Computer Networks 203 (2022) 108666

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

6Graph: A graph-theoretic approach to address pattern mining for Internet-wide IPv6 scanning

Tao Yang [a], Bingnan Hou [a,*], Zhiping Cai [a,*], Kui Wu [b], Tongqing Zhou [a], Chengyu Wang [a]

[a] College of Computer, National University of Defense Technology, Changsha 410073, China
[b] Department of Computer Science, University of Victoria, Victoria V8W 3P6, Canada

ARTICLE INFO

Keywords:
Pattern mining
Internet-wide scanning
IPv6
Outlier detection

ABSTRACT

IPv6 target generation is critical in fast IPv6 scanning for Internet-wide surveys and cybersecurity analysis. However, existing techniques generally suffer from low hit rates because the targets are generated from inappropriate address patterns. To address the problem, we propose 6Graph, a graph-theoretic method for IPv6 address pattern mining. It first divides the IPv6 address space into different regions according to the structural information of a set of known addresses. Then, 6Graph maps the addresses of each region into undirected graphs and conducts the density-based graph cutting for address clustering to mine IPv6 address patterns and detect the misclassified addresses iteratively. Besides, we exploit the random IPv6 target generation based on Hamming distance without additional and complicated target selection. Experiments on 11 large-scale candidate datasets show that the address patterns of 6Graph have a higher seed density than the existing methods. Further results over real-world networks indicate that 6Graph can achieve 12.6%–35.8% hit rates on the candidate datasets, which is an 8.8%–275.0% improvement over the state-of-the-art methods in Internet-wide scanning.

1. Introduction

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, is widely implemented and rapidly adopted in recent years due to strong demand. In October 2020, nearly 35% of Google users accessed their services via IPv6 [2]. Meanwhile, the number of active IPv6 BGP entries in the routing table is also increasing rapidly [3]. The Internet is moving unavoidably towards IPv6.

Even with its advantages, IPv6 also raises new challenges for effective network management, especially efficient Internet-wide scanning, which is essential to investigate the Internet for ISPs and researchers [4]. For traditional IPv4 networks, asynchronous scanning tools like ZMap [5] and Masscan [6] have drastically enhanced our capability of Internet-wide network surveys, including topology discovery [7,8], IP address analysis [9,10], and geolocation [11]. Meanwhile, network device search engines like Shodan [12] and Censys [13] can acquire Internet-wide asset data for evaluating network security, discovering vulnerabilities, and tracking remediations [14]. Nevertheless, these tools are not effective when applied for IPv6-based Internet because of IPv6's vast address space. Intuitively, based on the current brute-force approach, it would take tens of millions of years to render comprehensive scans of the IPv6 address space. How to evaluate the network assets and the status quo of usage on IPv6-based networks is the urgent concern of ISPs [15,16]. Besides, the researcher needs an effective method to obtain worldwide network measurements for IPv6 [17].

To this end, IPv6 target generation methods are designed and utilized as a universal method for efficient IPv6 scanning. By characterizing and modeling the structure of seed address space with a set of known addresses (i.e., address pattern mining), it can generate candidate addresses that may have a higher probability to be active [18], so as to reducing the probing scope and significantly accelerating the scanning. Yet, existing IPv6 target generation techniques commonly face the misclassified address challenge during address pattern mining, even for the newly invented methods 6Hit [19] and 6Tree [20]. Technically, they both consider the IPv6 address as a 32-dimensional vector in the address space and then conduct the space partition sequentially to construct the "space tree", a.k.a., divisive hierarchical clustering (DHC). During the "space tree" spanning, the misclassified address challenge happens frequently and will mislead the subsequent space partition. As shown in the running example in Fig. 1, Nos.

* Corresponding authors.
E-mail addresses: yangtao97@nudt.edu.cn (T. Yang), houbingnan19@nudt.edu.cn (B. Hou), zpcai@nudt.edu.cn (Z. Cai), wkui@uvic.ca (K. Wu), zhoutongqing@nudt.edu.cn (T. Zhou), chengyu@nudt.edu.cn (C. Wang).

https://doi.org/10.1016/j.comnet.2021.108666
Received 14 May 2021; Received in revised form 29 November 2021; Accepted 29 November 2021
Available online 18 December 2021
1389-1286/© 2021 Published by Elsevier B.V.

