SAND2016-10509C

A Zoning Algorithm for Dynamic Cyber Zone Defense

Marci McBride
Sandia National Laboratories
Albuquerque, NM 87185
Email: mmcbri@sandia.gov

Robert Mitchell
Sandia National Laboratories
Albuquerque, NM 87185
Email: rrmitch@sandia.gov

***Abstract*—Attacks on cyber systems continue to plague public and private sector enterprises. While cyber zone defense is an appealing strategy to prevent, disrupt and tolerate these attacks, existing approaches assign hosts to zones based on their function (for example, printer zones and sensor zones) or place in the architecture (for example, corporate zones and demilitarized zones). This leaves the large number of human-operated commodity workstations within an enterprise unaddressed. We propose a dynamic zoning algorithm which periodically or asynchronously assigns hosts to zones based on peer requests made by their human operators. The proposed algorithm runs quickly on basic hardware for a large enterprise, and the zone statistics converge to values that match what simple mathematical models predict. We conclude that dynamic cyber zone defense calls for additional research and is a candidate for technology transfer.**

I. Introduction

Mitchell, et al. introduce the concept of cyber zone defense in [8] and [7]; the authors propose a math model and simulation in order to predict its effectiveness. While these studies are focused on creating homogeneous zones, we investigate a dynamic zoning technique for the large class of devices comprising the human-operated commodity workstations of an enterprise.

Dynamic zoning connects the concept of cyber zone defense to the realm of moving target defense (MTD). In particular, a dynamic cyber zone defense is a network-based MTD. Figure 1 illustrates where dynamic cyber zone defense and other network-based approaches fit in the MTD literature. Okhravi et al. proposed this MTD taxonomy in [9].

The two components of a threat model are the capabilities and goals of a notional attacker. Our threat model assumes the adversary can implant malware in a host within the subject network, thus creating an insider threat. We assume the primary goals of an attacker are data exfiltration and lateral movement. We also assume command and control (C2) is an intermediate objective.

The rest of this paper is organized as follows: First, Section II surveys the state of the art for this topic. Next, Section III formulates the problem and proposes an algorithm which solves the problem. Third, Section IV presents numerical data and figures that quantify the algorithm's performance. Finally, Section V contains our conclusions from the study and identifies future lines of research.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. Approved for unlimited release: SAND2016-XXXX C.

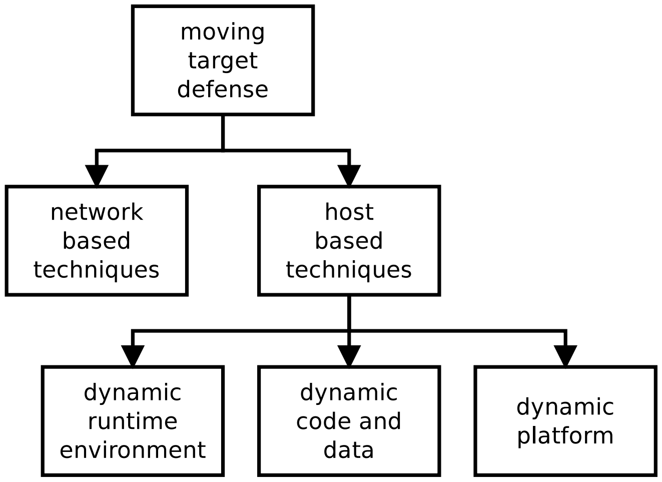

Fig. 1. Moving target defense taxonomy.

II. Literature Search

Calls for static cyber zone defense frequently come from the supervisory control and data acquisition (SCADA) community. Effendi and Davis survey the history of cyber attacks on industrial control systems (ICSs) and identify best practices for SCADA security in [4]. In particular, the authors advocate the familiar ICS security approach of establishing Internet-facing (Zones 1 and 2), demilitarized (Zone 3) and industrial (Zone 4) segments. They discuss man in the middle (MitM) attacks but do not provide any numerical data demonstrating the effectiveness of their best practices. In contrast, our investigation proposes a dynamic zoning strategy for enterprise networks faced with a specific threat model and provides quantitative data.

