Spatio-temporal Address Mutation for Proactive Cyber Agility against Sophisticated Attackers

Jafar Haadi Jafarian, Ehab Al-Shaer, Qi Duan
Department of Software and Information Systems
University of North Carolina at Charlotte
Charlotte, NC, USA
jjafaria, ealshaer, qduan@uncc.edu

ABSTRACT

The static one-to-one binding of hosts to IP addresses allows adversaries to conduct thorough reconnaissance in order to discover and enumerate network assets. Specifically, this fixed address mapping allows distributed network scanners to aggregate information gathered at multiple locations over different times in order to construct an accurate and persistent view of the network. The unvarying nature of this view enables adversaries to collaboratively share and reuse their collected reconnaissance information in various stages of attack planning and execution. This paper presents a novel moving target defense (MTD) technique which enables host-to-IP binding of each destination host to vary randomly across the network based on the source identity (spatial randomization) as well as time (temporal randomization). This spatio-temporal randomization will distort attackers' view of the network by causing the collected reconnaissance information to expire as adversaries transition from one host to another or if they stay long enough in one location. Consequently, adversaries are forced to re-scan the network frequently at each location or over different time intervals. These recurring probings significantly raises the bar for the adversaries by slowing down the attack progress, while improving its detectability. We introduce three novel metrics for quantifying the effectiveness of MTD defense techniques: deterrence, deception, and detectability. Using these metrics, we perform rigorous theoretical and experimental analysis to evaluate the efficacy of this approach. These analyses show that our approach is effective in countering a significant number of sophisticated threat models including collaborative reconnaissance, worm propagation, and advanced persistent threat (APT), in an evasion-free manner.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: General—*Security and protection*

MTD'14, November 3, 2014, Scottsdale, Arizona, USA.

http://dx.doi.org/10.1145/2663474.2663483.

General Terms

Security

Keywords

IP address randomization; adversary-awareness; reconnaissance; moving target defense (MTD)

1. INTRODUCTION

Static nature of network configurations allows adversaries to perform thorough and collaborative reconnaissance with low cost and in a short time. This is due to the fixed, global and persistent host-IP mappings that allow attackers to distribute scanning tasks and aggregate reconnaissance information from different locations and at different scanning intervals.

Although enterprise networks with static configurations could be secured by perimeter reactive security devices, such as firewall and IDS/IPS, adversaries can still infiltrate these networks by compromising vulnerable public hosts, and then traversing the internal network host-by-host until the target is reached. Moreover, these reactive defense mechanisms suffer from several drawbacks, including their inability to discover novel attacks, their susceptibility to evasion, and their latency in attack detection [17]. These weaknesses entails the design of proactive defense countermeasures that (1) thwart insider attacks in enterprise networks, and (2) provide mitigation against stealthy and unknown attacks, without necessarily relying on observability of attack behavior.

In this paper we present a proactive MTD technique that provides mitigation for enterprise networks against such persistent threat models. Our approach introduces dynamicity into network by not allowing the reconnaissance information collected at one host to be usable at other times and other network hosts. This objective is achieved by varying the host-IP binding dynamically based on location and time parameters. Our approach is host- and network-transparent as it requires no changes to the actual IP addresses or platforms of the end-hosts. Each host is associated with a unique set of IP addresses, called ephemeral IP addresses (eIP), to reach other hosts in the network. More specifically, to reach host B, host A is given an eIP. This eIP only allows A to reach B (spatial constraint) and only during a specific interval (temporal constraint). However, using this eIP by any other host or during other time intervals constitutes an invalid communication. This spatial host-IP binding is unique in the network (*i.e.*, each host has a unique set of eIPs) and it will be

frequently changed after random intervals. The appropriate host-IP binding is determined based two strategies: (1) *random mutation* in which the eIP is selected randomly from the unused address space based on uniform distribution, and (2) *deceptive mutation* in which the mapping is constructed in order to deceive the adversary and either deter or allow detection of her penetration in the network.

Our approach offers significant contribution over previous IP hopping techniques. Unlike previous works on IP hopping which are only based on temporal mutations [1, 3], our approach presents a novel and more efficient IP mutation approach using the spatial dimension as the main mutation parameter. This significantly improves effectiveness and avoids the limitations of the temporal-based IP mutation/hopping [1]. In addition, this scheme is shown to be more effective against various sophisticated reconnaissance attacks, including cooperative reconnaissance and advanced persistent threat. Moreover, unlike previous approaches [19], it can be deployed in existing IPv4 and IPv6 networks incrementally and transparently without requiring any modification to existing end-host platforms and network infrastructures, or interrupting active sessions during the mutation operation.

The MTD defense techniques can either (1) deter the the attack progress by reconfiguring system resources, (2) deceive the attacker by providing a false view of the target system, and (3) increase attacker's detectability by deceitfully forcing her to commit more mistakes. We introduce three metrics, namely *deterrence*, *deception* and *detectability* to quantify each of these mitigation vectors respectively. Using these metrics, we evaluate the effectiveness of our approach against intended threat models. Our analyses and simulation shows that this spatio-temporal address randomization approach can thwart a diverse variety of threat models, including collaborate reconnaissance, propagation of stealthy worms, and advanced persistent threat.

The rest of this paper is organized as follows. Section 2 reviews the related work. In Section 3 the basic methodology of our approach is discussed. Section 4 presents architecture and protocols for deployment of our approach on both legacy and software-defined networks. Section 5 shortly describes the implementation details. In Section 6 the effectiveness and overhead of our approach is evaluated, and Section 7 concludes the paper.

2. RELATED WORKS

The APOD scheme [4] uses *hopping tunnels* based on temporal cooperative address/port randomization to disguise the identity of end parties from sniffers. Similarly, DyNAT [11] provides a transparent approach for IP hopping by translating the IP addresses before packets enter the public network to defeat sniffers. Both techniques require cooperation of both client and server networks for randomization and none of them address attack models such as scanners which rely on probe responses for discovering the end-hosts.

NASR [3] is an address space randomization scheme that offers a temporal IP hopping approach to defend against hitlist worms, based on DHCP updates. In addition to limited unpredictability, NASR is not transparent to the end-hosts because DHCP changes are applied to the end-host itself which results in disruption of active connections during address transition.

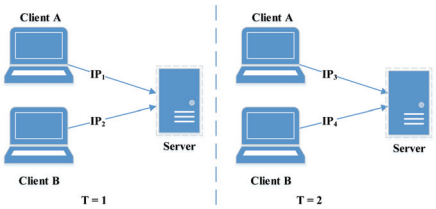

Figure 1: Illustration of spatiotemporal IP assignment

RHM [1], and OF-RHM [9] provide temporal mutation of host IP addresses to disrupt external and internal scanners. In a similar work, Albanese *et al.* [2] propose a mechanism for periodically changing the virtual identity of nodes in a MANET in order to defeat the attacker's reconnaissance efforts. The approaches are transparent to end-hosts, but do not consider spatial mutation which restricts their applicability against coordinated attack models.