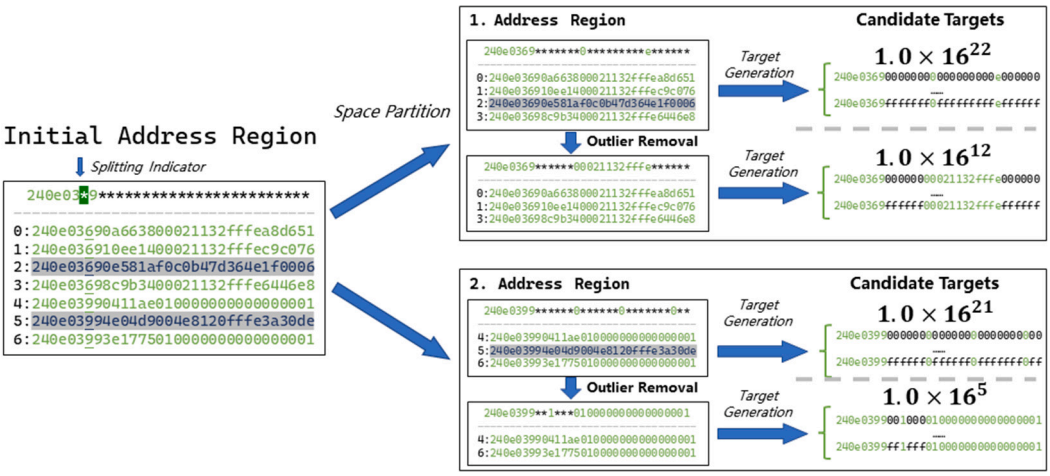


Fig. 1. Naive space partition cannot cope with the misclassified seeds (gray background) and cause the inflation of scanning scope, a.k.a., scanning space.

0,1,2,3 address vectors are classified into one subtree (No. 1 address region), while Nos. 4,5,6 address vectors belong to another one (No. 2 address region). However, the misclassified address vector (i.e., Nos. 2,5 address vectors) are divided into wrong address regions because they cause many unassigned dimensions (*) and such vast scanning scope, which results in the waste of scanning budgets and a low hit rate of about 11.5% [19]. In other words, if we could remove those misclassified addresses (outliers) in advance, the scanning scope will be reduced to a narrow range and the probing budgets will be used efficiently. However, automatic outlier detection in address regions is still an open problem.

We are thus motivated to propose 6Graph,[1] a graph theoretic IPv6 address pattern mining method that is integrated with the clustering for unsupervised outlier detection and the density-based graph cutting algorithm. In this work, 6Graph uses the sets of known IPv6 addresses, also called *seeds*, for address pattern mining and efficient target generation. Following the convention, 6Graph first considers IPv6 addresses as high-dimensional vectors in the address space and performs an enhanced divisive hierarchical clustering [21] on the given seeds for the partition of the address space. In detail, 6Graph adopts a Breadth-First Search strategy to improve efficiency. Then, 6Graph maps the IPv6 seeds of a region into an undirected graph without the edges after the space partition and recursively adds the current shortest edge when the two vertices are not reachable to each other and the density of the graph would increase. Meanwhile, those connected components of the graph including only one seed are not rare and their unique seeds are just the misclassified seeds, i.e., outliers. Finally, those outliers generated from the iterative graph cutting will be used as the seeds for the next round space partition. More details of IPv6 address pattern mining will be presented in Section 4.3.

To demonstrate the wide application scenarios of the address patterns, we employ 11 large-scale candidate datasets and exploit a simple targeted IP generation based on Hamming distance to evaluate the performance (i.e., hit rate). The results show that 6Graph outperforms the state-of-the-art methods in Internet-wide scanning.

