Predictability of IP Address Allocations for Cloud Computing Platforms

Hussain M.J. Almohri, *Member, IEEE,* Layne T. Watson, *Life Fellow, IEEE*, David Evans

***Abstract*—One way to combat denial of service attacks on cloud-based virtual networks is to use unpredictable network addresses, aiming to increase attacker effort by requiring attackers to search a large IP address space to find a target host. IP address randomization is used by several moving target defenses, relying on the assumption that it is difficult for an attacker to predict newly allocated IP addresses. This work analyzes whether IP addresses used by cloud providers are unpredictable enough in practice. We analyze the IP address allocation behaviors in two major cloud computing providers (Amazon Web Services and Google Cloud Platform) and find that the actual entropy provided by allocated IP addresses is limited. We evaluate several prediction models, including a simple frequency-based model as well as a Markov process model that produces an address prediction set from time series data of collected IP addresses. Our results show that simple models can reduce the search space for allocated IP addresses and diminish the effectiveness of randomization defenses.**

***Index Terms*—Moving target defenses, Randomization, Network security, Security management, Computer network management, Unsupervised learning**

I. Introduction

Denial-of-service (DoS) attackers exhaust a target set of services in a network to deny access to benign clients. DoS attacks at the network, transport, or application layers can target specific IP addresses of the services and exploit vulnerabilities at the targeted layer. To mitigate denial of service attacks, researchers have explored solutions such as inspecting and filtering and identifying attack paths [1], detecting anomalies [2], [3], balancing work loads [4], and limiting request traffic [5]. In recent years, the idea of moving target defenses such as IP address randomization has emerged as a promising approach [6]–[10]. IP address randomization enables moving target defense systems to mitigate DoS attacks on hosts that are exposed to the Internet. The idea is to confuse the attacker by periodically moving services to newly allocated IP addresses, forcing the attacker to spend time and effort locating the target service at a new IP address. In such systems, registered clients may be redirected to the new addresses, while unknown and potentially malicious clients temporarily lose contact with the target. For example, consider a cloud-based network with a target server that uses a public static IP address. The server runs a service, such as a VPN, for a large number of clients that are authenticated a priori, and disallows anonymous clients from using the service. To keep potential denial of service attackers distracted from the target server, another private machine in the same network selects a time and requests a fresh IP address, and updates the target server's IP address with the fresh IP address. Then, the private machine securely notifies authenticated clients of the new IP address. The process of refreshing IP addresses over time is referred to as *IP randomization*.

A moving target defense, such as the one proposed by Al-Shaer et al. [11], senses that a host is being targeted by a DoS attack, requests a new IP address from the pool of addresses of the cloud provider, moves the targeted service to the new IP address, and informs the legitimate clients about the new IP address. Assuming the attacker has no control over the communication between a benign client and the host, the attacker must blindly search the IP address space to find the new address of the target. Thus, the IP address of the target service serves as a shared secret among authenticated (legitimate) users of the service. Even though the IP address is open to Internet traffic, attackers need to search through a large IP address space to find the target service. That said, the IP address space is limited and there are tools for efficiently searching through an IP address space. To remedy this, moving target defense systems frequently refresh the IP address for the target service to waste the adversary's search effort. Maintaining an unpredictable series of IP addresses for target servers is a core security requirement in moving target defense systems since the adversary's cost depends on the expected number of attempts needed to locate the IP address of the target.

Although previous works address design issues for building moving target defense systems, no previous work has systematically tested the critical underlying assumption, namely that cloud providers allocate IP addresses in a way that is unpredictable to an adversary. The goal of this work is to investigate the effectiveness of IP address randomization in practice, by analyzing the predictibility of IP addresses allocated by prominent cloud computing platforms.

The main goal of this work is to design a practical attack that could be realized given the available tools and APIs from cloud computing providers. We consider an attacker who is able to generate and maintain a dataset of IP addresses and uses that dataset to predict the IP address of a moved service. Generating the dataset should be scalable, allowing

H. M. J. Almohri is with the Department of Computer Science, Kuwait University, Kuwait. Email: almohri@ieee.org

L. T. Watson is with the Departments of Computer Science, Mathematics, and Aerospace and Ocean Engineering, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061. Email: ltw@ieee.org

David Evans is with the Department of Computer Science, University of Virginia, Charlottesville, VA 22903. Email: evans@virginia.edu

This work was supported and funded by Kuwait University, Research Project No. (RQ01/18) and by a grant from the National Science Foundation (1422332).

the attacker to collect a large fraction of available IP addresses. The attacker should model the prediction based on observable evidence from the target network. Given recent observations from the target network, the attacker should generate a reasonably small set of candidate IP addresses that is likely to include the IP address allocated to the target service. We focus on IPv4 networks, even though IP Version 6 (IPv6) exponentially increases the address search space for attackers [12]. Unfortunately, IPv6 is still not fully deployed, and most services need to externally maintain support for IPv4 addresses. Further, IPv6 has numerous vulnerabilities [13] and does not necessarily help in preventing IP address prediction by attackers. Surprisingly, there are previous works on finding patterns in IPv6 addresses [14], [15], while IPv4 is less studied.

A. Contributions

The focus of this study is to investigate how widely-used cloud computing platforms allocate IP addresses from the perspective of an attacker who wants to predict IP address assignments. Given partial knowledge of recent IP addresses used by the target service, the attacker must predict the next set of IP addresses that will be assigned. This prediction requires narrowing the search space for IP addresses based on knowledge gained from the target service and cloud provider.

We assume attackers have no control over the cloud computing platform nor a system owner's cloud subscription. The attacker must predict the IP address allocation behavior based on events in time, using tools that can retrieve data from the cloud computing providers. The predictability of IP addresses depends on both the application's configuration and the cloud provider. For example, Amazon Web Services (AWS) allows users to allocate IP addresses in specific geographic regions, separating virtual machine creation from IP address allocation; Google Cloud Platform (GCP) only allocates a new address when a new virtual machine is created. The cloud computing platforms can also enforce limits on the number of IP addresses possessed by an account, as is the case with AWS.

As observed from the collected data, IP address allocations in AWS and GCP are not random. We collected a large set of IP addresses from each provider by allocating and releasing addresses in multiple regions for both AWS and GCP. We trained an address prediction algorithm on this data, mimicking an attacker's effort in overcoming IP address randomization employed by a cloud computing user.

In summary, the contributions of this work are:

1) an overview of the available cloud technologies for IP address randomization and the limitations and policies imposed by major platforms (Section II) and an analysis of the IP addresses they allocate from collecting data from several regions over several weeks (Section V-B),
2) formal attack and defense models for understanding the impact of IP address randomization (Section III),
3) three heuristic search algorithms for predicting IP address segments allocated by cloud computing platforms, including a frequency-based search and a model that uses clustering and a Markov transition matrix (Section IV), and
4) an open source implementation and experimental evaluation of the prediction process (Section V).

The results demonstrate that popular cloud provided may allocate addresses in predictable ways. Hence, it is important that moving target defenses based on IP address randomization are implemented to mitigate predictable address allocations, and their security is evaluated in light of an adversary's ability to focus their search for the target IP address based on knowledge of IP address allocation.

II. Background and Related Work

This section provides background on moving target defenses and an overview of cloud-based technologies that enable rapid IP address movement. Section II-C summarizes related work on evaluating the effectiveness of moving target defenses.

A. Moving Target Defenses

A moving target defense attempts to thwart attacks by continually changing some property of the target on which the attack depends. The goal is to stretch the attack's required time and resources through a series of changes in the network or machine configurations, exhausting the resources available to the attacker. Moving target defenses do not tackle specific technical vulnerabilities that allow an attack, but defend the system by making exploitation more difficult for all attacks that depend on a particular property. For example, a denial of service attack requires knowledge of the victim's network addresses. Moving target defenses attempt to prevent an adversary from knowing those addresses. Several authors have presented overviews and formal analyses of moving target defenses [16]–[21]; here, the focus is limited to moving target defenses that attempt to hide network endpoints.