Within the realm of network-based MTD, researchers have pursued many approaches, ranging from the crowded topic of Internet Protocol (IP) address randomization [2], [12], [5], [13] to dynamic protocol stacks [10], [11].

Ehab Al-Shaer proposes an MTD architecture called Mutable Networks (MUTE) in [2]. MUTE is an MTD which allows hosts to change configurations randomly and dynamically. In [2], host configuration consists of IP addresses, routing tables and "fingerprint." The author defines fingerprint as an outsider's perception of a host's operating system (OS) and application load. Their threat model includes scanning, denial of service (DoS) attacks and botnets. This study includes a thought experiment and a brief formal analysis; Al-Shaer does not provide any numerical data. While we also propose a network-based MTD technique, our approach uses dynamic zone memberships and includes quantitative evidence.

Van Leeuwen, et al. propose a rich set of metrics for characterizing the impact of an MTD for the defender in [12]. The authors demonstrate how to calculate two of these metrics (usable network capacity and jitter) for a network-based MTD that uses IP randomization. They do not consider a specific threat model. Our work expands on [12] by proposing a network-based MTD that uses dynamic zone memberships, studies new defensive work factors and identifies a specific threat model.

Jafarian, et al. investigate a network-based MTD called OpenFlow Random Host Mutation (OF-RHM) in [5]. The authors' MTD is based on changing the IP addresses of hosts using OpenFlow-based Software Defined Networking (SDN). They measure consistency with ground truth (percentage) over time, worm propagation (host count) over time, size of IP address pool over mutation interval and average flow table size over session establishment rate; this is a rich mix of defender and attacker-based statistics. The threat model of Jafarian, et al. focuses on worms. While we also propose a network-based MTD technique, our study measures different defender-based statistics, uses dynamic zone memberships and considers a different threat model.

Wang, et al. propose another IP address rotation scheme in [13]. Like [5], the authors' network-based MTD is also implemented in OpenFlow. Their main contribution is to expand the pool of available IP addresses used in previous work. While Wang, et al. do instrument an experiment that proves the concept, the authors do not provide any numerical data.

Pohly, et al. investigate Dynamically Insertable Bumps in the Stack (DIBS) in [10]. The authors' design goals are transparency, flexibility and agility, and they do not consider a specific threat model. Pohly, et al. conduct a rigorous experiment as part of their investigation: The authors measure connection throughput (MB/s) and microprocessor load (percentage used) for a TCP/IP stack, a DIBS stack and a stack including a Virtual Private Network (VPN). While they study a dynamic protocol-based MTD on mobile networks, our work investigates dynamic zone memberships on wireline networks and has a specific threat model.

Kaliappa Ravindran studies a model for a network operator to manage multiple protocols in [11]. While the author's primary goal is to increase network performance, they demonstrate how the same solution can improve security with a case study. Ravindran does not consider a threat model. The author measures resource usage (unitless) for various levels of ambient hostility, utility (unitless) for various quality of service (QoS) settings and traffic latency (ms) and network overhead (message count) for various levels of traffic intensity. While we don't want our security technique to degrade the performance of the subject network, our primary goal is security, and we have a threat model.

Although it relies heavily on prior work, Kampanakis, et al. distinguish themselves from other IP randomization and dynamic protocol studies in [6]. The authors study two previously published SDN-based MTD techniques: Random Host Mutation (RHM) and Random Route Mutation (RRM). Their threat model focuses on reconnaissance; specifically, network mapping and software version detection. Kampanakis, et al. propose a mathematical model for the attacker cost of scanning. The authors report the total and differential amount of time and traffic Nmap required to determine service versions and OS. Our work uses a dynamic zone membership approach instead of mutating hosts or routes, measures defender-based metrics and considers a different threat model.

