6Hit: A Reinforcement Learning-based Approach to Target Generation for Internet-wide IPv6 Scanning

Bingnan Hou*, Zhiping Cai*, Kui Wu†, Jinshu Su* and Yinqiao Xiong*‡
*School of Computer Science, National University of Defense Technology, China
†Department of Computer Science, University of Victoria, Canada
‡Department of Electronic Information and Electrical Engineering, Changsha University, China
E-mails: {*houbingnan19, zpcai, sjs*}*@nudt.edu.cn, wkui@uvic.ca, yq.xiong@ccsu.edu.cn*

IEEE INFOCOM 2021 - IEEE Conference on Computer Communications | 978-1-6654-0325-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488794

***Abstract*—Fast Internet-wide network measurement plays an important role in cybersecurity analysis and network asset detection. The vast address space of IPv6, however, makes it infeasible to apply a brute-force approach for scanning the entire network. Even worse, the extremely uneven distribution of IPv6 active addresses results in a low hit rate for active scanning. To address the problem, we propose 6Hit, a reinforcement learning-based target generation method for active address discovery in the IPv6 address space. It first divides the IPv6 address space into different regions according to the structural information of a set of known seed addresses. Then, it allocates exploration resources according to the reward of the scanning on each region. Based on the evaluative feedback from existing scanning results, 6Hit optimizes the subsequent search direction to regions that have a higher density of activity addresses. Compared with other state-of-the-art target generation methods, 6Hit achieves better performance on hit rate. Our experiments over real-world networks show that 6Hit achieves 3.5% - 11.5% hit rate for the eight candidate datasets, which is 7.7% - 630% improvement over the state-of-the-art methods.**

***Index Terms*—IPv6, Internet-wide scanning, Network measurement.**

I. Introduction

The Internet is moving unavoidably towards IPv6, as the Internet Assigned Numbers Authority (IANA) allocated the last blocks of IPv4 address space to the Regional Internet Registries (RIRs) in early 2011 [1]. IPv6, the next generation Internet Protocol, has a huge address space, and it is widely implemented and rapidly adopted in recent years. In October 2019, nearly 30% of Google users accessed their services via IPv6 [2]. In the meantime, the number of active IPv6 BGP entries in routing table is also increasing rapidly [3].

IPv6, however, poses a new challenge in Internet measurement, which is critical to all ISPs. In the past, Internet measurement has greatly benefited from the advances of modern hardware and computational power that facilitate effective Internet-wide scanning. Asynchronous scanning tools like ZMap [4] and Masscan [5] have drastically enhanced our capability of conducting Internet-wide network surveys, e.g., topology discovery [6], [7], IP address analysis [8], [9], and geolocation [10]. As to cybersecurity, network device search engines like Shodan [11] and Censys [12] can acquire Internet-wide asset data for evaluating network security, discovering vulnerabilities, and tracking remediations [13]. Nevertheless, these tools are not effective when applied for IPv6-based Internet, because the huge address space of IPv6 renders comprehensive scans nearly impossible.

This raises the question for ISPs on how to quickly evaluate their network assets and the status quo of customer usage. This also presents a challenge for measurement researchers on how to obtain worldwide network measurements for IPv6. Efficient address discovery in IPv6 address space is the key to answering the above challenges and also the foundation for many IPv6-based network services such as neighbor discovery [14] and security analysis [15]. We are thus motivated to develop an efficient and effective approach for Internet-wide IPv6 scanning.

We propose 6Hit, a reinforcement learning-based target generation method which uses a set of known active addresses, also called *seeds*, to efficiently discover active addresses in the IPv6 address space. Instead of randomly generating addresses, 6Hit utilizes the information provided by the seeds and learns the better search directions according to the feedback during the scanning process. To be more specific, it utilizes the seeds to partition the address space and takes the scanning result of each step as a reward to determine better search directions on the next step. Meanwhile, to manage the huge scale of the IPv6 address space, 6Hit incorporates the key notion of a probe budget that specifies the constraint on the number of total probe packets. Note that the performance of address discovery algorithms is generally evaluated by the *hit rate*, defined as the ratio of the total number of discovered active addresses over the total number of probe packets. Clearly, a high hit rate is needed for an effective solution to Internet-wide IPv6 scanning. In this context, our experiments show that the state-of-the-art can only achieve about 6% hit rate at most. 6Hit aims to push this limit.

In a nutshell, 6Hit considers IPv6 addresses as high-dimensional vectors in the address space and adopts a reinforcement learning approach. It first performs divisive hierarchical clustering on the seeds to construct a tree structure named a space tree, which represents a partition of the address space. Each region in the address space is associated with a prior of the scanning reward according to the distribution of seeds. Then, 6Hit distributes the number of probe packets proportional to the expected reward of each region. 6Hit adopts a sequential search strategy, and in each iteration, the expected reward of each region is modified based on the the evaluative

978-1-6654-0325-2/21/$31.00 ©2021 IEEE

feedback from existing scanning results, to obtain the better directions for subsequent probes. This reinforcement learning approach makes 6Hit resilient to the initial seeds, while the performance of other state-of-the-art target generation methods are highly subjective to the initial seeds [16], [17].

Like all model-free reinforcement learning approaches, 6Hit also needs to make a balance between exploration and exploitation. In particular, the probability of generating the target addresses in the high density region (of active addresses) may become higher and higher and the algorithm may lose the capability of exploring other high density regions. To avoid this, 6Hit adopts a variety of methods, e.g., stopping exploring some high density regions and regenerating the space tree.

The main contributions of the paper are as follows:

- We present a dynamic target generation method, 6Hit, for Internet-wide IPv6 scanning space. With a small number of seeds, it can quickly generate a large number of active addresses. This capability lays a foundation for other follow-on research such as topology discovery, security scanning in the IPv6 Internet.
- To the best of our knowledge, 6Hit is the first to apply reinforcement learning approach to IPv6 active scanning. By taking advantage of the real-time feedback from the historical scanning results (i.e., reward from the environment), it dynamically adjusts the search directions towards regions that contain more active addresses.
- Real network experimental results show that 6Hit achieves the highest hit rate compared with state-of-the-art target generation methods. In particular, 6Hit fulfills this achievement with very little knowledge on the network under detection.

II. Related work

IPv6 target generation using seeds were first studied by Barnes et al. [18] in 2012. They assumed that the known active addresses provide information on the use of addressing schemes. Subsequent researches on target generation are all based on this hypothesis, that is, the seed information is helpful in discovering more new addresses. So far, related research has exploited both the structural information and the statistical information in the seeds. Accordingly, we group related research into two classes: structural information-based and statistical information-based.