Jafarian et al. proposed a method for untraceable IP address randomization in response to the attacker's behavior [22]. The proposed work suggested hypothesis testing as a basis for forming the attacker's decisions. While an interesting approach, the aforementioned work does not address the specific characteristics and tools available for networks in cloud-based virtual networks. Another closely related work by Achleitner et al. suggested a formal model of network deception to defend against reconnaissance attacks [23]. The proposed Reconnaissance Deception System (RDS) confuses attackers by presenting virtual information to attackers using software-defined networking. The goals behind RDS are similar to those of *misery digraphs* [24]. A similar work uses network simulation to achieve deception by simulating virtual network topologies [25] and using the simulation results to delay reconnaissance attempts. The presented work also targets reconnaissance attempts by demonstrating the effectiveness of such attempts when attacking address randomization in cloud. The general shared goal is to combat reconnaissance attacks, however, the recent work in the area does not specifically target address randomization, nor provides a systematic modeling of reconnaissance attacks on cloud-based virtual networks.

B. Cloud-based Virtual Networks

A cloud computing provider, such as Amazon Web Services, provides infrastructure as a service. The user is a person that owns or manages an account with the cloud computing provider. The user has access to all services available for the account and can add, edit, or remove any service, host, or security rule in the account. Users can thus create virtual networks of hosts with routing rules, serving one or more applications, with specific Internet gateways. The user's network is referred to as the *target network* throughout this work.

Cloud computing platforms provide virtualized computing services that can be rapidly deployed for production. These services include virtual machines created from predefined or customized machine images, public static or ephemeral IP addresses that could be dynamically assigned to virtual machines, virtual networks with internal IP addresses and routing tables, and finally security rules that span one or more virtual machines. A virtual machine is usually created on a physical machine in a specific geographic region. The region of a virtual machine affects the services and the IP subnets available to it.

Amazon Web Services. Creating a virtual machine (referred to as an EC2 instance on Amazon Web Services) implies joining the virtual machine in a default virtual private cloud network with a set of predefined security rules (i.e., accessibility of ports) and routing tables. After an EC2 instance is launched, it is assigned an internal IP address as well as an ephemeral external IP address, which is subject to change whenever the EC2 instance is stopped (stopping an EC2 instance lowers or suspends the due hourly charges, but does not lose the data or its configuration).

An EC2 instance could be connected to an Internet gateway and serve requests from outside the network, thus exposing the host to the external network. With an elastic IP address, AWS allows the cloud user to allocate permanent static IP addresses that are not tied to specific EC2 instances, but are only visible in a single geographic region. A cloud user can choose to allocate up to five elastic IP addresses (per current policies, with an option to request an increased limit) without associating them with EC2 instances. When an EC2 instance receives one of the available elastic IP addresses, it can be stopped and restarted without changing its network address.

To efficiently randomize addresses, a network manager unit must employ an address allocation policy that randomly selects from a pool of elastic IP addresses, so that the connection status of EC2 instances and their relations with the network's surface can be modified in real time. What facilitates this swift movement is the range of coarse-grained and efficient application programming interfaces (APIs) that are available to cloud users for dynamically creating or deleting EC2 instances, allocating or releasing elastic IP addresses, and associating instances with elastic IP addresses.

Google Cloud Platform. Google Cloud Platform (GCP) allows for creating virtual machines in a cloud account, which are automatically given ephemeral IP addresses that are accessible through ports 80 and 22 as a default setting. The newly created virtual machine also receives an internal IP address and can join a virtual network, similar to the mechanisms in Amazon Web Services. The main difference is that, currently, GCP limits static external IP addresses to a single IP address per geographic region. API calls may be used to allocate a new IP address for each region. However, for a diverse set of IP addresses, the cloud user must create new virtual machines, or stop and start existing machines hoping for fresh IP addresses to be assigned to the corresponding virtual machines.

C. Evaluating Moving Target Defenses

Here the most closely related works are considered, which also aim to evaluate the effectiveness of randomization in moving target defenses.

Most of the previous work has focused on memory addresses because of the widespread deployment of address randomization in operating systems (for example, [26]–[28]), which prevent exploitation of memory corruption vulnerabilities. Address randomization or obfuscation in operating systems is analogous to randomizing IP addresses for a network, but uses the available space of memory addresses. When searching for a memory vulnerability in a stack pointer, address randomization helps, for example by randomizing the base address of a stack frame. Randomizing IP addresses in a cloud-based virtual network shares a similar goal with randomizing stack addresses in operating systems. Several works, starting with Shacham et al.'s seminal paper [29], have evaluated the effectiveness of address randomization defenses and found them to be vulnerable to derandomization attacks because of the limited entropy used in selecting addresses. Other works, such as the one by Conti et al. [30] demonstrate techniques to predict stack addresses.

Another proposed moving target defense, instruction set randomization [31], [32], seeks to disrupt code injection attacks by randomizing the instruction set. Evaluations of such defenses have also found them to be vulnerable to derandomization attacks [33], [34].

While the approach here has a similar goal to the previous works on evaluating the effectiveness of memory address space and instruction set randomization in practice, the focus of the presented work is on network address randomization and develop novel prediction methods specific to this domain.

Other works have taken an analytical approach to evaluating moving target defenses based on assumptions about adversaries and randomization. Carter et al. presented a game theoretic approach for comparing various move scheduling strategies for moving target defenses [35]. The work argues that simple randomization techniques could be easily detected by adaptive adversaries. The presented study shares a similar hypothesis, however, the focus is on developing a data-oriented approach for analyzing randomization using actual cloud-based IP address allocation. In a subsequent work, Winterrose and Carter developed a generic algorithm for modeling adaptive attackers [36]. They show that random defense strategies increase uncertainty, even when the defense uses a strategy based on a predefined model. The presented study sheds light on the diversity of IP addresses and ways to exploit IP address allocation to the attacker's advantage.

III. A Model of Attack and Defense

This section formalizes the problem of moving target addresses in the context of cloud-based virtual networks (also called target networks in this work) on cloud computing systems.

A. Definitions

Assume an attacker who controls of a set A of nodes that are connected to the Internet. The attacker is limited by the number of controlled nodes (which is proportional to the cost of the attack) and the time duration τ available for the attack.

The *attack surface* of the target network, S, is a set of hosts (represented by their IP addresses) that are externally exposed to the Internet. These hosts provide a port that responds to requests from any origin address. The target network's topology and relation to its clients are modeled as a connectivity digraph [24]. The *connectivity digraph* of the target network with respect to the nodes reachable from the Internet is a bipartite digraph $G = (V_1, V_2, E)$, such that $S \subset V_1$, $A \subset V_2$, and each $v \in V_2$ is bidirectionally reachable (using a TCP or UDP socket) from each $u \in S$. V_1 is the entire address space available to the target network for which a new virtual host could be created and assigned, while S represents virtual hosts created and connected to the Internet through G and also connected to an internal connectivity digraph G^*. We assume no host in the surface is an attacker ($A \cap S = \emptyset$). In reality, a host in the surface could be compromised, but we do not consider this case.

A *denial-of-service attack* (or simply an *attack*) is represented as a binary relation $R \subset A \times S$. An attack is a process to overload software on a host $u \in S$ so that u can no longer respond to legitimate clients.

B. The Attack Model

Consider an attacker whose goal is to deny service to the maximum number of hosts in the target network for the longest possible duration given the available (limited) resources. The attacker's goal could also be to gain remote access to some of the hosts in the target network through remote privilege escalation. Although the nature of a remote access attack and a denial of service attack are different, from an IP address randomization perspective, both attacks depend on knowing the target IP addresses for some period of time needed to carry out the attack. Hence, defenses based on hiding and moving IP addresses will be similarly effective.

A denial-of-service attack is performed at the application layer or at the link layer, for example by flooding the HTTP service with too many unsolicited requests. Attackers can communicate with Internet gateways in the target network and can set up experimental accounts in the target network's cloud computing system. The attacker is assumed to know the target network's cloud computing provider, and has identified an initial set of IP addresses that serve as the Internet gateway to the target network.

Assume that the target network's surface changes over time based on a proactive schedule with random time intervals. The attacker faces a problem when the target network enlarges S. Assuming IPv4 addresses are used in the target network, to detect an increase in number of hosts, the simplest method is scanning the possible IP addresses V_1 (or a similarly large subset of it), to find new hosts in S.