Two recent papers pursued dynamic platform technique (DPT) based MTD research that relates to our work:

Zhuang, et al. investigate an MTD framework in [14]. The authors consider attack graphs in their analysis. Their simulation uses a DPT-based MTD that simply resets virtual machines (VMs) periodically. Zhuang, et al.'s metrics of interest are percentage of successful attacks and attack prosecution time. We will incorporate attacker-based metrics such as these in an extension of this article.

Anderson, et al. study models to measure the effectiveness of DPT-based MTDs in [3]. The authors derive closed-form mathematical and stochastic Petri net (SPN) models to predict the probability of an attack's success. Our paper will lead to future work that will extend [3] with a closed-form mathematical model, SPN model and simulation that predicts the effectiveness of a network-based MTD. This will increase the coverage of predictive models for the MTD taxonomy illustrated in Figure 1.

III. Algorithms

This section discusses the metrics of interest, describes the problem at hand and proposes a solution to this problem.

A. *Metrics of Interest*

The metrics used to gauge the quality of our solution are the following.

1) average number of hosts assigned to a zone
2) average number of zones a host is assigned to
3) number of disruptions

Managing the number of hosts per zone is important because if any zone grows too large, this zone becomes a high value target (HVT). Likewise, managing the number of zones per host is important because if any host achieves too much access, this host becomes an HVT. The number of disruptions is a practical matter of interest. This metric can estimate the number of service calls the enterprise information technology (IT) manager can expect.

B. Formulation

Previous cyber zone defense work [8], [7] proposes limiting malware communication without interfering with useful work by organizing similar devices or activities into the same zone and by allowing only essential communication within and between those zones. We seek to address the large fraction of human-operated commodity hosts in the enterprise by allowing these humans to request additional network visibility on demand. Our proposed algorithm will broker these requests and grant the network visibility required to accomplish the mission while thwarting or disrupting cyber attackers.

C. Solution

Algorithm 1 fulfills peer requests and requires five inputs: reqs is the list of peer requests, mhpz is the maximum number of hosts per zone, mzph is the maximum number of zones per host, mz is the maximum number of zones and ad is the assignment dictionary. For ad, the zone identifier is the key and the value for each key is a list of hosts assigned to the zone. The algorithm iterates over every pair of hosts (p) in reqs. If the components of the pair are the same, the request is degenerate or if the hosts are already members of the same zone, then there is nothing more to be done and the algorithm continues on to the next request pair. If there are less zones than mz allows, the algorithm creates a new zone with the two hosts as members. (new_zone_id simply generates a unique new zone name.) Otherwise, the algorithm identifies the smallest zone and places the two hosts in this zone. For each host added to the smallest zone, if necessary, the algorithm will remove the oldest member of that zone and increase the disruption count. Note that the algorithm applies the first in, first out (FIFO) method by adding hosts to the end of the list for a specific zone and, when necessary, removing the oldest host in the zone.

Algorithm 2 aggregates some repetitive logic in Algorithm 1. This is the procedure for removing a host from one of the zones it is a member of, if necessary. It requires three inputs: host is the subject device, ad is the assignment dictionary and mzph is the maximum number of zones per host. If the host is a member of more zones than mzph allows, the algorithm finds the largest zone and removes the host from that zone.

