6Forest: An Ensemble Learning-based Approach to Target Generation for Internet-wide IPv6 Scanning

Tao Yang, Zhiping Cai, Bingnan Hou and Tongqing Zhou
College of Computer, National University of Defense Technology, China
E-mails: {*yangtao97, zpcai, houbingnan19, zhoutongqing*}*@nudt.edu.cn*

IEEE INFOCOM 2022 - IEEE Conference on Computer Communications | 978-1-6654-5822-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/INFOCOM48880.2022.9796925

Abstract—**IPv6 target generation is the critical step for fast IPv6 scanning for Internet-wide surveys. Existing techniques, however, commonly suffer from low hit rates due to inappropriate space partition caused by the outlier addresses and short-sighted splitting indicators. To address the problem, we propose 6Forest, an ensemble learning-based approach for IPv6 target generation that is from a global perspective and resilient to outlier addresses. Given a set of known addresses, 6Forest first considers it as an initial address region and then iteratively divides the IPv6 address space into smaller regions using a maximum-covering splitting indicator. Before a round of space partition, it builds a forest structure for each region and exploits an enhanced isolation forest algorithm to remove the outlier addresses. Finally, it pre-scans samples from the divided address regions and based on the results generates IPv6 addresses. Experiments on eight large-scale candidate datasets indicate that, compared with the state-of-the-art methods in IPv6 worldwide scanning, 6Forest can achieve up to 116.5% improvement for low-budget scanning and $15\times$ improvement for high-budget scanning.**

Index Terms—**Internet-wide scanning, IPv6, outlier detection, ensemble learning**

I. Introduction

Since World IPv6 Day in 2011 and Launch in 2012 [1], IPv6 is widely implemented and adopted in recent years due to the strong demands from mobile Internet and cloud computing [2]. Over 30% of Google users are accessing their services via IPv6 in July 2021 [3]. As a result, the Internet is experiencing a rapidly increasing number of IPv6 routing entries [4].

Conducting Internet-wide surveys that probe and analyze such a vast address space is non-trivial, which hinders the effective network assets evaluation and risk analysis for IPv6-based Internet [5]. Traditional asynchronous scanning tools, such as ZMap [6] and Masscan [7], involve the advanced techniques of topology discovery [8], [9], IP address analysis [10], [11], and geolocation [12], to facilitate high-performance network surveys [13]. However, these tools are designed for IPv4 networks and generally face efficiency problems when dealing with IPv6 address space. In fact, based merely on these brute-force approaches, it would take tens of millions of years to render comprehensive scanning of the entire IPv6 address space [14].

To relieve this pitfall, IPv6 target generation is utilized as an essential step for efficient IPv6 scanning. By characterizing and modeling the structure of IPv6 address space with a set of known addresses (i.e., the seeds), it can generate candidate addresses that present a higher probability to be active, so as to narrowing down the probing scope and significantly accelerating the scanning [15]. Following this idea, existing target generation methods (e.g., 6Hit [16] and 6Tree [17]) generally treat an IPv6 address as a 32-dimensional vector and sequentially and iteratively divide the current seeds into different seed clusters (a.k.a., address region) until the cluster size is smaller than a pre-defined threshold. After this space partition process, the dimensions with different nibbles (i.e., the free dimensions) in each cluster are assigned with hexadecimal values to construct a probing address space for scanning. Intuitively, the probing scope would expand exponentially with larger free dimensions in the final clusters, i.e., the inflation of scanning space. It has been found that the existing proposals fail to handle the outlier addresses defined in § III-B, which would, unfortunately, enlarge the free dimensions and result in a low hit rate, about 11.5% [16].

Using an example based on the common algorithm of 6Tree and 6Hit, we illustrate the outlier address challenge [18] and analyze the reason for its occurrence. As shown in Fig. 1, the seeds are divided into 4 clusters after 6Tree & 6Hit space partition by the leftmost free dimension (i.e., splitting indicator). Unfortunately, weak characterization is the common drawback of those 4 clusters because Nos. 0, 6 isolated seeds cannot guide target generation due to their fixed nibbles on all the dimensions, and Nos. 1, 2 address regions have large intra-cluster distances (e.g., No. 4 and No. 5 seeds) but small inter-cluster distances (e.g., No. 3 and No. 4 seeds) which intuitively results in many free dimensions for target generation and the inflation of candidate targets' number (i.e., scanning space). The budgets within the capability of Internet-wide network surveys are just a drop in the bucket for such vast scanning space (2.0×16^{18}), where there are two underlying reasons: 1) those seeds (outliers) with unique structures (i.e., address patterns) cause the explosion of free dimensions, e.g., No. 2 seed in No. 1 address region. 2) the inappropriate splitting indicator (leftmost free dimension) could divide some seeds with different structures into one cluster, e.g., Nos. 4, 5 seeds in No. 2 address regions. If those outliers could early be detected and removed, and an appropriate splitting indicator could be exploited, the ideal address regions are able to reduce the probing scope to a narrow range ($\approx 1.0 \times 16^{8}$). However, automatic outlier detection in address regions and appropriate splitting indicator selection are still open problems.

We are thus motivated to propose 6Forest, an efficient IPv6 target generation approach that is integrated with a maximum-covering splitting indicator for space partition and