This blind attack method faces challenges. First, the time when new hosts are added to S is unknown and can be selected randomly by the server (or adjusted based on detection of potential attacks). Second, given a policy with short time intervals for changing S, the attacker must waste resources blindly scanning the network because V_1 could be very large. The viable alternative to an exhaustive search is to attempt a reconnaissance attack in preparation for launching a denial-of-service attack. In this work, the reconnaissance effort is realized by collecting and analyzing data from the target network's cloud computing provider.

The attacker's goal is to exploit the entire surface S in the target virtual network. As $S \subset V_1$, and V_1 is a large address space, the core challenge is to reduce the search space for S to a small subset of V_1. Thus, the attacker attempts a reconnaissance attack by collecting data from the target network's cloud computing provider. As part of a reconnaissance attack, the attacker knows a subset of $s \subset S$. Also, the attacker collects IP addresses from the cloud computing provider by requesting and releasing IP addresses as a normal cloud user. This data collection forms a dataset $\mathcal{D}$ of IP addresses (Section IV-A).

At the start of the attack, the attacker knows s and has collected $\mathcal{D}$. At some later time, a secure system trusted by the target network changes the surface S to S' (The attacker cannot tamper with or control the target network's account with the cloud computing provider.) $S \cap S'$ is not necessarily empty. The precise goal of the attacker is to use s and $\mathcal{D}$ to predict all the values in S'.

C. The Defense Model

Target networks serve a specific audience by hosting applications that are accessible through the Internet. While many applications use the HTTP (or the TLS-based HTTPS) protocol, this work makes no specific assumptions on the choice of network protocol. The target network has at least one host for the application with no firewall rules restricting access to the application. Cloud computing platforms provide ways to ban unwanted IP addresses from reaching the target network, which do not seem practical. With respect to the defense model, we assume:

1) the target network has a total of N hosts in the surface,
2) according to a schedule, a total of N addresses are requested from the cloud computing platform, which replace the IP addresses for the N hosts,
3) for each change, the IP addresses are requested once and are assigned to the N hosts,
4) the target network does not memorize the previously allocated IP addresses, and
5) the target network does not react to denial of service attempts.

D. IP Address Allocation

Cloud computing providers create and control networks, which host virtual networks created by cloud users. A cloud computing provider serves many cloud users. A subset of cloud users request public IP addresses either by asking for an elastic IP address that can be associated with a virtual machine or by creating Internet accessible virtual machines. In both cases, a function, denoted Ω, within the cloud computing platform allocates an IP address for the cloud user. The IP address is reserved for the user until either the user releases the elastic IP addresses or destroys the virtual machine for which an IP address was allocated.

A denial-of-service attacker's goal is to learn the behavior of Ω and use the gained knowledge to attack a target network in the cloud computing provider. The internal logic of Ω is not disclosed to any user, but we can make some assumptions about its behavior:

1) cloud computing providers assign IP addresses based on the region in which the target network is hosted,
2) a large number of cloud users coexist in each region,
3) regions vary in the size of the set of available IP addresses and the IP address diversity,
4) an IP address allocated for a cloud user is only released after an explicit release request from the user, and
5) IP addresses are selected from a pool of available IP addresses in a region via the unknown random function Ω.

Cloud computing providers may employ various implementations of Ω which, due to IP address capacity and availability in regions, ultimately result in address allocations with low entropy when used in moving target defenses.

IV. Attack Strategies

This section presents an overview of the learning and prediction model, and a generic attack algorithm (Section IV-A), followed by the design of three different heuristic search methods for finding newly allocated IP addresses: random (Section IV-B), frequency-based (Section IV-C), and a clustering model (Section IV-D).

A. Overview

Our high-level approach is to analyze the collected data (the learning phase) and generate predictions for the next IP addresses in the target network's surface S, which will be used by the attacker to deny access to the hosts with newly allocated IP addresses (the attack phase).

The Learning Phase. The process of learning and predicting the next set of IP addresses by the target network is depicted in Figure 1. The cloud computing platform uses an IP address allocation algorithm, which employs a function Ω that outputs an IP address, given a requesting client's region, and other *hidden* parameters that are obscure to the requesting client. The Ω function responds to IP address allocation requests from a data collector (which can be implemented using tools available from the cloud computing provider, as discussed in Section V). The IP addresses received from Ω form the attacker's dataset $\mathcal{D}$, comprised of rows of N IP addresses with the time of allocation. Each data point (referred to as an allocation) is a set of N (distinct) IP address assignments. Data points are recorded when the user requests N new IP addresses, and an IP address allocation algorithm by the cloud computing platform provides the N addresses to the user. The collected data is used to train models for predicting IP addresses to search in the attack.

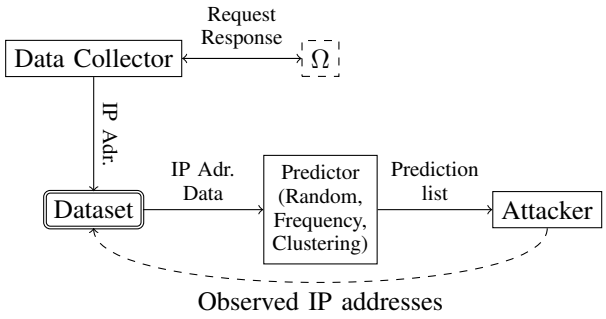


Fig. 1: The process of predicting IP addresses in the target network. Ω is a function used by the cloud computing provider to allocate IP addresses. The attacker's aim is to predict its behavior using either a random, a frequency, or a clustering strategy (Section IV). Observed IP addresses are recorded and used in consequent predictions.

Attack Algorithm. Algorithm 1 summarizes one attack iteration. We assume each attack iteration starts immediately after the target network finishes assigning N fresh IP addresses to the target hosts, and the attackers goal is to find those N IP addresses. The algorithm receives the number N of servers to target and a prediction model, M, which is generated using one of the attack strategies presented later.

The outer for-loop iterates in order over the predictions made by M. The `predict` function returns an ordered list of the predictions make by the model M. Each prediction is an IP address prefix; in our experiments, we predict the first three bytes of the IP address. The unspecified bits will be searched by brute force. The `complete` function returns all possible IP addresses that can be produced by completing the predicted prefix. We assume the attacker can detect whether the tried IP address belongs to the target network, and that the attacker has no interest in attacking other networks in the same cloud computing platform. If the attack succeeds, the attacker records the IP address in the observation list O. The attack completes once N IP addresses were successfully attacked or after exhausting all the predicted prefixes. The model may be updated based on information learned from the attack, for example, updating the frequency counts of prefixes based on success or failure in the attack.

Predicting Addresses. The prediction model uses the collected data to form a list of predicted IP addresses, sorted in decreasing order of priority, that are likely to be allocated in a consequent call to the Ω function. The priority of an IP address in a prediction list could be based on several strategies. As depicted in Figure 1, this work investigates three strategies for forming the prediction list. The simplest strategy is to form

Algorithm 1 One iteration of a denial of service attack on N hosts in the target network.

Require: N, M
$O \leftarrow [\,]$
for $A \in \texttt{predict}(M)$ **do**
  for $A' \in \texttt{complete}(A)$ **do**
    attack the server on A'
    if attack on A' succeeded **then**
      append(A', O)
      if $|O| = N$ **then**
        return
      end if
    end if
  end for
end for
update(M) with information from this iteration

the prediction list by randomly selecting IP prefixes from the dataset and exhaustively searching the last byte of each of those prefixes. The random strategy (Section IV-B) gives equal priorities to all observed prefixes in the prediction list. The hypothesis for the random strategy is that Ω behaves semi-randomly, subject to unknown constraints. Another strategy is to assume the distribution of IP prefixes in the dataset predicts the future IP addresses, so the frequency strategy (Section IV-C) gives prediction priority to IP prefixes that appear most frequently in the dataset. Finally, IP addresses can have temporal relationships, that is, occurrence of one may indicate occurrence of another in a subsequent allocation by Ω. In a clustering strategy (Section IV-D), a Markov transition matrix can capture such relations among clusters of IP addresses.

In all prediction strategies the attacker receives a prediction list and attacks the IP addresses in decreasing order of priority. The random and frequency strategies can directly form prediction lists from the dataset. The clustering strategy clusters the dataset first and moves to generate a Markov transition matrix (Section IV-D). Our experimental results (Section V) find that the simple frequency-based predictions are most successful, and substantially reduce the expected cost of the attack. The clustering approach can be useful when Ω produces more time-dependent IP address allocations, although our experimental results do not find this to be common in the collected dataset.