Algorithm 1 Peer Request Fulfillment

```
function FULFILL_REQUESTS(reqs, mhpz, mzph, mz, ad)
    disruptions ← 0
    flag ← 0
    for $p$ in reqs do
        if $p_0$ == $p_1$ then
            continue
        end if
        for hl in ad.values() do
            if hl.count($p_0$) and hl.count($p_1$) then
                flag ← 1
                break
            end if
        end for
        if flag then
            flag ← 0
            continue
        end if
        if $|\text{ad.keys}()| <$ mz then
            ad.update(new_zone_id(ad): [$p_0$, $p_1$])
            continue
        end if
        smallest_zone_size ← MAX_ZONE_SIZE
        for zone in ad do
            if $|\text{ad}[\text{zone}]| <$ smallest_zone_size then
                smallest_zone ← zone
                smallest_zone_size ← $|\text{ad}[\text{zone}]|$
            end if
        end for
        for host in $p$ do
            if host not in ad[smallest_zone] then
                ad[smallest_zone].append(host)
            end if
            if $|\text{ad}[\text{smallest\_zone}]| >$ mhpz then
                ad[smallest_zone].pop(0)
                disruptions += 1
            end if
            remove_if_necessary(host, ad, mzph)
        end for
    end for
    return disruptions
end function
```

IV. Results

A. Method

We obtained a list of the host names for all network devices for an enterprise. We used an information technology management database and a human resources database to add human and department fields to this host name list. The resulting list of $<$ host name, human name, department name $>$ tuples was our raw data set. For the initial conditions, we assigned each host to one zone which corresponded with the department of its human owner. Our assumption was that people typically work with everybody in their department. To generate each day's peer requests, we simulated humans forming new collaborations at some rate. We assumed this rate was guided by an exponential distribution because this distribution describes the time between events that occur continuously and independently at a constant average rate. [1] A single parameter, λ, informs the exponential distribution. For each day's peer requests, we iterated over each human in the enterprise and used an exponentially-distributed random variable to determine if they began collaborating with another human that day. If so, we used uniformly-distributed random variables to determine which of their hosts needed to peer

Algorithm 2 Remove If Necessary

```
function REMOVE_IF_NECESSARY(host, ad, mzph)
    zones ← zone_memberships(host, ad)
    if $|zones| > mzph$ then
        largest_zone_size ← -1
        for zone in zones do
            if $|ad[zone]| >$ largest_zone_size then
                largest_zone ← zone
                largest_zone_size ← $|ad[zone]|$
            end if
        end for
        ad[largest_zone].remove(host)
    end if
end function
```

TABLE I
System parameters.

Parameter Name	Description	Source
$a_{\max}$	maximum hosts per zone	System Manager
$b_{\max}$	maximum zones	System Manager
$c_{\max}$	maximum zones per host	System Manager
h	number of hosts	Enterprise
u	number of humans	Enterprise
λ	new collaboration frequency	Enterprise
$\bar{a}$	average hosts per zone	Zoning Algorithm
$\bar{c}$	average zones per host	Zoning Algorithm
$\bar{d}$	average disruptions per day	Zoning Algorithm

with another and which of the other hosts in the enterprise this host needed to peer with.

B. Steady State Metrics

We hypothesized that the metrics of interest for a dynamically zoned enterprise approach some limit over time, and these steady state trends are important to us. Our simulations matched our theory; we found that such a system does converge. Figures 2 through 4 show how various input parameters effect where critical metrics for such a system converge. To generate the steady state data, we considered a number of parameterizations which simulated the proposed algorithm's performance for different maximum numbers of hosts per zone ($a_{\max}$), maximum numbers of zones ($b_{\max}$), numbers of hosts (h), numbers of humans (u), and new collaboration rates (λ). Our domain for λ extended from 1/day to 1/month. For each simulation, we randomly selected a subset of the enterprise hosts based on a uniform distribution and assigned each host's initial zone based on its human owner's department. The size of these subsets ranged from 2% to 80% in order to explore the domains of interest. For each day, we created a number of peering requests, passed these requests and the current zone assignments to the zoning algorithm and calculated $\bar{a}$ and $\bar{c}$. We simulated additional days until the these metrics converged.