Cai *et al.* [6] and Yegneswaran *et al.* [20] present techniques for defending honeynets from systematic mappings by IP shuffling. These techniques are reactive and are not effective against coordinated information gathering and persistent threats such as worm propagation or advanced persistent threat.

SDNA [19] inserts a hypervisor within each network node to make host appearance dynamic to observers while at the same time retaining transparency to OS, applications, and end-users. The main shortcoming of SDNA architecture emanates from the costly reconfiguration of each and every network node to include hypervisor technology.

3. BASIC METHODOLOGY

Regular users usually reach network hosts by their names. Host names are resolved to their corresponding IP addresses by naming services such as DNS. In legacy networks, a name is usually mapped to one (or several) rather static addresses which can be used globally (by any host) to reach the target. In our approach, name-to-address mapping occurs dynamically and as a function of source (requestor) identity as well as time. Individualized mapping based on source identity is referred to as *spatial* mutation, while dynamic (*i.e.*, shifting) mapping based on time is referred to as *temporal* mutation. Specifically, host h_i must use different eIP addresses to reach a host h_j at various time intervals. The lifetime during which an eIP can be used by h_i to reach h_j is denoted as *mutation interval*. At the end of each mutation interval, this eIP is updated. A trivial example of spatiotemporal IP mutation is depicted in Figure 1. As an example of spatial mutation note that during the first mutation interval ($T = 1$), client A reaches the server via IP_1, while B reaches the same server via IP_2. As an example of temporal mutation note that during the first mutation interval, A reaches the server via IP_1, while during the second mutation interval the same server is reached via IP_3.

In our approach, the IP address assigned to a host network interface remains untouched. We refer to these addresses as real IP addresses or rIPs of network hosts. Except for authorized network administrators, no entity is allowed to reach a host by its rIP. Instead, each rIP is mapped to a short-lived ephemeral IP address, called eIP, which is determined based on the source identity and time. The new eIPs are provided

to users by DNS responses and the time-to-live (TTL) value is set to the expiration time of the new rIP-eIP mapping.

Assume $\mathbf{R}$, $\mathbf{E}$, and $\mathbf{N}$ denote the set of rIPs, eIPs, and names of the network respectively. The DNS functionality in static networks could be described by two functions. The first function defines the mapping from name to rIP address of the queried host:

$$f_{mapping} : \mathbf{N} \rightarrow \mathbf{R}$$
$$f_{mapping}(n_d) \mapsto i_d \quad (1)$$

where n_d and i_d denote the name and rIP address of the queried host respectively. The second function defines the TTL value of the DNS reply; *i.e.*, the expiration time of the mapping. The TTL value could be different for each host.

$$f_{TTL} : \mathbf{N} \rightarrow \mathbb{N}$$
$$f_{TTL}(n_d) \mapsto \tau \quad (2)$$

In our approach, the mapping function is changed to reflect the spatio-temporal IP address assignment:

$$f_{mapping} : \mathbf{N} \times \mathbf{N} \times \mathbb{N} \rightarrow \mathbf{E}$$
$$f_{mapping}(n_d, n_s, t) \mapsto i_d \quad (3)$$

where n_s and t denote the querying host (client) name and current time respectively. Also, the TTL value of each name-eIP mapping is different for each mutation interval. Therefore, the TTL function is defined in the following manner:

$$f_{TTL} : \mathbf{N} \times \mathbb{N} \mapsto \mathbb{N}$$
$$f_{TTL}(n_d, t) \mapsto \tau \quad (4)$$

Note that the source identity is known to the authoritative DNS server for intra-enterprise DNS queries. For external clients, although there are proposals to extend DNS resolvers to pass in part of the clientŠs IP address in the DNS message [5], this information is not currently available in DNS queries. Therefore, for external clients our mutation is restricted to the temporal dimension and limited spatial mutation (based on identity of resolvers).

The eIPs must be chosen from the largest possible candidate set to minimize IP reuse and maximize adversary's search space. IPv6 will provide enough IP addresses per host to simplify the IP mutation problem. However, because the IP mutation is confined to intra-enterprise communications, we use both unused public as well as private IP addresses, such as address ranges 192.168.0.0/16, or 10.0.0.0/8. Therefore, the eIPs are chosen from a relatively large set of IP addresses (larger than /8 even in IPv4 scheme).

3.1 Temporal Mutation

Temporal mutations are performed by associating an expiration time to each mapping; *i.e.*, via the TTL function f_{TTL}. Current time of the system is synchronized by a central entity, referred to as *controller* (Section 4). More specifically, current time is defined as a nonzero integer which is incremented after a relatively short interval (*i.e.*, every one minute).

We define parameter λ as the average mutation interval. Determining effective lifetime of mappings or the length of a mutation interval is a trade-off between performance and robustness. Short durations will result in higher number of DNS queries, while longer durations decrease the degradation rate of attacker's information.

The TTL value of the k^{th} eIP mapping for a client host h_i to reach a server host h_j, denoted as $T^k_{i \rightarrow j}$ is determined based on the identity of both end-parties. To maximize unpredictability (Markov property) while achieving the expected average length, each such TTL value is chosen based on a Poisson arrival process with mean λ.

3.2 Spatial Mutation

When the temporal mutation interval of a mapping from a host h_i to h_j is expiring, the controller must provide a new mapping. In fact, to reach each destination host h_j, each source host h_i is associated with an eIP, such that this eIP could be only used by h_i to reach h_j. The distribution based on which these new mappings are determined can be either *uniform* or *deceptive (adversary-adaptive)*. The controller uses a strategy selection algorithm to determine the appropriate strategy at any given time by analyzing the behavior of potential adversaries in the network.

For random mutation, to determine a new mapping for a host h_i to reach another host h_j, the controller first excludes currently in-use eIPs and simply chooses a random eIP from remaining eIPs in $\mathbf{E}$ with uniform probability. The eIP that host h_i can use to reach h_j during the k^{th} mutation interval, is denoted as $e^k_{i \rightarrow j}$.