978-1-6654-5822-1/22/$31.00 ©2022 IEEE

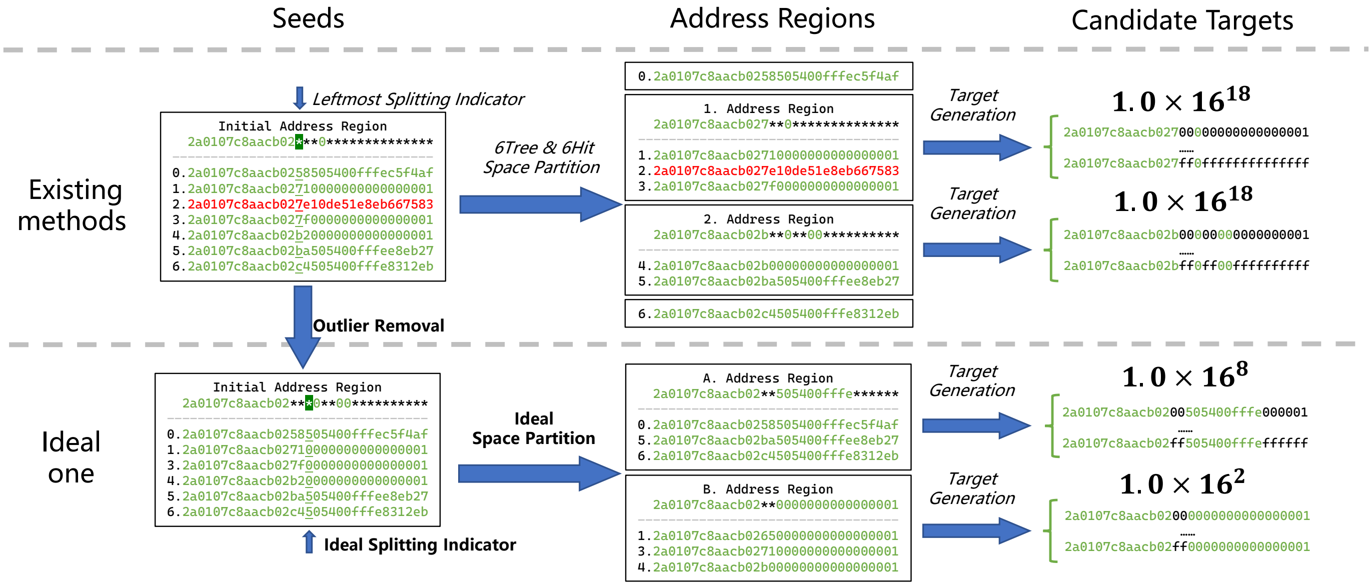

Seeds
Address Regions
Candidate Targets
Existing methods
Leftmost Splitting Indicator
Initial Address Region
2a0107c8aacb02***0***************
0.2a0107c8aacb0258505400fffec5f4af
1.2a0107c8aacb0271000000000000000001
2.2a0107c8aacb027e10de51e8eb667583
3.2a0107c8aacb027f0000000000000001
4.2a0107c8aacb02b20000000000000001
5.2a0107c8aacb02ba505400fffee8eb27
6.2a0107c8aacb02c4505400fffe8312eb
6Tree & 6Hit Space Partition
0.2a0107c8aacb0258505400fffec5f4af
1. Address Region
2a0107c8aacb027**0**************
1.2a0107c8aacb0271000000000000000001
2.2a0107c8aacb027e10de51e8eb667583
3.2a0107c8aacb027f0000000000000001
2. Address Region
2a0107c8aacb02b**0**00**********
4.2a0107c8aacb02b00000000000000001
5.2a0107c8aacb02ba505400fffee8eb27
6.2a0107c8aacb02c4505400fffe8312eb
Target Generation
1.0×16^{18}
2a0107c8aacb02700000000000000001
......
2a0107c8aacb027ff0ffffffffffffff
Target Generation
1.0×16^{18}
2a0107c8aacb02b00000000000000001
......
2a0107c8aacb02bff0ff00ffffffffff
Outlier Removal
Ideal one
Initial Address Region
2a0107c8aacb02***0**00**********
0.2a0107c8aacb0258505400fffec5f4af
1.2a0107c8aacb0271000000000000000001
3.2a0107c8aacb027f0000000000000001
4.2a0107c8aacb02b20000000000000001
5.2a0107c8aacb02ba505400fffee8eb27
6.2a0107c8aacb02c4505400fffe8312eb
Ideal Splitting Indicator
Ideal Space Partition
A. Address Region
2a0107c8aacb02**505400fffe******
0.2a0107c8aacb0258505400fffec5f4af
5.2a0107c8aacb02ba505400fffee8eb27
6.2a0107c8aacb02c4505400fffe8312eb
B. Address Region
2a0107c8aacb02**0000000000000001
1.2a0107c8aacb0265000000000000000001
3.2a0107c8aacb0271000000000000000001
4.2a0107c8aacb02b00000000000000001
Target Generation
1.0×16^{8}
2a0107c8aacb0200505400fffe000001
......
2a0107c8aacb02ff505400fffeffffff
Target Generation
1.0×16^{2}
2a0107c8aacb02000000000000000001
......
2a0107c8aacb02ff0000000000000001

Fig. 1. An example of target generation for the given seeds of 6Tree & 6Hit space partition and an ideal space partition. Note that the outlier (red) has been removed in the ideal space partition.

the enhanced isolation forest algorithm [19] for outlier address detection. Following the convention, 6Forest considers IPv6 addresses as 32-dimensional vectors in the address space. In this work, it first utilizes the given seeds as the initial address region and iteratively divides the current region into seed clusters until the size of clusters' free dimensions is smaller than a threshold according to the maximum-covering splitting indicators instead of the leftmost splitting indicators [16], [17]. Intuitively, each free dimension of the address region will be allocated with the covering values whose formal definition will be disclosed in § III-B. Given an address region, its seeds having the same nibble on the maximum-covering free dimension will be divided into one cluster. Additionally, it is critical to detect the outlier seeds and remove them before seed division for an address region. Specifically, 6Forest exploits the nibbles of every free dimension to build a depth-restricted isolation forest and score seeds by their average depth in the forest structure, and filter out addresses whose scores exceed the threshold (i.e., outliers). Finally, 6Forest randomly samples few addresses from each appropriate address region and conducts the pre-scanning to estimate the hit rates. 6Forest will exploit the regions with a higher estimated hit rate for target generation. More details of 6Forest will be presented in § IV.

To demonstrate the wide application scenarios of 6Forest, we employ six multiple-size candidate datasets for low budget (i.e., 10M) tests to evaluate the performance (i.e., hit rate) referring to the state-of-the-art methods [16]. Additionally, we also utilize two large-scale candidate datasets for high budgets (i.e., 500M) tests to illustrate the performance on worldwide scanning. The results show that 6Forest outperforms the state-of-the-art methods in Internet-wide scanning.

The main contributions of the paper are as follows:

- We present 6Forest, an ensemble learning-based IPv6 target generation approach for Internet-wide scanning, which aims to figure out the open problem even in the state-of-the-art methods: disturbance of outlier seeds and inappropriate splitting indicators. To the best of our knowledge, 6Forest is the first to propose the maximum-covering splitting indicators for space partition and apply the isolation forest to outlier seed detection for target generation.
- Based on 6Forest's appropriate address space partition, we realize efficient IPv6 scanning with just a few additional budgets ($< 1\%$ of given budgets) for pre-scanning. Additionally, we also provide a mathematical guarantee.
- Real network experimental results on eight large-scale datasets show that 6Forest achieves up to 48.19% hit rate in the low budget scanning, which is a more than 116.5% improvement over the state-of-the-art methods. Additionally, in the Internet-wide scanning with high budgets (500M), 6Forest also outperforms the existing methods.

The authors have provided public access to their code at *https://github.com/Lab-ANT/6Forest*.

II. Related Work

The research on IPv6 target generation using seeds started early in 2012. Barnes et al. [20] assume that the known active addresses provide information on the use of addressing schemes. This hypothesis, that the seed information is helpful in discovering more new addresses, has become the foundation of subsequent researches [21] and is strengthened by the recent experimental results. So far, related researches have exploited both the semantic information and the structural information in the seeds.

In the semantic information-based methods, such as 6GC-VAE [22], 6VecLM [23] and 6GAN [24], the seeds are first converted into the vectors according to the IPv6 vector space mapping technology (i.e., IPv62Vec). After the IPv62Vec, the vector datasets will be used to train the deep neural network (e.g., Transformer, Variational Autoencoder, and Generative Adversarial Network). Furthermore, each nibble in the IPv6 addresses and its location will be regarded as a word, and an IPv6 address will be constructed as a sentence. IPv6 target generation problems are converted to the solved text generation problems. However, the huge computing cost of deep neural networks means that these methods cannot scale to large-scale scanning.

For the second class, the structural information of seeds is mainly used to determine the scanning area or to guide the target generation. Foremski et al. [25] introduce Entropy/IP, an algorithm for learning patterns from seeds, which utilizes empirical entropy to group adjacent nibbles 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. Murdock et al. [26] propose 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. [17] propose 6Tree, which leverages a space tree formed from the given seeds' structure to divide the IPv6 address space. 6Tree calculates the density of the nodes on the space tree according to the number of the known active addresses. It then generates target addresses based on the density of the given nodes. Hou et al. [16] propose 6Hit and first apply reinforcement learning to IPv6 active scanning. 6Hit dynamically allocates the budget according to the reward of the scanning on all regions. Through feedback, 6Hit optimizes the subsequent search direction to high-density regions. However, the experiments show that the state-of-the-art methods can only achieve an 11.5% hit rate in the low budget scanning (10M) in the literature [16].

Considering that the semantic information-based model can hardly cope with large-scale IPv6 scanning and its interpretability challenges, we refer to the structural information of seeds to generate the targets. The main weakness of the second technology roadmap is the relatively low hit rate in large-scale scanning caused by the short-sighted leftmost splitting indicators and the "black sheep" (i.e., outlier seeds). To this end, this work is dedicated to addressing those challenges and pushing the limit of the hit rate.

III. Preliminaries

To help better understand the paper, we introduce the structures of IPv6 addresses (i.e., address patterns) in advance which is the theoretical foundation of active IPv6 addresses discovery, and then define some necessary metrics. Finally, we present a formal statement of IPv6 target generation.

A. IPv6 Address Patterns

Instead of a random uniform distribution in the entire address space, the active IPv6 addresses follow some potential address patterns [27]. An IPv6 address, consisting of a global routing prefix, a local subnet identifier, and an interface identifier (IID), is usually assigned to users by the ISPs within dynamical length prefixes. Furthermore, the IPv6 addresses could be categorized into Tab. I and their formal definitions are presented in [28]. In short, IPv6 target generation aims to mine these address patterns from the given seeds S and then conducts IPv6 scanning accordingly.

