A Moving Target Defense Approach to Mitigate DDoS Attacks against Proxy-Based Architectures

Sridhar Venkatesan*, Massimiliano Albanese*, Kareem Amin† Sushil Jajodia*, and Mason Wright†
* Center for Secure Information Systems
George Mason University, 4400 University Drive, Fairfax, VA 22030, USA
Email: {svenkate,malbanes,jajodia}@gmu.edu
† Division of Computer Science and Engineering
University of Michigan, 2260 Hayward Street, Ann Arbor, MI 48109, USA
Email: {amkareem,masondw}@umich.edu

Abstract—**Distributed Denial of Service attacks against high-profile targets have become more frequent in recent years. In response to such massive attacks, several architectures have adopted proxies to introduce layers of indirection between end users and target services and reduce the impact of a DDoS attack by migrating users to new proxies and shuffling clients across proxies so as to isolate malicious clients. However, the reactive nature of these solutions presents weaknesses that we leveraged to develop a new attack – the *proxy harvesting attack* – which enables malicious clients to collect information about a large number of proxies before launching a DDoS attack. We show that current solutions are vulnerable to this attack, and propose a moving target defense technique consisting in periodically and proactively replacing one or more proxies and remapping clients to proxies. Our primary goal is to disrupt the attacker's reconnaissance effort. Additionally, to mitigate ongoing attacks, we propose a new client-to-proxy assignment strategy to isolate compromised clients, thereby reducing the impact of attacks. We validate our approach both theoretically and through simulation, and show that the proposed solution can effectively limit the number of proxies an attacker can discover and isolate malicious clients.**

I. INTRODUCTION

In recent years, Distributed Denial of Service (DDoS) attacks have emerged as one of the biggest threats to Internet security, with attack volumes reaching 500 Gbps [1]. Moreover, with the pervasiveness of botnets and the declining cost of launching an attack (e.g., a 1-week DDoS can cost as low as \$150 [2]) the risk of DDoS attacks has dramatically increased.

Volumetric or bandwidth-based DDoS attacks disrupt the availability of a service to its legitimate users by flooding the target with large volumes of traffic that result in congesting the target's network infrastructure. While attackers are constantly developing new types of DDoS attacks (such as multi-vector attacks and application-layer attacks), volumetric attacks continue to be amongst the most common type of attack [1]. These attacks leverage the fact that the target's location can be easily identified (e.g., through DNS lookup), allowing the attackers to directly reach the target. A possible approach for preventing an attacker from directly reaching a target is to introduce indirection layers between users and target services [3], [4]. In this case, in order to access a protected service, users need to authenticate with an intermediate layer of systems acting as proxies between the clients and the services, and relay incoming requests to the servers. Additionally, protected services accept traffic only from designated systems in the indirection layer. Although these solutions do not require Internet-wide adoption to mitigate DDoS threats to a given enterprise, the indirection layer needs to be well-provisioned to effectively out-muscle the attacks. However, as we cannot indefinitely add resources to the indirection layer and keep adding proxies, we need to investigate how to increase the complexity for the attacker without further increasing the number of resources the defender has to deploy.

This work was partially supported by the Army Research Office under grants W911NF-13-1-0421 and W911NF-13-1-0317, and by the Office of Naval Research under grant N00014-13-1-0703.

Before launching a DDoS attack, an attacker needs to collect information about the *entry points* to the target system (e.g., the IP addresses of targets or proxies). Existing defensive mechanisms focus on curbing the persistence of an attack while making it relatively easy to discover these entry points. As a result, these techniques *react* to a DDoS attack by, for instance, filtering the attack traffic. Therefore, the attack surface of the protected service remains static before the onset of an attack. In order to introduce dynamism to the attack surface and increase the overall effort for the attacker, Moving Target Defense (MTD) techniques have emerged as a new cyber defense paradigm [5], [6]. These techniques provide a principled approach to shift or change a system's attack surface so as to introduce uncertainty for the attackers, thereby compromising their ability to plan effective attacks.

Existing MTD-based architectures [7], [8], [9] leverage cloud platforms to host a small set of active proxies. Incoming connection requests are validated by a well-provisioned lookup server which then redirects each authorized user to one of the secret active proxies to serve the user's subsequent requests. When under attack, a central server instantiates new proxies and clients associated with attacked proxies are *moved* to these newly instantiated proxies. To weed out compromised clients (assumed to be insiders), existing architectures shuffle the clients before assigning them to new proxies such that the insiders are eventually isolated from the innocent clients. Existing architectures assume that insiders persist in the system during an attack and hence, initiate the movement only *after* the attack has occurred. Such a reactive mechanism presents weaknesses that can be exploited to diminish the protection offered by these architectures.

To illustrate current limitations, we first present a new type of attack – the *proxy harvesting attack* – which can be used to collect information about a possibly large number of proxies before launching a massive DDoS attack against all known proxies. We show that state-of-the-art solutions are vulnerable to this attack. Next, we present a simple attacker isolation technique, BIND-SPLIT, to counter proxy harvesting attacks. We prove that, under the described attack model, the number of users that are affected by a DDoS attack can be minimized when a system is utilizing BIND-SPLIT. One limitation of BIND-SPLIT is that it requires the lookup server to maintain a static mapping between clients and proxies, which may lead to an uneven load distribution across proxies, thereby adversely affecting a system's performance. To address this issue, while simultaneously reducing the impact of DDoS attacks, we propose a lightweight defense technique, PROTAG, which proactively reconfigures the attack surface by periodically replacing one or more proxies and reassigning clients to new proxies. Finally, we show how BIND-SPLIT and PROTAG can be combined in a hybrid approach which integrates proactive cyber defense and attacker isolation.

The paper is organized as follows. Section II discusses relevant related work, whereas Section III describes the threat model we assume in our work. Section IV examines the limitations of current approaches and introduces the proxy harvesting attack. We then present our solution to address these limitations, by discussing BIND-SPLIT in Section V, PROTAG in Section VI, and the proposed insider isolation algorithm in Section VII. Finally, we present experimental results in Section VIII and give concluding remarks in Section IX.

II. Background and Related Work

Since the first appearance of DDoS attacks in 2000 [10], several defense mechanisms based on filtering attack traffic or limiting a client's bandwidth share have been proposed to mitigate attacks at the Internet scale [11]. For an in-depth survey of DDoS flooding attacks and existing approaches to mitigate them, we refer the reader to [12]. Unfortunately, in order to be effective, these solutions rely on large-scale adoption and coordination among different network elements. These limitations promoted overlay-based defense approaches that hide the location of target servers behind a well-provisioned, distributed overlay network [13]. For instance, MOVE [14] protects services against attackers who control and disrupt only a subset of network elements. In MOVE, target services accept traffic only from a subset of overlay nodes (the *secret servlets*) and when a DoS attack is detected, the target service is migrated to a new host to mitigate the impact of the attack.