In contrast, deceptive mutation characterizes the adversarial behavior and reconfigures the addresses in order to deceive attackers about identity of network hosts. Specifically, assume an adversary has probed some part of network address space and the rest of addresses are yet to be visited. As a result of these probes, some network hosts are already probed by the attacker. The deceptive mutation reconfigures network hosts such that if the attacker randomly chooses an IP address from the unvisited address space, this IP address is most probably mapped to a host which is already probed or visited by the attacker. For example in Figure 1 assume at $T = 1$, an attacker which resides in A probes IP_1 and gathers information about the server. At $T = 2$ the attacker probes an unvisited IP such as IP_3, but deceptive mutation leads the attacker to the same server.

Deceptive mutation is designed based on two observations. Firstly, security properties of a network and hosts allow us to conjecture the probability that a host h_j is chosen or compromised as the next target, when the attacker resides in h_i. In other words, network access control policies as well as the running services and vulnerabilities of h_j allow us to predict how attractive a target such as h_j would be to an attacker in h_i. Secondly, persistent threat models such as worm propagation, or advanced persistent threat result in inflation of weakly connected components (WCC) in the connectivity graph of the network [7].

The connectivity graph represents communications between network hosts during a specific interval. In this graph, nodes are network hosts and edges represent flows between hosts. A connectivity record is a tuple of the form $\xi = \langle h_i, h_j, t \rangle$, where h_i is the source (referred to as $\xi[src]$), h_j is the destination host ($\xi[dst]$), and t shows the last time h_i *attempted* to communicate with h_j ($\xi[t]$). At each interval, the controller receives at most n^2 connectivity records, where n denotes the number of hosts.

Gateways are the distributed entities which are responsible for handling rIP-eIP translations of physical subnets (Section 4). Each gateway sends its connectivity records to the controller at regular intervals. By aggregating connectiv-

ity records of all gateways, the controller establishes/updates the network connectivity graph. Assume the set of all connectivity records are represented by $\Lambda = \{\xi_1, \ldots, \xi_z\}$. Given Λ, we define a directed graph $G(\Lambda) = \langle V(\Lambda), E(\Lambda)\rangle$, where $V(\Lambda) = \bigcup_{\xi\in\Lambda}\{\xi[src], \xi[dst]\}$ and $E(\Lambda) = \bigcup_{\xi\in\Lambda}\{(\xi[src], \xi[dst])\}$. The controller periodically receives the connectivity reports from the gateways and updates the connectivity graph.

The objective is to determine the mapping that h_i must use to reach h_j. Alg. 1 describes this procedure. To this aim, the controller first determines the weakly connected components (WCC) of the graph using a depth-first search, and groups the hosts that belong to the same WCC together. This is because *if* an adversary resides in any of the hosts in a WCC, her adversarial reconnaissance or activity within the monitoring period is definitely confined to the hosts in the same WCC. For example, if a worm is propagating in a network, all infected hosts will be in the same WCC. However, these infected hosts are not attractive targets for the worm.

Deceptive mutation aims to assign potentially unattractive IP addresses to potentially attractive targets for an attacker residing in h_i. By establishing the graph and determining the WCCs, each host h_i is associated with a set of hosts that reside in its WCC. This set is denoted as $\boldsymbol{WCC_i} = \{h_k\}$ and γ is defined as the size of this set; *i.e.*, $\gamma = |\boldsymbol{WCC_i}|$. The last eIP addresses through which h_i reached the hosts in $\boldsymbol{WCC_i}$ are included in a set $\boldsymbol{E_i} = \{IP_k\}$.

The probability that a attacker residing in h_i moves to h_j is denoted as *transition score* from h_i to h_j. The procedure for determining the transition scores is described later. By considering the transition scores of h_i to all the hosts with different WCCs, the controller determines the top γ hosts with highest transition scores which are *not* in the same WCC as h_i, denoted as $\boldsymbol{K_i}$. If h_j belongs to $\boldsymbol{K_i}$, the controller randomly chooses an address from $\boldsymbol{E_i}$ as eIP. This is because, firstly h_j is not in the same WCC with h_i, and therefore a potential attacker residing in h_i has not attempted to communicate with it in the near past. Secondly, the transition score of h_i to h_j shows that for an attacker in h_i, h_j is an easy and attractive target. However, if h_j is not in $\boldsymbol{K_i}$, or it is within the same WCC as h_i, the new mapping is determined using the random mutation.

Algorithm 1 deceptive mutation algorithm: determines mapping for h_i to reach h_j for kth mutation ($e^k_{i\to j}$)

```
detect WCC of h_i (WCC_i) using DFS on G(Λ)
include current eIP of all hosts in WCC_i in E_i
select K_i as the top γ hosts s.t. h_k ∉ WCC_i based on
transition scores.
if host h_j ∈ K_i then
    randomly choose e ∈ E_i and set e^k_{i→j} = e
else
    randomly choose e ∈ (E − E_i) and set e^k_{i→j} = e
end if
```

Transition scores from a host h_i to all other hosts such as h_j is determined beforehand. In order to calculate transition scores, we first need to establish *transition probability* between each two hosts. Parameter $p_{i,j}$ shows the transition probability from h_i to h_j and it is determined based on CVSS scores of all vulnerabilities of all running services of h_j which are accessible to h_i [18]. However, these transition probabilities only reflect immediate interest of attackers, but the transition score $y_{i,j}$ denotes the final steady-state probability that an attacker who resides in h_i ends up (*i.e.*, compromises or probes) in h_j. The initial transition score vector is denoted as $\boldsymbol{y_i^0}$, where $\boldsymbol{y_i^0}[i] = 1$, and $\boldsymbol{y_i^0}[j] = 0, j \neq i$; *i.e.*, initially the attacker is only in h_i. We define the matrix $\boldsymbol{P}_{n\times n}$ as the transition probability matrix where $\boldsymbol{P}[i,j] = p_{i,j}$. To accurately calculate steady-state transition scores, we use a modified version of Google PageRank algorithm [15]. Let $\boldsymbol{y_i^l}$ be a column vector representing the transition score of the hosts after l^{th} repetition. The steady-state transition score is determined by iterative calculation of $\boldsymbol{y_i^{l+1}}$ until the vector stabilizes [15]:

$$\boldsymbol{y_i^{l+1}} = \beta \boldsymbol{P}\boldsymbol{y_i^l} + (1-\beta)\boldsymbol{y_i^0} \tag{5}$$

where β is the damping factor [15].

3.2.1 Strategy Selection

Appropriate strategy is chosen as a trade-off between performance and security. While deceptive mutation inflicts more computational overhead than random mutation, it is more detrimental to persistent attackers (refer to Section 6). Therefore, the IP mutation takes place according to the random strategy, unless we observe abnormal inflation in the size of connected components in the network connectivity graph.