TABLE I
Categories of Address Patterns

Address Patterns	IID Examples	Comments
Embedded-IPv4	c0:a8:2:a & 192:168:2:10	embed IPv4 address 192.168.2.10
Embedded-Port	0:0:0:50 & 0:0:0:80	embed port 80 for HTTP services
IEEE-derived[a]	0250:56ff:fe89:49be	embed MAC address 02:50:56:89:49:be
Low-byte	0:0:0:1 & 0:0:0:2	embed all zero prefixes except the low byte
Pattern-bytes	face:b00c:3333:1b26 & face:b00c:3333:2c45	common pattern-bytes face:b00c:3333
Randomized[b]	10de:51e8:eb66:7583	pseudorandom

[a] The inserted word "fffe" is so-called the Organizationally Unique Identifier [29].

[b] It characterizes the pseudorandom by using Stateless Address Autoconfiguration (SLAAC) [30].

B. Definition

Definition 1: IPv6 address vectors. A 128-bit IPv6 address whose binary integer is N can be converted to a 32-dimensional vector v, each dimension of which is a hexadecimal integer, and the δ^{th} dimension can be presented as follows:

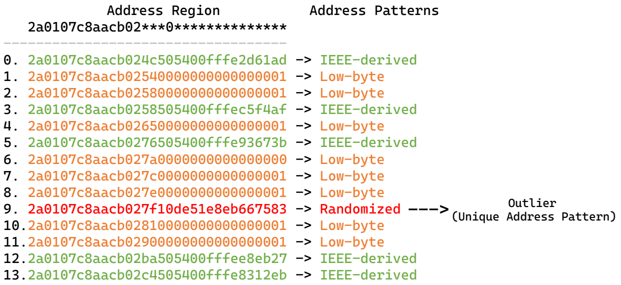
$$v[\delta] = (N >> (0\text{x}80 - \delta << 0\text{x}2)))\&0\text{x}f \qquad (1)$$

Definition 2: Seed (address) regions. After space partition, the set of all given seeds S is divided into the small-scale seed clusters, which usually are the so-called seed (address) regions except for those clusters consisting of only one seed. The isolated seed has the fixed nibbles on all dimensions and cannot guide target generation.

Definition 3: Free dimensions and fixed dimensions. We consider that there are 32 dimensions and each dimension ranges from 0x0 to 0xf in the entire IPv6 address space. For a given address region, those dimensions where all the seeds have the same nibble values are the so-called fixed dimensions otherwise they are the free dimensions. Following the convention, we use the wildcard symbol "*" to denote an uncertain nibble (free dimension).

Definition 4: Scanning space. Obviously, the maximum number of generated IPv6 targets are based on the size of the given region's free dimensions ζ and the scanning space is equal to 16^{ζ}.

Definition 5: Outlier seeds. In a given address region, those seeds having unique address patterns will be regarded as the outliers.



```
              Address Region            Address Patterns
   2a0107c8aacb02***0**************
   --------------------------------
0. 2a0107c8aacb024c505400fffe2d61ad -> IEEE-derived
1. 2a0107c8aacb02540000000000000001 -> Low-byte
2. 2a0107c8aacb02580000000000000001 -> Low-byte
3. 2a0107c8aacb0258505400fffec5f4af -> IEEE-derived
4. 2a0107c8aacb02650000000000000001 -> Low-byte
5. 2a0107c8aacb0276505400fffe93673b -> IEEE-derived
6. 2a0107c8aacb027a0000000000000000 -> Low-byte
7. 2a0107c8aacb027c0000000000000001 -> Low-byte
8. 2a0107c8aacb027e0000000000000001 -> Low-byte
9. 2a0107c8aacb027f10de51e8eb667583 -> Randomized --->      Outlier
                                                    (Unique Address Pattern)
10.2a0107c8aacb02810000000000000001 -> Low-byte
11.2a0107c8aacb02900000000000000001 -> Low-byte
12.2a0107c8aacb02ba505400fffee8eb27 -> IEEE-derived
13.2a0107c8aacb02c4505400fffe8312eb -> IEEE-derived
```

Fig. 2. An outlier example for a given address region.

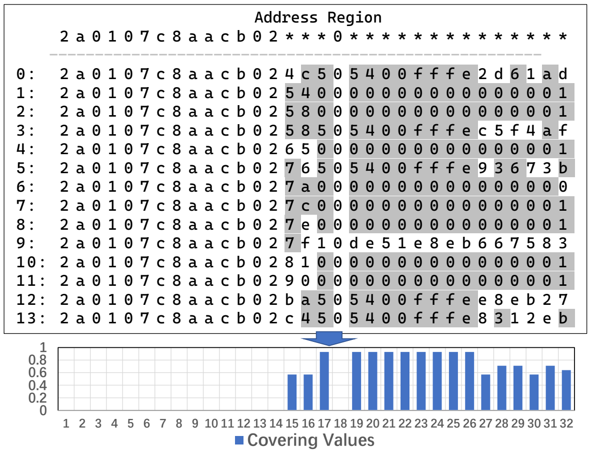
Definition 6: Covering. The frequency sum of non-unique nibbles for a free dimension of the given address region. Intuitively, the purpose of the covering aims to find the splitting indicator that minimizes the number of isolated seeds. Suppose a region that includes K seeds and whose i^{th} dimensions nibble vector is V_i, the frequency of the nibble j is defined as $|V_i == j|$:

$$K = \sum_{j=0}^{15} |V_i == j| \tag{2}$$

After the normalization for evaluation, the covering $Cover_i$ of i^{th} dimensions could be shown as follows:

$$Cover_i = \frac{\sum_{j=0,|V_i==j|>1}^{15} |V_i == j|}{K} \tag{3}$$

As shown in Fig. 3, 6Forest allocates a covering value for each free dimension and adopts the first maximum-covering free dimension (17th) rather than leftmost free dimension (15th) as the splitting indicator for subsequent seed division. More details will be presented in § IV-B.



```
                          Address Region
     2 a 0 1 0 7 c 8 a a c b 0 2 * * * 0 * * * * * * * * * * * * * *
     ---------------------------------------------------------------
0:   2 a 0 1 0 7 c 8 a a c b 0 2 4 c 5 0 5 4 0 0 f f f e 2 d 6 1 a d
1:   2 a 0 1 0 7 c 8 a a c b 0 2 5 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
2:   2 a 0 1 0 7 c 8 a a c b 0 2 5 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
3:   2 a 0 1 0 7 c 8 a a c b 0 2 5 8 5 0 5 4 0 0 f f f e c 5 f 4 a f
4:   2 a 0 1 0 7 c 8 a a c b 0 2 6 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
5:   2 a 0 1 0 7 c 8 a a c b 0 2 7 6 5 0 5 4 0 0 f f f e 9 3 6 7 3 b
6:   2 a 0 1 0 7 c 8 a a c b 0 2 7 a 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7:   2 a 0 1 0 7 c 8 a a c b 0 2 7 c 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
8:   2 a 0 1 0 7 c 8 a a c b 0 2 7 e 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
9:   2 a 0 1 0 7 c 8 a a c b 0 2 7 f 1 0 d e 5 1 e 8 e b 6 6 7 5 8 3
10:  2 a 0 1 0 7 c 8 a a c b 0 2 8 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
11:  2 a 0 1 0 7 c 8 a a c b 0 2 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
12:  2 a 0 1 0 7 c 8 a a c b 0 2 b a 5 0 5 4 0 0 f f f e e 8 e b 2 7
13:  2 a 0 1 0 7 c 8 a a c b 0 2 c 4 5 0 5 4 0 0 f f f e 8 3 1 2 e b
```

Fig. 3. A covering example of the address region. Note that only the free dimensions are allocated with the covering values and all the non-unique nibbles (grey background) of one free dimension are colored.