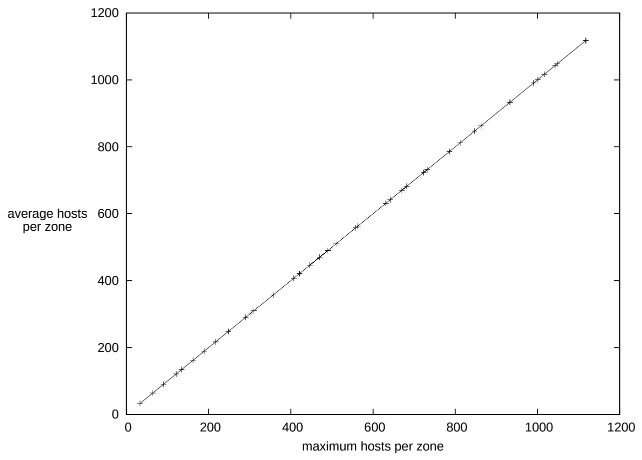

Fig. 2. Average hosts per zone versus maximum hosts per zone.

Figure 2 illustrates the impact of maximum hosts per zone on the average hosts per zone. As expected, when the system converges, $\bar{a}$ approaches $a_{\max}$. Figure 2 does not show separate curves for different settings of λ because λ does not impact steady state $\bar{a}$ in the domains of interest.

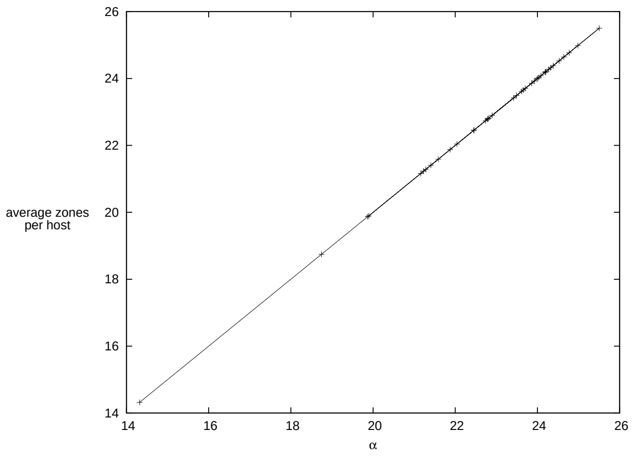

Fig. 3. Average zones per host versus α.

We found that the average zones per host approached a function of three system parameters which we call α. Specifically,

$$\alpha = \frac{b_{\max} \cdot a_{\max}}{h}$$

Figure 3 visualizes the impact of α on $\bar{c}$. When the system converges, $\bar{c}$ approaches α. This suggests that α is a useful math model for predicting the steady state $\bar{c}$ of a dynamically zoned enterprise. Figure 3 does not show separate curves for different settings of λ because λ does not impact steady state $\bar{c}$ in the domains of interest.

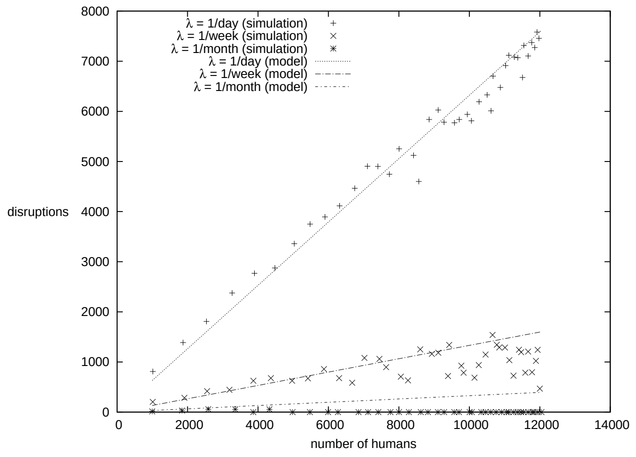

Fig. 4. Disruptions versus humans.