We define Δ as a random variable of which all normal WCCs are observations. We denote the mean and standard deviation of Δ as $\mu(\Delta)$ and $\sigma(\Delta)$ respectively. Our observation of the data collected in a /16 network in a period of 7 days shows that the distribution of Δ can be satisfactorily modeled as *normal* for appropriate choices of sampling intervals (We used 30 minute intervals in our work). This result is also confirmed by [7] for worm propagation.

The three-sigma rule states that for a normal distribution, nearly all values (99.7%) lie within 3 standard deviations of the mean. We use the three-sigma rule as the basis for strategy selection. Specifically, after each sampling interval, the controller establishes the graph and determines all WCCs. Then it selects the random mutation strategy, if and only if the following condition holds for each and every WCC w_k of current graph:

$$\forall w_k \in W_i, |w_k| \leq \mu(\Delta) + 3\sigma(\Delta) \tag{6}$$

Otherwise if any WCC violates this condition, the controller selects the deceptive strategy. Note that although component inflation may demonstrate propagation of a threat in the network, it might as well results from activities of benign users. However, since such false positives only change the IP mutation strategy of our technique, its does not have any effect on benign users. This is, in fact, a basic difference between our proactive approach and intrusion detection techniques.

4. ARCHITECTURE AND PROTOCOLS

The architecture of our approach for deployment on legacy networks is depicted in Figure 2. The deployment is accomplished via two components: gateways and controller. Gateways are distributed components which are located in front of subnet switches and perform rIP to/from eIP translations according to the recent mappings announced by the controller. For DNS responses, it replaces each rIP with

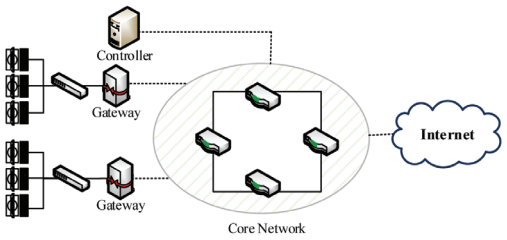

Figure 2: The architecture for deployment of our approach on legacy networks

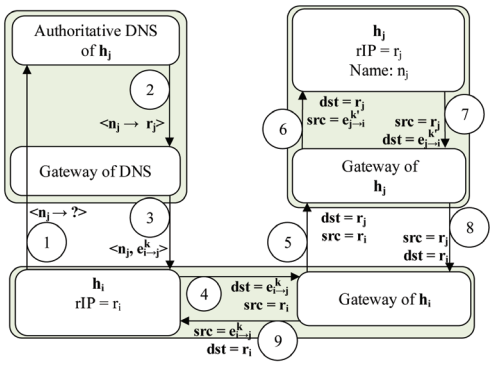

Figure 3: The protocol for communication via name

its corresponding eIP. Controller, on the other hand, is the central vintage point of the network. It determines new mappings for each and every host and announces them to the corresponding gateways.

Our scheme can also be implemented on software-defined networks in a straightforward manner. For SDN, the controller responsibilities are implemented in the SDN controller, and the translations are performed by associating actions with installed flows in OF-switches.

Specifically, gateways have four major responsibilities: (1) performing rIP-eIP translations for communications, (2) detection, filtering and reporting of illegitimate connections to the controller, (3) reporting connectivity records to the controller at designated interval, and (4) updating DNS replies to reflect mutations. Controller has three major responsibilities: (1) determining the IP mutation strategy based on connectivity records, (2) determining mapping and distributing them to appropriate gateways, and (3) authorization of rIP accesses based on access control policy of the network.

A host can be reached either via name or real IP address (rIP). However, only authorized entities such as administrators are allowed to communicate with a host via rIP. When a gateway receives a packet destined to rIP of a host, it authorizes the flow according to its access control policy.

To access host h_j using its name at time t, a source host h_i sends a query to the DNS. As depicted in Figure 3, DNS performs the standard translation (*i.e.*, resolves n_j to r_j) and sends the DNS reply to h_i. The gateway of the subnet where DNS resides in modifies the DNS reply and replaces the provided r_B with $e^k_{i\to j}$ where k denotes the last mutation interval, and provides it to h_i. The gateway also modifies the TTL value of DNS reply based on the expiration time associated with this mapping.

Host h_i uses this eIP to communicate with h_j. Upon receipt of the initial packet, the source gateway authorizes it. If h_i uses an illegitimate IP, translation will fail and the flow is dropped. Otherwise, gateway translates $e^k_{i\to j}$ to r_j and forwards the packet. The gateway performs the reverse translation for outbound packets of the flow. The destination gateway also performs the same translation for h_i.

Note that *no* session is broken as a result of temporal mappings. Although a rIP is mapped to two different eIPs at two different times, but the sessions which are using former eIPs will be continued as before.

5. IMPLEMENTATION

To study and demonstrate the feasibility of our approach, we implemented it in a designated class C network within university campus network. The network was divided into two subnets, and each subnet included 2 hosts and one gateway. The authoritative DNS (*BIND* package on Ubuntu) was located in a separate subnet and behind a gateway. The first subnet included two machines hosting a Web Server, and an FTP server. The second subnet only included clients. Temporal mutation interval was set to 10 minutes. We performed several network activities during mutation including downloading files from FTP server, and web browsing. Availability of these services was not affected and long-lived connections functioned soundly and accurately.

To evaluate effectiveness of our approach for large-scale attacks, we implemented it on emerging software-defined networks (OpenFlow [14]). We implemented the algorithms in POX [13], a network SDN controller written in Python that communicates with OpenFlow 1.0 switches. In our implementation on SDN, the POX controller performs responsibilities of the controller while IP translations are performed by associating actions with flow entries in OF-switches. The network was simulated using a random topology generator (based on Erdos-Rényi model) on Mininet [12]. The flexibility and scalability of mininet allowed us to investigate several attack scenarios including worm propagation and advanced persistent threat in relatively large networks.

6. EVALUATION

In this section we evaluate the effectiveness and overhead of our approach against various threat models.

6.1 Assumptions and Metrics

6.1.1 Attacker Model

In our evaluation, we consider two types of attackers: (1) naive non-adaptive attacker who is either uninformed about our approach, or is incapable of adapting to it; and (2) adaptive attacker, who knows about the dynamics of out network, and adapts the attack to its dynamics.

6.1.2 Metrics

MTD approaches defeat the attack either by deterring it, deceiving the attacker, or by increasing attacker's detectability. Therefore, to evaluate theoretical and practical effectiveness of our approach we introduce the following metrics:

- *Deterrence (Π)* metric quantifies the cost exerted on the attacker in terms of time by comparing completion time of an attack in our network (referred to as MTD network) and legacy networks (referred to as static networks). Assume the attack completion time in MTD and static networks are denoted as T_m and T_s respectively. The deterrence metric is calculated as $\Pi = T_m/T_s$.