In the first class, the structural information of seeds is mainly used to determine the scanning area [17]–[20]. Murdock et al. [17] proposed 6Gen, which assumes that the address space with high density seeds is more likely to have undiscovered active addresses. 6Gen greedily expands each seed as a center of each cluster to generate the target addresses by maintaining the maximal seed density and the minimal scale. Liu et al. [20] proposed 6Tree, which takes advantage of a space tree formed from seeds' structure to divide the IPv6 address space. 6Tree calculates the density of active nodes on the space tree according to the known active addresses that are loaded on the node. It then generates target addresses based on the density of active nodes. In the second class, the statistical information extracted from seeds is exploited to guide the target address generation [16], [21], [22]. Foremski et al. [16] introduced Entropy/IP, an algorithm for learning patterns from seeds, which utilizes empirical entropy to group adjacent nybbles of IPv6 addresses into segments and uses Bayesian network to model the statistical dependencies between values of different segments. This learned statistical model is used to generate target addresses for scanning.

Considering that the statistical model usually needs a large number of seeds to achieve a better performance, we adopt the strategy of utilizing the structural information of seeds to generate the targets. A main drawback of this type of method is that the initial seeds usually have a large impact on the performance (i.e., hit rate). 6Hit successfully solves this problem with several novel ideas, e.g., adjusting search directions based on existing scanning feedback, and introducing several new ideas in the scanning to strike a good balance between exploration and exploitation.

III. Background on IPv6 Target Generation

A. Problem Statement

Each network device has at least one allocated IP address and opens certain ports to the Internet for communication. These ports correspond to certain protocols. If a scanner sends packets following a certain protocol to a target address and the target responds, we consider the target address to be active under this protocol. Without loss of generality, we assume that the scanner sends probing packets following the same protocol type and assume that the set of all active addresses under this protocol in the address space X is $\mathcal{A}$. Assume that we already know seeds C, i.e., a subset of active addresses in $\mathcal{A}$. Assume that we are given a fixed budget on the total number of probing packets. *The problem of target generation is to find the active addresses in $\mathcal{A}$ as many as possible within the budget constraint.*

B. Structural Information and Space Partition

An IPv6 address can be represented as a hexadecimal string with 32 nybbles. Alternatively, we can consider an IPv6 address as a vector, called *address vector*, in a 32-dimensional space (i.e., address space). The value in each dimension of the vector is an integer in $[0, 15]$ (i.e., the value range of a nybble). Following the convention, we use the wildcard symbol "*" to denote a nybble which can be any value in $[0, 15]$.

Structural information-based approaches partition the address space X into regions. In the following we use regions and subspaces interchangeably. Since the density of active addresses in each region is initially unknown, structural information-based approaches should find a way to learn the address density in each region and generate targets in high-density regions to increase the hit rate. To this end, they assume that the seeds are random samples from the set of all active addresses $\mathcal{A}$ (in the address space X) and explore the hidden information in the seeds to facilitate the partitioning of X.

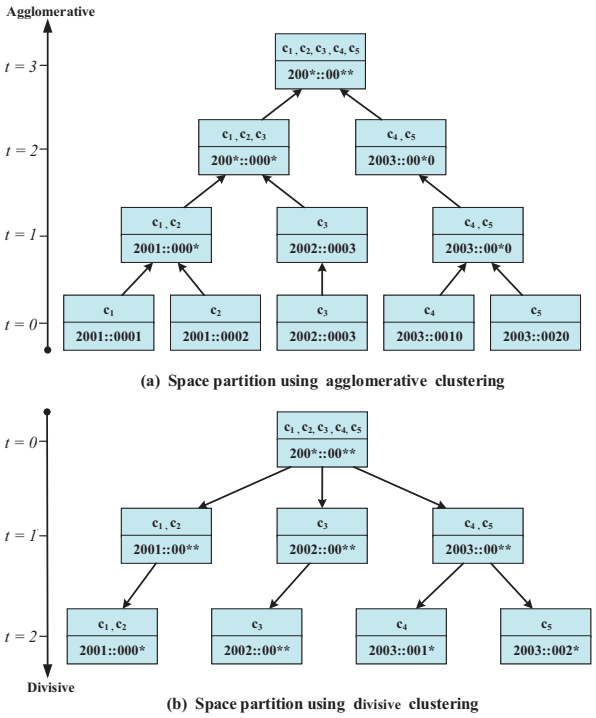

Fig. 1. Examples of space partition using hierarchical clustering.

Existing research mainly adopts a hierarchical clustering algorithm to group seeds into a hierarchy of "tree" of clusters, each cluster representing a region in the address space X. This clustering algorithm can use either agglomerative or divisive clustering, depending on whether the hierarchical decomposition is formed in a bottom-up (merging) or top-down (splitting) approach [23].

Examples of agglomerative and divisive clustering of seeds are shown in Fig. 1 (a) and (b), respectively. As shown in Fig. 1 (a), the agglomerative hierarchical clustering (AHC) algorithm starts by instantiating with a cluster for each seed ($c_i, i \in [1, 5]$). In each iteration, it calculates the Hamming distance [24] between clusters and merges two clusters with the minimum distance. One representative method using AHC algorithm is 6Gen [17]. It is worth noting that different from the general AHC algorithm, 6Gen does not explicitly merge similar clusters. Instead, each cluster grows independently and it allows a seed address to belong to multiple clusters. This may result in some overlapped subspaces. For example, two clusters (200*::000* and 2003::00*0) formed at $t = 2$ in Fig. 1 (a) contain the same IPv6 address 2003::0000.

The divisive hierarchical clustering (DHC) algorithm adopts a top-down clustering approach, as shown in Fig. 1 (b). One example is 6Tree [20]. It starts with a single cluster containing all seeds. After each iteration, it splits the clusters from the left-most variable dimension (i.e., dimension with value of "*"). It stops when each seed is in its own singleton cluster or the cluster can no longer split (i.e. only one variable dimension left). Note that subspaces generated by this cluster splitting method may result in a loss of address space. As shown in Fig. 1 (b), address space 2003::00** is split into subspaces 2003::001* and 2003::002*, which results in the loss of subspace 2003::00[0, 3-15]*, where the dimension with value $[0, 3-15]$ indicates the value in this dimension can be any value except 1 and 2.

6Hit also uses the DHC algorithm, but we enhance it to avoid space loss during cluster splitting. In addition, the generated regions do not overlap with each other and thus allow us to explore each region efficiently with few probes. Space partition with 6Hit will be disclosed in Sec. IV-B.