The main contributions of the paper are as follows:

- We present 6Graph, a graph-theoretic IPv6 address pattern mining approach for IPv6 target generation. To the best of our knowledge, 6Graph is the first to apply the unsupervised outlier detection to IPv6 address pattern mining.
- Based on 6Graph's fine-grained address patterns, we adopt a simple-yet-effective distance-based target generation method to realize efficient IPv6 scanning.
- Real network experimental results on all the large-scale datasets show that 6Graph achieves 12.6%–27.7% hit rate, which is an 8.8%–275.0% improvement over the state-of-the-art methods in Internet-wide scanning.

2. Related work

2.1. IPv6 target generation

IPv6 target generation using seeds is first studied by Barnes et al. [22] in 2012. They assume 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 semantic information and the structural information in the seeds. Accordingly, related efforts can be classified into two categories: semantic information-based and structural information-based.

In the information-based methods, such as 6GCVAE [23], 6VecLM [24] and 6Gan [25], the seeds are firstly converted to 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. [26] 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. [27] 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. [20] propose 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. Hou et al. [19] propose

[1] Code is available at https://github.com/Lab-ANT/6Graph.git.

6Hit and first apply reinforcement learning approach to IPv6 active scanning. 6Hit dynamically allocates the budget according to the reward of the scanning on each region. Through feedback, 6Hit optimizes the subsequent search direction to high-density regions. However, the experiments show that the prior works can only achieve an 11.5% hit rate in large-scale scanning in the literature [19].

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 drawback of this type of method is the relatively low hit rate in large-scale scanning. To this end, this work is dedicated to pushing this limit and mining appropriate space partition (i.e., address patterns).

2.2. IPv6 address patterns

For such vast address space, the rule-based allocation of IPv6 active addresses is necessary. Related works illustrate that the IPv6 active addresses in the one domain follow the common IPv6 addressing schemes, i.e., address patterns [28].

An IPv6 address consists of a global routing prefix, a local subnet identifier, and an interface identifier (IID) [28]. While the global routing prefix and the local subnet identifier are usually assigned by the ISPs, the IID could be categorized into the following patterns [29]:

- **Embedded-IPv4.** The low bytes of the IID embeds the IPv4 address of the network interface. For instance, c0:a8:2:a or 192:168:2:10 could embed an IPv4 address 192.168.2.10.
- **Embedded-Port.** The low bytes of the IID embeds the service ports. For instance, 0:0:0:80 or 0:0:0:50 could embed decimal or hexadecimal port 80 for HTTP services.
- **IEEE-derived.** The IID embeds the Ethernet address and the inserted word "fffe" as so-called the Organizationally Unique Identifier [30] and then flips the U/L (7th) bit. For instance, 0250:56ff:fe89:49be could embed the MAC address 00:50:56:89:49:be.
- **Pattern-bytes.** The IIDs of a set of IPv6 have the common prefix or pattern and only differ on several nibbles. However, the different nibbles are not always in the low bytes. For instance, face:b00c:3333:1b26 and face:b00c:3333:2c45 have the common bytes,
  face:b00c:3333.
- **Randomized.** For users' privacy, the IID keeps a pseudorandom representation according to the privacy extensions for IPv6 SLAAC [30,31].

Besides, previous research [22] has shown that the allocation of both the IIDs, the global routing prefixes, and the local subnet identifiers follow certain rules (e.g., 2a02:6d40:30c4:7000::1 and 2a02:6d40:30c2:8000::1), and the seeds C can indicate the information on the use of addressing schemes. In other words, those address patterns are the key to IPv6 target generation. However, the prior methods can hardly figure out this problem.

3. Preliminaries

3.1. Definitions

We define some necessary metrics in advance.