C. Problem Statement

To clarify the boundary of the IPv6 target generation, we define that an active IPv6 address should meet the following requirements, namely alive host, Internet access, and probe response. Intuitively, if a scanner sends packets following a certain protocol to a target address and the target responses, 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 the set of all active addresses under this protocol in the address space is A. The seeds S is a set of known active IPv6 addresses under the same protocol in the address space, which can usually be collected from Internet traffic, DNS records, and the existing IPv6 scanning methods. Considering the algorithm τ with the given scanning budget b, the output can be defined as $\tau(S,b)$ and the goal of IPv6 target generation is stated as follows:

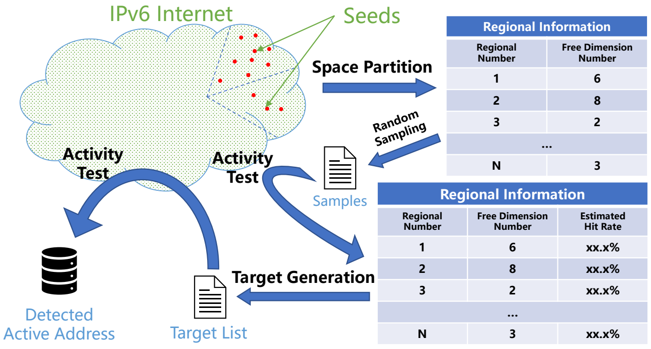
$$\max \quad \frac{\tau(S,b) \cap A}{b} \qquad s.t. \quad \tau(S,b) \cap S = \emptyset \tag{4}$$

Where $\tau(S,b) \cap A$ could be verified through the existing scanner [6] and $\frac{\tau(S,b)\cap A}{b}$ is the formal definition of hit rate. Besides, the generated target should exclude known seeds, that is $\tau(S,b) \cap S = \emptyset$.

IV. Design of 6Forest

We first present the system overview of 6Forest and then describe the details of its key technical components: outlier seed detection, space partition, and hit rate estimation.

A. System Overview



Regional Information	
Regional Number	Free Dimension Number
1	6
2	8
3	2
...	
N	3

Regional Information		
Regional Number	Free Dimension Number	Estimated Hit Rate
1	6	xx.x%
2	8	xx.x%
3	2	xx.x%
...		
N	3	xx.x%

Fig. 4. The workflow of 6Forest.

Fig. 4 illustrates the main workflow of 6Forest. It first conducts the space partition of the entire IPv6 address space with a set of known active IPv6 addresses (i.e., seeds) by iteratively dividing the seeds into smaller clusters based on the maximum-covering splitting indicators. After the space partition, 6Forest exploits the isolation forest algorithm to detect the outlier seeds and remove them from the address region. Finally, 6Forest samples a few targets ($< 1\%$ of the given budgets) from the ultimate address regions and pre-scans them for the estimation of hit rates in the address regions. Those regions with a high estimated hit rate will be prioritized for target generation.

B. Space Partition

The space partition in state-of-the-art methods [16], [17] is realized by recursively dividing the seeds into smaller seed clusters to construct a tree structure namely "space tree". Apart from the outlier seeds challenge, the inappropriate splitting indicator (rough leftmost free dimension) is also the drawback through the entire space partition.

To this end, we proposed a new metric to evaluate the priority of all the free dimensions as the splitting indicator, i.e., covering. The covering represents the number of non-unique nibbles of all the seeds on the given free dimension. For the full use of seeds, the maximum covering highlights a free dimension where there are the fewest isolated seeds after the seed division. Furthermore, the free dimensions in an ideal seed region are not strictly behind the fixed dimensions and the simple leftmost splitting indicator could result in the misdivision of seeds. For example in Fig. 3, if the leftmost free dimension (15th dimension) was adopted as the splitting indicator for space partition of the seeds rather than the maximum covering one (i.e., 17th dimension), those seeds belonging to different address patterns would be divided into one address regions and cause the inflation of scanning space, e.g., No. 2 seed (Low-byte) and No. 3 seed (IEEE-derived).

Additionally, the existing methods adopt a Depth First Search strategy to recursively split the current leaf nodes for expansion of "space tree", which means the preservation of the entire space tree structure and can be hardly performed in parallel. To this end, we exploit the Breadth-First Search strategy and a FIFO structure (i.e., Queue) rather than the tree structure to conduct the space partition. During the space partition, only those address regions that can be used for subsequent division are saved in the FIFO structure, which reduces space complexity theoretically. Furthermore, the FIFO structure in 6Forest can be shared by multiple CPUs (i.e., resource pooling), which could improve the efficiency of space partition significantly.

Algorithm 1 Space Partition

```
Require: the set of seeds $S$, the seed number threshold $\beta$.
Ensure: the set of seed regions $SR$
initialize the FIFO structure $NQ$
$SR = \emptyset$
$NQ$.put($S$)
while not $NQ$.empty() do
    $cur$ = $NQ$.get()
    if $cur$.size()< $\beta$ then
        $SR$.add($cur$)
        continue
    end if
    $clusters = \text{SEEDCLUSTERING}(cur)$
    for $\delta \in clusters$ do
        $NQ$.put($\delta$)
    end for
end while
return $SR$
function SEEDCLUSTERING($Seeds$)
    $maxcovering = -1$
    for $i \in Seeds.freedimensions$ do
        $count, V_i = \text{BINCOUNT}(Seeds, i, 16)$
        $covering = \frac{\sum_{j=0,count[j]>1}^{15} count[j]}{|Seeds|}$
        if $covering > maxcovering$ then
            $clusters = V_i$
            $maxcovering = covering$
        end if
    end for
    return $clusters$
end function
```

The pseudocode of 6Forest's space partition is shown in Alg. 1. In a round of seed division, 6Forest first takes the node from the FIFO structure and respectively allocates all the free dimensions with the covering values. Leveraging the nibbles of the free dimension with maximum covering value as the cluster representative, 6Forest divides the current seeds of the node into the corresponding child nodes, i.e., clusters with the same nibble values, and then pushes them in the FIFO structure. Additionally, if there are several free dimensions are commonly allocated with the maximum covering value, 6Forest chooses the leftmost one.

Fig. 5 shows an IPv6 space partition example of a candidate dataset C_1 described in § V and the tracking of a seed (2a0225a9000209120000000000000001). It is worth highlighting that the space partition of 6Forest is similar to that of 6Tree & 6Hit for the large address regions (e.g., Nodes 0, 5) because the splitting indicators are commonly the leftmost free dimension of the current region which could fast present the rough classification of the initial seeds. For a small address region (e.g., Node 508), 6Forest chooses the splitting indicator that minimizes the number of isolated seeds, regardless of its location. In a nut, 6Forest's space partition is conducted from the entire seeds' perspective.

Complexity analysis: Let m be the number of input seed addresses. The algorithm first sorts the m seeds, which have the worst-case time complexity of $O(mlogm)$. Then, the space partition will traverse each address vector once per free dimension and which will be not beyond $O(m)$ time consumption totally, and the worst-case time complexity is $O(mlogm)$. Besides, 6Forest will only retain the entire seeds once and not reserve the interim nodes during space partition which reduces the space complexity to $O(m)$ while existing methods [16], [17] need to keep the entire tree structure and their worst-case space complexity is $O(mlogm)$.