- *Deception (Ω)* metric quantifies the ratio of targets that the attacker misses due to deception. Specifically, it denotes the percentage of resources that evade the attack by deceiving attackers about the attack outcome. If the total number of resources are denoted as N and the total number of undiscovered resources is denoted as M, the deception is quantified as $\Omega = N/M$. Note that $\Omega \in [0, 1]$.

- *Detectability (Ψ)* metric compares the rate of committing illegitimate actions (*e.g.*, probe to nonexisting destinations) during attack in MTD network and static networks. Assume R_m and R_s denote rates of committing illegitimate actions by the attacker in MTD and static networks respectively. Detectability is calculated as $\Psi = R_m/R_s$.

6.1.3 Network Settings

We assume that the network consists of n hosts, m IP addresses and s subnets, such that each subnet includes m/s IP addresses and n/s hosts. Each host is allowed (by network access control policy) to reach $n\phi$ other hosts, such that α percent of these $n\phi$ hosts' rIPs are located in the same subnet, and the remaining $1-\alpha$ percent are located in other subnets. We also assume that scanning each IP address on average requires τ seconds.

6.2 Mitigation against IP Sharing/Mapping

The effect of temporal mutation on defeating IP sharing techniques such as hitlist worms has been studied in the literature [3,9]. However, effectiveness of temporal mutation depends on the interval between IP gathering and attack. If this interval is short then the temporal mutation will fail to defeat IP sharing. However, spatial mutation completely defeats IP sharing because the IP list will become invalid after sharing.

Disruption of IP sharing has significant repercussions for many attack classes. For example, consider systematic honeynet mapping techniques which aim to systematically identify and blacklist the addresses that are monitored [6]. Our approach completely thwarts honeynet mapping attacks, because due to individualized views of the network, an unused IP from one host's perspective may be considered used from the perspective of another host.

Our approach defeats IP sharing techniques via deception. Assume host h_i shares a list of ν IP addresses with host h_j. Since each host can uniformly access $n\phi$ network hosts, the average number of unreachable IPs will be $\nu' = (1-\phi)\nu$. For random spatial mutation, the probability that an IP address that h_i uses to reach any host is included in the provided list is ν'/m. If h_j uses these ν IP addresses to reach the destinations, the probability that k IP addresses (out of ν) are also valid for h_j follows binomial distribution $B(\nu, \nu'/m)$. So, the expected number of shared hosts between h_i and h_j is $\frac{\nu\nu'}{m}$ where $\nu' \leq \nu \leq n$. Therefore for any two hosts that share ν IP addresses, the deception Ω is quantified using the following formula:

$$\Omega = \frac{\nu - \frac{\nu^2(1-\phi)}{m}}{\nu} = 1 - \frac{\nu(1-\phi)}{m} \tag{7}$$

Figure 4 shows the deception exerted on this threat model model. Note that almost all network hosts evade the attack as a result of spatial mutation. However, as the address space size increases, Ω approaches 1.

6.3 Mitigation against Scanners and Worms

Scanners and worms (or any other automated malware) use various scanning techniques to identify vulnerable hosts. In the following sections, we study effectiveness of our approach against cooperative and local-preference scanning worms both for naive attackers. However, note that in addition to the scanning techniques investigated here, all dynamics-aware scanning techniques are susceptible to our approach. For instance, histlist worms [3], or flash worms [21] are easily thwarted because IP addresses are individualized and short-lived.

6.3.1 Cooperative Scanners

Cooperative scanners aim to minimize the amount of generated traffic by minimizing repeated probing of the same IP address. Spatial mutation vector falsifies cooperative scanning assumption, because two scanners at two different locations will reach a host via two different IP addresses. Furthermore, the deceptive mutation approach severely reduces propagation of cooperative worms by restricting the growth of weakly connected components of the network.

Theoretical Analysis: Initially, we ignore the effect of temporal mutation for simplicity, and only study the effect of random spatial mutation on a cooperative worm. Note that a host in our network may be scanned by none of the cooperative scanners because an eIP scanned by some scanner A actually is used by another scanner B to reach a target. Assume q scanners are scanning the address space such that each scanner can reach $n\phi$ hosts and probes m/q addresses. If all hosts are uniformly distributed in the address space, each scanner will discover $n\phi/q$ hosts, and so the probability that a host is missed by a scanner is ϕ/q. The probability that a host is missed by all scanners is $(1-\frac{\phi}{q})^q = e^{-\phi}$ and therefore:

$$\Omega = e^{-\phi} \tag{8}$$

The worst case scenario occurs when $\phi = 1$; *i.e.*, each host is allowed to reach all hosts. Even in this case, at least 37% of hosts are saved due to random spatial mutation. This is the minimum proportion because even more are saved by deceptive and temporal mutations.

Eq. 8 also represents the deception that is exerted on cooperative worms. To prove this, we use mathematical propagation modeling. Assume $S(t)$, and $I(t)$ represents the susceptible, and infected populations at time t respectively. Parameter η denotes the scanning rate of each scanner. In a legacy network, we have $dI(t)/dt = \frac{n\phi}{m}\eta I(t)$ [1]. For MTD network, assume $Q(t)$ represents the quarantined population at time t; *i.e.*, susceptible hosts that they are not accessible to scanners as a result of mutation. With respect to infected population, the probability that a probe targets a host is $\frac{((n-Q(t))\phi)^2}{m}$ and the probability of probe success

is $(n - Q(t))\phi$. Therefore:

$$\frac{dI(t)}{dt} = \frac{(n - Q(t))\phi}{m}\eta I(t) \quad (9)$$

Without temporal mutation, a host might be quarantined because an eIP which is unused for one scanner is considered used for another one. For spatial mutation, on average every $(n - I(t))\phi$ unsuccessful scans move one host to quarantined state with probability $\frac{((n-I(t))\phi)^2}{m}$. The probability emanates from the fact that some of the hosts may already be infected, and so only the portion of hosts which are not infected must be considered. Therefore, the number of quarantined hosts due to spatial mutation is:

$$\frac{dQ(t)}{dt} = \frac{(n - I(t))\phi}{m}\eta I(t) \quad (10)$$

Numerical analysis of these equations (using Matlab Symbolic Math Toolbox) shows that as t grows $I(t)$ approaches $(1 - e^{-\phi})n$, while $Q(t)$ approaches $ne^{-\phi}$ which is consistent with Eq. 8. Figure 5 shows the theoretical infected population growth for various ϕ and only considering random spatial mutation. Note that the deception for legacy networks is 0 (all hosts get infected). For MTD network, as the percentage of reachable hosts (ϕ) decreases, deception increases. For example, if every host on average can reach one third of other hosts in the network, then at least (*i.e.*, only with random mutation) 72% of network hosts will evade detection.