Recently, Wang *et al.* [8] designed MOTAG to protect web services by hiding the application server's location behind a large pool of proxy servers. Originally designed to support services that require client authentication and later extended to support anonymous users [7], MOTAG leverages the on-demand availability of resources in a cloud environment to spawn new proxy servers when attacked. To limit the impact of an attack, MOTAG migrates clients to new proxies and shuffles the client-to-proxy assignment to isolate insiders who divulge the location of the secret proxy servers. However, the shuffling process employed by MOTAG to isolate insiders does not consider the overhead associated with instantiating and maintaining new proxies. To address this issue, Wood et al. [9] proposed DoSE, a cloud-based architecture that provides a cost-effective mechanism to isolate insiders and confine an attack to a few proxies. In DoSE, each client is associated with a risk value which captures the likelihood that the client will indulge in a DoS attack. Additionally, each proxy server has an upper bound on the risk that it can tolerate. During an attack, DoSE assigns clients to proxies based on the corresponding risk parameters and updates the risk value of clients associated with attacked proxies. Similar to MOTAG, DoSE instantiates new proxies to reduce the impact of the attack and migrates victims to new proxies based on their updated risk values. By maintaining a state for each client, DoSE limits the number of proxies needed to identify insiders, thereby reducing the cost to maintain such an architecture.

Besides overlay-based approaches, *moving target defenses* have been employed to limit the impact of DoS attacks in other settings. For instance, Duan et al. [15] proposed a proactive technique to minimize packet loss due to DoS attacks by randomly choosing a routing path which has minimal overlap with recently used routing paths. Similarly, Herzberg and Shulman [16] present a challenge-response mechanism to mitigate DNS-amplification DoS attack. In this scheme, the local resolver is challenged to resolve the domain name from a verifier whose IP address is chosen at random from an address block. The verifier resolves the domain name only if it receives the request from the local resolver within a predefined time interval. Although these techniques are effective against DDoS, they are limited by their assumptions on either the attackers capabilities or the employed attack vectors (e.g., DNS amplification). In this paper, we consider a more general threat model and design a moving target defense architecture to mitigate bandwidth-based DDoS attacks under this model.

III. Threat Model

In this section, we briefly describe the threat model we consider in our work. We primarily focus on protecting Internet services that require client authentication, such as online banking and e-commerce portals. In our threat model, the attacker employs bandwidth-based or volumetric DoS attacks – such as SYN flood and DNS amplification – to disrupt the availability of target services. We assume a persistent attacker possessing sufficient capabilities (e.g., controlling a well-provisioned botnet) to simultaneously attack multiple proxies and any subsequent new proxy that may be spawned to mitigate the impact of the attack. Such repeated attack behavior using botnets is one of the emerging trends in DDoS attacks [1], [17]. In a typical attack, bots spoof the IP addresses of traffic generated to congest network links of the target network. However, we assume that the attacker does not have sufficient capabilities to congest backbone links such as ISP networks or cloud infrastructures. In addition to leveraging a botnet, attackers may compromise the credentials of legitimate clients or eavesdrop on legitimate client's network connections and use them as *insiders*. They can then use this capability to learn the location of secret proxies and feed this information to the botnet in order to launch a DDoS attack. We assume that the number of such insiders is much smaller than the number of legitimate clients served by the target system.

IV. Limitations of Current MTD Architectures

Existing MTD architectures [7], [8], [9] hide the location of an application server behind a layer of proxies. These proxies receive requests on behalf of the application server and forward requests (responses) from (to) the clients. Fig. 1 gives an overview of state-of-the-art MTD architectures. Essentially, they include three main components: a well-provisioned lookup server, proxy servers, and the application server. A description of each component is provided below.

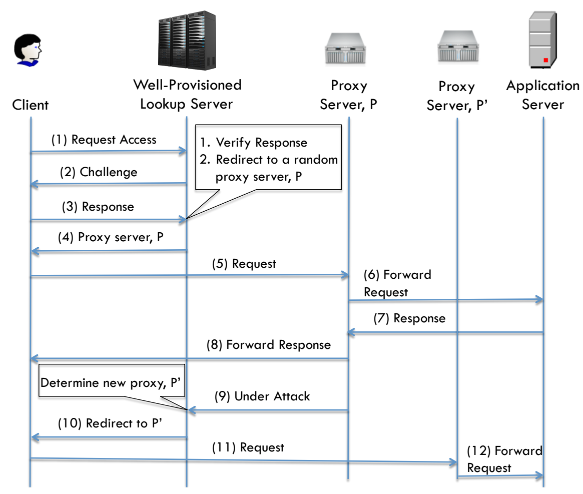


Fig. 1. Overview of MTD-based Architecture

Well-Provisioned Lookup Server: To communicate with the application server, the client sends a request to a lookup server (step 1). The lookup server responds to the client's request by issuing a challenge (step 2) such as user credentials [8] or Proof-of-Work (PoW) [9]. These challenge-response mechanisms aim to filter out clients who are either unauthorized to use the system or impersonate a legitimate client by IP spoofing. Upon solving this challenge (step 3), the client is redirected to a random active proxy server (step 4) that relays requests from the client to the application server. To ensure that the lookup server (whose IP address is public) is not susceptible to DoS attacks, existing architectures employ either a well-provisioned server – such as Content Distribution Networks [9] or a load-balancing DNS [7] – or leverage existing PoW schemes to force clients to solve cryptographic puzzles before they can consume the lookup server's resources.

Proxy Servers: Existing architectures assume the availability of a large pool of proxy servers – with only a small number of them active at any point in time. The IP addresses of active proxy servers are kept *secret*, i.e., they are not disclosed to clients who are unable to solve the challenge issued in step 2. *Authorized* clients contact the corresponding proxy server which in turn relays requests (responses) to (from) the application server (steps 5 through 8). As the location of the application server is known only to these proxy servers, the attack surface shifts from the application server to the active proxies. Existing architectures assume that proxies are equipped with a detection mechanism that enables them to detect an attack. When attacked, the proxy servers first inform the lookup server (step 9). Then, the lookup server instantiates new proxy servers and redirects clients associated with the attacked proxies to the new proxies (step 10). Additionally, an attack on a proxy implies that one of the clients connected to the attacked proxy is an insider and is divulging the proxy address to the attacker. Therefore, before redirecting the victims to new proxies, the lookup server uses a client-to-proxy shuffling strategy to segregate innocent clients from insiders and allocates them to different proxies. As a result of this shuffling process, the number of innocent clients impacted during subsequent flooding attacks is reduced.

Application Server: The application server hosts one or more services and maintains a state for each client connected to it. As the application server stores all the session information, the proxies are lightweight and only implement a simple traffic indirection logic to relay requests/responses.