Remark 1: There has been work trying to avoid the drawback of leftmost splitting indicators, such as DET [15]. DET utilizes the "heuristic" indicators based on the smallest but non-zero entropy for the space partition, which could address the inappropriate space partition of small regions partly. However, there are two weaknesses in this method: 1) For the large seed regions, the nibble values are evenly distributed on the many free dimensions considering its large number of seeds. The smallest entropy, as a splitting indicator, is so sensitive that it could lead to the local optima. 2) For the small-scale

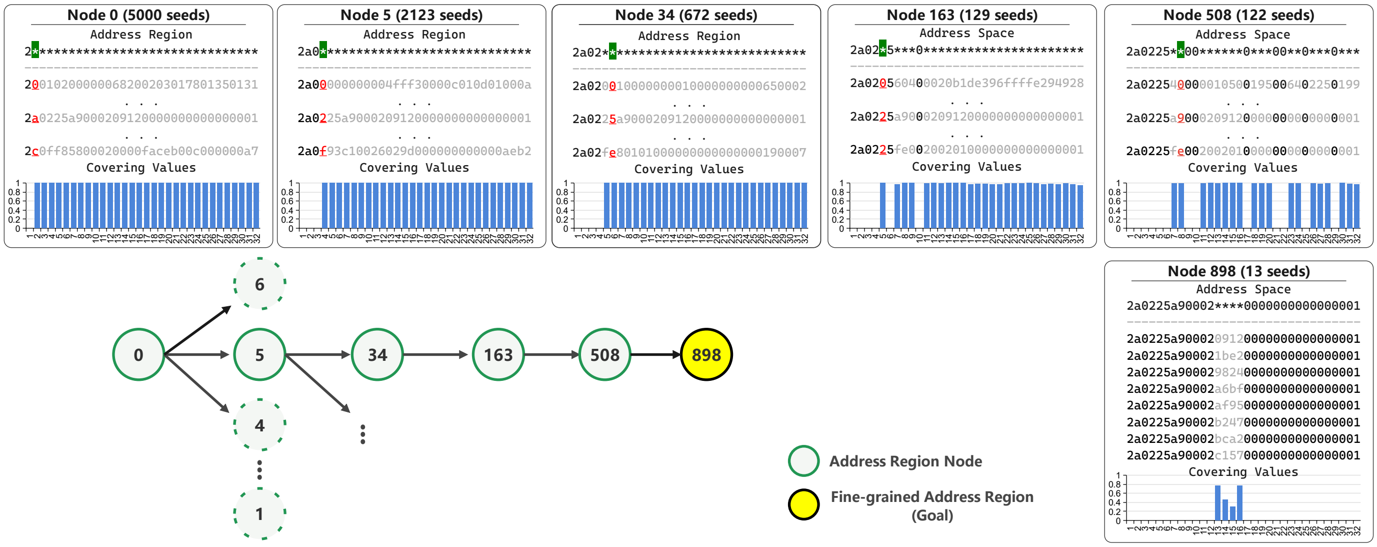

Node 0 (5000 seeds)
Address Region
2*******************************
20010200000068200203017801350131
. . .
2a0225a900020912000000000000000001
. . .
2c0ff85800020000faceb00c000000a7
Covering Values
Node 5 (2123 seeds)
Address Region
2a0*****************************
2a0000000004fff30000c010d01000a
. . .
2a0225a9000209120000000000000001
. . .
2a0f93c10026029d000000000000aeb2
Covering Values
Node 34 (672 seeds)
Address Region
2a02****************************
2a020010000000010000000000650002
. . .
2a0225a9000209120000000000000001
. . .
2a02fe80101000000000000000190007
Covering Values
Node 163 (129 seeds)
Address Space
2a02*5***0**********************
2a020560400020b1de396ffffe294928
. . .
2a0225a9000209120000000000000001
. . .
2a0225fe002002010000000000000001
Covering Values
Node 508 (122 seeds)
Address Space
2a0225**00******0***00**0***0***
2a022540000010500195006402250199
. . .
2a0225a9000209120000000000000001
. . .
2a0225fe002002010000000000000001
Covering Values
Node 898 (13 seeds)
Address Space
2a0225a90002****0000000000000001
2a0225a9000209120000000000000001
2a0225a900021be20000000000000001
2a0225a9000298240000000000000001
2a0225a90002a6bf0000000000000001
2a0225a90002af950000000000000001
2a0225a90002b2470000000000000001
2a0225a90002bca20000000000000001
2a0225a90002c1570000000000000001
Covering Values
0 0.2 0.4 0.6 0.8 1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
0
5
6
4
1
34
163
508
898
Address Region Node
Fine-grained Address Region (Goal)

Fig. 5. The instance of space partition is based on the maximum covering splitting indicators (green background), which generates 1585 address space nodes and 1065 seed regions (partially shown) from the dataset C_1 including 5000 seeds.

seed regions, the unique nibbles of the seeds on the given dimensions can still be added to the entropy, which means that DET prefers to divide the seeds into a large cluster with many isolated seeds instead of several clusters without isolated seeds. Unfortunately, the isolated seeds cannot be used for target generation.

C. Outlier Seed Detection

We have defined the outlier seed and pointed out that it partly causes the inflation of the scanning space. However, how to automatically detect these anomalies is still unresolved. In this section, 6Forest exploits an isolation forest algorithm to filter out the outliers and provide the address regions without outliers for IPv6 target generation.

Algorithm 2 Outlier Detection

```
Require: the seeds $S$ of given region
Ensure: new address regions $NS$, outlier seeds $OS$
$threshold = \frac{|S.freedimensions|}{|S|}$
for $i \in S.freedimensions$ do
    IsolationTree($S, i$)
end for
for $s \in S$ do
    if $s.weight >> threshold$ then
        $OS$.add($s$)
    else
        $NS$.add($s$)
    end if
end for
return $NS, OS$
function IsolationTree($Seeds, splitindex$)
    $count, clusters$ = BinCount($Seeds, splitindex, 16$)
    ▷ A nibble typically is notated in hexadecimal format
    $k$ is the number of 1 in list $count$
    for $\delta \in clusters$ do
        if $|\delta| == 1$ then
            $seed = \delta[0]$
            $seed.weight = seed.weight + \frac{1}{k}$
        end if
    end for
end function
function BinCount($Seeds, dimension, maxlength$) ▷ Divisive clustering and frequency counting
    initial $count, clusters$ with two $maxlength$ lists
    for $s \in Seeds$ do
        $count[s[dimension]] + = 1$
        $clusters[s[dimension]]$.add($s$)
    end for
    return $count, clusters$
end function
```

The basic hypothesis is that the outliers are the first to be isolated and their nibble values are unique on most of the free dimensions. The pseudocode of outlier seed detection is shown in Alg. 2. 6Forest first builds a depth-restricted isolation tree on each free dimension. Then, the weights from the isolation forest will respectively be summed up as anomaly scores of the seeds. Finally, those seeds whose anomaly scores are above the threshold will be regarded as the outliers.

We use an example for clarity: Given a seed region 2a0107c8aacb02***0************** in Fig. 2, 6Forest first builds the corresponding isolation forest. In Fig. 6 three typical isolated trees represent the seed division results on the 15th, 17th, and 28th dimensions. And the isolated seeds of a restricted depth tree will be scored for the representation of outliers. Note that the outlier (i.e., No. 9) has a unique address pattern and could be isolated in most trees but not all (e.g., the isolation tree of the 15th dimension).