B. Random Attacker

In this strategy, the model M for Algorithm 1 just returns a list of all observed prefixes in random order. The rationale for the random attacker is to make a simple and fast attack procedure that can be realized in practice, and also depends on the data collected for the clustering attacker.

C. Frequency Attacker

The frequency attacker draws predictions from a sorted (in decreasing order) list of prefixes based on the frequency of appearance in the dataset. The frequency attacker's hypothesis is that address prefixes with high frequency are likely to reappear. Initially, M will return all unique prefixes (first 24-bits of IP addresses) sorted in decreasing order of their frequencies in the dataset. After executing an attack iteration, the model updates the frequencies for the next attack iteration.

D. Clustering Attacker

The clustering attacker executes an attack when the target network's surface changes. Ideally, the attacker knows the precise timing of assigning new IP addresses to hosts in the surface. The attack iteration predicts the set Q of IP addresses recently assigned to the hosts using the most likely transition from the N IP addresses that were observed in a prior (ideally, the previous) attack iteration. To establish the most likely transitions, the attacker clusters the dataset and produces a Markov transition matrix as described next.

Computing Clusters. The aim is to predict the leading 24 bits of the address, leaving the last byte for a brute force attack. The proposed method can generalize to clustering the leading $x \leq 32$ bits of the address. The clusters are expected to represent various networks within the cloud computing infrastructure. A centroid-based clustering scheme using the Hausdorff distance function [37], [38] is used on the collected data points. Consider the historical data as a time series $\mathcal{S}$ of sets of IP addresses $\mathcal{A}_k = \{A_{k1}, \ldots, A_{kN}\}$, where each A_{ki} consists of the numerical value of the leading 24 bits of an IP address. The problem is, given $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m$, predict the next set $\mathcal{A}_{m+1}$. Suppose that all the sets $\mathcal{A}_k$, $1 \leq k \leq m$, cluster into ℓ coherent clusters $\{C_i\}_{i=1}^{\ell}$ with cluster prototypes $\{P_i\}_{i=1}^{\ell}$. Cluster C_i is, by definition, all (sets) $\mathcal{A}_j$ closer to P_i than to any other $P_k \neq P_i$. (In case of a tie for the minimum distance, arbitrarily assign $\mathcal{A}_j$ to the cluster with the lowest index.)

The distance between two nonempty finite sets (of integers) $\mathcal{A}$ and $\mathcal{B}$ is the Hausdorff distance $\rho(\mathcal{A}, \mathcal{B})$, defined as follows. For integer x, define

$$d(x, \mathcal{B}) = \min_{y \in \mathcal{B}} |x - y|$$

and the directional Hausdorff distance

$$H(\mathcal{A}, \mathcal{B}) = \max_{x \in \mathcal{A}} d(x, \mathcal{B}).$$

The Hausdorff distance (which is a metric in the sense of topology) is then

$$\rho(\mathcal{A}, \mathcal{B}) = H(\mathcal{A}, \mathcal{B}) + H(\mathcal{B}, \mathcal{A}).$$

The prototype P_k for cluster C_k has the property that

$$P_k \in C_k \text{ and } \sum_{\mathcal{A} \in C_k} \rho(P_k, \mathcal{A}) = \min_{\mathcal{B} \in C_k} \sum_{\mathcal{A} \in C_k} \rho(\mathcal{B}, \mathcal{A}).$$

The clustering can be done using a standard iterative k-means procedure. During the clustering, $\ell = \lfloor 0.05m \rfloor$ clusters are created, where m is the number of IP address sets. The value of ℓ can be adjusted to improve the prediction accuracy. Initially, the available data points (sets of IP addresses) are equally distributed among all clusters (except possibly for the last cluster) by randomly selecting data points from the dataset and filling up the clusters 1, 2, …, ℓ. Then, the prototypes

are computed based on the Hausdorff distance function, as described earlier.

Computing the Transition Matrix. The hypothesis is that the allocation of some IP addresses precedes the allocation of others in a repeated trend. Thus, to form a prediction set, a Markov transition matrix is used to capture the most likely transition of IP address sets $\mathcal{A}_k$ among clusters by observing IP addresses in the time series. Using the empirical probability in the transition matrix, one can predict the next group of IP addresses that are likely to be used by the IP address assignment function Ω.

Define a discrete Markov process transition matrix T, whose (i, j) element T_{ij} is the (empirical) probability of a transition from a set in cluster C_i to a set in cluster C_j:

$$T_{ij} = \big(\text{number of transitions } \mathcal{A}_k \to \mathcal{A}_{k+1} \mid \mathcal{A}_k \in C_i,\ \mathcal{A}_{k+1} \in C_j,\ 1 \le k < m\big)/\ell.$$

Thus, if $\mathcal{A}_m \in C_r$, the maximum element T_{rs} in the rth row of T gives the most likely transition from $\mathcal{A}_m$ to be a set in C_s. The entire cluster C_s is selected to predict the next IP address set. T can be large and sparse for large datasets, so a row-based sparse matrix storage format is used for T.

Producing Predictions. The core strength of the clustering attacker is in using empirical probabilities of transitions among sets of IP addresses as observed in the collected dataset. Precisely, the most likely transition is found by first finding the cluster C_R containing the previous set of N IP addresses. For each empirical probability $T_{Rj} > \tau$ in column j and row R of the transition matrix T, sorted in descending order, the attack iteration retrieves the cluster C_j. The success of the attack is measured as the cardinality of the union of the sets $y \cap Q$ taken over all $y \in C_j$ and over all j for which $T_{Rj} > \tau$, where τ is a probability threshold. For our experiments, we use $\tau = 0$, so the attacker eventually tries all distinct addresses in all sets $y \in C_j$ for all j, continuing until this cardinality reaches N or the union is complete.

V. Evaluation

The implementation of Algorithm 1 and the collected data set are available under an open source license at https://github.com/kussl/ClusterAttack. Next, the procedure for data collection is described. Section V-B presents an analysis of the collected IP address data, and Section V-C reports on results from simulated attack experiments. The effect of increasing the size of address space by combining IP addresses from multiple regions is studied in Section V-D. Finally, parallelized attacks are considered in Section V-E.

A. Data Collection

Using the tools available for normal users, we used an AWS user account and a GCP user account to allocate and record IP addresses at fixed time intervals (the GCP user account was created at the time of study and a previously-created AWS account was used). Simultaneous IP allocation is limited to a specific number in both platforms, but can be programmed using the software development kits available from each one. During data collection, neither AWS nor GCP generated any security alerts nor did they limit the data collection scripts (developed for this study) except for general limitation rules that applies to any normal user.

Amazon Web Services. AWS only allows users to allocate a maximum of five *elastic IP addresses*, which can be associated with an EC2 instance (a virtual machine in AWS), and later disassociated from that instance and reused for another one. The default limit of five elastic IP addresses may be increased by a request from AWS. Note that an AWS account can create many EC2 instances and receive new static IP addresses for each one. Thus, if more than five IP addresses are needed, EC2 instances have to be created and destroyed, which is a time consuming process.

We collected data from several geographically-separated regions in AWS. Regions were selected from various continents (`AP_NORTHEST-1` (Tokyo) in Asia; `CA-CENTRAL-1` (Canada), `US-EAST-1` (Virginia), and `US-WEST-1` (California) in North America; `EU-WEST-1` (Ireland) and `EU-WEST-3` (Paris) in Europe; and `SA-EAST-1` (São Paolo) in South America), but the choice of regions was arbitrary. Before collecting data, there was not enough information about regions to decide on the quality of data being collected. However, as the results show (Section V-B), the selected regions exhibit diverse behaviors.