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 dimension can be present as follows:

$$v[\delta] = (N \gg (128 - 4\delta))\ 0xf, \quad \delta = 1, 2, \ldots, 32 \tag{1}$$

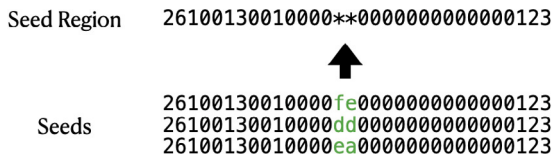


Fig. 2. A seed region example. Note that the $15th, 16th$ dimensions are the free dimensions and the others are the fixed dimensions.

$$distance\left(\begin{matrix}2a026d4034504200000000000000000001\\ 2a026d4030c47000000000000000000001\end{matrix}\right) = 5$$

$$distance\left(\begin{matrix}2a026d403*504*00000000000000000001\\ 2a026d4030c4700000000000000000001\end{matrix}\right) = 3$$

$$distance\left(\begin{matrix}2a026d403*504*00000000000000000001\\ 2a026d4030c**000000000000000000001\end{matrix}\right) = 1$$

Fig. 3. The examples of distances including IPv6 seeds and seed regions.

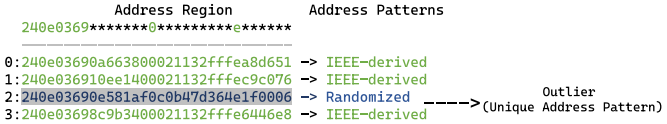


Fig. 4. An outlier example for a given address region.

Definition 2. Free dimensions and fixed dimensions. After IPv6 space partition, the generated regions consist of a set of seeds, which have the same values on the fixed dimensions and differ on the free dimensions. Following the convention, we use the wildcard symbol "*" to denote an uncertain nibble (i.e., free dimension) (see Fig. 2).

Definition 3. Scanning space. Obviously, the maximum number of generated IPv6 targets are based on ξ the size of given region's free dimensions and equal to 16^{ξ}.

Definition 4. Seed regions and address patterns. The seed regions are the subsets of the candidate datasets after the IPv6 space partition. Then, 6Graph mines the address patterns based on the graph cutting algorithm, and the address patterns should meet the characteristics described in Section 2.2 and have rare free dimensions to guarantee high-density scanning space.

Definition 5. Seed density. For a seed region or address pattern with ξ size of free dimensions and the seed set C, the corresponding density is

$$density = \begin{cases} \frac{C.size()}{\xi}, & C.size() > 1 \\ 0, & C.size() == 1 \end{cases} \tag{2}$$

Additionally, the region consisting of only a seed is considered as the outlier and its density is zero for calculation.

Definition 6. Distance. Without loss of generality, the distance can evaluate the similarity metric between regions, region and IPv6 address vector, or IPv6 address vectors. 6Graph adopts the Hamming distance [32] between the nybble-level representation of addresses and regions. And the distance between the wildcard "*" and any values is zero (see Fig. 3).

Definition 7. Outlier seeds. In a given address region, those seeds having unique address patterns will be regarded as the outliers (see Fig. 4).

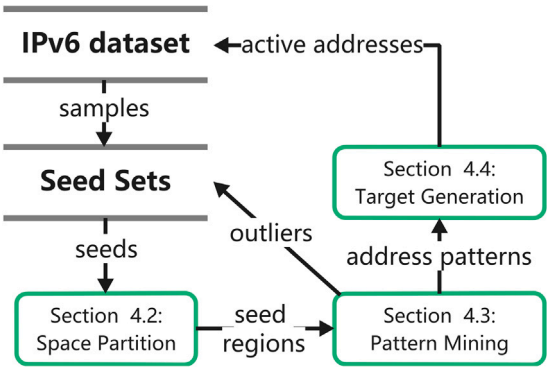


Fig. 5. The workflow of 6Graph.

3.2. Problem statement

An active IPv6 address should meet the following requirements, namely alive host, Internet access, and probe response. 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 the set of all active addresses under this protocol in the address space X is A. Seeds C is a set of known active IPv6 addresses in the address space X, which can usually be collected from network traffic, DNS records, and existing IPv6 scanning methods. Considering the target generation algorithm τ with the given probing budget b, this problem is stated as follows:

$$\begin{aligned} \max \quad & \tau(C, b) \\ s.t. \quad & \tau(C, b) \subset A, \quad C \subset X, \quad \tau(C, b) \cap C = \emptyset \end{aligned} \tag{3}$$

4. Design of 6Graph

We first present the system overview of 6Graph and then describe the details of its key technical components: space partition, graph-theoretic pattern mining, and distance-based target generation.