Existing MTD architectures are limited by their threat model, which assume that insiders (i.e., malicious clients able to solve the challenge) *persist* in the network during an attack. This assumption facilitates effective isolation of innocent clients from insiders. However, under a more general threat model, in which an insider's behavior is unknown, we show that it is possible to circumvent existing moving target defenses. To this end, we present a novel attack, the *proxy harvesting attack*, and propose a solution to counter this attack.

A. Proxy Harvesting Attack

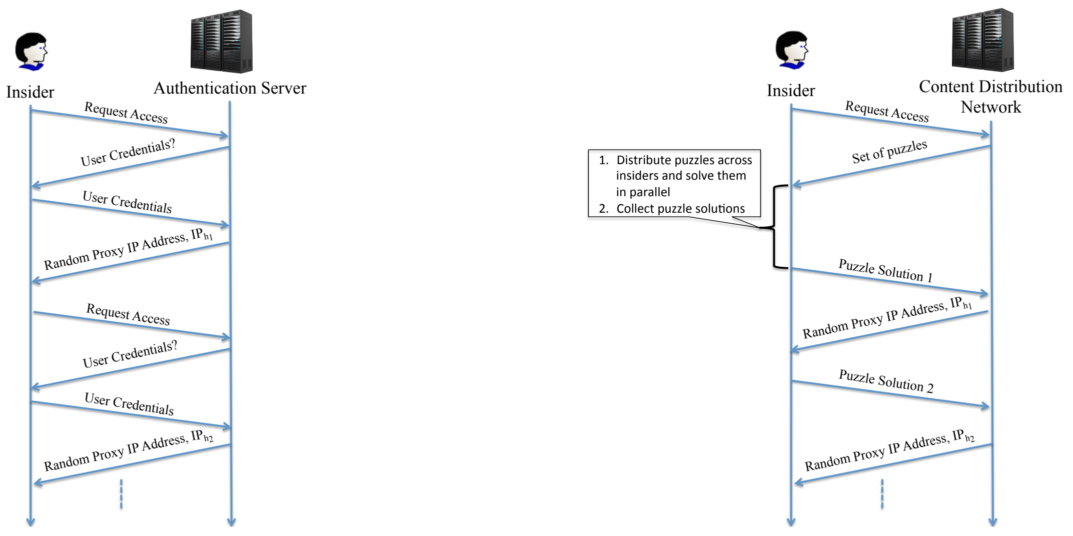
In this section, we present the *proxy harvesting attack*. The attacker's goal is to gather IP addresses of as many proxy servers as possible before launching a coordinated attack against multiple proxies. We make no assumption regarding an insider's behavior. In particular, we do not assume that the insiders persist in the system during or after a DDoS attack.

In MOTAG, the proxy harvesting attack works as shown in Fig. 2(a). After an insider authenticates, the authentication server assigns it to one of the active proxies at random. Through this process, the insider learns the IP address of an active proxy. Then the insider authenticates again with the authentication server and learns a new IP address. Ideally, this process is repeated until the IP addresses of all active proxies are collected. For DoSE, the proxy harvesting attack can be modified as shown in Fig. 2(b). An insider first collects a set of puzzles from the CDN. A benign client should choose one of the puzzles at random and solve it. Upon submitting its solution to the CDN, the client will be redirected to a secret proxy. However, an insider, after collecting the set of puzzles, can distribute them to other insiders and solve multiple puzzles in parallel. The insider can then collect these solutions and submit them to learn the location of multiple secret proxies.

In both architectures, once all or a significant number of active proxies have been discovered, the insider can outsource this information to external attackers (e.g., botmasters) to launch a coordinated DDoS attack against all known active proxies. As mentioned earlier, when proxies are under attack, the lookup server shuffles affected clients and moves them to new proxies. However, if the insider is not present in the system, the shuffling operation simply increases the overhead without providing any benefit in identifying the insiders.

B. Analysis of the Proxy Harvesting Attack

In this section we analyze the proxy harvesting attack against MOTAG. For the sake of brevity, we omit the analysis


Insider
Authentication Server
Request Access
User Credentials?
User Credentials
Random Proxy IP Address, IP_{b1}
Request Access
User Credentials?
User Credentials
Random Proxy IP Address, IP_{b2}
Insider
Content Distribution Network
Request Access
Set of puzzles
1. Distribute puzzles across insiders and solve them in parallel
2. Collect puzzle solutions
Puzzle Solution 1
Random Proxy IP Address, IP_{b1}
Puzzle Solution 2
Random Proxy IP Address, IP_{b2}

(a) Proxy Harvesting Attack against MOTAG

(b) Proxy Harvesting Attack against DoSE

Fig. 2. Proxy Harvesting Attack on existing moving target-based architectures

for DoSE, which, however, is straightforward to conduct similarly to what we show below for MOTAG.

In MOTAG, the insider needs to interact only with the lookup server (here, authentication server) to harvest IP addresses of active proxies. As the lookup server assigns the clients to proxies randomly, the insider may be assigned to a previously known proxy. The number of requests that must be made by an insider to discover all the IP addresses can be modeled as a classical variation of the Coupon Collector's Problem [18]. Hence, if there are N active proxies, the expected number of requests the insider must make to the lookup server is $N \cdot \log(N) + \gamma \cdot N + o(1)$ where $\gamma \approx 0.5772156$ is the Euler-Mascheroni constant and $o(1) \approx 0.5$. Therefore, if there are 50 active proxies, on average the insider needs to make 225 requests to learn all the IP addresses. It should be noted that, when multiple insiders collude to harvest IP addresses, the expected number of requests per insider decreases.

In practice, insiders may not know the exact number of proxies that are active at a given time, but they may know the range of IP addresses used by active proxies. Additionally, if an insider makes a large number of requests to the lookup server, an intrusion detection system may flag the situation as suspicious. Therefore, the attacker needs to determine the optimal number of authentication requests to strive a balance between stealthiness and harvesting enough proxies. We can model this problem as an optimization problem with the objective to maximize the number of active proxies discovered while simultaneously minimizing the likelihood of detection. Using the dynamic programming formulation in [19], the Bellman equation yielding the optimal stopping criterion is

$$A(r,n)=\min\{D(n+1),c+p_{r,n}\cdot A(r+1,n+1)+(1-p_{r,n})\cdot A(r,n+1)\} \quad (1)$$

where,

- $A(r, n)$ is the expected cost incurred by the insider to learn r distinct IP addresses by making n requests;
- $D(n)$ is the expected cost incurred by the insider when the attack is detected after making n requests;
- c is the cost of a single request to the lookup server;
- $p_{r,n}$ is the conditional probability of discovering a new IP address given that r distinct IP addresses were discovered from n requests: from [19], $p_{r,n} = \frac{C(r+1,n+1)}{C(r,n)}$ where $C(r,n) = \sum_{j=0}^{\infty} \frac{(r+j)!}{j!(r+j)^n}$

As the number of IP addresses harvested increases with the number of requests to the lookup server, intuitively Eq. 1 maximizes the number of requests, n, made by an insider subject to the constraint that the total cost of making n requests does not exceed the cost of detection, $D(n)$. Therefore, based on Eq. 1, the optimal probing strategy is as follows: if the insider detection cost $D(n)$ is less than the cost to discover new proxies, then make new request, otherwise stop.