In AWS, regions can either use *classic* elastic IP addresses allocated using `EC2-Classic`[1], or *default* virtual private cloud (VPC) addresses allocated in `Default-VPC`. In `EC2-Classic`, IP addresses are allocated in a flat network shared by other users. In `Default-VPC`, an EC2 instance receives a new IP address in a default VPC for the region as soon as the instance is created. The description from AWS does not provide details on the differences of IP address prefixes in either mode. AWS also does not provide details on restricting the actual subnetworks from which IP addresses are allocated. However, `EC2-Classic` is no longer supported for accounts that were created after 4 December 2013. Since the account used for experiments was created before then, the data collection script could collect data from `EC2-Classic` in several regions of AWS (`US-EAST-1`, `AP-NORTHEAST-1`, `EU-WEST-3`, and `SA-EAST-1`). Other regions received IP allocations from `Default-VPC`.

Google Cloud Platform. GCP imposes a different policy: only a single global elastic IP address can be created at a time. Thus, at the time of this study, requesting elastic IP addresses in GCP for the purpose of collecting a diverse dataset of IP addresses wasn't viable. Instead, the process was to create five virtual machines to record IP addresses, destroy the virtual machines, and then recreate new ones.

B. Analyzing Collected IP Addresses

This section presents a statistical overview of the collected data, revealing some aspects of how AWS and GCP allocate IP addresses. Throughout this section, the goal is to analyze data patterns primarily for three-byte prefixes (first 24 bits of

[1] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-classic-platform.html

Region	Days	Total IP Addresses	Different IP Addresses	3-byte prefixes
AP-NORTHEAST-1	46	30,689	7,236	235
CA-CENTRAL-1	42	28,578	15,836	439
EU-WEST-1	109	120,142	7,812	550
EU-WEST-3	43	29,309	13,766	203
SA-EAST-1	45	30,187	2,143	88
US-EAST-1	63	44,405	38,744	1666
US-WEST-1	46	30,965	3,907	258
GCP	42	29,025	42	42

TABLE I: Summary of IP addresses collected.

IP addresses). Later in Section V-C, the collected data is used to predict three-byte prefixes.

Number of IP Addresses and Prefixes. Table I shows the summary of data collected in all regions for the entire period of data collection. The goal is to show the dataset sizes and the number of unique prefixes and IP addresses in each dataset. All datasets were collected during the same period of time, except for EU-WEST-1, which was collected in two separate time intervals. The reason for variability in dataset sizes is due to occasional failure of the data collection scripts.

Completeness Analysis. A complete dataset is one that includes all possible three-byte prefixes in a given cloud computing platform region during a time interval, subject to constraints imposed by Ω (one such constraint could be limiting prefixes per groups of users). Analysis of completeness for full IP addresses is practically unattainable. Since an attacker can have access to a few cloud accounts at a time, and because other users do reserve IP addresses, attackers need long time intervals to collect all possible unique IP addresses. As Table I demonstrates, three-byte prefixes are far more limited and a complete dataset of all possible prefixes is a realistic possibility. Without internal information about Ω, proving completeness, even for three-byte prefixes, is not practical. Instead, we analyze the completeness of the available dataset *relative* to the specific time interval in which data was collected. The analysis is done in two ways: (1) counting the number of new data points observed each day (Figure 2), and (2) counting the number of possible values in the last byte for each observed three-byte prefix (Table III).

First, the goal is to estimate completeness by measuring how much new data is observed after a number of days during the course of data collection. The available dataset is sorted in an ascending order of full calendar days. Starting with the earliest day on which IP addresses were allocated, the number of new three-byte prefixes is shown in in Figure 2. The reported number counts the prefixes that were not seen in any previous day. The number of new prefixes converges to zero for all regions, but the time to converge varies by region. This analysis is useful for attackers to estimate the time interval required to observe and record a large number of prefixes in the dataset. Table II complements Figure 2 by showing the number of days until no more new three-byte prefixes were observed for the rest of days in the data collection period. Table II also captures the number of individual days on which no previously unseen data points were observed.

Table III shows the number of distinct values for the last byte for each observed three-byte prefix. On average 10% of the 255 possible values for the last byte are covered across the datasets. This supports the decision in this work to focus only on predicting the prefixes and enumerating the last byte. After predicting the prefixes, the attacker can start by enumerating the observed values for the last byte, and in case of trying all observed last byte values without success, the attack continues by enumerating the rest of the possible values.

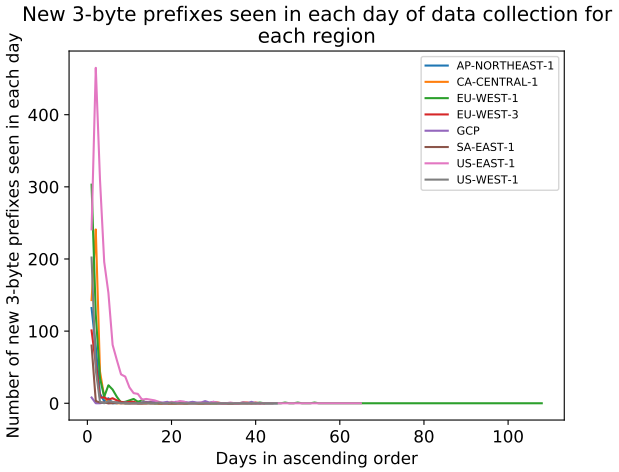


Fig. 2: Daily new three-byte prefixes in each dataset.

Region	Days	Days to observe all 3-byte prefixes	Days with no new 3-byte prefix
AP-NORTHEAST-1	46	11	38
CA-CENTRAL-1	42	6	35
EU-WEST-1	109	54	88
EU-WEST-3	43	40	23
SA-EAST-1	45	14	37
US-EAST-1	63	34	38
US-WEST-1	46	20	37
GCP	42	39	17

TABLE II: The number of days until no more new three-byte prefixes are observed in the remainder of the dataset (second column), and the number of days in which no previously-unobserved three-byte prefixes are observed (third column).

Frequency Distribution. To estimate the effectiveness of a simple attack strategy of just predicting the most frequently-seen three-byte prefixes, the datasets were analyzed to compute the relative frequencies of each observed prefix. A relative frequency is the number of appearances of a unique three-byte prefix divided by the sum of all frequencies. The summary

Region	Max.	Min.	Mean	Median
AP-NORTHEAST-1	75	1	31	30
CA-CENTRAL-1	66	8	36	36
EU-WEST-1	56	1	14	12
EU-WEST-3	174	1	68	63
SA-EAST-1	139	1	24	14
US-EAST-1	55	1	23	22
US-WEST-1	91	1	15	14
GCP	1	1	1	1

TABLE III: Statistical summaries for the number of values observed in the last byte for each three-byte prefix.

Region	Max.	Min.	Mean	Median
AP-NORTHEAST-1	0.01	≈ 0	0.00425	0.004
CA-CENTRAL-1	0.005	0.001	0.00229	0.002
EU-WEST-1	0.007	≈ 0	0.00179	0.0015
EU-WEST-3	0.012	≈ 0	0.0049	0.005
SA-EAST-1	0.067	≈ 0	0.01136	0.007
US-EAST-1	0.002	≈ 0	0.00066	0.001
US-WEST-1	0.024	≈ 0	0.00384	0.003
GCP	0.076	≈ 0	0.02379	0.018

TABLE IV: Relative frequencies of three-byte IP prefixes. The value in each cell is computed by dividing the number of occurrences of each unique IP prefix in a region by the total number of recorded IP prefixes. ≈ 0 indicates less than 0.0001.

Region	H_1	H_2	H_3	H	Max.
AP-NORTHEAST-1	1.106	1.846	4.632	5.342	5.46
CA-CENTRAL-1	0.692	0.699	5.462	6.049	6.084
EU-WEST-1	0.961	2.162	5.06	6.058	6.31
EU-WEST-3	0.565	0.565	5.05	5.06	5.313
SA-EAST-1	0.654	0.661	3.724	3.887	4.477
US-EAST-1	1.495	2.738	5.668	7.577	8.012
US-WEST-1	1.16	1.62	4.993	5.364	5.553
GCP	0.439	1.513	3.182	3.364	3.738

TABLE V: Shannon entropy values provide a measure of diversity in the dataset. H_b is the entropy of byte b, while H is the entropy of the first three address bytes. The last column shows the maximum possible entropy for each region's observations.

statistics for the computed relative frequencies are given in Table IV.