A straightforward idea is using the comprehensive results of the entire forest, i.e., ensemble learning. However, it is

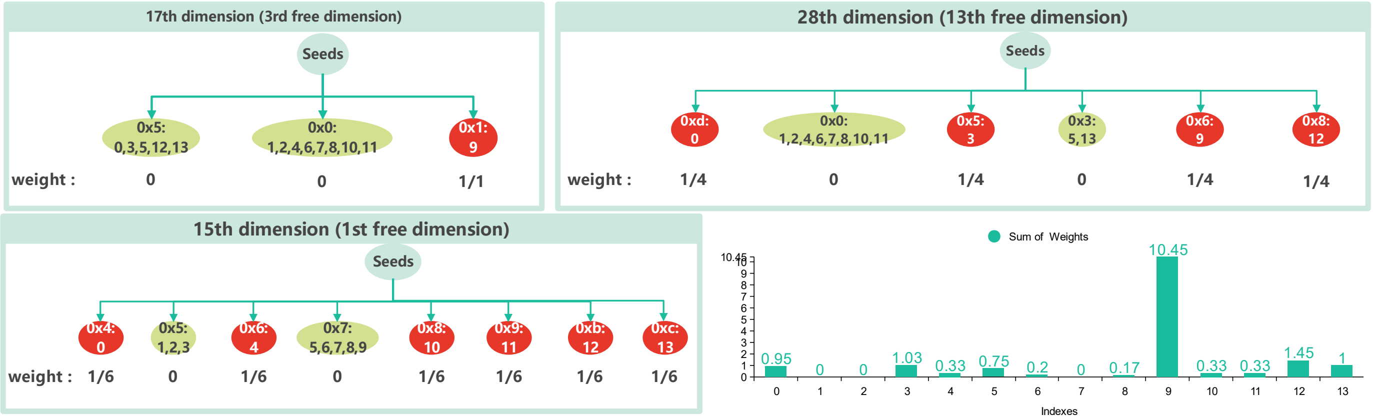


Fig. 6. The example of the isolation forest (partially shown) for outlier detection of the seed region (2a0107c8aacb02***0**************) in Fig. 2, where the splitting indicators including the corresponding indexes and the weights of unique seeds (red) are displayed. Note that the No. 9 seed has the highest score (10.45) when the seeds' weights are summed respectively.

not wise to simply accumulate scores for representation of outliers because the normal address regions also have different nibbles on a few free dimensions where the isolation tree usually mistakes their seeds as outliers. In other words, the splitting indicators (e.g., 15th dimension) themselves could be incorrect whose weights should decrease if the most of seeds are isolated.

To facilitate the calculation, the weights are dynamically set to $\frac{1}{k}$ assuming that there are k isolated seeds in a isolation tree. Considering an address region with f free dimensions and m seeds, the sum of scores for outliers representation on all free dimensions is equal to f and the expectation score of one seed is $\frac{f}{m}$. Accordingly, those seeds far away from the expectation score of the seed region are the outliers and need to be removed. For the given region with 13 seeds and 17 free dimensions in Fig. 2 and Fig. 6, 6Forest regards those seeds weighing over the threshold $\frac{17}{13} << 10.45$ as the outliers (e.g., No. 9 seed with a 10.45 weight), which consists with the truth.

Remark 2: For a seed region with m seeds as input, the time complexity of Alg. 2 is $O(m)$, because the number of isolation trees is always less than the constant 32 and the time consumption of the nibble count does not exceed $O(m)$. The time complexity of outlier detection is still $O(m)$ for a given address region with m seeds.

D. *Estimation of Hit Rate*

Considering that the seed density in a region may not align well with the density of active addresses in the region [16], it is not wise to allocate the budgets according to the number of seeds or randomly, even in the regions with narrow probing scope. Thus, 6Forest pre-scans the samples from the address regions to estimate the hit rates and then prioritizes large-scale scanning in those regions with high estimated hit rates.

The basic institution is that the sampling hit rates could represent the hit rates of entire regions within the permissible error. Assume that the hit rate of the given seed region is p with ζ free dimensions and n addresses are sampled for pre-scanning in which A addresses are active and respond to the scanning probes. An address in the region is active or not which could be regarded as a random variable $Y \in \{0, 1\}$ and the Y_i is the i^{th} address of n sampling addresses. Thus, $Y_1, Y_2, ..., Y_n$ follow the independent Bernoulli distributions:

$$P(\sum_{i=1}^{n} Y_i \leq k) = \sum_{j=0}^{k} \binom{n}{j} p^j (1-p)^{n-j} \tag{5}$$

Let $k = (p - \varepsilon)n$, ε is the permissible error. According to the Hoeffding's inequality [31]:

$$P(\sum_{i=1}^{n} Y_i \leq (p - \varepsilon)n) \leq e^{-2\varepsilon^2 n} \tag{6}$$

Let $\varepsilon = \sqrt{\frac{\ln n}{n}}$:

$$P(\sum_{i=1}^{n} Y_i \leq (p - \sqrt{\frac{\ln n}{n}})n) \geq 1 - \frac{2}{n^2} \tag{7}$$

Note that $\frac{\sum_{i=1}^{n} Y_i}{n}$ is the estimated hit rate, namely $\hat{p}$.

$$P(p \geq \hat{p} + \sqrt{\frac{\ln n}{n}}) \geq 1 - \frac{2}{n^2} \tag{8}$$

We can conclude that with probability at least $1 - \frac{2}{n^2}$, $p \geq \hat{p} + \sqrt{\frac{\ln n}{n}}$. In other words, we can guarantee that the hit rates of sampling pre-scanning can be used to guide the worldwide IPv6 scanning.

Technically, not all the address regions are exploited for the hit rate estimation, especially those with few free dimensions ($\zeta \leq 3$) because the full scanning directly is a good choice.

V. Evaluation

We compare the performance of 6Forest and other existing methods, including 6Hit [16], 6Tree [17], 6Gen [26] and Entropy/IP [25], with real-world test. Excluding 6Gan [24] from the baselines is mainly due to the unavailability of its source code.

A. Dataset Description

Gasser et al. [32] have collected active IPv6 addresses from multiple sources (e.g., websites, DNS records and TLS certificates) whose range from 2001:1210:100:1::17 to 2c0f:ffc8:4001:4::2. We used the data published in July 1st, 2021 as the source of address set, including $15.1M$ detected IPv6 addresses. Additionally, not all seeds are "active addresses" as defined in § III-C and respond to the ICMP scanning probes. After the necessary active test of the seeds, there are $2.8M$ "active" IPv6 addresses, i.e., seeds S.

Considering that the fast scanning with high budgets is beyond the capacity of some existing methods (e.g., 6 Hit and Entropy/IP) due to the efficiency limitation, we respectively exploit two different scale seed sets for hit rate evaluation both on high and low budgets. To investigate the impact of initial seed on the performance of different methods in a real-world test, we adopt two strategies to form seed sets from S: downsampling and biased-sampling. In downsampling, we randomly sample a certain number of addresses from S as candidate seed set $C_i, i \in \{1, 2, 3\}$. In biased-sampling, we order the addresses in C and then extract a certain number of adjacent addresses as seed set $C_i, i \in \{4, 5, 6, 7, 8\}$. More details of these seed sets are shown in Tab. II.

TABLE II
CHARACTERISTICS OF SEED SETS

Scanning Budget	Seed Set	Number	Selection Strategy	Range
Low	C_1	$5k$	Down Sampling	2001:200:0:1008::1 - 2c0f:feb0:13:2::9
	C_2	$30k$	Down Sampling	2001:200:0:1cd1::13 - 2c0f:feb0:e::ff7f
	C_3	$0.1M$	Down Sampling	2001:200:0:801::2:5 - 2c0f:ff40:1:1012::1
	C_4	$5k$	Biased Sampling	2001:200:0:1::1 - 2001:240:bb40:4235::1
	C_5	$30k$	Biased Sampling	2001:200:0:1::1 - 2001:380:4f:b80::2
	C_6	$0.1M$	Biased Sampling	2001:200:0:1::1 - 2c0f:fef8:0:102f::2
High	C_7	$1.4M$	Biased Sampling	2001:200:0:1::1 - 240e:38b:86c4:6d01::
	C_8	$1.4M$	Biased Sampling	240e:38b:86ff:1f01:: - 2c0f:ffc8:4001:4::2

B. Real-world Test Results

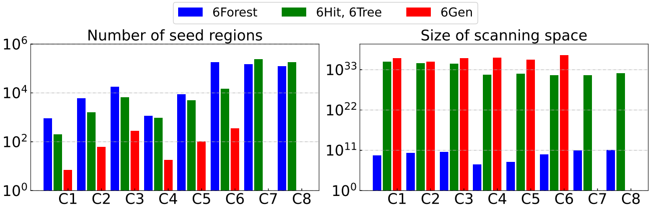


Fig. 7. The numbers of generated seed region and their entire scanning space (logarithmic scale) on the candidate datasets.