V. BIND-SPLIT STRATEGY

The proxy harvesting attack exploits the lookup server's random assignment scheme – i.e., an incoming client request is randomly assigned to one of the active proxies – to discover multiple proxies before launching an attack. One of the approaches to overcome the proxy harvesting attack is to maintain a predefined mapping of clients to proxies, thus limiting the number of IP addresses that can be harvested by insiders. In this section, we present BIND-SPLIT, a simple, yet efficient, client-to-proxy mapping strategy that can be used to isolate insiders even if they do not persist in the system.

In the BIND-SPLIT strategy, the lookup server is configured to maintain a static mapping between clients and active proxies. Let $N = C_p \cdot P_0$ be the total number of clients with each one of the P_0 proxy servers initially serving C_p clients. Each client has a binding to a particular server, which persists even if the client logs out and logs back in. Also, let I be the number of insiders, wherein each insider can only discover the IP address of the server to which it is currently assigned. If the goal of the defender is to minimize the number of times any server can be attacked, the BIND-SPLIT strategy we propose appears to be asymptotically optimal. We assume the defender does not have sufficient server resources to preemptively assign every client its own server, which clearly would minimize attacks. Under the BIND-SPLIT strategy, clients remain bound to their assigned server until that server is attacked. When any server is attacked, the attacked server is shut down and two new servers are spawned in its place. The clients from the attacked server are split into two equal-size groups and migrated to the new servers. If the attacked server has only one client, then that client must be an insider.

We analyze BIND-SPLIT by considering a tree T. Each vertex of T represents a proxy spawned by BIND-SPLIT. Initially T consists of a root note, with edges to each of the initial P_0 proxies. When an attack occurs on some proxy p, we augment T by creating two new nodes, corresponding to the new proxies spawned by BIND-SPLIT. We make these nodes descendants of the attacked proxy. Notice that a node in T is a leaf node only if the proxy corresponding to it is never attacked. Therefore, to bound the total number of proxies attacked by I insiders, it suffices to bound the total number of non-leaf nodes in T. Now consider the proxies at some depth $d > 0$ in T. We argue that there can be only I non-leaf nodes at depth d. By definition of BIND-SPLIT, the clients served by two different proxies at depth d must be disjoint. Therefore at most I of the proxies at depth d can contain an insider, and so at most I proxies at depth d can have children. It is clear that T can also have depth at most $O(\log C_p)$, and therefore contains at most $O(I \cdot \log C_p)$ non-leaf nodes.

The maximum number of proxies used by the BIND-SPLIT strategy is $O(I \cdot \log C_p)$. This is the same order of growth as the number of attacks. The simple reason is that each time there is an attack on a server, the server is retired and two new servers spawned, increasing the server count by one. Note that, if BIND-SPLIT uses too many proxies to be practically applicable in some settings, it is possible to modify it to use fewer proxies. Specifically, the defender can merge proxies that have not been attacked for a long time. This modification has the drawback of increasing the time to isolate an attacker on the merged proxies, in case the merged proxies did contain an insider and the new proxy is later attacked.

VI. Proactive Proxy Migration Architecture

A limitation of BIND-SPLIT is the tight coupling of clients to proxies which may lead to suboptimal load distribution across proxies. Furthermore, the effectiveness of existing MTD-based architecture against DDoS attack degrades in the presence of proxy harvesting attacks: moving and shuffling clients among proxies occur *only after* a DDoS attack has been detected. To strike a balance between the performance and service availability, we propose PROTAG (PROactive proxy-based moving TArGet architecture). Inspired by Fast-Flux Service Networks (FFSN) – a popular network architecture used by botnets, phishers and spammers to provide high availability for their services [20] – we periodically migrate clients to new proxies irrespective of whether an attack has been detected. Such proactive movement would disrupt the reconnaissance efforts of insiders by invalidating all or part of the information they were able to gather. In fact, as implied by our reasoning in Section IV-A, in order to remain stealthy, attackers will first gather IP addresses of multiple proxies over a possibly extended period of time, and only at a later stage they will use that information to launch a DDoS attack. If, in the meantime, many clients have been migrated to new proxies, the attack will be either disrupted or at least mitigated. Finally, after an attack, PROTAG harnesses the insider isolation principle of the BIND-SPLIT strategy to partition the clients across proxy pools and eventually isolate the insiders.

Within the existing MOTAG architecture, a proactive proxy movement strategy can be defined in terms of two primary factors: proxy selection and movement frequency. Proxy selection involves determining *which* proxies to replace, whereas a strategy's movement frequency determines *when* to replace the proxies. In this paper, we consider a simple, yet effective, strategy in which a subset of active proxies is chosen uniformly at random for replacement. Effectiveness of this strategy is evaluated by computing the expected number of active proxies that can be discovered in comparison with existing architectures where no proactive action is implemented.

A. Proxy Movement

Let $\mathcal{P} = \{P_1, P_2, ..., P_K\}$ be the set of all available proxies. We assume that, at any time $t \in T$ – where T is a set of discrete time points – only k (with $k < K$) proxies are active, and we use $\mathcal{P}_t = \{P_{i_1^t}, P_{i_2^t}, ..., P_{i_k^t}\}$ – where $i_j^t \in [1, K]$ for all $j \in [1, k]$ and all $t \in T$ – to denote the set of active proxies at time t.

In PROTAG, the authentication server initiates the movement procedure every Δt time units – we refer to Δt as the *reconfiguration period* and to $f = \frac{1}{\Delta t}$ as the *reconfiguration frequency*. At that time, the authentication server chooses a subset of m active proxies $\mathcal{P}_r \subseteq \mathcal{P}_t$, with $|\mathcal{P}_r| = m$, uniformly at random. Additionally, the authentication server chooses m available (but currently not active) proxy servers $\mathcal{P}'_r$ uniformly at random from the set $\mathcal{P} \backslash \mathcal{P}_t$. All the clients associated with a proxy in $\mathcal{P}_r$ are then migrated to a proxy in $\mathcal{P}'_r$.

