targeted attacks is to change one or more features of the attack surface over time in an unpredictable fashion [5]. By doing so, the network ensures that attacks based on the previous attack surface are ineffective against the system's updated attack surface. Mechanisms for changing the attack surface are classified as *moving target defenses* (MTD).

One widely used approach of MTD makes use of a large number of decoys that are virtual machines [6]. Each of these decoys is given a unique and valid IP address and implements common network protocols, and hence appears to an adversary as a valid system [7]. By observing the interactions between the adversary and the decoys, the network monitoring system gains information about the adversary.

While virtualization enables the creation of a large number of decoys, the capabilities of the decoy nodes are limited by constraints on memory and processing power of the system in which the virtual machines are hosted, as hundreds or even thousands of virtual decoys may reside on a single physical machine [8]. As a result, these low-end virtual decoy nodes employ simplified versions of network protocols such as TCP/IP, do not provide higher-layer services, and often have longer response time for queries made to them. An intelligent adversary can make use of such features to distinguish between real and decoy nodes [9].

After a node has been detected as a decoy by an adversary, the adversary will blacklist that IP address. If the IP addresses of each node remains fixed, blacklisting the IP addresses of decoys will allow identifying the IP address of a real node and penetrating the system. The IP address of the decoy nodes must therefore be randomly refreshed and reassigned over time [10]. On the other hand, if only the decoy node IP addresses are refreshed and reassigned while the addresses of real nodes remain fixed, the adversary will determine that long-lasting IP addresses belong to real nodes. Hence, the network needs to randomize the IP addresses of all real and decoy nodes. Since higher-layer services such as TCP rely on maintaining a static IP address, IP randomization will disrupt and degrade the performance of the services provided by the real nodes.

Deciding when to randomize to prevent detection while minimizing the service disruption has been identified in the literature [11]. Extensive empirical efforts are underway and systems are being built to study the trade off between randomization and performance degradation [8], [12]. Such approaches however are limited to specific and small systems

and hence do not allow the development of general design principles for effective randomization. On the other hand, at present, a suitable analytical approach for deciding when and how often to randomize is missing, thus preventing any effective performance evaluation of decoy-based MTD.

In this paper, we present an analytical approach to evaluating the effectiveness of MTD. We make the following specific contributions:

- We develop a model for the interaction between a virtual node and an adversary. We model the adversary's actions as a sequential detection problem of deciding whether the node is real or a decoy based on its response times, and determine the optimal detection strategy. When the response times to queries are exponential, we develop closed-form expressions for the probability of the real node being identified.
- For the network, we formulate the problem of IP address randomization as an optimal stopping problem and derive the optimal randomization policy. We analyze the performance and security of our proposed randomization policy, including the probability that the adversary determines the IP address of the real node, the expected number of connections that are interrupted, and the expected time between randomizations under the proposed policy.
- We evaluate our approach through a numerical study, which is based on real data from NMAP, a widely-used tool for identifying the operating system and services running on a node. Our simulation results characterize the effectiveness of the decoys in mimicking the real nodes on the system performance.

The paper is organized as follows. Section II reviews the related work. Section III states our assumptions on the system and adversary models. Section IV contains an analytical approach to modeling the interaction between a virtual machine and an adversary, who attempts to determine whether the VM is a decoy. Section V presents a design approach to randomizing the IP address space based on the observed behavior of the adversary. Section VI contains a numerical evaluation. Section VII concludes the paper.

II. Related Work

Decoy-based defenses, such as honeypots, that are based on virtualization have been extensively studied [6]. Decoy designs vary in capability. Decoy nodes can be further classified as low-interaction and high-interaction decoys. High-interaction decoys implement the full protocol stack, including some application-layer services such as fake databases or web servers [13]. Low-interaction decoys emulate a partial network stack, sometimes consisting only of the TCP/IP layer and below [7]. Due to their limited implementation, low-interaction decoys require fewer resources such as bandwidth, CPU, and memory, but are also easier for an adversary to detect because of their lack of ability to provide valid network services and unusually long response time [9], [14].

Address space randomization has been proposed as a diversification technique, and evaluated through implementation studies [11]. In spite of this, the security benefit from address space randomization, as well as the resource cost, are not well-understood, and hence techniques for adaptively choosing a randomization strategy in response to sensed data is not available.

A control-theoretic approach to designing protocol diversification mechanisms, which create diversity by introducing new, randomized protocol states and are distinct from the IP randomization considered in this work, was studied in [1]. The authors formulate the problem of estimating whether the system is in a secure state and discuss principles for designing a control-theoretic modeling approach, but do not formulate such an approach or derive the optimal control policy.

III. Adversary Model and Preliminaries

In this section, we describe the system and adversary models considered in this work.

A. System Model

We consider a network consisting of N virtual machines (VMs). The VMs are classified as real or decoy nodes. Real nodes implement all layers of the network protocol stack and are used to provide applications and services to the system owner and valid users. The function of the decoy nodes is to distract adversaries from targeting real nodes as well as to gather information on adversaries.

The virtual network consisting of decoys is monitored and managed by a hypervisor [15]. The tasks of the hypervisor include creating and removing virtual nodes and assigning IP addresses and routing tables to the nodes. In addition, the hypervisor receives logged information on attempted connections to the decoy nodes, including the nature and timing of any queries from the adversary and the timing of the node response.