C. Seed Density and Activity Density

A structural information-based approach treats the seeds as independently and identically distributed (iid) random samples from the set of active addresses $\mathcal{A}$ in the address space X. As such, the regions of address space with the highest density of active addresses will likely have the highest density in seeds. In other words, target generation based on the structural information in seeds is density-driven, and the activity density (i.e., the number of active addresses) in each region is estimated by the seed density in this region. For example, 6Gen determines the sequence of regions to be scanned according to the seed density in the regions. over-rely on Seed, and need be updated frequently

One main pitfall of the above method is that the seed density in a region may not align well with the activity density in the region. In practice, samples may have noise and may be biased. If we take the fitting degree between the distribution of seeds and the distribution of real active addresses as the only basis for target generation, the density-driven method would depend too much on the quality of seeds. The critical issue is to avoid over-reliance on seeds and find the real high-density regions within the probing budget.

6Hit tackles the above problem with a bandit algorithm [25], a reinforcement learning method. It introduces the action-reward mechanism into the scanning process, and treats the allocation of probe packets in each region as an action and calculates the reward based on the number of active addresses detected in each region. On the initial allocation of exploration resources, 6Hit utilizes regional seed density to allocate probing resources. The evaluation of the return at each region determines the subsequent scanning direction. Thus, 6Hit constantly uses the discounted cumulative reinforcement reward to obtain more and more accurate activity density information. More details will be disclosed in the next section.

IV. 6Hit Design

We first present the system overview of 6Hit and then describe the details of its three key technical components: space partition, target generation, and avoidance of premature convergence.

A. System Overview

Fig. 2 illustrates the main workflow of 6Hit. It first divides the entire IPv6 address space X using an enhanced DHC algorithm. Taking advantage of the existing knowledge (i.e., regional seed density), it iteratively explores the active addresses.

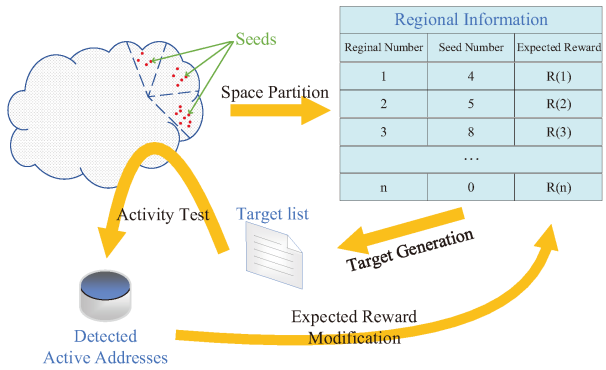

Fig. 2. The workflow of 6Hit.

Meanwhile, 6Hit defines a regional scanning expectation reward (i.e., $R(i)$, $i \in [1, n]$) to estimate the return of active addresses in each region. After each iteration of scanning, 6Hit updates the expected regional reward according to the probing result so far, and adjusts the future probing packets in each region accordingly. This dynamic scan-update-adjust process cycle continues until the total number of probing packets reaches the budget limit.

B. Space Partition

The partition of the address space X is realized by utilizing the DHC algorithm to construct a tree structure named the "space tree". The tree is built from the root to leaf by hierarchically dividing the 32-dimensional space X. Meanwhile, in order to overcome the problem of space loss during cluster splitting (as per Sec. III-B), we added a new type of leaf node called "R-node" in the construction of space tree. The purpose of adding an "R-node" for each non-leaf node as its child node is to record the space loss during its splitting process. In this way, 6Hit can ensure that the exploration scope covers the entire address space X.

Which factor can decide the scope of 6Decoder? Vocabulary size?

The pseudocode is shown in Alg. 1. At the beginning of the space tree construction, the root node contains all address vectors in X, i.e., the 32 dimensions of the root are all marked with "*". A dimension marked with "*" is also called variable dimension. Since the space tree is constructed according to the structural information embedded in the seed addresses, each node on the tree contains two attributes: (1) the assigned dimension ($assignedDimension$) which is used to represent the address space allocated to the node, and (2) the assigned seed ($assignedSeed$) which is used to calculate the regional seed density associated with the node. In the process of cluster splitting, DHC performs seeds segmentation according to whether the leftmost variable dimension has the same value, that is, the addresses associated with the same child node have the same value on their assigned dimensions. Finally, we add an R-node ($AddRnodes()$) for each non-leaf node as its child node, which records all the residual addresses not covered by its other child nodes.

Algorithm 1 Space Partition

Input: The set of seed addresses C;
Output: The root node of the space tree d_0;

$d_0 = InitializeRoot()$; ▷ Initializes the root node, specifying that its child is d_1. The address space of root is the full space.
$d_1 = CreateNode(C)$;
$DHC(d_1)$;
$AddRnodes()$; ▷ Adds an R-node as a child node for each non-leaf node.
return d_0
function $CreateNode(c)$
▷ Creates a node and assigns the seeds c and dimension to it.
  return $newNode$
end function
function $DHC(node)$
  if $|node.assignedDimension| \geq 31$ **then**
    return
  end if
  for $\delta = 1$ to 32 **do**
    if $\forall c_i, c_j,\ c_i, c_j \in node.assignedSeed,\ c_i[\delta] == c_j[\delta]$ **then**
      $node.assignedDimension.Append(\delta)$;
    end if
  end for
  for $i = 1$ to 32 **do**
    if $i \notin node.assignedDimension$ **then**
      $\delta^* = i$;
      Break;
    end if
  end for
  $subsequences = Parition(node.assignedSeed, \delta^*)$;
▷ Splits out subsequences of the assigned seeds on node where the seed vectors have the same value in dimension δ^*.
  for $\sigma \in subsequences$ **do**
    $newNode = CreateNode(\sigma)$;
    $node.childNodes.Append(newNode)$;
  end for
  for $child \in node.childNodes$ **do**
    $DHC(child)$;
  end for
end function

Fig. 3 shows an example of building a space tree with 500 seed addresses. The final tree contains 147 nodes in total, but we only draw some of the nodes for clear illustration. The blue nodes represent non-leaf nodes and the red nodes are leaf nodes. It can be seen that 6Hit applies a top-down clustering method. Advancing from the root to leaf nodes, the number of variable dimensions is gradually reduced. Meanwhile, each non-leaf node has an R-node as its child, which covers the address space not covered by its other child nodes. As shown in the figure, the root node d_0 has two child nodes

d_1 and d_{R0}. The $assignedDimension$ of d_1 represents the address subspace whose first 6 dimensions are 200112 and the remaining dimensions are variable dimensions (marked with "*"). The $assignedDimension$ of d_{R0} represents the address subspace whose first 6 dimensions are !200112 and the remaining dimensions are variable dimensions (marked with "*"). Note that !200112 denotes any value not equal to 200112. In other words, the union of the child nodes' address spaces is equal to the parent node's address space.