A frequency attack (Section IV-C) can use a sorted list of prefixes and their frequencies in the dataset to generate predictions. For example, given the collected dataset, picking the IP prefix with the maximum relative frequency in GCP, the attacker's best chance of correctly predicting an IP prefix in the next IP address allocation is 7.6%, given that Ω does not introduce a significant number of new IP prefixes within a time delta that would result in incorrect assumptions about the frequencies in the dataset.

Data Entropy. The difficulty of predicting the next address prefixes used by the target network is assessed now. The predictability of the first, second, and third bytes is captured by their (Shannon) entropy, $H_b = -\sum_i p_i \ln p_i$, $b = 1, 2, 3$, where p_i is the relative frequency of the ith byte value.

Table V shows Shannon entropy values. H_b is the entropy for byte b, while H is the entropy for the first three bytes taken together. The last column shows the maximum possible entropy $H = \ln K$ for K observations, which varies by region. More predictable prefixes correspond to entropy values much smaller than the maximum entropy.

Repetition Rates. Repetition in the dataset benefits the clustering attacker, which uses a transition matrix to predict the target server's choices. The rate of repetition for unique three-byte prefixes was measured. For each region R, the three-byte prefixes were collected from the dataset. Then, to measure the repetition, for each unique three-byte prefix A_i, the number $d_R^{A_i}$ of three-byte prefixes observed in the dataset between two occurrences of A_i in R was recorded. Since A_i can repeat more than once in R, Table VI shows summary statistics (maximum, minimum, mean, and median) for all $d_R^{A_i}$ values in each region R.

Region	Max.	Min.	Mean	Median
AP-NORTHEAST-1	13711	1	233	125
CA-CENTRAL-1	7346	1	432	293
EU-WEST-1	40916	1	546	277
EU-WEST-3	26801	1	198	98
SA-EAST-1	8917	1	87	21
US-EAST-1	31383	1	1582	1013
US-WEST-1	14203	1	255	141
GCP	27165	1	25	7

TABLE VI: Gaps between three-byte prefixes' repetitions. Within each region, for each unique three-byte IP prefix, the reported statistic is for the number of three-byte prefixes recorded between every repetition of the three-byte IP prefix.

C. Attack Simulation

Moving target defenses employ cycles of refreshing IP addresses to distract attackers. In each refreshing cycle, a set of N IP addresses S' are requested from Ω to replace IP addresses S in the surface. Recall that the goal of this study is to use data collected from AWS and GCP to predict IP addresses in S', in each refreshing cycle. In the rest of this section, three-byte prefixes are referred to as prefixes. A full search of the last byte is not always necessary. As the data in Table III shows, the average number of unique values for the last byte across various datasets is 1–68, and the maximum recorded number of unique values for the last byte is 174.

Attacker's guesses. The performance of each attack strategy is measured by the number of guesses required to predict one or more prefixes (depending on whether $N = 1$ or $N > 1$) in the target network during an attack iteration. An attack iteration starts at the beginning of each refreshing cycle and terminates if all N hosts are successfully discovered and attacked, or there are no more prefixes to try. The source of the guesses varies with the attacker. For the frequency and random attackers, the guesses come from the list U of unique prefixes, and differ only in how U is accessed (cf. Sections IV-C and IV-B). For the clustering attacker, the guesses come from the prefix sets in selected clusters, chosen in an order determined by transition matrix probabilities (Section IV-D). For all attackers, a given prefix is tried at most once — for the random (choose N prefixes at random from U) and clustering (examine each prefix in each prefix set in each selected cluster) attackers, a prefix may occur more than once.

The attack process consists of selecting an as yet untried prefix, completing it with a value in the last byte (searching through all possible values in the last byte), and attempting a denial of service attack on the formed IP address. As searching for the last byte, given a prefix, is trivial, the simulations only include the number of guesses for predicting prefixes.

Simulation method. The goal of simulations is to estimate how long an attacker would find N hosts. In our attack simulation, the dataset is divided into a training sequence with 70% of the original dataset, in the order collected from the IP address allocator of the region. Then, the remaining 30% of

the collected data are used as the test sequence. A complete attack simulation iterates through all prefix sets in the test sequence, with each *attack iteration* involving attacker guesses for a prefix set from the test sequence.

During each attack iteration, the prediction list L is formed according to one of the attack strategies: clustering (Section IV-D), frequency (Section IV-C), or random (Section IV-B). L is ordered according to priorities set by the attack strategies. The attacker tries prefixes with the highest priorities first. Prefixes are selected in order from L, representing attacker guesses (described earlier). Guessing continues until either all prefixes in $\mathcal{A}^*$ are correctly guessed or there are no more candidate prefixes in the prediction list. Each trial of a prefix increments a guess counter. At the end of each iteration, a guess counter value close to N is desired by the attacker. We do not include the cost of scanning the last byte in the IP address in the results, as we are assuming a naïve strategy of just scanning all addresses over the last byte.

Simulation results. Simulation results should reveal the quality of prediction and indicate whether solely requesting IP addresses from Ω in a single region is an effective IP randomization technique. The box plots summarizing the results are shown in Figure 3. The results shown are for $N = 5$, which is the original number of IP addresses allocated in each iteration of collecting data. Note that modeling the prediction of a *set* of $N > 1$ IP addresses is qualitatively different from just iterating the prediction of $N = 1$ address multiple times.

There are several observations from the results of Figure 3, showing the summary statistics of simulations for all datasets. First, the frequency attacker achieves the lowest median number of correct guesses in all datasets, although the performance is generally close to that of the clustering algorithm. This suggests that the time series model behind the clustering is not necessary, i.e., the address prefix set $\mathcal{A}_{m+1}$ depends less on $\mathcal{A}_m$ than on the individual address prefix frequencies, independent of time. Second, the median number of guesses is a function of the unique number of prefixes observed in the dataset. For example, US-EAST-1 has 1666 unique prefixes and achieves a median of 1229, 1363, or 1445 guesses, depending on the attack strategy. Third, increasing the number of unique prefixes alone does not necessarily increase the entropy in all attack iterations, since for example, the first quartile of the box plot is just below 1,000 guesses. Finally, the total estimated number of guesses for full IP addresses is low and can be quickly checked by systems such as Zmap [39]. For example, taking the median number of possible values for the last byte from Table III, one can estimate the number of guesses to predict five *IP addresses* in US-EAST-1 as $1229 \times 22 = 27,038$, a reduction of approximately 30% from trying all 38,744 unique IP addresses observed in US-EAST-1. This number of guesses is not a significant burden on attackers, since checking IP addresses can be fast and can potentially be parallelized to bypass anomaly detectors.

A prediction list can fail to predict all N prefixes in an attack iteration. This happens with the clustering attacker when the clusters are too small. It can also happen with the random and frequency attackers if the actual prefixes were not observed during the data collection phase. To measure the occasional failures to detect prefixes, we divide the total number of predicted prefixes by the total number of prefixes used as choices of the target server in all iterations, forming the hit rate. In all the experiments, the three attack strategies had a hit rate of at least 90%, mostly above 95%, during all iterations, except for the random attacker that performed poorly with GCP, with a hit rate of 33% over the entire attack simulation.

D. Enlarging the Address Space

As the results of the previous section demonstrate, three-byte prefixes are frequently reused by the Ω function. Although resolving this vulnerability is out of the scope of this work and requires a deep analysis, an immediate remedy might be to increase the size of the available address space. One possible solution is to randomize the IP address of the target server by requesting addresses from multiple regions for which an experiment was designed. The goal is to simulate a defense strategy where IP addresses are requested from multiple regions. The attacker uses a dataset of addresses collected from all regions used by the defense, assuming the attacker's correct guess of all the used regions. However, the attacker does not necessarily know from which region the next IP address of the target server is selected. The advantage of the clustering approach is that in addition to predicting the next IP address, the data also assists the attacker in guessing the next region from which the IP address was requested.

In this experiment, IP addresses of multiple regions (all the regions in the collected datasets) were combined in a time series list:

$$(A_{11}, A_{21}, \ldots, A_{mn}),$$

where A_{ij} is the jth IP address from the ith region, forming a new dataset with an increased number of unique IP addresses. In a loop that starts with an empty dataset $\mathcal{D}^*$, in each iteration, a region's dataset $\mathcal{D}$ is randomly chosen and the next IP address from D is recorded in the new dataset $\mathcal{D}^*$. The IP addresses in $\mathcal{D}^*$ are recorded in the order they appeared at the time of collecting the original datasets. The new dataset is as large as the smallest original dataset.