We conducted the Internet-wide scanning experiments in July 2021 and limited the probing rate to 200 million bps for strictly ensuring Internet citizenship as suggested by Partridge and Allman [33]. The experiments including performance evaluation of 6Forest and existing methods were performed at the Hangzhou data center of Alibaba Cloud (AS37963). An IPv6 single-threaded address scanner was deployed on a Linux platform with an Intel Platinum 8260 Processor (2.50GHz, 4 core) and 16GB RAM.

For fair comparison, we conducted the preprocessing step (i.e., space partition of 6Tree and 6Hit, seeds clustering of 6Gen, seeds statistics of Entropy/IP) offline. This is due to the following considerations: some methods (e.g., 6Gen) require huge computing resources and the main metrics of IPv6 target generation are the hit rates of large-scale scanning.

1) Scanning Space: Compared with the existing whose target generation is based on seed regions, 6Forest successfully refined the seed regions and reduced the entire scanning space by several orders of magnitude, which can account for the high performance of 6Forest. Note that 6Gen cannot finish the space partition (seeds clustering) within the reasonable time limit on the candidate set $C_i, i \in \{7, 8\}$ due to its complexity.

2) Scanning with Low Budgets: We compared the numbers of detected active addresses of 6Forest and the baselines using the different seed set $C_i, i \in [1, 6]$ introduced above. The results are shown in Fig. 8 where we can see that the methods (i.e., 6Forest, 6Hit, and 6Tree) based on space partition are much higher than others. Furthermore, 6Forest outperforms the state-of-the-art methods on all downsampling datasets $C_i, i \in [1, 3]$ and the largest biased sampling dataset C_6, which demonstrates the powerful ability of 6Forest. We cannot deny that 6Forest does not always lead in all scenarios and is slightly weaker than the newest invented method (6Hit) especially for those small biased-sampling datasets (C_4, C_5). The underlying reason is that the seeds in small biased-sampling datasets overlap very few address regions with the high-density and 6Hit's conservative budget allocation for dynamic scanning takes a weak lead in this case.

3) Scanning with High Budgets: The results on the sampling seed sets are unstable. It is necessary to exploit the entire seed set (2M) for the evaluation with high budgets (500M). However, the requirements of the high budget scanning are beyond the capability of some existing methods: Entropy/IP [25] and 6Gen [26] are not yet able to process the entire seed set (2.8M) considering their huge computing requirements. 6Hit [16] determines the next scanning direction based on the result of probing which demands a lot of time consumption to wait for the probing response and is not able to finish the 500M budget scanning within the reasonable time limit. Thus, the entire seeds are divided into two large seed sets $C_i, i \in \{7, 8\}$ and 6Tree is utilized as the baseline of high budget scanning. As shown in Fig. 9, we can see that the hit rates both decrease during scanning, and 6Forest's performance is maintained at a higher level than 6Tree's in general.

Finally, Tab. III presents all the real-world test results. Following the existing methods [16], 6Forest has removed the IPv6 aliases from results according to the known alias prefixes [32] but does not ensure that there will be no unknown

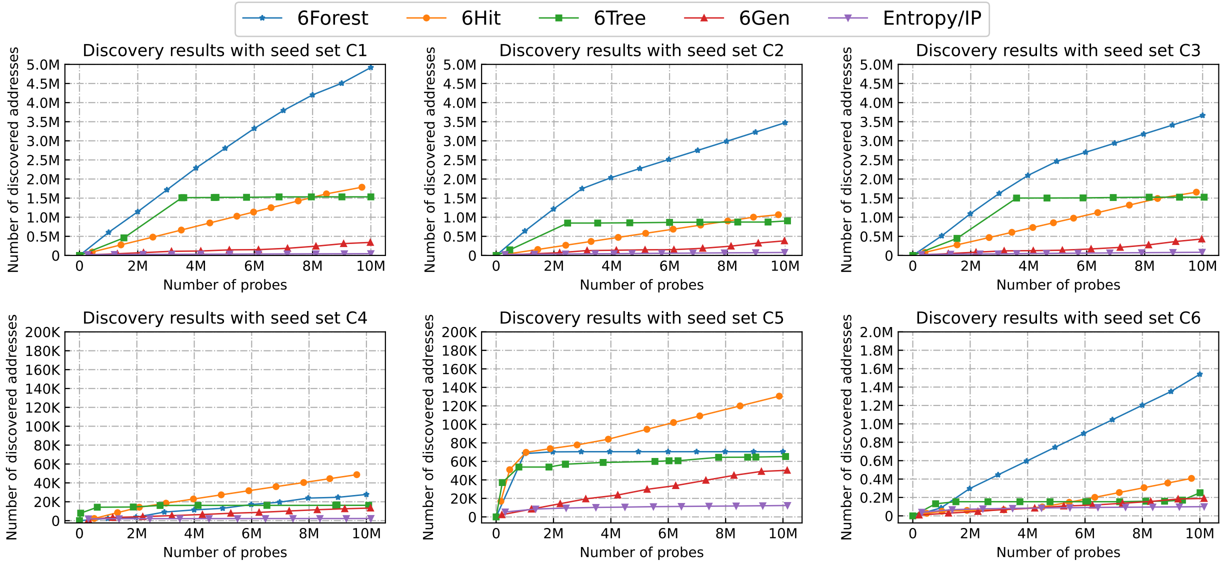


Fig. 8. Discovery of active addresses with the low budget of 10M on seed set $C_i, i \in [1, 6]$. 6Forest discovered more active addresses on most seed sets $C_i, i \in \{1, 2, 3, 6\}$.

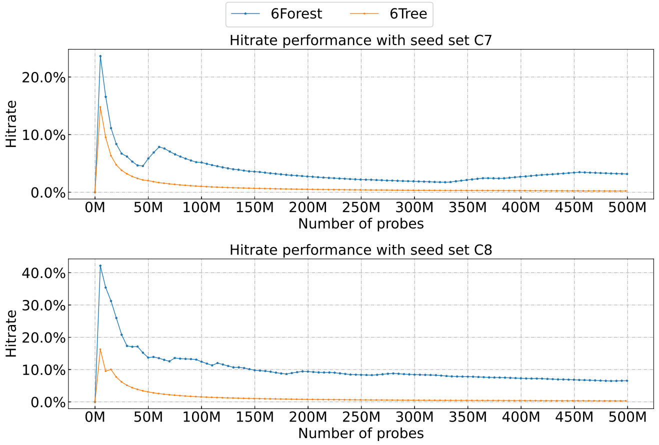


Fig. 9. The hit rates of 6Forest and 6Tree with a total budget of 500M on seed set $C_i, i \in \{7, 8\}$.

aliases because IPv6 alias resolution is not within our work. It is worth highlight that the high hit rates of 6Forest, 6Hit, 6Tree do not represent the real-world average density of active IPv6 addresses. In fact, active IPv6 addresses are still sparse for the vast IPv6 address space. Besides, the results on the sampling-based datasets (i.e., $C_i, i \in [1, 6]$) are significantly affected by the sampling seeds. Therefore, it makes no sense to simply compare the hit rates of different experiments and we should focus on the performance improvement of 6Forest over the existing methods. In summary, large-scale scanning on the IPv6 Internet shows that 6Forest is comparable to the state-of-the-art methods (6Hit) for the candidate datasets $C_i, i \in \{4, 5\}$ and achieved 3.17%-48.19% hit rates for all other candidate datasets, which is respectively a 116.5%-266.7% improvement for low budget scanning and about 15x-21x improvement for high budget scanning over the state-of-the-art methods.