Complexity analysis: Let m be the number of input seed addresses. The algorithm first sorts the m seeds, which has the worst-case time complexity of $O(m \log m)$. Then it performs the space partition by finding the leftmost variable dimension δ^*. The partition can be finished by traversing each address vector once per dimension, and at most 32 dimensions need to be traversed. So the worst-case time complexity of this step is $O(32m)$. In summary, the worst-case time complexity of space partition is in the order of $O(m \log m)$.

Remark 1: The number of regions generated by Alg. 1 is no larger than twice the number of seed addresses. This is because each seed belongs to only one region and each seed may split off at most one region associated with an R-node.

C. Reinforcement Learning-based Target Generation

After space partition, the critical issue is appropriate allocation of the probes to achieve a high hit rate within the given budget. To be formal, we can re-state the problem as a combinatorial optimization problem. Assume that the total probing budget is β. A model $P = (\mathcal{S}, f, \Omega)$ of a combinatorial optimization problem consists of:

如何使有限的子弹发挥最大效能

- $\mathcal{S}$: the set of candidate solutions which define over a finite set of discrete decision variables $X_i^j, i \in [1, n], j \in [0, \beta]$ where X_i^j indicates that there are j different addresses generated in the i_{th} region and n is the total number of regions.
- $f(s)$: the objective function that returns $|s \cap \mathcal{A}|$, where s denotes the set of generated target addresses and thus $|s \cap \mathcal{A}|$ indicates the number of hit active addresses. Our goal is to maximize this objective function.
- Ω: the constraint, i.e., $|s| \leq \beta$.

A feasible solution $s \in \mathcal{S}$ is a complete assignment of values to variables that satisfy the constraint Ω. A solution $s^* \in S$ is called a global optimum if and only if: $f(s^*) \geq f(s), \forall s \in \mathcal{S}$.

Obviously, there are various allocation strategies for the probes sent to different regions $X_i^j, i \in [1, n], j \in [0, \beta]$. For example, 6Gen sends all the probes to the region with the maximum seed density $X_{i^*}^{\beta}, \tau_{i^*} = max(\tau_i), i \in [1, n]$ where τ_i indicates the seed density of the i_{th} region. This strategy can achieve a good result only when the distribution of seeds is close to the distribution of real active addresses in the regions. This condition, however, might not be true in many real-world networks.

We adopt a reinforcement-learning approach for target generation that evaluates the feedback from existing probes to adjust probing directions. This is based on two considerations: *First,* the seed density may largely differ from the activity density in a region. Hence we should not rely on seed density too much but instead only use seed density as a prior for the allocation of probes in the initial scan. The expected reward from each region should be gradually revised according to the evaluative feedback. *Second*, the activity density of a region changes over time. This is because when some addresses in a region are detected to be active, these active addresses should be recorded and removed from the region to keep them from being probed again. To this end, 6Hit utilizes a penalty return when it generates duplicate target addresses to reduce over-exploration of some regions.

The target generation algorithm is shown in Alg. 2. The algorithm takes as input a partitioned address space $X = \{X_1, \ldots, X_n\}$ and a specified number of targets generated per iteration b. In fact, every X_i has a bijective relationship with every leaf node on the space tree. The size of X_i can be calculated by the $|unassignedDimension|$ attribute of its corresponding leaf node, and the seed density in the region can be measured by $|assignedSeed|$. Then, the algorithm iterates over three main steps: (1) allocate b target addresses according to the weight of regional expected reward ($TargenGen()$); (2) determine whether these targets are active (through $\mathcal{A}Test$); (3) update the expected regional reward (in procedure $UpdateReward()$). The following is a more detailed description of the three main steps.

Algorithm 2 Target Generation

Input: A partitioned address space X. The number of targets generated per iteration b;
Output: Active address set A;

```
ConsumedBudget = 0;
A = ∅;
while ConsumedBudget ≤ β do
    A* = ∅;
    {t_1, ..., t_b} = TargetGen(X, b);
    for i = 1 to b do
        if t_i ∉ A & t_i ∈ A then                ▷ Atest()
            A*.add(t_i);
        end if
    end for
    A = A ∪ A*
    UpdateReward(A*);
    ConsumedBudget += b;
end while
return A
```

Step 1 ($TargetGen$): As mentioned earlier, the information about the environment (i.e., activity density in each region), may not be accurate. 6Hit must explore its environment in order to see which probe allocation strategy achieves a better hit rate. 6Hit uses the soft-max action selection algorithm to increase opportunities of exploring unknown regions. This algorithm selects the explorative action according to Boltzmann distribution as shown in the follows:

$$P(i) = \frac{e^{R(i)}}{\sum_{j=1}^{n} e^{R(j)}},$$

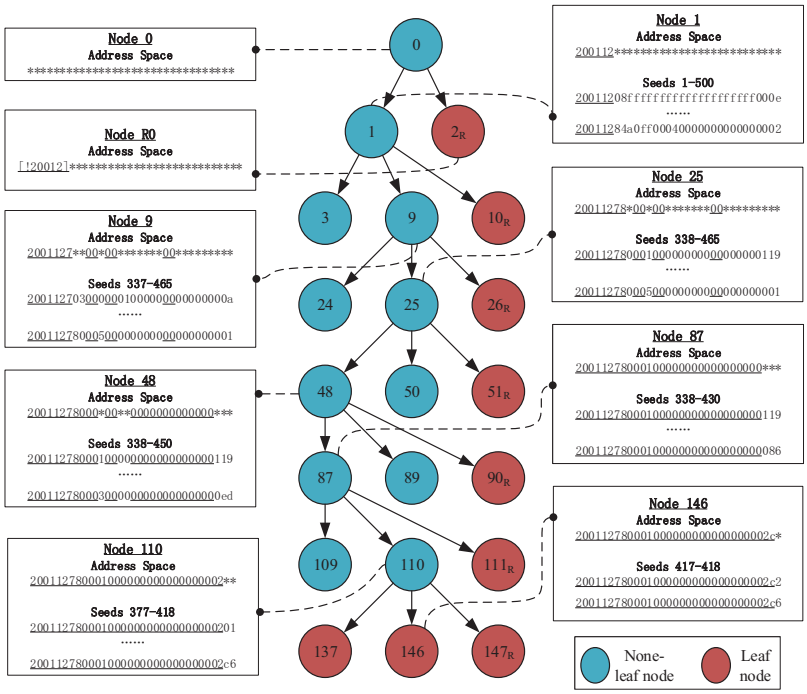

Fig. 3. Example of space tree generation. It generates a 147-node tree (partially shown) from 500 seed addresses.

where $R(i)$ indicates the expected reward of region i and $P(i)$ indicates the probability of generating target addresses in region i. Thus, there will be $b \cdot P(i)$ target addresses randomly generated in the i_{th} region in this iteration.