4.1. System overview

Fig. 5 illustrates the main workflow of 6Graph. It first samples the IPv6 datasets to generate multi-size seed sets. Then it conducts the space partition of the entire IPv6 address X on the seeds of candidate seed sets. 6Graph exploits the graph-theoretic algorithm to mine the address patterns from the seed regions while the misclassified seeds (outliers) in the seed regions will be removed. Meanwhile, some outliers will be attached to their zero-distance address patterns, i.e., seed rejoining, while others will be used as the new seed sets for the next round. Finally, efficient IPv6 target generation and scanning are conducted and based on the address patterns.

4.2. Space partition

The partition of the address space X in existing methods [19,20] is realized by recursively utilizing the divisive hierarchical clustering [21] algorithm to construct a tree structure namely "space tree", that means the state-of-the-art methods will recursively divide the entire seed set into some small clusters instead of clustering the single seeds into the seed regions. However, there are two disadvantages to these methods: (1) time-consuming space tree construction. Limited to the recursion and the tree structure, parallel space partition is hardly able to implement. (2) unnecessary storage. the existing methods adopt a Depth First Search strategy to recursively find the leaf nodes of the space tree. Meanwhile, the allocated regions of a child node only depend on its parent node, instead of its ancestor node. Accordingly, the preservation of the entire space tree will cause huge storage usage.

Algorithm 1 Space Partition

Require: the set of seeds C, the seed number threshold β
Ensure: address regions AS
1: $root = C, NQ = Queue(), AS = \emptyset$
2: $NQ.put(root)$
3: **while not** $NQ.empty()$ **do**
4: $cur = NQ.$get() ▷ FIFO Queue for Breadth-First Search.
5: **if** $cur.$size() $\leq \beta$ **then**
6: $AS.add(cur)$
7: **continue**
8: **end if**
9: $splits =$ LEFTMOST(cur) ▷ Split the current seeds on the leftmost free dimension.
10: **for** $s \in splits$ **do**
11: $NQ.put(s)$
12: **end for**▷ Split out subsequences of the assigned seeds on node where the seed vectors have the same value.
13: **end while**
14: **return** AS
15:
16: **function** LEFTMOST$(Seeds)$
17: **for** $\delta = 1 \rightarrow 32$ **do**
18: $clusters = List()$
19: **for** $s \in Seeds$ **do**
20: $clusters[s[\delta]].add(s)$
21: **end for**
22: **if** $len(clusters) > 1$ **then**
23: **break** ▷ There are different nibbles on the δ^{th} dimension (leftmost).
24: **end if**
25: **end for**
26: **return** $clusters$
27: **end function**

To this end, we exploit the Breadth-First Search strategy and a FIFO structure rather than the tree structure to generate target address regions. The pseudocode is shown in Algorithm 1. 6Graph first takes the node from the FIFO structure and the leftmost free dimension will be the splitting indicator. According to the values on the splitting indicators, the seeds of the current node will be associated with the corresponding child nodes which will be pushed in the FIFO structure.

Fig. 6 shows an example of building a space tree with a candidate dataset of seeds. The final tree contains 144 nodes in total, but we only draw some of the nodes for a clear illustration. The leaf nodes correspond to the seed regions defined in Definition 4 and the non-leaf nodes would be reserved in tree structures [19,20] while 6Graph exploits the FIFO structure and discards the interim nodes to reduce storage usage. Besides, the FIFO structure in 6Graph can be shared by multiple CPUs (i.e., resource pooling).

It is worth highlighting that the non-leaf nodes (seed regions) are not always perfect after the space partition. For example, Node 6 only includes one seed that should be added to the set of outlier seeds and those outlier seeds mislead the subsequent space partition which results in the vast scanning space (e.g., the first and last seeds in Node 120). The problem will be solved in Section 4.3.

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 dimension which will be not beyond $O(32\ m)$ time consumption totally and the worst-case time complexity is $O(mlogm)$. Besides, 6Graph 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 [19,20] need to keep the entire tree structure and their worst-case space complexity is $O(mlogm)$.