The authentication server maintains each client-proxy association for a period of time Δt^*, which we refer to as the *association period*. If a client assigned to a proxy P_i closes the connection with P_i and re-authenticates with the authentication server before Δt^* time units since it was originally assigned to P_i, then the authentication server assigns it to the same proxy. This mechanism imposes an upper bound on the rate at which an insider may discover new proxies. In fact, the insider may not discover more than $\frac{1}{\Delta t^*}$ proxies per time unit by doing repeated authentication requests. The time interval Δt^* is determined by the network administrator by taking into account the desired load per proxy (in terms of number of clients) and the expected arrival rate of clients, thereby ensuring that proxies are not overloaded. The frequency with which the proxies are moved depends on the number of active proxy addresses that an insider can learn in Δt time units. In other words, if proxies are proactively moved using this strategy every Δt time units, then an insider would have discovered at most $\lceil \frac{\Delta t}{\Delta t^*} \rceil$ proxies. Determining the value of Δt is critical to ensure that the strategy is effective: the smaller the value of Δt, the higher the insider's effort to harvest proxies. To this end, we derive the expected number of active proxies that can be discovered by multiple insiders as a function of the number of authentication requests and the reconfiguration interval Δt.

B. *Analysis of PROTAG*

In this section, we model the problem as a game between insiders and authentication server, such that, at each round, the insiders send multiple independent probes to discover proxies while the authentication server moves multiple proxies. Let I be the number of insiders and m be the number of active proxies moved by the authentication server at each round of the game. For the sake of analysis, we assume that the reconfiguration period Δt is a multiple of the association period Δt^*, that is $\Delta t = z \cdot \Delta t^*$ where $z \in \mathbb{Z}^+$ is a positive integer. Therefore, at each round, the authentication server first moves m of the k active proxies – using the strategy described earlier – followed by each insider making z authentications at times that are multiple of Δt^*. For the purpose of this analysis, we consider a finite number $q \in \mathbb{Z}^+$ of rounds. The defender moves m proxies at times $0, \Delta t, 2 \cdot \Delta t, \dots, (q-1) \cdot \Delta t$, and, for each round $r \in [1, q]$, each insider makes an authentication request at times $(r-1) \cdot \Delta t, (r-1) \cdot \Delta t + \Delta t^*, \dots, (r-1) \cdot \Delta t + (z-1) \cdot \Delta t^*$. The total number of authentication requests made by all the insiders during the q rounds is then $n_{tot} = q \cdot z \cdot I$. As each probe at times $(r-1) \cdot \Delta t + x \cdot \Delta t^*, x \in [0, z-1]$ is assigned an active proxy uniformly at random – independently of the specific client – the expected number of distinct proxies discovered by I insiders at the end of q rounds by making z probes at each round is equal to the number of distinct proxies discovered by an insider making $z \cdot I$ probes at each round.

An example of the timeline of an authentication server-insider game is shown in Fig. 3. In this example, $q = 3$, $z = 3$ and $I = 1$. The insider makes a move every $\Delta t^* = 2$ time units while the authentication server makes a move every $\Delta t = z \cdot \Delta t^* = 6$ time units.

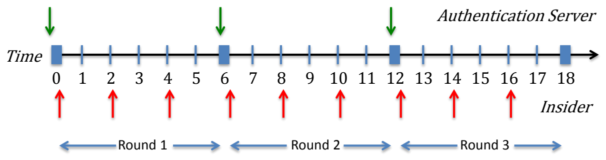


Fig. 3. Example of authentication server-insider game

Let X_r be a random variable representing the number of distinct active proxy IP addresses discovered at the end of round $r \in [1, q]$. The effectiveness of the strategy can be evaluated by computing the expected value of X_q, that is the expected number of distinct proxies that the insider discovers by the end of the q-th round of the game. This can be computed as $E[X_q] = \sum_{i=1}^{k} (i \cdot Pr[X_q = i])$. For the first round, the probability that the insider discovers x proxies out of k active proxies after $z \cdot I$ authentications, $Pr[X_1 = x]$, is given by the number of ways to arrange x distinct proxies from k active proxies across $z \cdot I$ authentications divided by the number $k^{z \cdot I}$ of possible outcomes. Therefore, the probability distribution for X_1 is given by

$$Pr[X_1 = x] = \frac{P_{(k,x)} \cdot S_{(z \cdot I)}^{(x)}}{k^{z \cdot I}}, \ x \in [1, \min(z \cdot I, k)] \quad (2)$$

where,

- $S_n^{(r)} = \sum_{j=1}^{r} \frac{(-1)^{r-j} j^n}{j!(r-j)!}$ is a Stirling number of the second kind, which is the number of ways to divide a set of n objects into r nonempty subsets. Intuitively, this captures the notion that multiple authentication requests may yield the same IP address;
- $P_{(k,x)}$ is the number of ways to arrange x-element subsets from a k-element set.

Moving proxies invalidates knowledge acquired by the insider. Let A_r be a random variable which represents the number of known proxies that are removed when the authentication server moves m proxies at round r. Also, let D_r be the maximum number of active proxies that the insider might have already discovered by the end of round $r-1$. Such number cannot exceed the number $x + A_r$ of proxies discovered at the end round r. Therefore, the value of D_r depends on the outcome of A_r, that is

$$D_r(A_r = l) = \min((r-1) \cdot z, x + l, k) \quad (3)$$

Hence, for $x \in [1, \min(r \cdot z \cdot I, k)]$, the probability distribution of X_r is given by

$$Pr[X_r = x] = \sum_{y=1}^{D_r(A_r = l)} Pr[X_r = x | X_{r-1} = y] \cdot Pr[X_{r-1} = y] \quad (4)$$

Then, combining Eq. 3 and Eq. 4, we can write

$$Pr[X_r = x] = \sum_{l=0}^{m} \sum_{y=\max(l,1)}^{D_r(A_r = l)} \begin{pmatrix} Pr[X_r = x | X_{r-1} = y, A_r = l] \cdot \\ \cdot Pr[A_r = l | X_{r-1} = y] \cdot Pr[X_{r-1} = y] \end{pmatrix} \quad (5)$$

In Eq. 5, $Pr[A_r = l | X_{r-1} = y] = \frac{C_{(y,l)} \cdot C_{(k-y,m-l)}}{C(k,m)}$. If the insider has discovered y and x proxies by the end of rounds $r-1$ and r respectively, then during the $z \cdot I$ authentications at round r, the insider should have discovered $x - (y - l)$ new proxies where $l \in [0, m]$. However, the insider may encounter s already known proxies, where $s \in [0, y - l]$. Therefore, for a given s, $Pr[X_r = x | X_{r-1} = y, A_r = l]$ is the number of ways of arranging $x - (y - l)$ distinct proxies from $k - (y - l)$ unknown active proxies and s distinct proxies from $y - l$ discovered active proxies across $z \cdot I$ authentications out of $k^{z \cdot I}$ possible outcomes. Hence,

$$Pr[X_r = x | X_{r-1} = y, A_r = l] =$$

$$= \frac{\sum_{s=0}^{y-l} P_{(k-(y-l), x-(y-l))} P_{(y-l,s)} C_{(x-(y-l)+s, s)} S_{z \cdot I}^{(x-(y-l)+s)}}{k^{z \cdot I}} \quad (6)$$

where,

- $S_n^{(r)}$ is a Stirling number of the second kind
- $P_{(n,r)}$ is the number of ways of arranging r-element subsets of an n-set
- $C_{(n,r)}$ is the number of ways of choosing r objects from a set of n objects

By substituting Eq. 6 in Eq. 5 and recursively solving for $r = Q$ with Eq. 2 as the base case, we can derive $Pr[X_q = x], \forall x \in [1, k]$.