Step 2 ($\mathcal{A}Test$): Once the targets have been generated, we need to detect whether or not the generated target addresses are active. We record the detected active addresses and the regions where the addresses are located, so that we can use this information to update the regional reward.

Step 3 ($UpdateReward$): We update the expected regional reward to give high-hit regions a better chance for next-round probes. To be more specific, the (initial) prior expected regional reward $R(i)^1$ is calculated based on the seed density:

$$R(i)^1 = \frac{|X_i.assignedSeed|}{|X_i.unassignedDimension|},$$

where $|X_i.assignedSeed|$ and $|X_i.unassignedDimension|$ represent the number of seeds and the number of variable dimensions on the leaf node corresponding to the region X_i, respectively. After each iteration, the expected regional reward $R(i)^t$ is updated as follows:

$$R(i)^{t+1} = (1-\alpha) \cdot R(i)^t + \alpha \cdot r(i)^t \quad (t \geq 1),$$

where $\alpha (0 < \alpha \leq 1)$ is the learning rate, and $r(i)^t$ is the reward associated with region X_i from the scanning results. Note that α is a constant step-size parameter that determines the update speed of $R(i)$. $r(i)^t$ is calculated as follows:

$$r(i)^t = \frac{\mathcal{I}_{X_i}(A^t) - \mathcal{I}_{X_i}(D^t)}{|X_i.unassignedDimension|},$$

where A^t denotes the set of active addresses probed during time step t and D^t is the duplicate targets during the probing process thus far in time steps 0 through t. $\mathcal{I}_{X_i}$ is the indicator function returning the number of (active or duplicate) addresses in region X_i. It can be seen from the reward function that 6Hit not only rewards the active addresses $\mathcal{I}_{X_i}(A^t)$ generated in the region X_i but also penalises the duplicate addresses $\mathcal{I}_{X_i}(D^t)$ to avoid the waste of budget. Furthermore, 6Hit utilizes $|X_i.unassignedDimension|$ to normalize the rewards because different regions may have quite different size (i.e., the range of addresses).

Remark 2: It is easy to see that the time complexity of Alg. 2 is $O(\frac{\beta}{b} \cdot n)$, where n is the number of regions. Regarding the number of targets generated per iteration, we can set $b = \gamma \cdot m$, where m is the total number of seed addresses and $\gamma (\geq 2)$ is a constant. Based on Remark 1, the time complexity of target generation is thus in the order of $O(\beta)$.

D. Avoidance of Premature Convergence

With the iterations of the algorithm, the expected reward of high-hit regions will become higher and higher, resulting in more targets generated in these regions. This will eventually lead to the algorithm's convergence. Nevertheless, it is possible that some high-density regions initially may not be given enough chance for probing, and the algorithm may miss these high-density regions if it converges too early on other regions. This is the typical exploration-exploitation dilemma, and the problem of early convergence into local maximum is called premature convergence. In addition to the penalty policy for duplicate address generation, 6Hit adopts two other mecha-

nisms to avoid premature convergence: (1) nodes chipping and (2) space repartition. The two mechanisms occur at different stages of the algorithm. Nodes chipping is performed as soon as the address space is partitioned and space repartition is applied only when the risk of premature convergence appears.

*1) **Nodes Chipping**:* In essence, the target generation algorithm dynamically adjusts the generated region of targets according to the expected reward of the region. Probes are always attracted to regions with high activity density. To prevent the algorithm from getting caught into local minimum at the beginning of execution, 6Hit calculates the seed density of each region as soon as the address space is divided. If the seed density of a region exceeds a certain threshold, the chipping operation will be carried out, that is, all the addresses in the region will be scanned and the leaf node corresponding to this region will be removed from the space tree.

In our experiments (Section V), we set the upper limit of prior expected regional reward $R(i)$[1] as:

它通过提前处理掉最明显的高价值目标**，来避免强化学习算法在游戏初期就被这些目标"带偏"，从而保证了算法有足够的机会去探索那些更广阔、更未知的区域。

$$\frac{16^{|X_i.unassignedDimension|-1}}{|X_i.unassignedDimension|}.$$

When the regional seed density exceeds this value, the whole address space on this node will be scanned and then the node will be removed. For example, the initial expected reward on node d_{146} in Fig. 3 is 2 which exceeds the upper limit of seed density 1 in regions with one dimension. 6Hit will scan the d_{146}'s address space 2001:1278:1::2c* and then cut d_{146} off.

*2) **Space Repartition**:* This mechanism was proposed as an exploration enhancement, applied every time when the premature convergence is "detected", i.e., at the current iteration the number of regions that generate targets is smaller than a threshold. Empirically, we set this threshold to $\frac{1}{20}n$ where n is the total number of regions.

The space repartition process utilizes addresses chosen randomly from the detected active addresses to rebuild the space tree and then makes a follow-up scanning. Meanwhile, it adopts heuristic operations to make a wider exploration of the search space. Alg. 3 shows the basic three steps of space repartition: (1) *selection*, (2) *crossover*, and (3) *mutation*. The *selection* step is to form a new-generation seeds from all the available active addresses known so far. The random selection from a larger range of active addresses ensures the new-generation seeds embed more structural information. The *crossover* and *mutation* operations are carried out with probability and both aim at changing the search interval to encourage more exploration. *Crossover* focuses on manipulating the lower dimensional interval of the address vector (i.e., more towards re-dividing the regions), while *mutation* focuses on manipulating the higher dimensional interval of the address vector (i.e., more towards exploring unknown regions).

$Selection$: This process randomly selects m active addresses from all the known active addresses so far, where m is set to the same number of initial input seeds. Note that we do not increase m in order to control the running time as well as the probing budget in each iteration.

$Crossover$: This operation is a probabilistic process that exchanges information between two "parent" addresses for generating two "child" seeds. As to IPv6 addresses, the representation of the address vector (i.e., 32 dimensions) facilitates the crossover and mutation steps. 6Hit utilizes a single-point crossover with a fixed probability p_c. For the "parent" addresses, a random integer, called the crossover point, is generated in the lower dimensional interval of $[17, 32]$. The portions lying to the right of the crossover point are switched to produce two offspring. Fig. 4 shows an example of the crossover operation on two addresses.