Experimental Analysis: Our analysis so far only considers spatial mutation. Due to complications raised from inter-dependency of temporal and deceptive spatial mutations, we use experimentation to study the complete effect of our approach on non-adaptive cooperative worms. Figure 6 shows propagation of cooperative worms in static and MTD networks with $n = 2^{10}$ and $m = 2^{16}$. Note that while random spatial mutation alone rescues up to $e^{-\phi}$ of network hosts, the number of saved hosts are more in practice due to effect of temporal and deceptive spatial mutation. Our experimentation shows that for $\phi = 1/3$, $\Omega = 0.82$. However, practical deception depends on λ (average mutation interval), because as the mapping changes more frequently, more and more hosts will be saved, especially after the size of scanned address space exceeds the size of unscanned address space.

6.3.2 Local-Preference Scanning

Local-preference scanning is a dynamics-aware scanning approach which enhances possibility of evasion, because local traffic is usually less prone to inspection. Many types of recent worms (*e.g.*, Stuxnet, TDSS [16]) use local scanning as a propagation vector.

Theoretical Analysis: Our approach diminishes the distinction between local and non-local IPs. Therefore, number of illegitimate scans generated by a local-preference worm in the MTD network is significantly more than that of static networks. Assume a local-preference scanner which scans a local IP address (in the same physical subnet) with probability β. In a static network, the probability that a probe is hit is the probability that a local scan is hit plus the probability that a non-local scan is hit. Therefore, the probability that a probe is missed is:

$$p_s = 1 - \beta \cdot \frac{\alpha n\phi}{m/s} - (1 - \beta) \cdot \frac{(1 - \alpha)n\phi}{m - m/s}$$

In the MTD network, since eIP addresses are randomly chosen from address space the probability that a probe is missed is $p_m = 1 - n/m$.

If a local-preference scanner issues W probes per second, the probability that w probes are missed follows binomial distribution $B(W, p_s)$ and $B(W, p_m)$ for static and MTD networks respectively. Therefore, the average rate of issuing missed probes when a scanner scans W addresses/second will be $r_s = W \cdot p_s$ and $r_m = W \cdot p_m$. Assuming the same scanning rate for both networks, we have:

$$\Psi = \frac{r_m}{r_s} = \frac{p_m}{p_s} \quad (11)$$

Figure 7 denotes detectability for networks with $n = 2^{10}$, and $m = 2^{16}$. Note that our approach can increase attacker's detectability up to 5 times. Also note that as the local scan probability (β) increases, attackers's detectability increases. Moreover, note that if the percentage of locally reachable hosts (α) increases, attacker's detectability also increases.

The increased probe missing rate also deters the attack by increasing attack completion time. For a static network, a local-preference scanner requires $\frac{n\phi}{(1-p_s)}$ scans to detect all reachable hosts. For MTD network, the attacker requires $\frac{n\phi}{(1-p_m)}$ scans to this aim. Therefore:

$$\Pi = \frac{1 - p_s}{1 - p_m} \quad (12)$$

For example, for $n = 2^{10}$, $m = 2^{16}$, $s = 2^6$, $\phi = 1$, $\alpha = 0.75$ and $\beta = 0.9$, we have $\Pi = 51.58$.

6.4 Advanced Persistent Threat

Advanced persistent threat (APT) refers to a broad class of attacks which follow a long-term pattern of sophisticated hacking aimed at compromising a specific target. The targeting is conducted through continuous and low-and-slow monitoring and interaction in order to achieve the defined objectives. The attackers are usually external entities. Due to zoning and deployment of defense-in-depth strategies in networks, the attackers usually initiate their attack by compromising a publicly accessible host. They gradually continue compromising hosts, and moving from one host to another until they finally discover a path to the target [8].

In legacy networks, as the adversary gathers more information and compromises new hosts, the search space shrinks and her knowledge about network enhances, which allows her to gradually move toward the target. However, in our MTD network all gathered information is invalidated by moving from one host to another. This invalidation (1) forces adversary to recollect information, and (2) obfuscate adversary's view of the network, making her to blindly attack already targeted hosts.

Theoretical Analysis: The effectiveness of our approach against APT depends on the attacker's adaptability. For a naive non-adaptive attacker it is very straightforward to show that our approach is effective, especially in terms of deterrence, because the attacker will keep randomly moving between network hosts, and the deceptive spatial mutation deteriorates the situation for the attacker. In contrast, an adaptive attacker will redo scanning whenever she moves to

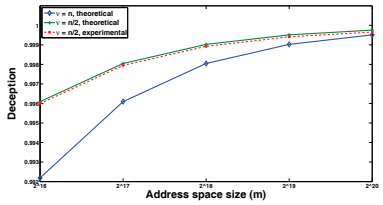

Figure 4: Theoretical and experimental deception against IP sharing/mapping, $n = 2^{10}$

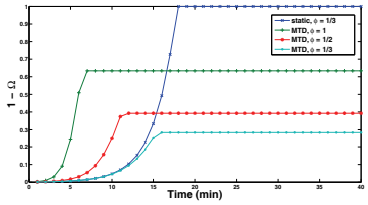

Figure 5: Theoretical deception against cooperative scanners, $n = 2^{10}$, $m = 2^{16}$

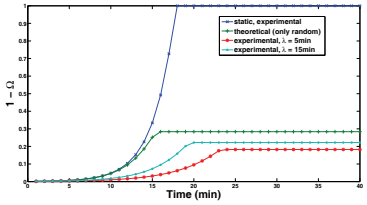

Figure 6: Experimental deception against cooperative scanners, $n = 2^{10}$, $m = 2^{20}$, $\phi = 1/3$

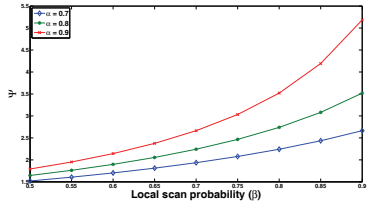

Figure 7: Detectability against local-preference scanners, $n = 2^{10}$, $m = 2^{16}$, $s = 2^6$, $\phi = 1$

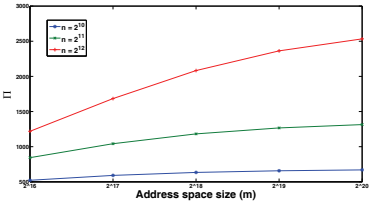

Figure 8: Theoretical deterrence against adaptive APT, $\phi = 0.5$, $\tau = 1$, $\varphi = \lambda = 300$

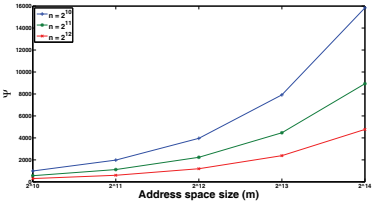

Figure 9: Theoretical detectability of adaptive APT attacker, $\phi = 0.5$, $\tau = 1$, $\varphi = \lambda = 300$

a new host, or stays in a host for a relatively long time. The effectiveness of our approach against adaptive APT can be measured in terms of deterrence and detectability.