VII. Insider Isolation Algorithm

PROTAG aims at mitigating the impact of a DDoS attack, which can be defined as the average number of innocent clients who suffer from degraded performance during the attack. In the proposed architecture, the impact of a DDoS attack on a given number of active proxies, on average, remains constant as all the clients are distributed uniformly at random across the same pool of active proxies. To reduce the impact of subsequent DDoS attacks, we propose a simple insider isolation algorithm which incorporates the isolation principles of the BIND-SPLIT strategy to partition clients into different proxy pools and eventually, isolate insiders into proxy pools with fewer innocent clients. In this algorithm, the lookup server maintains a mapping for each client to a set of proxies. Before the onset of an attack, all the clients are associated with a single pool of active proxies. After an attack, for each attacked proxy pool $\mathcal{P}$, the lookup server will first spawn two new proxy pools $\mathcal{P}_1$ and $\mathcal{P}_2$ with the same size as the attacked proxy pool. Next, the server will randomly partition the clients associated with the attacked proxy pool into two groups $\mathcal{C}_1$, $\mathcal{C}_2$ and re-assign clients in $\mathcal{C}_i$ to proxy pool $\mathcal{P}_i$, $i = 1, 2$. For subsequent connection requests by any client in $\mathcal{C}_i$, the lookup server will re-direct it to proxy pool $\mathcal{P}_i$, $i = 1, 2$.

A. Analysis

We analyze the proposed insider isolation algorithm and provide a theoretical upper bound on the number of attacks and the number of proxies necessary to completely isolate the insiders. To analyze the algorithm, consider a tree $\mathcal{T}$ where the nodes of the tree $\{\mathcal{P}, \mathcal{C}\}$ represent the identity of the proxy pool $\mathcal{P}$ and the set of clients $\mathcal{C}$ associated with it. An edge from $\{\mathcal{P}, \mathcal{C}\}$ to $\{\mathcal{P}', \mathcal{C}'\}$ exists if and only if proxies in $\mathcal{P}$ were attacked and the lookup server spawned proxies in $\mathcal{P}'$ and assigned clients in $\mathcal{C}' \subset \mathcal{C}$ to proxies in $\mathcal{P}'$ where $|\mathcal{C}'| = \frac{|\mathcal{C}|}{2}$.

It can be seen that at any depth d all sets of clients at depth d are disjoint and hence, the resulting tree $\mathcal{T}$ has a total depth of $\log(N)$, where N is the total number of clients. The depth of the tree provides an upper bound on the number of DDoS attacks experienced by a client before isolating the insiders. In other words, the number of attacks experienced by any user of the system is bounded by $\log(N)$. Furthermore, the number of proxies spawned by the algorithm can be calculated by counting the number of leaf nodes in $\mathcal{T}$. As a base case, consider a scenario where there is exactly one insider in the system among N' clients associated with a proxy pool $\mathcal{P}'$. In this case, the resulting tree $\mathcal{T}'$ will be a skewed binary tree with $\log(N')$ leaf nodes. When there are I insiders, then in the worst case, after each attack, the insiders will be evenly distributed across all the proxy pools. Consider the set $\mathcal{F}$ of subtrees rooted at nodes at depth $\log(I)$. The root of each subtree $\mathcal{T}' \in \mathcal{F}$ represents a proxy pool with $\frac{N}{I}$ innocent clients and one insider. As there are at most I such subtrees, the maximum number of proxies is $\log\left(\frac{N}{I}\right) \cdot I \cdot |\mathcal{P}_i|$ where $|\mathcal{P}_i|$ is the number of proxies in the initial active proxy pool.

VIII. Simulation Results

In this section, we study the proxy harvesting attack, validate the proactive movement and the insider isolation algorithm presented in the previous section through simulations.

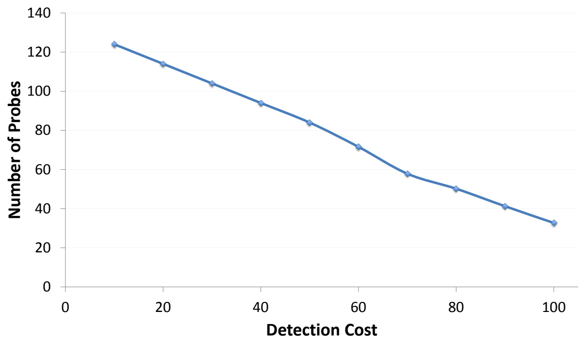


Fig. 4. No. of probes by an insider for different detection costs

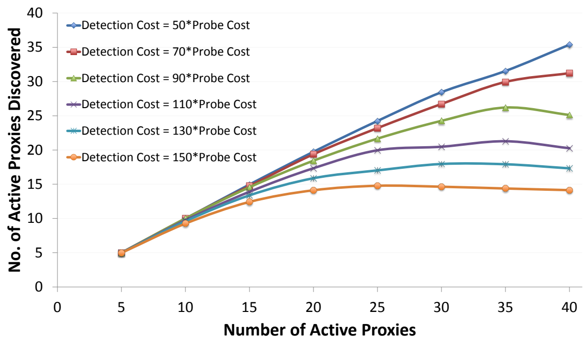


Fig. 5. No. of discovered proxies vs. number of active proxies

A. Proxy Harvesting Attack

In order to illustrate the trade-off in the attacker's actions in MOTAG-based architectures (whether to authenticate or not), we simulated Eq. 1 for various values of the detection cost and number of active proxies. In the simulation, we assumed, $D(n) = P(n) \cdot d$, where $P(n) = (1 - e^{-(\alpha \cdot n)})$ is the probability that the attacker makes n probes and d is the cost for an insider to be detected. Such exponential distributions have been used to model the effort required by an attacker to successfully mount an attack [21]. In our simulations, we set $\alpha = 0.05$.