TABLE III
AVERAGE HIT RATES OF TARGET GENERATION ALGORITHM

Approach	Low Budget Tests						High Budget Tests	
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
Entropy/IP [25]	0.46%	0.78%	0.84%	0.02%	0.12%	0.99%	-	-
6Gen [26]	3.42%	3.84%	4.32%	0.13%	0.50%	1.86%	-	-
6Tree [17]	15.33%	8.98%	15.17%	0.16%	0.65%	2.51%	0.21%	0.31%
6Hit [16]	18.42%	10.91%	16.92%	**0.50%**	**1.32%**	4.20%	-	-
6Forest*	**48.19%**	**34.74%**	**36.64%**	**0.28%**	**0.70%**	**15.40%**	**3.17%**	**6.54%**

* Our work

VI. Conclusion

In this work, we illustrate the low hit rate challenges of the IPv6 target generation caused by the “black sheep” (outlier seeds) and the leftmost splitting indicator for space partition. To this end, we propose 6Forest, an ensemble learning-based approach for discovering active IPv6 addresses. 6Forest adopts a novel maximum-covering metric for the choice of splitting indicators and an enhanced isolation forest algorithm to automatically filter out the outlier seeds, which generates fine-grained address regions and effectively reduces the size of the scanning space. With the real-world test, 6Forest has achieved much better performance on hit rate than the state-of-the-art solutions, 6Hit and 6Tree, 6Gen, and Entropy/IP.

Acknowledgment

This work is supported by the National Key Research and Development Program of China (2018YFB1800202). The corresponding authors are Zhiping Cai and Bingnan Hou.

References

[1] M. Bakardjieva, *Internet society: The Internet in everyday life.* Sage, 2005.

[2] S. Pack, X. Shen, J. W. Mark, and J. Pan, “Adaptive route optimization in hierarchical mobile ipv6 networks,” *IEEE transactions on mobile computing*, vol. 6, no. 8, pp. 903–914, 2007.

[3] Google, “Ipv6 adoption statistics,” 2021. [Online]. Available: https://www.google.com/intl/en/ipv6/statistics.html

[4] G. Huston., “Ipv6 bgp table reports,” 2021. [Online]. Available: https://bgp.potaroo.net/index-v6.html

[5] K. Borgolte, S. Hao, T. Fiebig, and G. Vigna, “Enumerating active ipv6 hosts for large-scale security scans via dnssec-signed reverse zones,” in *Proceedings of the IEEE Symposium on Security and Privacy*. San Francisco, CA, USA: IEEE Computer Society, 2018, pp. 770–784.

[6] Z. Durumeric, E. Wustrow, and J. A. Halderman, “Zmap: Fast internet-wide scanning and its security applications,” in *Proceedings of the {USENIX} Security Symposium*. Washington, DC, USA: USENIX Association, 2013, pp. 605–620.

[7] D. Myers, E. Foo, and K. Radke, “Internet-wide scanning taxonomy and framework,” in *Proceedings of the Australasian Information Security Conference*, vol. 27. Sydney, Australia: Australian Computer Society, Inc, 2015, p. 30.

[8] R. Beverly, “Yarrp’ing the internet: Randomized high-speed active topology discovery,” in *Proceedings of the Internet Measurement Conference*. Santa Monica, CA, USA: ACM, 2016, pp. 413–420.

[9] R. Beverly, R. Durairajan, D. Plonka, and J. P. Rohrer, “In the ip of the beholder: Strategies for active ipv6 topology discovery,” in *Proceedings of the Internet Measurement Conference*. Boston, MA, USA: ACM, 2018, pp. 308–321.

[10] D. Plonka and A. Berger, “Kip: A measured approach to ipv6 address anonymization,” *arXiv preprint arXiv:1707.03900*, 2017.

[11] J. Czyz, M. Luckie, M. Allman, M. Bailey *et al.*, “Don’t forget to lock the back door! a characterization of ipv6 network security policy,” in *Proceedings of the Network and Distributed Systems Security*. San Diego, CA, USA: The Internet Society, 2016.

[12] Q. Scheitle, O. Gasser, P. Sattler, and G. Carle, “Hloc: Hints-based geolocation leveraging multiple measurement frameworks,” in *Proceedings of the Network Traffic Measurement and Analysis Conference*. Dublin, Ireland: IEEE, 2017, pp. 1–9.

[13] A. S. A. M. 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.

[14] O. Gasser, Q. Scheitle, S. Gebhard, and G. Carle, “Scanning the ipv6 internet: towards a comprehensive hitlist,” *arXiv preprint arXiv:1607.05179*, 2016.

[15] G. Song, L. He, Z. Wang, J. Yang, T. Jin, J. Liu, and G. Li, “Towards the construction of global ipv6 hitlist and efficient probing of ipv6 address space,” in *Proceedings of the IEEE/ACM International Symposium on Quality of Service*. Hangzhou, China: IEEE, 2020, pp. 1–10.

[16] B. Hou, Z. Cai, K. Wu, J. Su, and Y. Xiong, “6hit: A reinforcement learning-based approach totarget generation for internet-wide ipv6 scanning,” in *Proceedings of the IEEE Conference on Computer Communications*. Toronto, Canada: IEEE, 2021.

[17] 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.

[18] T. Yang, B. Hou, Z. Cai, K. Wu, T. Zhou, and C. Wang, “6graph: A graph-theoretic approach to address pattern mining for internet-wide ipv6 scanning,” *Computer Networks*, vol. 203, p. 108666, 2021.

[19] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *Proceedings of the IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.

[20] R. Barnes, R. Altmann, and D. Kerr, “Mapping the great void: Smarter scanning for ipv6,” 2012. [Online]. Available: http://www.caida.org/workshops/isma/1202/slides/aims1202_rbarnes.pdf

[21] J. Ullrich, P. Kieseberg, K. Krombholz, and E. Weippl, “On reconnaissance with ipv6: a pattern-based scanning approach,” in *Proceedings of the International Conference on Availability, Reliability and Security*. Toulouse, France: IEEE, 2015, pp. 186–192.

[22] T. Cui, G. Gou, and G. Xiong, “6gcvae: Gated convolutional variational autoencoder for ipv6 target generation,” in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Singapore: Springer, 2020, pp. 609–622.

[23] T. Cui, G. Xiong, G. Gou, J. Shi, and W. Xia, “6veclm: Language modeling in vector space for ipv6 target generation,” in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Ghent, Belgium: Springer, 2021.

[24] T. Cui, G. Gou, G. Xiong, C. Liu, P. Fu, and Z. Li, “6gan: Ipv6 multi-pattern target generation via generative adversarial nets with reinforcement learning,” in *Proceedings of the IEEE Conference on Computer Communications*. Toronto, Canada: IEEE, 2021.

[25] P. Foremski, D. Plonka, and A. Berger, “Entropy/ip: Uncovering structure in ipv6 addresses,” in *Proceedings of the Internet Measurement Conference*. Santa Monica, CA, USA: ACM, 2016, pp. 167–181.

[26] A. Murdock, F. Li, P. Bramsen, Z. Durumeric, and V. Paxson, “Target generation for internet-wide ipv6 scanning,” in *Proceedings of the Internet Measurement Conference*. London, UK: ACM, 2017, pp. 242–253.

[27] R. Hinden and S. Deering, “Ip version 6 addressing architecture,” Internet Requests for Comments, RFC Editor, RFC 4291, February 2006.

[28] F. Gont and T. Chown, “Network reconnaissance in ipv6 networks,” Internet Requests for Comments, RFC Editor, RFC 7707, March 2016.

[29] T. Narten, R. Draves, and S. Krishnan, “Privacy extensions for stateless address autoconfiguration in ipv6,” Internet Requests for Comments, RFC Editor, RFC 4941, September 2007.

[30] S. Thomson, T. Narten, and T. Jinmei, “Ipv6 stateless address autoconfiguration,” Internet Requests for Comments, RFC Editor, RFC 4862, September 2007.

[31] W. Hoeffding, “Probability inequalities for sums of bounded random variables,” *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, 1963.

[32] O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczyński, S. D. Strowes, L. Hendriks, and G. Carle, “Clusters in the expanse: Understanding and unbiasing ipv6 hitlists,” in *Proceedings of the Internet Measurement Conference*. Boston, MA, USA: ACM, 2018, pp. 364–378.

[33] C. Partridge and M. Allman, “Ethical considerations in network measurement papers,” *Communications of the ACM*, vol. 59, no. 10, pp. 58–64, 2016.