Moving target defense (MTD) mechanisms are controlled by the hypervisor. Based on the received information on the network state, the hypervisor can decide to reassign a random IP address to each virtual node. To ensure sufficient entropy in the randomization, a large address space, such as the IPv6 address space, must be used. After each node has been given a new IP address that is independent of its previous address, any information gathered by the adversary relating to the identities of real and decoy nodes becomes obsolete. We assume that all nodes are assigned new IP addresses at once, since otherwise the adversary can differentiate between real and decoy nodes if they are assigned new addresses at different times. When a node is given a new IP address, any connections to the node from outside the virtual network are lost and must be reestablished, potentially interrupting the network services provided by the real node. Legitimate nodes elsewhere in the network are assumed to have mechanisms for identifying the real node and reestablishing connections after randomization has occurred.

B. Adversary Model

The adversary is assumed to know the range of IP addresses occupied by the real node and the decoys; however, it

does not know which IP address is currently used by the real node. We assume that IP addresses are randomly assigned to real and decoy nodes, and the IP address alone does not give any information of whether the node is real or a decoy to the adversary. The goal of the adversary is to determine the IP address of a real node, which can then be targeted for further attacks, and in the process detect and discard the decoys from consideration.

The adversary is capable of sending messages to probe IP addresses and observing the response. Possible messages include ICMP echo packets. The adversary can send queries to the IP addresses in any order. The number of nodes that can be queried at a given time depends on the number of parallel sessions that the adversary can open, which is a function of the adversary's resource capabilities. To simplify our analytical model, we assume there is only one adversary. We assume that the IP address of the real node is not known to the decoys, which do not interact with the real node, so that the adversary cannot automatically obtain the real node's IP address after compromising one decoy. Furthermore, we assume that the rate at which probe messages are sent is limited, as any unknown IP address that sends probe messages at a high rate will be blocked by the hypervisor.

In general, the adversary can determine whether a node is real or a decoy based on knowledge of the implementation differences between the real node and the decoys. First, due to resource constraints, a decoy's network protocols (e.g., TCP/IP) are simplified and thus behave different from those used by the real node. Second, decoy nodes may be detected due to abnormalities in application-layer services such as web servers. This detection method, however, requires semantic analysis of the decoy responses and hence is more time- and resource-intensive. Third, since decoys typically have a longer response time than real nodes due to their limited computation capabilities, the adversary can differentiate between real and decoy nodes by sending probing messages to a node and observing the response time.

In this paper, we focus on timing-based decoy detection, since it is independent of the type of decoy, and hence is the most popular detection method in practice [9], [14].

IV. Analytical Approach to Interaction Between Adversary and Single VM

In this section, we first give a model for the interaction between a virtual machine and an adversary attempting to determine whether the virtual machine is real or a decoy. We then analyze the probability that the adversary detects whether the node is real or a decoy, as well as the expected time for the entire interaction.

A. Single VM Interaction Model

In what follows, we give a model for the interaction of a single virtual machine with the adversary. The adversary sends a series of ping messages to the virtual machine and observes the response times, denoted $Z_1, Z_2, \ldots$. Based on the response times, the adversary reaches a decision as to whether the node is real or a decoy. Let H_0 (resp. H_1) denote the event where the node is a decoy (resp. real). Define the probability distributions $P_0(\cdot) \triangleq P(\cdot|H_0)$ and $P_1(\cdot) \triangleq P(\cdot|H_1)$. Let $P(H_1) = \pi$ and $P(H_0) = (1-\pi)$.

After observing T messages, where T is a random stopping time chosen by the adversary, the adversary outputs a random variable $X = f(Z_1, \ldots, Z_T) \in \{0,1\}$, where $X = 0$ represents a belief that the node is a decoy and $X = 1$ represents a belief that the node is real. If $X = 0$ and the node is real, then the adversary has failed to detect the real node and will scan all of the remaining decoys before returning to reinspect the real node, incurring additional delay c_R. If $X = 1$ and the node is a decoy, then the adversary spends time c_D determining that the node is a decoy via semantic analysis. The total time expended by the adversary interrogating the node is therefore equal to

$$C_{\mathscr{A}}(T, f) = c_D(1-\pi)p_0(X=1) + c_R\pi p_1(X=0) + \mathbf{E}\left(\sum_{k=1}^{T} Z_k\right).$$

The goal of the adversary is to choose T and f in order to satisfy

$$\inf_{T,f} C_{\mathscr{A}}(T, f). \tag{1}$$

Equation (1) defines a sequential detection problem for the adversary. In order to determine the time required for the adversary to choose and the probability of error, the first step is to compute the optimal decision for a given value of T, as follows.

Lemma 1: Given observations $Z_1, \ldots, Z_k$, the optimal decision $X = f(Z_1, \ldots, Z_k)$ is given by

$$X = \begin{cases} 0, & Pr(H_1|Z_1, \ldots, Z_k) < \frac{c_D}{c_D + c_R} \\ 1, & Pr(H_1|Z_1, \ldots, Z_k) \geq \frac{c_D}{c_D + c_R} \end{cases}$$

Proof: We let $p_k = Pr(H_1|Z_1, \ldots, Z_k)$ denote the probability that a node is a decoy conditioned on the observed response times and $\mathbf{E}_k(\cdot) = \mathbf{E}(\cdot|Z_1, \ldots, Z_k)$. The adversary will decide that the node is a decoy and choose $X = 0$ when it minimizes the expected cost, i.e., when

$$c_D Pr(Z_1, \ldots, Z_k|H_0)(1-\pi) > c_R Pr(Z_1, \ldots, Z_k)\pi.$$

Rearranging terms gives the desired result. ■

It remains to find the optimal stopping time T, which determines the number of pings sent by the adversary before determining whether a node is real or a decoy. This stopping time is quantified by the following proposition.