The cost of detection, d, and the cost of probing, c, were normalized by setting $d = z \cdot c, z \in \mathbb{Z}^+$. In our simulations, we assumed $c = 1$. Fig. 4 shows the number of authentication probes as a function of the detection cost d. As expected, the number of authentication probes made by an insider decreases as the detection cost increases. The rationale behind this trend is that, as the detection cost d increases, the expected cost of insider detection $D(n)$ increases for every subsequent probe. This restricts the number of authentication probes for the insider. As shown in Fig. 5, for lower detection costs, an insider can discover at least 70% of the active proxies and hence, can launch a *significant* DDoS attack against known proxies.

B. PROTAG

To assess the performance of PROTAG, we simulated Eq. 5 for 40 active proxies with different numbers of insiders. In our simulation, the insiders made a total of 100 probes. As discussed earlier, the number of proxies discovered by an attacker using I insiders is equivalent to the number of

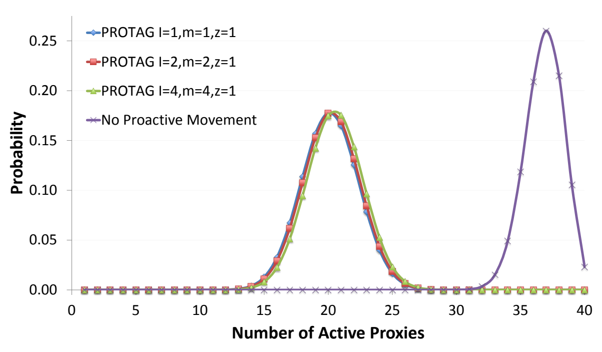


Fig. 6. Probability distribution of the number of discovered proxies when authentication server moves $m = I$ proxies

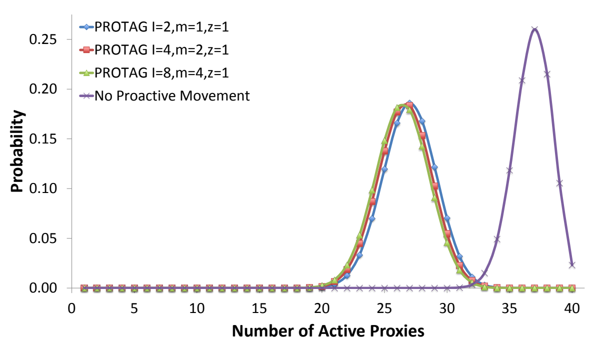


Fig. 7. Probability distribution of the number of discovered proxies when authentication server moves $m = 0.5 \cdot I$ proxies

proxies discovered by one insider sending I probes. Fig. 6 and Fig. 7 depict the probability distribution of the number of proxies discovered by multiple insiders with and without applying PROTAG, where z is the number of authentications between consecutive proxy movements by the authentication server. As shown in the figures, when there is no proactive movement, there is a high probability of discovering most of the proxies with an expected value of 37. However, with proactive movement, the probability distribution *shifts* such that the probability of discovering more than 80% of the proxies reduces significantly. In particular, for proactive movement, the expected number of proxies discovered is 20 and 26 respectively, reducing the number of discovered proxies by 46% and 30% respectively. Furthermore, we observed that by increasing the number of probes made by the insider, the probability distribution of the number of proxies discovered by an insider did not change.

On the other hand, if the authentication server moves multiple proxies ($m > 1$), the expected number of discovered proxies can be reduced. Fig. 6 and Fig. 7 depict the probability distribution of discovered proxies when the authentication server moves $m = I$ and $m = 0.5 \cdot I$ proxies, respectively. It can be seen that the expected number of discovered proxies can be restricted to the case when the authentication server moved only one proxy. These results also show that, if the cost of frequently moving a proxy is high, then the authentication server can move less frequently and compensate for the additional information gained by an attacker by moving multiple proxies (proportional to the number of probes) later.

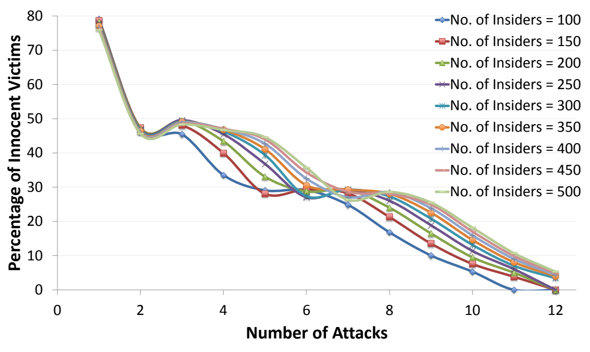


Fig. 8. Percentage of innocent users affected during a sequence of attacks

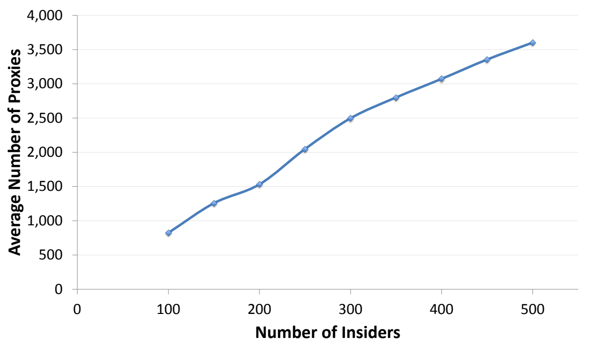


Fig. 9. Average number of proxies required to isolate 90% of innocent users

C. Insider Isolation Algorithm

To study the performance of the proposed insider isolation algorithm, we considered a system serving 10,000 users with a varying number of insiders. In order to simulate the user's behavior, in our environment, after every hour, the users (if not connected) log into the system with a probability of 0.8 and if connected, they disconnect with a probability of 0.5. These simulations were conducted for a period of 500 hours ($\sim$ 3 weeks of uptime) starting with a proxy pool of 2 active proxies. In our simulation, we assumed that the attacker's objective is to discover at least 50% of the active proxies before launching the attack. Against such a threat model, we studied the performance of the proposed algorithm to save at least 90% of innocent users with and without proactive movement. All simulations were repeated 30 times and the results were averaged over these independent trials.