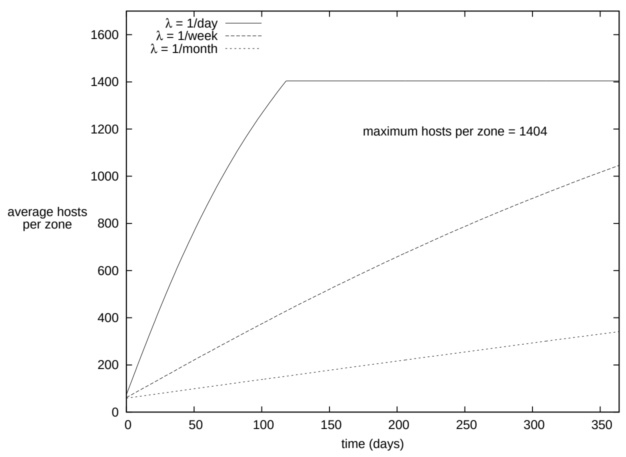

Fig. 5. Average hosts per zone over time.

We propose the following theoretical model for the average number of disruptions per day for a converged system:

$$\begin{aligned} \bar{d} &= u \cdot P(\text{peer request per human per day}) \\ &= u \cdot CDF(\lambda, 1) \\ &= u \cdot (1 - e^{-\lambda \cdot 1}) \\ &= u \cdot (1 - e^{-\lambda}) \end{aligned} \tag{1}$$

In other words, the average number of disruptions on a given day will be equal to the number of humans times the probability that a human will generate a peer request on that day. The first factor is u. The second factor is the cumulative distribution function (CDF) for an exponential distribution given $x = 1$. The CDF for an exponential distribution is $1 - e^{-\lambda x}$ which reduces to $1 - e^{-\lambda}$ given $x = 1$. Because peer requests are determined stochastically, the number of disruptions will not converge. Figure 4 shows that the simulations follow the trends predicted by this theoretical model. As the data for λ = 1/month show, this model is biased high; this is because peer requests will already be fulfilled by the existing zone assignments with some probability. We will remove this bias from $\bar{d}$ in future work.

C. Transient Metrics

While Figures 2 through 4 visualized the steady state trends for key metrics, Figures 5 and 6 show how quickly network parameters will converge. We do not show a figure for disruptions because this measurement moves little before $\bar{a}$ and $\bar{c}$ converge; however, $\bar{d}$ will converge as indicated in Figure 4 at the same time as $\bar{a}$ and $\bar{c}$. To generate the transient data, we used the full enterprise configuration and considered a number of parameterizations which simulated the proposed algorithm's performance for different new collaboration rates (λ). As with the steady state simulations, our domain for λ extended from 1/day to 1/month, and we assigned each host's initial zone based on its human owner's department. For each of 365 days, we created a number of peering requests, passed these requests and the current zone assignments to the zoning algorithm and calculated $\bar{a}$ and $\bar{c}$.

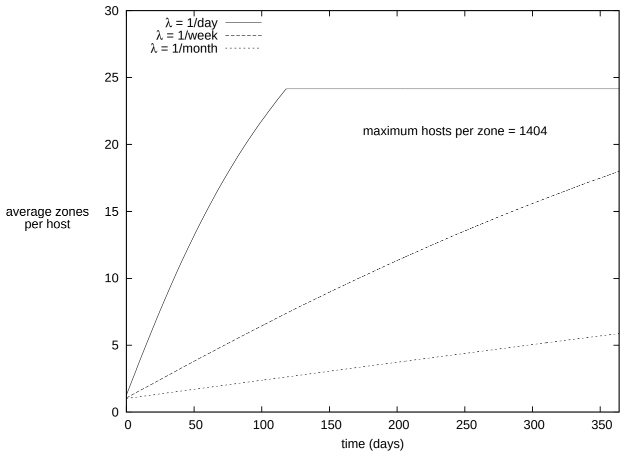

Fig. 6. Average zones per host over time.