Proposition 1: There exist constants $p_L, p_U \in (0,1)$ such that the optimal stopping time T is given by

$$T = \min\{k : Pr(H_1|Z_1, \ldots, Z_k) \notin (p_L, p_U)\}.$$

Proof: By [16], the optimal stopping time for a problem of the form (1) is equal to

$$\min\{k : \min\{c_R p_k, c_D(1-p_k)\} = v(Z_1, \ldots, Z_k)\},$$

where

$$v(Z_1, \ldots, Z_k) \triangleq \inf_T \left\{ \mathbf{E}_k \left(\min\{c_R p_T, c_D(1-p_T)\} + \sum_{i=1}^{T} Z_i \right) \right\}.$$

The function v can be rewritten as a function of p_k, equal to

$$s(p_k) = \inf_T \{\mathbf{E}_k(\min\{c_R p_T, c_D(1-p_T)\}) + (1-p_k)\mathbf{E}\left(\sum_{i=1}^{T} Z_i | H_0\right) + p_k \mathbf{E}\left(\sum_{i=1}^{T} Z_i | H_1\right)\}.$$

Now, since $s(p_k)$ is an infimum of linear functions of p_k, then it is concave as a function of p_k. Furthermore, $s(p)$ is bounded above by $\min\{c_R p, c_D(1-p)\}$, and hence the infimum is achieved when $s(p) = \min\{c_R p, c_D(1-p)\}$. We therefore have the desired value of T, with $p_L = \sup\{0 \le \pi \le 1/2 | s(\pi) = c_R \pi\}$ and $p_U = \inf\{1/2 \le \pi \le 1 | s(\pi) = c_D(1-\pi)\}$. ∎

Proposition 1 implies that the adversary's optimal strategy in differentiating between real and decoy nodes is to query the node and observe the timing of responses until $Pr(H_1|Z_1,\ldots,Z_k) < p_L$, in which case it terminates and outputs $X = 0$, or $Pr(H_1|Z_1,\ldots,Z_k) > p_U$, in which case it terminates and outputs $X = 1$. In what follows, we analyze the detection probability and time required for detection under basic assumptions regarding the responses of real and decoy nodes.

B. Analysis of Decoy Detection by Adversary

The effectiveness of the decoy nodes depends on their ability to hide the identity of the real node from the adversary. Two metrics that quantify the time required for the adversary to identify the real node are the probability that the adversary has identified the real node within time t, denoted $D(t)$, and the expected time for the adversary to identify the real node, denoted t^*.

In the following analysis, we compute $D(t)$ and t^* based on several assumptions of the network and adversary. We assume that the responses from the decoy and real nodes are both exponential. The mean response times for the decoy and real nodes are equal to $\frac{1}{\lambda_0}$ and $\frac{1}{\lambda_1}$, respectively. We assume that the adversary queries nodes according to a Poisson process with rate μN, so that the time in between queries by the adversary is $\frac{1}{\mu N}$. The parameter μ depends on the resources of the adversary and the adversary's willingness to avoid detection. Based on these assumptions, the following lemma defines the adversary's probability of successfully identifying the real node within time t.

Lemma 2: There exists a constant $B > 0$, with B a function of the constant p_U in Proposition 1, such that

$$D(t) = \sum_{k=k^*}^{\infty} \frac{e^{-\lambda_1 t}}{k!} \mu \int_0^t e^{(\lambda_1-\mu)\tau} (\lambda_1(t-\tau))^k \, d\tau,$$

where $k^* = \min\{k : \left(\frac{\lambda_1}{\lambda_0}\right)^k e^{-(\lambda_1-\lambda_0)t} > B\}$.

Proof: We let T_1 denote the time when the adversary first queries the real node and T_2 denote the time required for the adversary to identify a node as real after querying it the first time. $D(t)$ is therefore equal to $Pr(T_1 + T_2 \le t)$. Let r_1 and r_2 denote the probability density functions of T_1 and T_2, respectively, so that

$$D(t) = \int_0^t r_1(\tau) r_2(t-\tau) \, d\tau.$$

We now analyze r_1 and r_2. Since the adversary is assumed to query nodes according to a Poisson process with rate μN, and each node is uniformly likely to be queried, the real node is queried according to a Poisson process with rate μ. Thus the time until the first query of the real node is an exponential random variable with mean $\frac{1}{\mu}$, and so $r_1(\tau) = \mu e^{-\mu\tau}$.

In order to compute r_2, we first define q_0 and q_1 to be the probability density functions of the decoy and real node response times, respectively. We let Λ_k denote the likelihood ratio after k response times $Z_1,\ldots,Z_k$ have been observed, i.e.,

$$\Lambda_k = \prod_{j=1}^{k} \frac{q_1(Z_j)}{q_0(Z_j)} = \frac{\lambda_1^k e^{-\lambda_1 \sum_{j=1}^k Z_j}}{\lambda_0^k e^{-\lambda_0 \sum_{j=1}^k Z_j}}.$$

The detection rule of Proposition 1 is equivalent to detecting a real node at time k if $\Lambda_k > B$ for some $B > 0$ that is a function of p_U. Let K_t denote the number of responses that the adversary has received by time t. We have

$$Pr\left(\prod_{j=1}^{K_t} \frac{q_1(Z_j)}{q_0(Z_j)} > B\right) = \sum_{k=0}^{\infty} \left[Pr\left(\prod_{j=1}^{K_t} \frac{q_1(Z_j)}{q_0(Z_j)} > B | K_t = k, H_1\right) \cdot Pr(K_t = k|H_1)\right].$$