$Mutation$: Each selected address undergoes mutation with a probability $p_u (\ll p_c)$. Like crossover, the mutation point is randomly generated in the higher dimensional interval of $[1, 16]$. As to IPv6 addresses, nybbles (dimensions) are representation of chromosomes, a nybble (or gene) is mutated by randomly selecting a value in the range $[0, 15]$.

Algorithm 3 Space Repartition

Input: The detected active address set A, The number of the initial input seeds m;
Output: The root node of the rebuilt space tree d_0^{new};

$N = \emptyset$;
while $|N| \leq m$ **do**
  $a_i, a_j = Selection(A)$; ▷ Randomly selects two addresses in A.
  $a_i^c, a_j^c = Crossover(a_i, a_j, p_c)$; ▷ Performs crossover with probability p_c .
  $a_i^{cm} = Mutation(a_i^c, p_u)$; ▷ Performs mutation with probability p_u.
  $a_j^{cm} = Mutation(a_j^c, p_u)$;
  $N.add(a_i^{cm})$;
  $N.add(a_j^{cm})$;
end while
$d_0^{new} = SpacePartition(N)$; ▷ Alg. 1.
return d_0^{new}

V. Performance Evaluation

We compare the performance of 6Hit and other state-of-the-art target generation methods, including 6Tree [20], 6Gen [17] and Entropy/IP [16], with real-world test.

A. Dataset Description

Gasser et al. [26] have collected active IPv6 addresses from multiple sources, as shown in Tab. I. We used the data published on Oct. 5th, 2019 as the active address set $\mathcal{C}$, which has $\approx 3.4M$ detected IPv6 addresses.

To investigate the impact of initial seed on the performance of different methods in real-world test, we adopt two strategies to form seed sets from $\mathcal{C}$: down sampling and biased sampling. In down sampling, we randomly sample a certain number of addresses from $\mathcal{C}$ as seed set $C_x,\ x \in [1, 2, 3, 4]$. In biased sampling, we order the addresses in $\mathcal{C}$ and then extract a certain number of adjacent addresses as seed set $C_y,\ y \in [5, 6, 7, 8]$. More details of these seed sets are shown

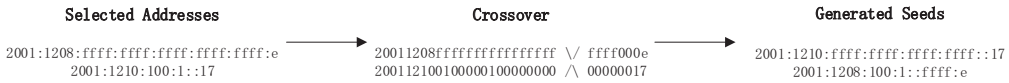

Fig. 4. An example of the crossover operation on two addresses.

TABLE I
MAIN SOURCES OF ACTIVE ADDRESS SET $\mathcal{C}$.

Main sources	Nature	Public
Alexa top website list [27]	Servers	Yes
Statvoo website list [28]	Servers	Yes
Cisco Umbrella website list [29]	Servers	Yes
Zone files for several top-level domains [30]	Servers	Yes
CAIDA IPv6 DNS names dataset [31]	Servers	Yes
TLS certificates in certificate transparency logs	Servers	Yes
Rapid7 FDNS ANY dataset [32]	Servers	Yes
RIPE Atlas dataset [33]	Routers	Yes
Scamper [34]	Routers	–

Address range: $2001:200:0:1::1 \sim 2c0f:ffc8:4001:1::2$
Total number of addresses $\approx 16^{30.898}$
Total number of active addresses $\approx 3.4M$

TABLE II
CHARACTERISTICS OF THE SEED SETS.

Seed set	Seed number	Selection strategy	Address range
C_1	$1k$	Down sampling	$2001:1388::1-2c0f:feb0::1$
C_2	$5k$	Down sampling	$2001:1284::1-2c0f:f470::1$
C_3	$30k$	Down sampling	$2001:1218::1-2c0f:fed8::1$
C_4	$0.1M$	Down sampling	$2001:1218::1-2f0c:ff00::1$
C_5	$1k$	Biased sampling	$2001:1208::1-2001:1291::1$
C_6	$5k$	Biased sampling	$2001:1208::1-2001:1328::1$
C_7	$30k$	Biased sampling	$2001:1208::1-2001:1460::1$
C_8	$0.1M$	Biased sampling	$2001:200::1-2001:2003::1$

in Tab. II. Intuitively, the "quality" of C_x should be better than that of C_y in the sense that poor "quality" means poor fitting degree between the distribution of seeds and the distribution of active addresses.

B. Default System Parameters

6Hit includes several parameters (introduced in Sec. IV) that need to be set empirically: the learning rate is $\alpha = 0.1$, the probability of crossover is $p_c = 0.6$ and the the probability of mutation is $p_u = 0.01$.

C. Real-world Test Results

We evaluated the performance of 6Hit and other target generation methods in a real IPv6 network environment, where we were careful to ensure good Internet citizenship as suggested by Partridge and Allman [35]. Internet-wide scanning experiments were performed at Changsha University through the China Education and Research Network version 2 (CERNET2). We did a single-threaded address scanning on a Linux platform with an AMD OPTERON X3216 (3.0GHz) and 16 GB memory. The probing rate was limited to 10 million bps (in AS 23910). Such amount of probing traffic would not cause any problem on the scanned devices. We only used the ICMPv6 packets to perform address scanning. The Internet-wide scanning experiments were performed in October 2019.

We compared the hit rate using the different seed set $C_i, i \in [1, 8]$ introduced above. With each seed set, we used 10 million ICMPv6 packets to discover active addresses. The results are shown in Fig. 5. We can see that the hit rate of methods that use the structural information (6Hit, 6Tree and 6Gen) is much higher than that of methods that use statistics information (Entropy/IP). In addition, 6Hit and 6Tree have a higher hit rate than 6Gen. During the scanning process, we set the number of addresses generated per iteration $b(= \gamma \cdot m_i)$, where $\gamma = 3$ and m_i denotes the number of addresses in seed set C_i. It is worth highlighting that 6Hit has a much higher hit rate than other methods when the quality of initial seeds is poor (C_5, C_6 and C_7). This demonstrates the advantage of 6Hit being able to exploring new regions and adjusting its probing direction based on the feedback. However, with the increase number of seeds m_i, the number of iterations in 6Hit decreases, resulting in a drop in hit rate. 6Hit and 6Tree tend to yield the same number of iterations using seed set C_4, which results in a similar hit rate in this setting. The number of targets generated on region X_j in $(t+1)_{th}$ iteration is $3m_i \cdot P^t(j)$. The expected reward of region X_j is estimated by this number of addresses rather than the entire node space.