The results of this experiment are depicted in Figure 4. As shown, the maximum number of steps to find the target three-byte prefixes is increased, as the number of unique combinations of three-byte prefixes is naturally increased due to combining a diverse set of addresses in the new dataset $\mathcal{D}^*$. Further, notice that the clustering approach performs with a better median, compared to the other two approaches. This is due to the ability of the clustering approach to predict the region and the three-byte prefix that are expected to be used by the target server. Comparing the results to the most diverse region, US-EAST-1, the attacker still has a considerable chance to find the next three-byte prefix in a small number of steps. The results suggest that lowering the predictability of IP addresses is not trivial and requires careful considerations.

E. Parallelized Attacks

A variation of the attack model is to parallelize the attack process to reduce the time required to predict the next IP

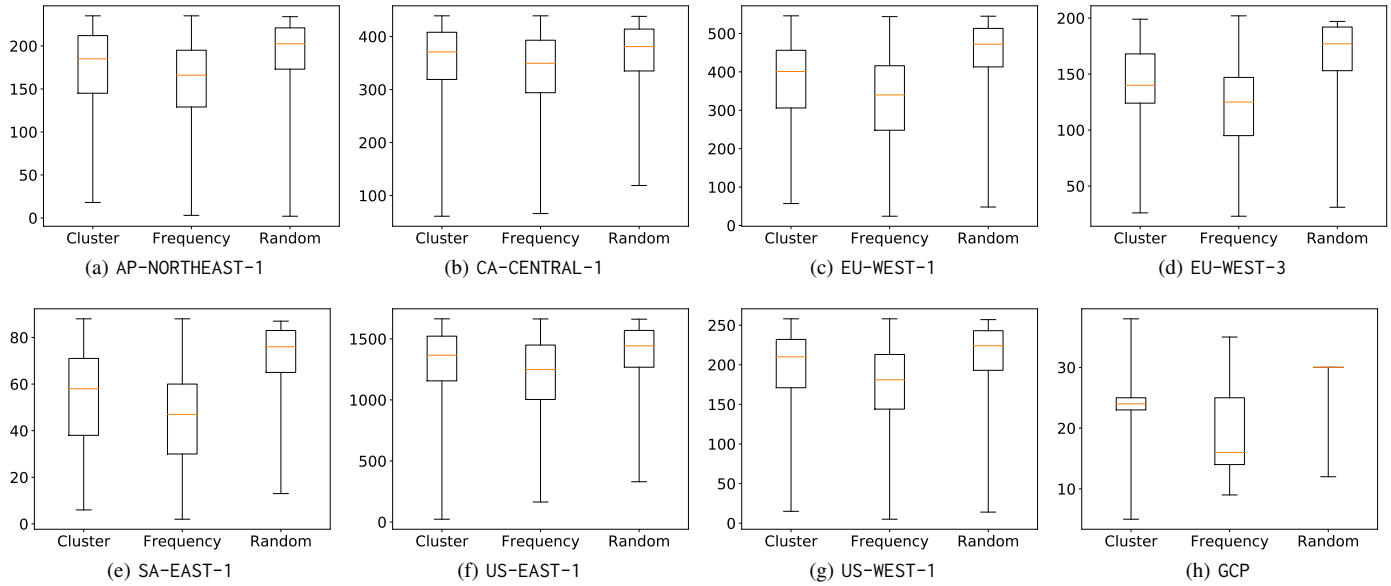


Fig. 3: Performance of simulated attack iterations. On each box plot, the x-axis shows the attack strategy, and the y-axis shows the number of guesses from the prediction list to predict the current prefixes used by the target server (note that the scales on the y-axis vary widely across the regions). Each box shows the minimum and the maximum (the whiskers), the median (the horizontal line in the rectangle), and the second and the third quartiles (the rectangle).

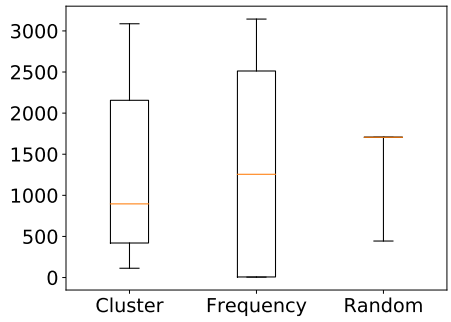


Fig. 4: Performance of simulated attack iterations with a dataset that combines IP addresses from multiple regions. For each IP address in the dataset, a region is randomly chosen and the next IP address in the time series is selected and recorded.

address. Parallelizing the attack on a specific target server can be done in three ways:

1) a single attacker with parallel attacking resources,
2) multiple independent attackers sharing no information, and
3) a coordinated attack by multiple attackers.

A parallel attack can reduce the time required to predict the target IP address but simultaneously executing attempts from multiple hosts. In cases where the attack is not limited by the response rate of the server, this can reduce the expected time of the attack by $1/N$ where there are N parallel attackers who are partitioning the address space to search (for example, by using the same prediction methods here, but dividing the guesses for the last byte of the IP address among the N attackers). This does not, however, reduce the resources required for a successful attack, just its expected time to success.

F. Limitations of the Study

The experiments and the dataset show a limited address space used in cloud computing platforms. Although the actual IP address allocation could be randomized, the availability of IP addresses affects the predictability of a moving target defense system. Given enough time and resources, attackers can design other data collection methods that would lead to collecting an even large dataset, revealing more patterns in the data. One limitation of this study is that only one data collection method was used in the learning phase. Using various data collection methods can be the subject of a future work, studying the effect of the various ways an attacker can learn about the behavior of Ω. Another limitation is the changing behavior of Ω. Attackers cannot guarantee a constant behavior in Ω, especially when such attacks become widespread. Thus, a continuous and adaptive learning process is required to keep attacks updated. Unless moving target defenses are redesigned to avoid relying solely on the behavior of Ω, the expectation is that attackers can design adaptive algorithms to chase the moves of target networks.

VI. Conclusion

The possibility of predicting IP addresses allocated by cloud computing platforms is alarming for moving target defenses that assume IP address allocations given by cloud services are highly unpredictable. As shown in Section V, an attacker can reliably predict cloud IP address allocations. Unless cloud computing platforms employ limitations on IP address allocation, attackers can conveniently update the database of

IP addresses and continue attacking various clients. Policies limiting IP address allocation would impose usability limitations that are difficult to implement when serving large virtual networks. Thus, designing moving target defense systems with a core mechanism that depends on freshly assigned IP addresses requires careful considerations to increase entropy in the selected IP addresses, disabling accurate predictions by attackers.

IPv6 addresses may be a secure alternative to IPv4 addresses as a platform for IP address randomization. Due to formatting differences and the IPv6 address space, evaluating the effectiveness of IPv6 addresses for randomization in cloud is a subject of future work.

REFERENCES

[1] A. Yaar, A. Perrig, and D. Song, "Pi: a path identification mechanism to defend against DDoS attacks," in *2003 Symposium on Security and Privacy, 2003.*, May 2003, pp. 93–107.

[2] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 217–228, Aug. 2005. [Online]. Available: http://doi.acm.org/10.1145/1090191.1080118

[3] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. ACM, 2002, pp. 71–82.

[4] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling "cool": Temperature-aware workload placement in data centers," in *Proceedings of the USENIX Annual Technical Conference*, ser. ATEC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 5–5. [Online]. Available: http://dl.acm.org/citation.cfm?id=1247360.1247365

[5] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 3, pp. 62–73, Jul. 2002. [Online]. Available: http://doi.acm.org/10.1145/571697.571724

[6] D. L. Cook, W. G. Morein, A. D. Keromytis, V. Misra, and D. Rubenstein, "WebSOS: protecting web servers from DDoS attacks," in *11th IEEE International Conference on Networks*, Sept 2003, pp. 461–466.

[7] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: an architecture for mitigating DDoS attacks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 176–188, Jan 2004.

[8] Q. Jia, K. Sun, and A. Stavrou, "MOTAG: Moving target defense against internet denial of service attacks," in *22nd International Conference on Computer Communication and Networks*, 2013.