Assuming that message K_t is received at time t implies that

$$\begin{aligned} r_2(t) &= \sum_{k=0}^{\infty} \left[Pr\left(\prod_{j=1}^{k} \frac{q_1(Z_j)}{q_0(Z_j)} > B \mid \sum_{j=1}^{k} Z_j = t, H_1\right) \cdot Pr(K_t = k|H_1)\right] \\ &= \sum_{k=0}^{\infty} \left[Pr\left(\frac{\lambda_1^k e^{-\lambda_1 \sum_{j=1}^k Z_j}}{\lambda_0^k e^{-\lambda_0 \sum_{j=1}^k Z_j}} > B \mid \sum_{j=1}^{k} Z_j = t, H_1\right) \cdot Pr(K_t = k|H_1)\right] \\ &= \sum_{k=0}^{\infty} Pr\left(\left(\frac{\lambda_1}{\lambda_0}\right)^k e^{-(\lambda_1-\lambda_0)t} > B\right) Pr(K_t = k|H_1). \end{aligned}$$

Now, $Pr\left(\left(\frac{\lambda_1}{\lambda_0}\right)^k e^{-(\lambda_1-\lambda_0)t} > B\right)$ is equal to 0 if $k < k^*$. Hence we have $r_2(t) = Pr(K_t \ge k^*|H_1)$. Combining the expressions of $r_1(t)$ and $r_2(t)$ gives the desired result. ∎

The second metric that we consider is the expected time until the adversary determines the identity of the real node.

Proposition 2: The expected time for the adversary to determine the identity of the real node, denoted t^*, is given by $t^* = \frac{1}{\mu} + \frac{k^*}{\lambda_1}$, where k^* is defined as in Lemma 2.

Proof: The time for the adversary to determine the real node's identity is equal to the time until the adversary queries the real node plus the time for the real node's identity to be found. This is equal to $T_1 + T_2$. Since the adversary queries the real node according to a Poisson process with rate μ, we have $\mathbf{E}(T_1) = \frac{1}{\mu}$. The time T_2 is equal to the time until k^* responses have been received by the adversary. Since the response times are independent and identically distributed with mean $\frac{1}{\lambda_1}$, we have $\mathbf{E}(T_2) = \frac{k^*}{\lambda_1}$. Hence

$$t^* = \mathbf{E}(T_1) + \mathbf{E}(T_2) = \frac{1}{\mu} + \frac{k^*}{\lambda_1}.$$

∎

The time required for the adversary to determine the identity of the real node affects the behavior of the hypervisor. Intuitively, if the adversary can quickly identify the real node, then the IP addresses of the real and decoy nodes must be randomized often, even in spite of lost connections at the real node. This trade-off is formulated explicitly in the following section.

V. Optimal Randomization by Hypervisor

In this section, we describe the optimal defense strategy, as characterized by the time at which the IP address space is randomized. For simplicity, we let $t = 0$ represent the time when the address space was last randomized, and denote by R the random variable representing the time when the next randomization takes place. The goal of the hypervisor is to minimize the probability of the adversary discovering the real node, which we denote $D(t)$ as in Section IV. At the same time, IP address randomization disconnects all users currently communicating with the real node, leading to a cost proportional to the number of connections Y_t. In what follows, we assume that the number of connections Y_t is defined by an M/M/1 queuing process, where connections form with rate λ and are served with rate ω. The selection of the next randomization time can therefore be formulated as the optimal stopping problem

$$\min_R \{\mathbf{E}(D(R) + \beta Y_R)\}, \tag{2}$$

where $\beta \geq 0$ is a parameter used to describe the trade-off between detection probability and performance.

We first describe how the hypervisor estimates $D(t)$ in order to solve (2). Since any connection to a decoy node likely comes from an adversary, and since the decoy nodes record information on when they have been scanned, the number of distinct nodes that have been scanned up to time t, denoted L_t is known to the hypervisor. Hence the probability that a random node, including the real node, has been scanned prior to time t is estimated as $\frac{L_t}{n}$. Furthermore, since the rate at which nodes are scanned prior to time t is $\frac{L_t}{t}$, the expected number of nodes that are scanned prior to time $t' > t$ is equal to $\mathbf{E}(L_{t'}|L_t) = \frac{L_t}{t}t'$. The solution to (2) is given by the following theorem.

Theorem 1: Let $T_c(y)$ denote the expected time to wait for the number of connections to the real node to reach zero, given that the current number of connections is y. The optimal randomization strategy for (2) is to randomize the IP address space at time t if one of the following conditions holds:

1) $L_t = n$
2) $\frac{L_t}{t} \geq n\beta(\omega - \lambda)$, and $\frac{L_t}{n} + \beta Y_t \leq \frac{L_t}{n}\frac{T_c(Y_t)}{t}$, or
3) $Y_t = 0$.

Proof: If $L_t = n$, then all virtual nodes have been scanned and hence the IP address space must be randomized to prevent targeting the real node. We first prove that if $Y_t = 0$, then it is optimal to randomize the address space. We then consider the case where $Y_t > 0$, and prove that it is never optimal to randomize unless $\frac{L_t}{t} \geq n\beta(\omega - \lambda)$. Finally, we show that when $\frac{L_t}{t} \geq n\beta(\omega - \lambda)$, the optimal strategy is either to randomize immediately or wait until the number of connections to the real node reaches zero.

First, note that since $D(t)$ is increasing in time, it is optimal to randomize the IP address space whenever the number of connections $Y_t = 0$. Furthermore, if $L_t = n$, then the adversary has already scanned the entire network and there is no additional benefit to randomizing, and so the hypervisor will wait until $Y_t = 0$.