Fig. 6 shows the hit rate performance of 6Hit per iteration. We can see that in general the hit rate of 6Hit is maintained at a higher level after the initial oscillation. The blue circles in Fig. 6 indicate when space repartition has been performed by 6Hit to avoid premature convergence. This process expands the scope of exploration and allows 6Hit to quickly discover new regions with high activity density. This is particularly important when the initial seed sets do not really reflect the actual activity density. Compared Fig. 5 and Fig. 6 (under seed sets $C_y, y \in [5, 6, 7, 8]$), we can see that space repartition indeed makes 6Hit significantly better than other methods.

In summary, discovery on the IPv6 Internet shows that 6Hit achieved 3.5% - 11.5% hit rate for the eight candidate datasets, which is 7.7% - 630% improvement over the state-of-the-art mthods. Especially in seed sets with poor seed quality, e.g., seed set C_5, 6Hit achieved 630% improvement compared to 6Tree which is the second best. This is due to the fact that 6Hit has more direction adjustments and more aggressive mechanisms for exploration, i.e., the space repartition. This exploration enhancement mechanism increases the exploration efforts, helps to quickly find new high-density regions, which greatly improves the hit rate.

VI. CONCLUSION

In this work, we proposed 6Hit, an efficient target generation method for discovering active IPv6 addresses. 6Hit adopts a

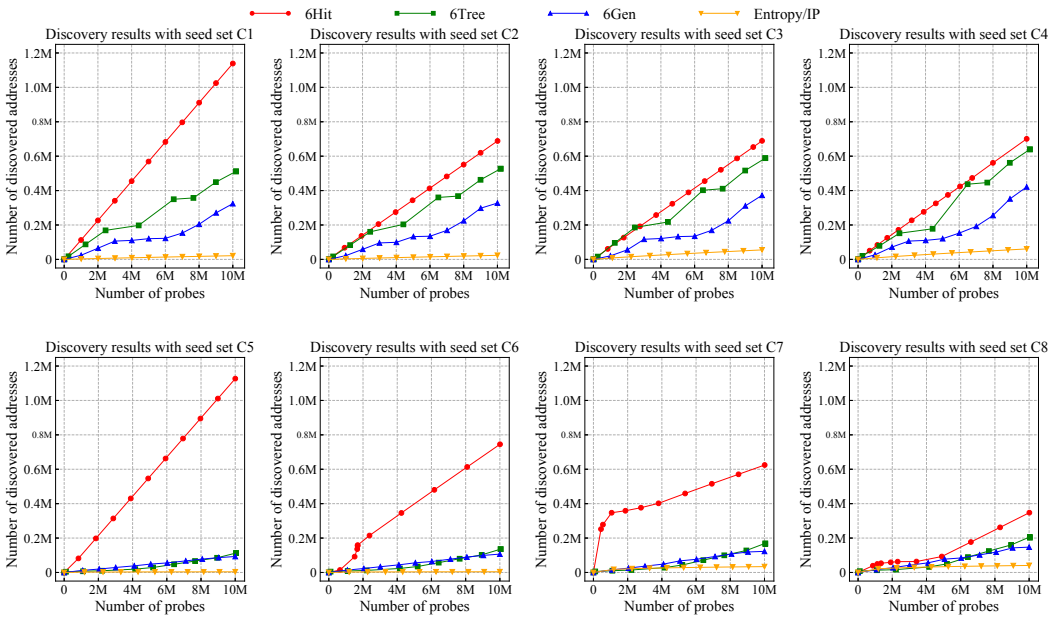

Fig. 5. Discovery of active addresses with a budget of $10M$ on each seed set. 6Hit discovered more active addresses. In particular, in the scenarios where small and biased sampling seed sets are used, 6Hit found much more active addresses than other methods.

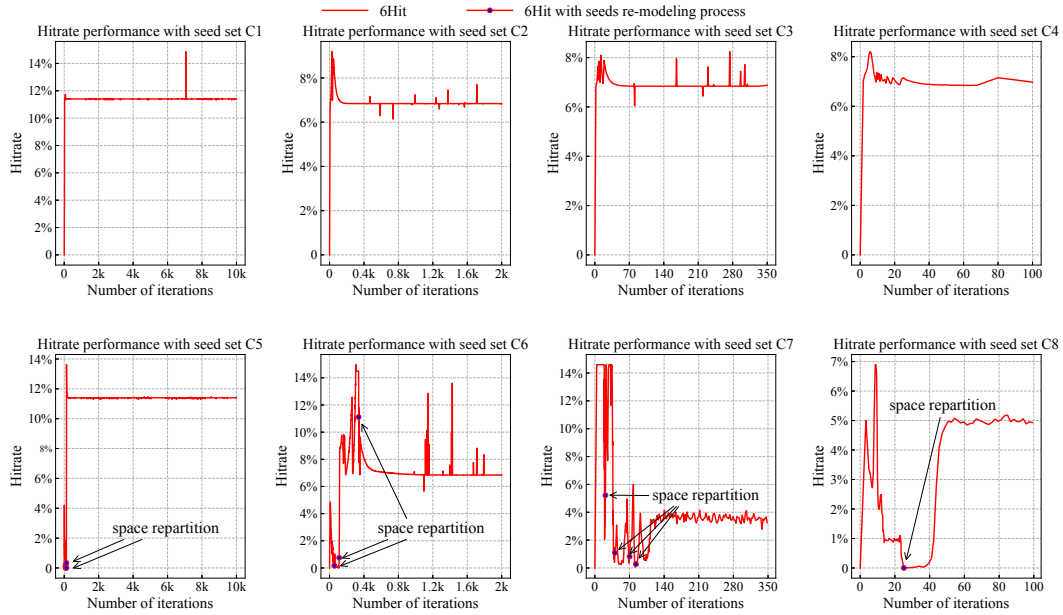

Fig. 6. Hit rate of 6Hit per iteration with a total budget of $10M$. The small blue circles indicate when space repartition occurred.

novel reinforcement learning-based approach to automatically learn the address space structure from the feedback of historical probes, based on which it dynamically adjusts subsequent address search direction towards the regions that have high activity density. This unique feature makes it more efficient and effective in target generation and more resilient to the quality of initial seed addresses. With real-world test, 6Hit has achieved much better performance on hit rate than the state-of-the-art solutions, 6Gen, 6Tree, and Entropy/IP.