As expected, the percentage of innocent users who are affected due to repeated DDoS attacks decreases with every subsequent attack (shown in Fig. 8). As described in section VII, the insider isolation algorithm spawns new proxies and partitions clients across these proxies after an attack. As the number of insiders increases, the average number of proxies instantiated also increases as depicted in Fig. 9. When deployed in a cloud environment such as Amazon EC2, the cost of on-demand instances to host the proxies (which is about $0.007 per hour for EC2 instances based on their spot pricing) would cost the enterprise less than $150 dollars (for 500 insiders) to mitigate attacks lasting for 100 hours and protect more than 90% of innocent users from subsequent attacks. To understand the impact of initial proxy pool size, we repeated

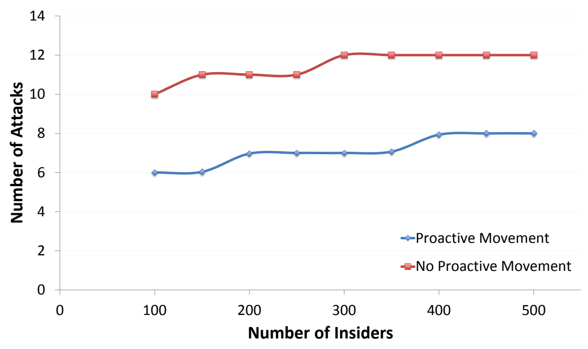


Fig. 10. Average number of attacks with and without proactive movement

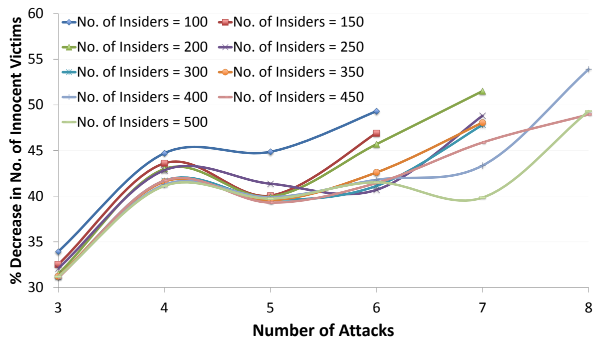


Fig. 11. Percentage decrease in number of innocent victims

the experiment with 10, 25 and 50 proxies and observed less than 5% decrease in the number of saved innocent users. In addition to the cost incurred from deploying the proxies, the users of the application incur an overhead cost due to frequent redirection to new proxies. As studied in [7], the clients are expected to experience less than 3 seconds overhead as a result of the redirection operation.

We also studied the impact of coupling PROTAG with the insider isolation algorithm. Here, we considered a movement strategy in which, from each active proxy pool, an active proxy was replaced on an hourly basis. As illustrated in Fig. 10, PROTAG, on average, reduced the number of attacks observed during an experiment's lifetime by 4. As a result, it was observed that the average number of innocent users who are affected due to an attack decreased by 40% in comparison to the setting without PROTAG (shown in Fig. 11).

IX. Conclusions

In this paper, we first presented a new type of attack, the proxy harvesting attack, and showed that state-of-the art-solutions are vulnerable to this attack. Then, to overcome the limitations of current approaches, we proposed a proactive strategy to periodically replace one or more proxies and remap clients to proxies, with the goal of disrupting the attacker's reconnaissance efforts, and an insider isolation approach to mitigate ongoing attacks. We validated our solution in a simulated environment, and simulation results confirmed that the proposed strategy is effective in reducing the number of proxies an attacker can discover in a given amount of time as well as the impact of attacks.

References

[1] "Worldwide infrastructure security report," Arbor Networks, Tech. Rep. Volume XI, November 2015.

[2] M. Goncharov, "Russian underground 101," Trend Micro Incorporated, Tech. Rep., 2012.

[3] D. G. Andersen, "Mayday: Distributed filtering for internet services," in *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS 2003)*, Seattle, WA, USA, March 2003, pp. 31–42.

[4] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: Secure overlay services," in *Proceedings of ACM SIGCOMM 2002*, Pittsburgh, PA, USA, August 2002, pp. 61–72.

[5] S. Jajodia, A. K. Ghosh, V. S. Subrahmanian, V. Swarup, C. Wang, and X. S. Wang, Eds., *Moving Target Defense II: Application of Game Theory and Adversarial Modeling*, 1st ed., ser. Advances in Information Security. Springer, 2013, vol. 100.

[6] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, Eds., *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*, 1st ed., ser. Advances in Information Security. Springer, 2011, vol. 54.

[7] Q. Jia, H. Wang, D. Fleck, F. Li, A. Stavrou, and W. Powell, "Catch me if you can: A cloud-enabled DDoS defense," in *Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2014)*, Atlanta, GA, USA, June 2014, pp. 264–275.

[8] H. Wang, Q. Jia, D. Fleck, W. Powell, F. Li, and A. Stavrou, "A moving target DDoS defense mechanism," *Computer Communications*, vol. 46, pp. 10–21, June 2014.

[9] P. Wood, C. Gutierrez, and S. Bagchi, "Denial of service elusion (DoSE): Keeping clients connected for less," in *Proceedings of the 34th IEEE Symposium on Reliable Distributed Systems (SRDS 2015)*, 2015, pp. 94–103.

[10] L. Garber, "Denial-of-service attacks rip the Internet," *Computer*, vol. 33, no. 4, pp. 12–17, April 2000.

[11] M. Geva, A. Herzberg, , and Y. Gev, "Bandwidth distributed denial of service: Attacks and defenses," *IEEE Security & Privacy*, vol. 12, no. 1, pp. 54–61, January/February 2014.

[12] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046–2069, Fourth Quarter 2013.

[13] A. Stavrou, *Encyclopedia of Cryptography and Security*, 2nd ed. Springer, 2011, ch. Overlay-Based DoS Defenses, pp. 891–897.

[14] A. Stavrou, A. D. Keromytis, J. Nieh, V. Misra, and D. Rubenstein, "MOVE: An end-to-end solution to network denial of service," in *Proceedings of the Network and Distributed System Security Symposium (NDSS 2005)*, San Diego, CA, USA, February 2005, pp. 81–96.

[15] Q. Duan, E. Al-Shaer, and H. Jafarian, "Efficient random route mutation considering flow and network constraints," in *Proceedings of the IEEE Conference on Communications and Network Security (IEEE CNS 2013)*, Washington, DC, USA, October 2013, pp. 260–268.

[16] A. Herzberg and H. Shulman, "DNS authentication as a service: preventing amplification attacks," in *Proceedings of the 30th Annual Computer Security Applications Conference (ACSAC 2014)*, 2014, pp. 356–365.

[17] A. Wang, A. Mohaisen, W. Chang, and S. Chen, "Delving into internet DDoS attacks by botnets: characterization and analysis," in *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2015)*, 2015, pp. 379–390.

[18] R. Sedgewick and P. Flajolet, *An Introduction to the Analysis of Algorithms*. Addison-Wesley, January 2013.

[19] P. R. Freeman, "Sequential estimation of the size of a population," *Biometrika*, vol. 59, no. 1, pp. 9–17, 1972.

[20] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and detecting fast-flux service networks," in *Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS)*, 2008.

[21] R. Ortalo, Y. Deswarte, and M. Kaâniche, "Experimenting with quantitative evaluation tools for monitoring operational security," *IEEE Transactions on Software Engineering*, vol. 25, no. 5, pp. 633–650, 1999.