If $L_t < n$ and $Y_t > 0$, we use the fact [16] that the solution to the optimal stopping problem (2) is given by

$$R^* = \min\{t : D(t) + \beta Y_t = w(L_t, Y_t)\},$$

where $w(L_t, Y_t)$ is defined as

$$w(L_t, Y_t) = \inf_{R' \geq t} \{\mathbf{E}(D(R') + Y(R')|L_t, Y_t)\}.$$

To evaluate whether $D(t) + \beta Y_t = w(L_t, Y_t)$, we first compute $\mathbf{E}(D(t + \delta t) + Y_{t+\delta t}|L_t, Y_t)$. Based on the above analysis, $\mathbf{E}(D(t + \delta t)|L_t, Y_t) = D(t) + \frac{L_t \delta t}{nt}$. Furthermore, for δt sufficiently small, $\mathbf{E}(Y_{t+\delta t}|Y_t, L_t) - Y_t = (\lambda - \omega)\delta t$. Hence we have

$$\mathbf{E}(D(t + \delta t) + \beta Y_{t+\delta t}|L_t, Y_t) - (D(t) + \beta Y_{t+\delta t}) = \frac{L_t \delta t}{nt} + (\lambda - \omega)\delta t$$

and it is therefore optimal to wait for at least time δt if $\frac{L_t}{t} < n\beta(\omega - \lambda)$.

To analyze the case where $L_t < n$ and $\frac{L_t}{t} > n\beta(\omega - \lambda)$, we let $A(t)$ denote the arrival process of connections to the real node and let $S(t)$ denote a Poisson process with rate ω that is independent of $A(t)$. We therefore have

$$\begin{aligned}\mathbf{E}(Y_{t'} - Y_t|Y_t) &\geq \mathbf{E}(A(t' - t) - S(t' - t)) \\ &= (\lambda - \omega)(t' - t),\end{aligned}$$

which implies that

$$\frac{\mathbf{E}(Y_t'|Y_t) - Y_t}{t' - t} \geq \frac{d}{dt}\mathbf{E}(Y_{t'}|Y_t)$$

and hence $\mathbf{E}(Y_{t'})$ is convex as a function of t. Thus $\mathbf{E}(D(t') + \beta Y_{t'}|L_t, Y_t)$ attains its minimum either at $t' = t$ or at the boundary point where $Y_{t'} = 0$, i.e., at time $T_c(Y_t)$. The expected probability of detection at this time is equal to

$$\mathbf{E}(D(T_c)|L_t, Y_t) = \frac{L_t}{nt}T_c(Y_t).$$

Thus, it is optimal to randomize at time t if $L_t < n$, $\frac{L_t}{t} \geq n\beta(\omega - \lambda)$, and $D(t) + \beta Y_t \leq \frac{L_t}{nt}T_c(Y_t)$, as desired. ∎

Theorem 1 states that the optimal IP randomization strategy for the hypervisor is the following. First, if the number of connections is 0, then the hypervisor randomizes the address space immediately, as there is no cost to randomization at that point. If the rate of scanning by the adversary is sufficiently low, then the hypervisor should wait until all connections to the real node have finished. Otherwise, the hypervisor should randomize immediately.

In what follows, we characterize the resulting system performance by analyzing the time between IP randomizations,

the probability that the real node is detected, and the expected number of connections that are disrupted.

A. Analysis of Detection Probability and Resource Cost

We begin with an analysis of the expected time between address randomizations. We state our results as two lemmas, which depend on the average scanning rate of the adversary. As a preliminary, we let $T' = \inf\{t' - t : Y_{t'} = Y_t - 1\}$ and $\tau_b = \mathbf{E}(T')$. We let $\Delta \triangleq \frac{L_t}{t}$ denote the scanning rate of the adversary.

Lemma 3: If the scanning rate Δ satisfies $\Delta < n\beta(\omega - \lambda)$ and the number of connections to the real node has reached its stationary distribution, then the expected time between randomizations of the address space is $\mathbf{E}(R) = \frac{\lambda\tau_b}{\omega - \lambda}$.

Proof: If the scanning rate satisfies $\Delta < n\beta(\omega - \lambda)$, then by Theorem 1, the hypervisor will not randomize the address space until $Y_t = 0$. Hence $\mathbf{E}(R) = \tau_b \mathbf{E}(Y_0)$ since we have assumed that connections to the real node follow an M/M/1 queuing process. Furthermore, if we assume that the connection process of the real node is in steady-state, then $\mathbf{E}(Y_0) = \frac{\lambda}{\omega - \lambda}$. ∎

The following lemma describes the expected number of connections broken and the probability of detection under the slow-scanning assumption.

Lemma 4: If the scanning rate Δ satisfies $\Delta < n\beta(\omega - \lambda)$, then the expected number of connections lost due to IP randomization is 0 and the probability of detection is $\frac{\Delta}{n}\frac{\tau_b\lambda}{\omega - \lambda}$.

Proof: By Theorem 1, IP randomization will not take place until $Y_t = 0$, and hence 0 connections are lost. The probability of detection is given by

$$\mathbf{E}(D(R)) = \mathbf{E}\left(\frac{L_R}{n}\right) = \frac{\Delta}{n}\mathbf{E}(R) = \frac{\Delta}{n}\frac{\tau_b\lambda}{\omega - \lambda},$$

giving the desired result. ∎

In the case where $\Delta \geq n\beta(\omega - \lambda)$, the following proposition describes the time between randomizations, probability that the real node is identified, and the expected number of dropped connections.