We see that the zone statistics for this enterprise will stabilize after 118 days given λ is 1/day. For smaller settings of λ, statistics will take longer to converge but will do so at the same values.

V. Conclusions

Dynamic zones are an important extension to static homogeneous [8], [7] or functional [4] zones because they bring the benefits of cyber zone defense to the enterprise at large.

In future work, we will pursue a number of refinements and enhancements to this study. First, we will remove the bias from the current model of $\bar{d}$ by considering the probability that some peer requests will already be fulfilled. Also, we will consider additional metrics of interest. Specifically, new metrics will address the attacker-based security aspects of a dynamic zoning approach. Third, we will identify a more sophisticated algorithm that will remove existing zone members in a more elegant fashion (for example, based on priorities and least recent use). Finally, we will survey implementation options (for example, layer 2 versus layer 3 and centralized versus distributed approaches).

References

[1] https://en.wikipedia.org/wiki/Exponential_distribution.

[2] Ehab Al-Shaer. Toward network configuration randomization for moving target defense. In *Moving Target Defense*, pages 153–159. 2011.

[3] Nicholas Anderson, Robert Mitchell, and Ing-Ray Chen. Parameterizing Moving Target Defenses. In *New Technologies, Mobility and Security*, Larnaca, Cyprus, November 2016.

[4] Asif Effendi and Robert Davis. ICS and IT: Managing Cyber Security Across the Enterprise. In *Society of Petroleum Engineers Middle East Intelligent Oil and Gas Conference and Exhibition*, Abu Dhabi, UAE, September 2015. Society of Petroleum Engineers.

[5] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. Openflow random host mutation: transparent moving target defense using software defined networking. In *First workshop on Hot topics in software defined networks*, pages 127–132, Helsinki, Finland, August 2012. ACM.

[6] Panos Kampanakis, Harry Perros, and Tsegereda Beyene. SDN-based solutions for Moving Target Defense network protection. In *IEEE 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–6, Sydney, Australia, June 2014. IEEE.

[7] Robert Mitchell and Paul Sery. Refining the Foundations for Cyber Zone Defense. In *New Technologies, Mobility and Security*, Larnaca, Cyprus, November 2016.

[8] Robert Mitchell, Paul Sery, and Tom Klitsner. Foundations for Cyber Zone Defense. In *International Conference on Computer Communication and Networks*, Waikoloa, Hawaii, USA, August 2016.

[9] Hamed Okhravi, MA Rabe, TJ Mayberry, WG Leonard, TR Hobson, D Bigelow, and WW Streilein. Survey of Cyber Moving Target Techniques. Technical report, DTIC Document, 2013.

[10] Devin J Pohly, Charles Sestito, and Patrick McDaniel. Adaptive protocol switching using Dynamically Insertable Bumps in the stack. In *IEEE Military Communications Conference*, pages 342–347, Tampa, FL, USA, October 2015. IEEE.

[11] Kaliappa Ravindran. Management Software for Protocol-level Adaptations in Dependable Network Services. In *IEEE International Parallel and Distributed Processing Symposium Workshops*, pages 1288–1297, Chicago, IL, USA, May 2016. IEEE.

[12] Brian Van Leeuwen, William MS Stout, and Vincent Urias. Operational Cost of Deploying Moving Target Defenses Defensive Work Factors. In *IEEE Military Communications Conference*, pages 966–971, Tampa, FL, USA, October 2015. IEEE.

[13] Shaolei Wang, Lei Zhang, and Chaojing Tang. A new dynamic address solution for moving target defense. In *IEEE Information Technology, Networking, Electronic and Automation Control Conference*, pages 1149–1152, Chongqing, China, May 2016. IEEE.

[14] Rui Zhuang, Su Zhang, Scott A DeLoach, Xinming Ou, and Anoop Singhal. Simulation-based approaches to studying effectiveness of moving-target network defense. In *National symposium on moving target research*, pages 1–12, Annapolis, MD, USA, June 2012.