ACKNOWLEDGMENT

We would like to thank Zhizhu Liu for his kind help to our work. This work is supported by the National Key Research and Development Program of China under Grant No. 2018YFB1800202, The NSF-HN under Grant No. 2020JJ5621 and the Outstanding Youth Research Project of Provincial Education Department of Hunan under Grant No. 20B064. The corresponding author is Zhiping Cai.

REFERENCES

[1] I. Livadariu, K. Benson, A. Elmokashfi, A. Dhamdhere, and A. Dainotti, "Inferring Carrier-Grade NAT Deployment in the Wild," in *IEEE Conference on Computer Communications (INFOCOM)*, 2018, pp. 2249–2257.

[2] Google. (2019) IPv6 adoption statistics. [Online]. Available: https://www.google.com/intl/en/ipv6/statistics.html

[3] G. Huston. (2019) IPv6 BGP table reports. [Online]. Available: https://bgp.potaroo.net/index-v6.html

[4] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast Internet-wide scanning and its security applications," in *USENIX Security Symposium*, 2013, pp. 605–620.

[5] R. D. Graham. (2019) MASSCAN: Mass IP port scanner. [Online]. Available: https://github.com/robertdavidgraham/masscan

[6] R. Beverly, "Yarrp'ing the Internet: Randomized high-speed active topology discovery," in *ACM Internet Measurement Conference (IMC)*, 2016, pp. 413–420.

[7] R. Beverly, R. Durairajan, D. Plonka, and J. P. Rohrer, "In the IP of the Beholder: Strategies for Active IPv6 Topology Discovery," in *ACM Internet Measurement Conference (IMC)*, 2018, pp. 308–321.

[8] D. Plonka and A. W. Berger, "kIP: a Measured Approach to IPv6 Address Anonymization," *arXiv: 1707.03900*, 2017.

[9] J. Czyz, M. J. Luckie, M. Allman, and M. Bailey, "Don't forget to lock the back door! A characterization of IPv6 network security policy," in *Network and Distributed System Security Symposium (NDSS)*, 2019.

[10] Q. Scheitle, O. Gasser, P. Sattler, and G. Carle, "HLOC: Hints-based geolocation leveraging multiple measurement frameworks," in *Network Traffic Measurement and Analysis Conference (TMA)*, 2017.

[11] J. Matherly. (2019) Shodan: The search engine for Internet-connected devices. [Online]. Available: https://www.shodan.io

[12] Z. Durumeric, D.Adrian, A.Mirian, M.Bailey, and J. A. Halderman, "A search engine backed by Internet-wide scanning," in *Computer and Communications Security (CCS)*, 2015, pp. 542–553.

[13] Z. Durumeric, F. Li, J. Kasten, N. Weaver, J. Amann, J. Beekman, M. Payer, D. Adrian, V. Paxson, M. Bailey, and J. A. Halderman, "The Matter of Heart-bleed," in *ACM Internet Measurement Conference (IMC)*, 2014.

[14] A. S. Ahmed, R. Hassan, and N. E. Othman, "IPv6 Neighbor Discovery Protocol Specifications, Threats and Countermeasures: A Survey," *IEEE Access*, vol. 5, pp. 18 187–18 210, 2017.

[15] K. Borgolte, S. Hao, T. Fiebig, and G. Vigna, "Enumerating active IPv6 hosts for large-scale security scans via DNSSEC-signed reverse zones," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 770–784.

[16] P. Foremski, D. Plonka, and A. Berger, "Entropy/IP: Uncovering Structure in IPv6 Addresses," in *ACM Internet Measurement Conference (IMC)*, New York, USA, 2016, pp. 167–181.

[17] A. Murdock, F. Li, P. Bramsen, Z. Durumeric, and V. Paxson, "Target Generation for Internet-wide IPv6 Scanning," in *ACM Internet Measurement Conference (IMC)*. New York, USA: ACM Press, 2017, pp. 242–253.

[18] R. Barnes, R. Altmann, and D. Kerr, "Mapping the Great Void Smarter Scanning for IPv6," BBN Technologies, Tech. Rep., 2012.

[19] W. Blake, "Methods for Intelligent Mapping of the IPv6 Address Space," Master's thesis, Naval Postgraduate School, Monterey, CA, 2015.

[20] Z. Liu, Y. Xiong, X. Liu, W. Xie, and P. Zhu, "6Tree: Efficient Dynamic Discovery of Active Addresses in the IPv6 Address Space," *Computer Networks*, vol. 155, pp. 31–46, 2019.

[21] M. D.Gray, "Discovery of IPv6 Router Interface Addresses via Heuristic Methods," Master's thesis, Naval Postgraduate School, Monterey, CA, 2015.

[22] J. Ullrich, P. Kieseberg, K. Krombholz, and E. Weippl, "On Reconnaissance with IPv6: A Pattern-based Scanning Approach," in *IEEE International Conference on Availability, Reliability and Security (ARES)*, no. 2, 2015, pp. 186–192.

[23] G. Shobha and S. Rangaswamy, "Machine Learning," in *Handbook of Statistics*, 1st ed. Elsevier B.V., 2018, vol. 38, pp. 197–228.

[24] R. W. Hamming, "Error Detecting and Error Correcting Codes," *Bell Labs Technical Journal*, 1950.

[25] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[26] O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczynski, S. D. Strowes, L. Hendriks, and G. Carle, "Clusters in the Expanse: Understanding and Unbiasing IPv6 Hitlists," in *ACM Internet Measurement Conference (IMC)*, 2018, pp. 364–378.

[27] Alexa. (2019) The top sites on the web. [Online]. Available: https://www.alexa.com/topsites

[28] Statvoo. (2019) Website analytics and reviews. [Online]. Available: https://statvoo.com

[29] Cisco. (2019) Cisco Umbrella. [Online]. Available: https://umbrella.cisco.com

[30] PremiumDrops. (2019) Domain zone file and zone changes downloads . [Online]. Available: https://www.premiumdrops.com/zones.html

[31] CAIDA. IPv6 topology dataset. [Online]. Available: http://www.caida.org/data/active/ipv6_allpref_topology_dataset.xml

[32] Rapid7. (2019) Forward DNS (FDNS). [Online]. Available: https://opendata.rapid7.com/sonar.fdns_v2/

[33] RIPE. (2019) RIPE NCC Atlas dataset. [Online]. Available: https://atlas.ripe.net

[34] M. Luckie, "Scamper: a Scalable and Extensible Packet Prober for Active Measurement of the Internet," in *ACM Internet Measurement Conference (IMC)*, 2010.

[35] C. Partridge and M. Allman, "Ethical considerations in network measurement papers," *ACM Communications*, vol. 59(10), pp. 58–64, 2016.