[9] S. Venkatesan, M. Albanese, K. Amin, S. Jajodia, and M. Wright, "A moving target defense approach to mitigate DDoS attacks against proxy-based architectures," in *IEEE Conference on Communications and Network Security*, 2016.

[10] Q. Jia, H. Wang, D. Fleck, F. Li, A. Stavrou, and W. Powell, "Catch me if you can: A cloud-enabled DDoS defense," in *44th IEEE/IFIP Conference on Dependable Systems and Networks*, 2014.

[11] E. Al-Shaer, Q. Duan, and J. H. Jafarian, "Random host mutation for moving target defense," in *Security and Privacy in Communication Networks*, A. D. Keromytis and R. Di Pietro, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 310–327.

[12] J. Sun and K. Sun, "DESIR: Decoy-enhanced seamless IP randomization," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.

[13] J. Ullrich, K. Krombholz, H. Hobel, A. Dabrowski, and E. Weippl, "IPv6 security: Attacks and countermeasures in a nutshell," in *8th USENIX Workshop on Offensive Technologies (WOOT 14)*. San Diego, CA: USENIX Association, 2014. [Online]. Available: https://www.usenix.org/conference/woot14/workshop-program/presentation/ullrich

[14] J. Ullrich, P. Kieseberg, K. Krombholz, and E. Weippl, "On reconnaissance with IPv6: A pattern-based scanning approach," in *2015 10th International Conference on Availability, Reliability and Security*, Aug 2015, pp. 186–192.

[15] P. Foremski, D. Plonka, and A. Berger, "Entropy/IP: Uncovering structure in IPv6 addresses," in *Proceedings of the 2016 Internet Measurement Conference*, ser. IMC '16. New York, NY, USA: ACM, 2016, pp. 167–181. [Online]. Available: http://doi.acm.org/10.1145/2987443.2987445

[16] M. Wright, S. Venkatesan, M. Albanese, and M. P. Wellman, "Moving target defense against DDoS attacks: An empirical game-theoretic analysis," in *ACM Workshop on Moving Target Defense*, 2016.

[17] E. Miehling, M. Rasouli, and D. Teneketzis, "Optimal defense policies for partially observable spreading processes on bayesian attack graphs," in *Second ACM Workshop on Moving Target Defense*, 2015.

[18] R. Zhuang, A. G. Bardas, S. A. DeLoach, and X. Ou, "A theory of cyber attacks: A step towards analyzing MTD systems," in *Second ACM Workshop on Moving Target Defense*, 2015.

[19] R. Zhuang, S. A. DeLoach, and X. Ou, "Towards a theory of moving target defense," in *First ACM Workshop on Moving Target Defense*, 2014.

[20] H. Maleki, S. Valizadeh, W. Koch, A. Bestavros, and M. van Dijk, "Markov modeling of moving target defense games," in *ACM Workshop on Moving Target Defense*, 2016.

[21] D. Evans, A. Nguyen-Tuong, and J. Knight, *Effectiveness of Moving Target Defenses*. Springer, 2011.

[22] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Adversary-aware IP address randomization for proactive agility against sophisticated attackers," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, April 2015, pp. 738–746.

[23] S. Achleitner, T. F. L. Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha, "Deceiving network reconnaissance using SDN-based virtual topologies," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1098–1112, Dec 2017.

[24] H. M. J. Almohri, L. T. Watson, and D. Evans, "Misery digraphs: Delaying intrusion attacks in obscure clouds," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 6, pp. 1361–1375, June 2018.

[25] S. Achleitner, T. La Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha, "Cyber deception: Virtual networks to defend insider reconnaissance," in *8th ACM CCS International Workshop on Managing Insider Security Threats*, 2016.

[26] P. Team, "PaX address space layout randomization (ASLR)," 2003. [Online]. Available: https://pax.grsecurity.net/docs/aslr.txt

[27] S. Bhatkar, D. C. DuVarney, and R. Sekar, "Address obfuscation: An efficient approach to combat a board range of memory error exploits," in *USENIX Security Symposium*, 2003.

[28] L. Li, J. E. Just, and R. Sekar, "Address-space randomization for windows systems," in *22nd Annual Computer Security Applications Conference*, 2006.

[29] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh, "On the effectiveness of address-space randomization," in *11th ACM Conference on Computer and Communications Security*, 2004.

[30] M. Conti, S. Crane, L. Davi, M. Franz, P. Larsen, M. Negro, C. Liebchen, M. Qunaibit, and A.-R. Sadeghi, "Losing control: On the effectiveness of control-flow integrity under stack attacks," in *22nd ACM Conference on Computer and Communications Security*, 2015.

[31] E. G. Barrantes, D. H. Ackley, S. Forrest, T. S. Palmer, D. Stefanovic, and D. D. Zovi, "Randomized instruction set emulation to disrupt binary code injection attacks," in *10th ACM Conference on Computer and Communications Security*, 2003.

[32] G. S. Kc, A. D. Keromytis, and V. Prevelakis, "Countering code-injection attacks with instruction-set randomization," in *10th ACM Conference on Computer and Communications Security*, 2003.

[33] A. N. Sovarel, D. Evans, and N. Paul, "Where's the FEEB? The effectiveness of instruction set randomization," in *USENIX Security Symposium*, 2005.

[34] Y. Weiss and E. G. Barrantes, "Known/chosen key attacks against software instruction set randomization," in *22nd Annual Computer Security Applications Conference*, 2006.

[35] K. M. Carter, J. F. Riordan, and H. Okhravi, "A game theoretic approach to strategy determination for dynamic platform defenses," in *First ACM Workshop on Moving Target Defense*, 2014.

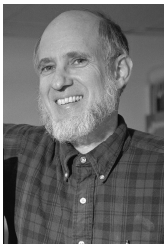
[36] M. L. Winterrose and K. M. Carter, "Strategic evolution of adversaries against temporal platform diversity active cyber defenses," in *Symposium on Agent Directed Simulation*, 2014.

[37] M. J. Atallah, "A linear time algorithm for the Hausdorff distance between convex polygons," *Information Processing Letters*, vol. 17, pp. 207–209, 1983.

[38] M. Chavent and Y. Lechevallier, "Dynamical clustering of interval data: optimization of an adequacy criterion based on Hausdorff distance," in *Classification, clustering, and data analysis*. Springer, 2002, pp. 53–60.

[39] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *22nd USENIX Security Symposium*, 2013.

Hussain M. J. Almohri received the BS degree in Computer Science from Kuwait University and the Ph.D. degree in Computer Science from Virginia Tech in 2013. He is currently an assistant professor of computer science at Kuwait University. He has co-founded a mobile payment startup and has advised a number of software startups in the Gulf region. His research focuses on systems and network security. He has served as a reviewer for several IEEE and IET journals and Kuwait Journal of Science.



Layne T. Watson (F '93) received the B.A. degree (magna cum laude) in psychology and mathematics from the University of Evansville, Indiana, in 1969, and the Ph.D. degree in mathematics from the University of Michigan, Ann Arbor, in 1974.

He has worked for USNAD Crane, Sandia National Laboratories, and General Motors Research Laboratories and served on the faculties of the University of Michigan, Michigan State University, and University of Notre Dame. He is currently a professor of computer science, mathematics, and aerospace and ocean engineering at Virginia Polytechnic Institute and State University. He serves as senior editor of Applied Mathematics and Computation, and associate editor of Computational Optimization and Applications, Evolutionary Optimization, Engineering Computations, and the International Journal of High Performance Computing Applications. He is a fellow of the National Institute of Aerospace and the International Society of Intelligent Biological Medicine. He has published well over 300 refereed journal articles and 200 refereed conference papers. His research interests include fluid dynamics, solid mechanics, numerical analysis, optimization, parallel computation, mathematical software, image processing, and bioinformatics.

David Evans is a Professor of Computer Science at the University of Virginia. He is the author of an open computer science textbook and a children's book on combinatorics and computability. He was Program Co-Chair for ACM Conference on Computer and Communications Security (CCS) 2017, and for the 31st and 32nd (2010) IEEE Symposia on Security and Privacy (where he initiated the SoK papers). He has SB, SM and PhD degrees in Computer Science from MIT and has been a faculty member at the University of Virginia since 1999.