Proposition 3: If the scanning rate Δ satisfies $\Delta \geq n\beta(\omega - \lambda)$, then the following hold. If $\frac{\Delta}{n}\tau_b - \beta < 0$, then the derivations of $\mathbf{E}(R)$, $\mathbf{E}(D(R))$, and $\mathbf{E}(Y_R)$ in Lemmas 3 and 4 hold. Otherwise, define $a = \frac{\Delta t}{n}\left(\frac{\Delta}{n}\tau_b - \beta\right)^{-1}$. We have

$$\mathbf{E}(R) \leq \min\left\{\frac{\tau_b\lambda}{\omega - \lambda}, \left[\tau_f\left(a - \frac{\lambda}{\omega - \lambda}\right)\right]_+\right\} \qquad (3)$$

and the probability of detection is bounded by

$$\mathbf{E}(D(R)) \leq \frac{\Delta}{n}\min\left\{\frac{\tau_b\lambda}{\omega - \lambda}, \left[\tau_f\left(a - \frac{\lambda}{\omega - \lambda}\right)\right]_+\right\}. \qquad (4)$$

Furthermore, the expected number of broken connections is given by

$$\mathbf{E}(Y_R) = a\frac{\pi_a}{\pi_0 + \pi_a},$$

where $\pi_a = \sum_{i \geq a} p(i)$, $\pi_0 = p(0)$, and $p(\cdot)$ is the stationary distribution of Y_k.

Proof: Theorem 1 implies that, if $\Delta \geq n\beta(\omega - \lambda)$, then the IP address space is randomized either when $Y_t = 0$ or when $\frac{L_t}{n} + \beta Y_t \leq \frac{L_t}{t}\frac{\tau_b Y_t}{n}$, which is equivalent to $\left(\frac{\Delta}{n}\tau_b - \beta\right)Y_t \geq \frac{\Delta t}{n}$. If $\frac{\Delta}{n}\tau_b < \beta$, then the condition never holds and the space is randomized when $Y_t = 0$, leading to the results of Lemmas 3 and 4.

Otherwise, R is reached when $Y_t = 0$ or $Y_t \geq a$, i.e.,

$$R = \inf\{t : Y_t \notin (0, a)\}.$$

Let $\tau_0 = \inf\{t : Y_t = 0\}$ and $\tau_a = \inf\{t : Y_t = a\}$. We have

$$\begin{aligned}\mathbf{E}(R) &= \mathbf{E}(\min\{\tau_0, \tau_a\}) \\ &\leq \min\{\mathbf{E}(\tau_0), \mathbf{E}(\tau_a)\} \\ &= \min\{\tau_b\mathbf{E}(Y_0), \tau_f(a - \mathbf{E}(Y_0))\} \\ &= \min\left\{\tau_b\frac{\lambda}{\omega - \lambda}, \tau_f\left(a - \frac{\lambda}{\omega - \lambda}\right)\right\}.\end{aligned}$$

As a result, $\mathbf{E}(D(R))$ is given by

$$\mathbf{E}(D(R)) = \frac{\Delta}{n}\mathbf{E}(R) \leq \frac{\Delta}{n}\min\left\{\tau_b\frac{\lambda}{\omega - \lambda}, \tau_f\left(a - \frac{\lambda}{\omega - \lambda}\right)\right\}.$$

Finally, the expected value $\mathbf{E}(Y_R)$ is derived as follows. Define $\tau_a = \min\{t : Y_t \geq a\}$ and $\tau_0 = \min\{t : Y_t = 0\}$. By Theorem 1, $R = \min\{\tau_0, \tau_a\}$. Hence

$$\mathbf{E}(Y_R) = 0 \cdot Pr(R = \tau_0) + a \cdot Pr(R = \tau_a).$$

Supposing that the system is in steady-state, we have

$$\begin{aligned}Pr(R = \tau_a) &= \int_{t=0}^{\infty} Pr(Y_R \geq a | R = t) Pr(R = t)\ dt \\ &= \int_{t=0}^{\infty} Pr(Y_t \geq a | Y_t = 0 \cup Y_t \geq a) Pr(R = t)\ dt \\ &= \int_{t=0}^{\infty} \frac{\pi_a}{\pi_0 + \pi_a} Pr(R = t)\ dt \\ &= \frac{\pi_a}{\pi_0 + \pi_a}.\end{aligned}$$

The expected number of disconnected sessions $\mathbf{E}(Y_R)$ is therefore given by $a\frac{\pi_a}{\pi_0 + \pi_a}$. ∎

VI. Simulation Study

In this section, we present a simulation study using Matlab. Our simulations addressed the adversary's querying process, described in Section IV, as well as the IP address randomization by the hypervisor in Section V. The setup of our threat model is motivated by the NMAP software tool [2], which performs automated scanning of a network in order to identify node configurations, including whether a virtual node is real or a decoy.

We consider a network with $N = 100$ nodes, including one real node and 99 decoys. Each real node was assumed to have an exponential response time with mean 100ms. The mean response times of the decoys varied from 120ms to 200ms; these parameters are based on the empirical study of [9], which indicates that the response time of a decoy node to a ping request is 20% to 50% greater than the response time of a real node in the same network. It was assumed that determining that a node is a decoy via semantic analysis required 120 seconds. Connections arrived at the real node according to an M/M/1 process with an average of