For a static network, the probability that a host is detected from any random host is ϕ. The probability that a host is detected after M rounds of scans is $1 - (1 - \phi)^M$. Let $\hat{M}$ denote the minimum value for which this probability approaches 1:

$$\hat{M} = \arg\min_{M} \left((1 - \phi)^M < \delta \right) \tag{13}$$

where δ denotes a very small number such as 10^{-4}. Note that $\hat{M}$ denotes the number of rounds of scans required to ensure that all hosts are detected. For example, for $\phi = 0.5$, $\hat{M} \simeq 15$.

In static networks since IP addresses do not change, the attacker can easily distinguish between network hosts; *i.e.*, the attacker can keep track of visited hosts. Therefore, the probability that the attacker hits the target follows hypergeometric distribution. So, in the worst case scenario the attacker needs to visit all n hosts to discover the intended target. Assume each visit requires κ seconds. Therefore the maximum attack completion time will be $T_s = n\kappa + \hat{M}m\tau$.

For our MTD network, when the attacker moves to a new host, all previous scanning information is deprecated and the attacker has to redo scanning. The probability that the attacker hits the target after R steps is $1 - (1 - \frac{1}{n})^R$. Let $\hat{R}$ denote the value for which this probability approaches 1; *i.e.*, the target is definitely hit:

$$\hat{R} = \arg\min_{R} \left((1 - \frac{1}{n})^R < \delta \right) \tag{14}$$

For example, for $n = 2^{10}$, $\hat{R} = 10 * 2^{10}$. Moreover, if κ is greater than average mutation interval (λ) the attacker has to repeat the scan. In other words, if the attacker stays in a host for a relatively long time the temporal mutation invalidates scanning result and attacker has to redo scanning. Therefore, the attack completion time for our MTD network is:

$$T_m = \hat{R}(m\lceil\frac{\kappa}{\lambda}\rceil\tau + \kappa)$$

Therefore:

$$\Pi = \frac{\hat{R}(m\lceil\frac{\kappa}{\lambda}\rceil\tau + \kappa)}{n\kappa + \hat{M}m\tau} \tag{15}$$

Figure 8 compares the detterence exerted on such APT attacker for various network settings. Firstly note that our approach is significantly effective against APT, and depending on the network size, it can deter the attack from 500 to $2,500$ times. Secondly note that as the address space size and number of network hosts becomes larger, the deterrence increases because the amount of extra scans required for our MTD network increases significantly.

Our approach can also enhance attacker's detectability. Detectability is defined as the rate of committing illegitimate actions by the attacker. In a static network, the attacker requires at most $\hat{M}m$ scans. In our MTD network, the attacker requires at most $\hat{R}m\lceil\frac{\kappa}{\lambda}\rceil$. To minimize detectability, we assume that the attacker uses cooperative scanning and therefore the probe missing probability is $1 - \frac{n\phi}{m}$ for both networks. Therefore:

$$\Psi = \frac{\hat{R}m\lceil\frac{\kappa}{\lambda}\rceil/T_m}{\hat{M}m/T_s} \tag{16}$$

Figure 9 shows that depending on network size and number of hosts, our approach can increase detectability from $2,000$ up to $16,000$ times.

To evade detection, the attacker can increase T_m (attack duration) in order to decrease the rate of committing illegitimate actions. For instance, the attacker can prolong κ; *i.e.*, the time she stays in each host. Figure 10 denotes detectability for various κ. Note that increasing κ does not help the

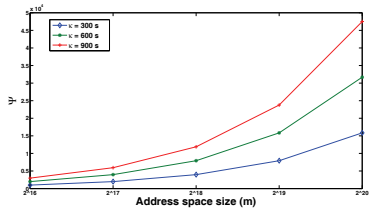

Figure 10: Theoretical detectability of non-adaptive and adaptive APT attackers, $\phi = 0.5$, $\tau = 1$, $n = 2^{10}$

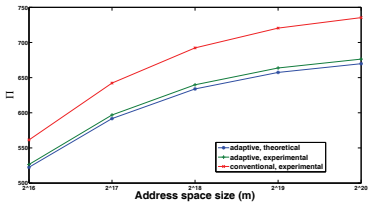

Figure 11: Theoretical and Experimental deterrence against APT, $\phi = 0.5$, $\tau = 1$, $\kappa = \lambda = 300$, $n = 2^{10}$

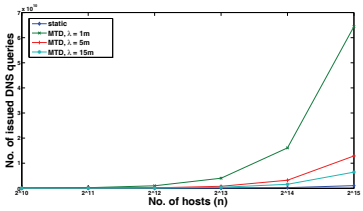

Figure 12: Average no. of DNS queries responded by auth. DNS in one hour

attacker since the temporal mutation vector of our approach invalidates previous scanning results and forces her to redo scanning. This is a very important result which shows that the interaction between temporal and spatial mutation vectors does not allow even persistent adaptive attackers to evade the detection by redesigning their attack strategies. In other words, the attacker has to either increase her detectability to expedite the attack, or waste her time to minimize her exposure. This trade-off between deterrence and detectability necessarily means that evasion from our approach is very challenging for the APT attacker.

Experimental Analysis: In theoretical analysis we assumed that the attacker is adaptive and after any spatial or temporal transition, she redoes reconnaissance again. In our experimental analysis, we confirm our result for adaptive APT attacker and also evaluate effectiveness for non-adaptive attackers. We use Monte Carlo method [10] to determine approximates for expected values of these variables. The details of this experimentation are omitted for brevity.

Figure 11 compares deterrence for both adaptive and non-adaptive attackers. Note that the experimental deterrence for adaptive attackers is very close to the theoretical result of Eq. 15. Also note that the deterrence which is exerted on non-adaptive APT attacker is even larger, because the deceptive as well as random spatial mutation vector of our approach constantly makes the attacker revisit already visited hosts. Therefore, the number of hosts visited by the non-adaptive attacker far exceeds $\hat{R}$ of Eq. 14.