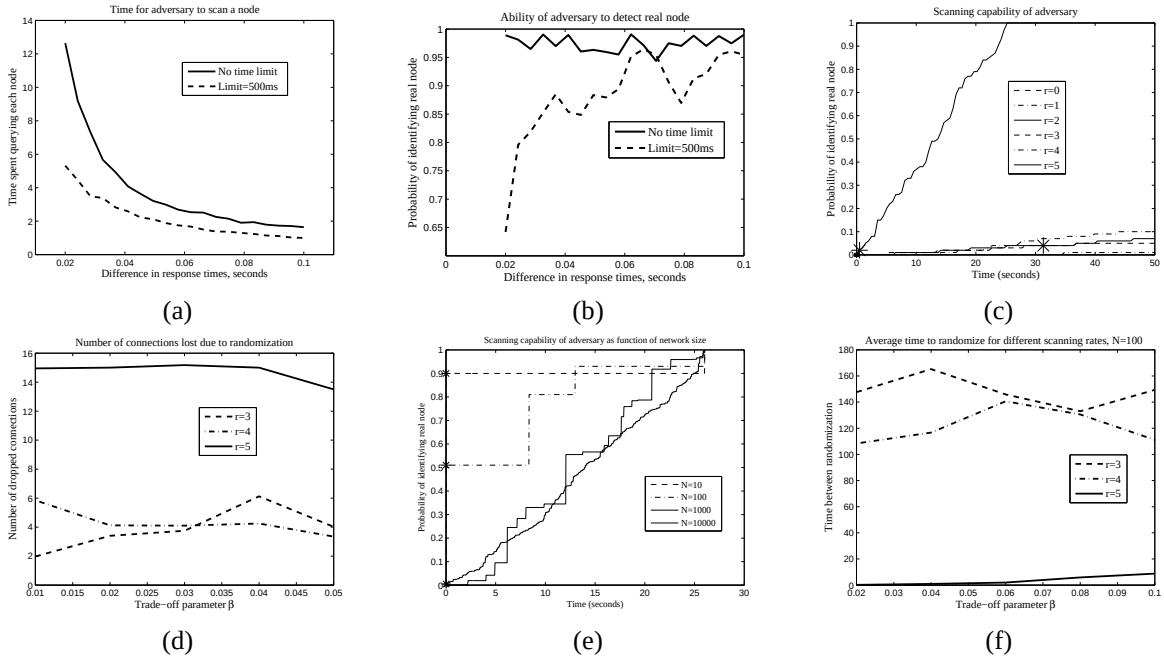

Fig. 1. Simulation results for network of $N = 100$ nodes, with one real node and 99 decoys. Real nodes have mean response time 100ms, while response time of decoys varies from 120 to 200ms. (a) Time for the adversary to identify whether a node is real or fake, either using the decision rule of Section IV or a rule where a node is a decoy if the response time exceeds a threshold of 500ms. The detection time is decreasing in the difference between real and decoy response times. (b) Probability that the adversary correctly identifies a node as real. While the decision rule of Section IV requires more queries per node, it results in higher detection probability. (c) Probability that the adversary has scanned the real node as a function of time, based on the scanning rate of the adversary (determined by parameter r). Asterisks indicate the time at which randomization occurs. (d) Number of dropped connections when randomization occurs. When the adversary employs a higher scanning rate, the network randomizes at a higher rate, thus increasing the number of connections that must be dropped and restored after randomization. (e) Effect of network size on the adversary's probability of identifying the real node. A larger number of decoys decreases the probability of identification and allows the hypervisor to wait longer before randomizing (randomization times shown as asterisks). (f) Expected time between IP randomization as a function of the trade-off β. If more weight is assigned to the cost of dropped connections (higher β), then the hypervisor will wait longer before randomizing. Additionally, a higher scanning results in shorter delay before randomization.

one new connection per second, and 1.1 existing connections terminated per second.

To evaluate the time required to determine whether a node is a decoy, we considered two querying strategies by the adversary. The first strategy, based on Proposition 1, terminates when the *a posteriori* probability that the node is real leaves a predetermined range (p_L, p_U). We chose $p_L = 6.7x10^{-4}$ and $p_U = 0.037$, ensuring that the probability of classifying a real node as fake is no more than 0.05 and the probability of classifying a fake node as real is no more than 0.25. In the second strategy, the adversary stops making queries either when the *a posteriori* probability that the node is real leaves the range (p_L, p_U), or when the response time of a query exceeds 500ms.

In order to analyze the hypervisor's randomization behavior, we simulated an adversary scanning the virtual network based on NMAP fingerprinting rules. In NMAP, the scanning rate varies according to a parameter $r \in \{0,1,\ldots,5\}$. Each value of r represents a different scanning rate, with $r = 0$ a slow scanning rate in order to avoid intrusion detection systems and $r = 5$ representing a rapid scanning rate. At $r = 0$, the adversary waits for 300 seconds (5 minutes) between scanning nodes, greatly increasing the time to identify the real node. When $r = 1$, the adversary waits 15 seconds in between scans, while $r = 2$ represents a waiting time of 0.4 seconds between scans. When $r = 3$, there is no waiting time. At $r = 4$, the adversary does not wait between scans, and stops querying a node when the response time exceeds 500ms. At $r = 5$, the adversary does not wait between scans, and stops querying a node when the response time exceeds 200ms.

The time required for the adversary to query each node and differentiate between real and decoy nodes is illustrated in Figure 1(a). When the response time of the decoy nodes is 120ms, compared to 100ms for the real nodes, it takes the adversary 14 seconds on average to distinguish between a real and decoy node using the strategy of Proposition 1. As the response time of the decoys diverges from the real node response time, however, the time required to diminish decreases. When the adversary times out after waiting 500ms for a response, the detection time is reduced, at the cost of increasing the probability of error.

Figure 1(b) shows the probability that a real node is correctly identified as a real node, as a function of the response time of the decoy nodes. The optimal detection strategy of Proposition 1 consistently provides a probability of correctness close to 1, at the cost of increased time to query each node. Imposing a timeout of 500ms, on the other hand, results in a low correctness probability when the response times of the real and decoy nodes are similar. The detection probability of this strategy converges to 1 because, when the average response time of the decoy is longer than

that of a real node, most responses that exceed the 500ms limit will be from decoy nodes.

The ability of the adversary to scan the entire virtual network and identify the real node is examined in Figure 1(c). Each curve represents a different value of the parameter r, which determines the delay between scans and the time allocated to each node. When $r = 0$ and $r = 1$, the adversary introduces delays between scans to avoid detection, increasing the time required between IP randomizations. In these cases, the hypervisor waits until all connections to the valid node have terminated (condition 1 of Theorem 1) before randomizing the address space. As r increases, the rate of scanning and hence the probability of identifying the real node at each time t increases. When $r \in \{4,5\}$, the hypervisor must randomize frequently in order to prevent the adversary from finding the real node.

The effect of randomization on the system performance is shown in Figure 1(d). When $r = 0,1,2$, the adversary's scanning rate is sufficiently low that the hypervisor can afford to wait until there are zero connections before randomizing. For the cases $r = 3,4,5$ shown, the hypervisor must randomize before all connections are concluded (condition 2 of Theorem 1), thus disrupting the performance of valid users. As the scanning rate increases ($r = 5$), the number of dropped connections increases due to the decreased time between randomization.

To mitigate the impact of a more powerful adversary, the system can deploy additional decoy nodes. Figure 1(e) shows the adversary's detection probability over time for different network sizes, N. Even if the adversary scans at the fastest possible rate ($r = 5$), the detection probability at the time of randomization is only 0.1 (when $N = 1000$) and 0.01 (when $N = 10000$). This allows the hypervisor to wait until the number of connections to the real node goes to zero. This improvement in availability to outside nodes, however, must be balanced with the resource cost in CPU and memory required to maintain a large number of decoys.

The average time between consecutive randomizations is illustrated in Figure 1(f). As the parameter β increases, additional weight is placed on the disruptions of valid connections caused by randomization, and hence the hypervisor will wait longer until more connections have been completed. On the other hand, a higher scanning rate r decreases the time between randomizations, since the detection probability increases at a faster rate.

VII. Conclusion

In this paper, we studied moving target defense mechanisms that use decoy-based deception and IP address randomization in order to thwart reconnaissance by adversaries. We first considered the interaction between a single virtual node and the adversary, in which the adversary attempts to determine whether a node is real or a decoy by observing the timing of the node responses to probe packets. We modeled the adversary's behavior as the solution to a sequential detection problem and derived closed-form solutions for the detection probability and expected time of detection for the case where the response times are exponentially distributed.

We then developed a design framework in which the hypervisor, which controls the virtual network, receives inputs from the decoy nodes, including which decoy nodes have been scanned by the adversary. We formulated the decision of when to randomize as an optimal stopping problem and derived the optimal policy for the hypervisor. We then analyzed probability of identifying the real node before randomization, the expected number of lost connections, and the expected time between randomizations as a function of the network parameters. Our results were further illustrated through simulation study, which was based on the NMAP network scanning tool.

In future work, we will extend our framework to consider design parameters other than the time of IP randomization, such as the number of decoys and the CPU and memory resources allocated to each decoy. We will also explore other decoy detection methods for the adversary, such as detection based on protocol implementations, using our framework.

References

[1] J. Rowe, K. Levitt, T. Demir, and R. Erbacher, "Artificial diversity as maneuvers in a control-theoretic moving target defense," *Moving Target Research Symposium*, 2012.

[2] M. Wolfgang, "Host discovery with NMAP," http://moonpie.org/writings/discovery.pdf, 2002.

[3] R. Chandrashekhar, M. Mardithaya, S. Thilagam, and D. Saha, "SQL injection attack mechanisms and prevention techniques," *Advanced Computing, Networking and Security*, pp. 524–533, 2012.

[4] P. Manadhata and J. Wing, "An attack surface metric," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 371–386, May-June 2011.

[5] S. Jajodia, A. Ghosh, V. Swarup, C. Wang, and X. S. Wang, *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. Springer, 2011.

[6] N. Provos and T. Holz, *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley Professional, 2007.

[7] N. Provos, "A virtual honeypot framework," *Proceedings of the 13th USENIX security symposium*, vol. 132, 2004.

[8] X. Liu, L. Peng, and C. Li, "The Dynamic Honeypot Design and Implementation Based on Honeyd," *Advances in Computer Science, Environment, Ecoinformatics, and Education*, pp. 93–98, 2011.

[9] S. Mukkamala, K. Yendrapalli, R. Basnet, M. Shankarapani, and A. Sung, "Detection of virtual environments and low interaction honeypots," *IEEE Information Assurance and Security Workshop (IAW)*, pp. 92–98, 2007.

[10] M. Abu Rajab, F. Monrose, and A. Terzis, "On the impact of dynamic addressing on malware propagation," *Proceedings of the 4th ACM Workshop on Recurring Malcode*, pp. 51–56, 2006.

[11] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis, "Defending against hitlist worms using network address space randomization," *Computer Networks*, vol. 51, no. 12, pp. 3471–3490, 2007.

[12] I. Kuwatly, M. Sraj, Z. Al Masri, and H. Artail, "A dynamic honeypot design for intrusion detection," *IEEE/ACS International Conference on Pervasive Services*, pp. 95–104, 2004.

[13] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. Snoeren, G. Voelker, and S. Savage, "Scalability, fidelity, and containment in the potemkin virtual honeyfarm," *SIGOPS Operating Systems Review*, vol. 39, no. 5, pp. 148–162, 2005.

[14] X. Fu, W. Yu, D. Cheng, X. Tan, K. Streff, and S. Graham, "On recognizing virtual honeypots and countermeasures," *IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pp. 211–218, 2006.

[15] D. Chisnall, *The Definitive Guide to the Xen Hypervisor*. Prentice Hall, 2007.

[16] G. Peskir and A. Shiryaev, *Optimal Stopping and Free-boundary Problems*. Birkhäuser Basel, 2006.