6.5 Overhead

To determine the time complexity of our approach, note that algorithm 1 is the costliest part of our mutation algorithm. Determining WCCs requires $O((n\phi)^2)$ steps, since the graph includes at most $(n\phi)^2$ edges. For Eq. 5, the algorithm requires $O(\sqrt{\log n})$ rounds [15], and each round requires n multiplications. Therefore, the deceptive mapping of each host can be determined in $O((n\phi)^2)$. The random mapping for a host can be determined in $O(n\phi)$. In terms of computational overhead, although the results are not presented here for brevity, our implementation showed that both mutation strategies are scalable for large network sizes.

With regard to DNS overhead, note that spatial mutation causes a marginal overhead on the number of issued DNS queries in the network. However, temporal mutation results in renewal of eIP mappings forcing all clients to query the authoritative DNS of the network. Assume n hosts are querying the authoritative DNS. For a traditional network, since the TTL of DNS replies are high, in any long interval of T, at most n^2 queries are responded by the authoritative DNS. In the MTD network, the authoritative DNS must at most respond to $n^2\lceil\frac{T}{\lambda}\rceil$ queries in the same interval. Figure 12 compares average number of DNS queries in static and MTD networks with various average temporal mutation durations, in an interval of 60 minutes.

In terms of configuration overhead for deployment of our approach note that this approach is transparent to clients, network hosts, routers, switches, DNS servers, and DHCP servers. Since gateways are located behind access control devices such as firewalls and IDSs and close to subnet switches, the mutation is transparent to them as well.

7. CONCLUSION

In this paper, we presented a spatiotemporal host IP mutation technique to deter, deceive and/or allow detection of attacks by invalidating their assumptions about the network. The integration of both spatial and temporal mutations can effectively defeat various reconnaissance-based threat models, ranging from naive IP sharing techniques to sophisticated advanced persistent threat. Our approach can characterize the behavior of potential attackers and adaptively reconfigure IP mutations to confine propagation of the attack in the network. To effectively evaluate our technique we introduced three metrics, namely deterrence, deception and detectability. We used mathematical worm propagation modeling and probability theory to theoretically evaluate the effectiveness of our technique. We also used our implementations on both legacy networks and software-defined networks to perform evaluation for large-scale attack scenarios.

In future, we plan to study and include other potential mutation vectors and develop a game-theoretic model for adversarial modeling and strategy selection. Moreover, we will investigate the effectiveness of our approach against certain classes of DDoS attacks.

8. ACKNOWLEDGMENTS

This research was supported in part by National Science Foundation under Grant No. CNS-1352238. Any opinions, findings, conclusions or recommendations stated in this material are those of the authors and do not necessarily reflect the views of the funding sources.

9. REFERENCES

[1] E. Al-Shaer, Q. Duan, and J. H. Jafarian. Random host mutation for moving target defense. In *Proceedings of the 8th International Conference on Security and Privacy in Communication Networks*, Padua, Italy, 2012.

[2] M. Albanese, A. De Benedictis, S. Jajodia, and K. Sun. A moving target defense mechanism for manets based on identity virtualization. In *Communications and Network Security (CNS), 2013 IEEE Conference on*, pages 278–286, Oct 2013.

[3] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis. Defending against hitlist worms using network address space randomization. *Comput. Netw.*, 51(12):3471–3490, 2007.

[4] M. Atighetchi, P. Pal, F. Webber, and C. Jones. Adaptive use of network-centric mechanisms in cyber-defense. In *ISORC '03*, page 183. IEEE Computer Society, 2003.

[5] W. v. d. G. C. Contavalli. Client subnet in dns requests. `http://tools.ietf.org/html/draft-vandergaast-edns-client-subnet-00`, 2011.

[6] J.-Y. Cai, V. Yegneswaran, C. Alfeld, and P. Barford. An attacker-defender game for honeynets. In *Proceedings of the 15th Annual International Conference on Computing and Combinatorics*, pages 7–16. Springer-Verlag, 2009.

[7] M. P. Collins and M. K. Reiter. Hit-list worm detection and bot identification in large networks using protocol graphs. In *Proceedings of the 10th international conference on Recent advances in intrusion detection*, RAID'07, pages 276–295, Berlin, Heidelberg, 2007. Springer-Verlag.

[8] V. N. L. Franqueira. *Finding multi-step attacks in computer networks using heuristic search and mobile ambients*. PhD thesis, University of Twente, Enschede, 2009.

[9] J. H. Jafarian, E. Al-Shaer, and Q. Duan. Openflow random host mutation: Transparent moving target defense using software defined networking. In *Proceedings of HotSDN workshop at SIGCOMM'12*, Helsinki, Finland, 2012.

[10] R. E. Kass, editor. *Markov Chain Monte Carlo in Practice (Chapman & Hall/CRC Interdisciplinary Statistics)*. Chapman and Hall/CRC, 1 edition, Dec. 1995.

[11] D. Kewley, R. Fink, J. Lowry, and M. Dean. Dynamic approaches to thwart adversary intelligence gathering. In *DARPA Information Survivability Conference Exposition II, 2001. DISCEX '01. Proceedings*, volume 1, pages 176 –185 vol.1, 2001.

[12] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets '10, pages 19:1–19:6, New York, NY, USA, 2010. ACM.

[13] OpenFlow group at Stanford University. *POX Wiki*, 2013. `https://openflow.stanford.edu/display/ONL/POX+Wiki`.

[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008.

[15] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[16] K. security lab. Securelist. `http://www.securelist.com/en/analysis`, 2012.

[17] S. Stafford and J. Li. Behavior-based worm detectors compared. In S. Jha, R. Sommer, and C. Kreibich, editors, *Recent Advances in Intrusion Detection*, volume 6307 of *Lecture Notes in Computer Science*, pages 38–57. Springer Berlin Heidelberg, 2010.

[18] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. An attack graph-based probabilistic security metric. In *Proceeedings of the 22nd annual IFIP WG 11.3 working conference on Data and Applications Security*, pages 283–296, Berlin, Heidelberg, 2008. Springer-Verlag.

[19] J. Yackoski, P. Xie, H. Bullen, J. Li, and K. Sun. A self-shielding dynamic network architecture. In *MILITARY COMMUNICATIONS CONFERENCE, 2011 - MILCOM 2011*, pages 1381 –1386, nov. 2011.

[20] V. Yegneswaran and C. Alfeld. Camouflaging honeynets. In *In Proceedings of IEEE Global Internet Symposium*, 2007.

[21] C. Zou and D. Towsley. Routing Worm: A Fast, Selective Attack Worm Based on IP Address Information. *Workshop on Principles of Advanced and Distributed Simulation*, pages 199–206, 2005.