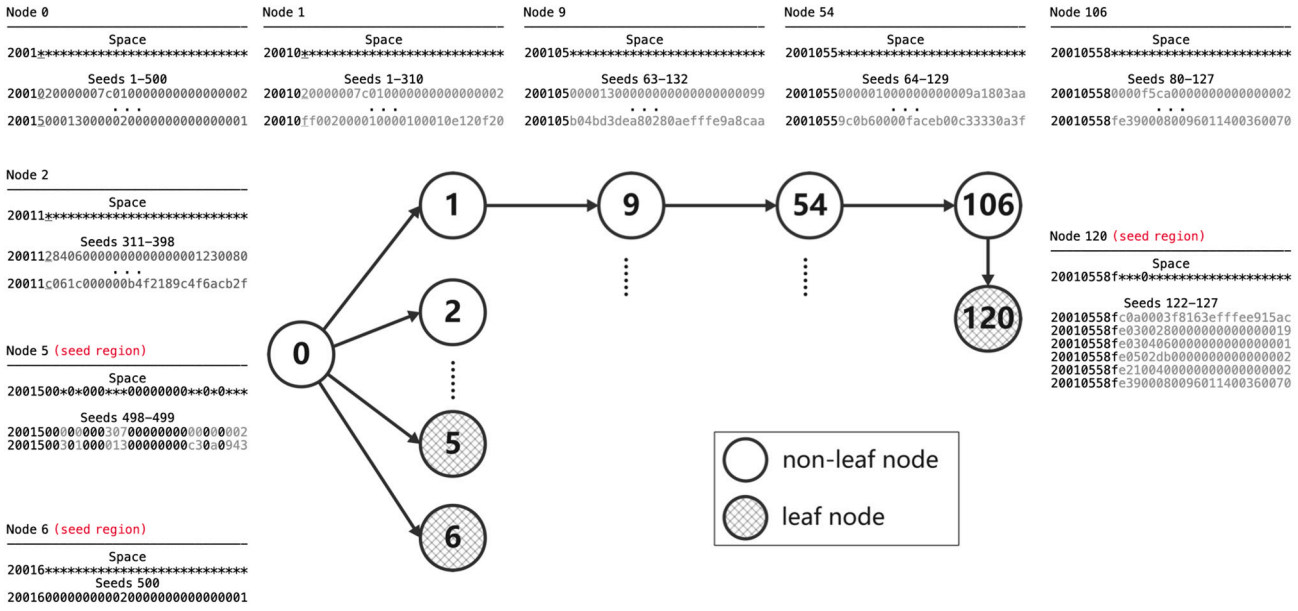


Fig. 6. The instance of space partition, which generates 144 nodes and 24 seed regions with 19 outlier seeds from 500 seeds.

4.3. Graph-theoretic pattern mining

After space partition, 6Graph will not allocate the scanning budgets directly according to the seed regions from inappropriate space partition which results in a low hit rate. As explained above, the problem of space partition is the interference of the outlier seeds. Thus, 6Graph adopts the graph-theoretic pattern mining algorithm to remove misclassified seeds and divide the seed regions into address patterns.

We restate the problem for clarity: the pattern mining aims to find the real IPv6 address patterns defined in Section 2.2 whose inputs are the inappropriate seed regions and outputs are the address patterns with the detected outlier seeds. The pseudocode of pattern mining is shown in Algorithm 2.

Given the seed region, 6Graph will evaluate the distances between all the seeds according to the Definition 6. It will sort the distances list and remove the duplicates and the overlong edges. Then, 6Graph constructs an undirected graph $G = (V, E)$ whose vertices are the seeds without edges. In the order of the sorted distance list, the edges will be added to the graph G when the related two vertices cannot be reachable for each one. To the end, the graph becomes connected, called the minimum spanning tree (MST).

Algorithm 2 Pattern Mining

Require: the given seed region X, the free dimension threshold α
Ensure: address patterns PS, outlier seeds OS

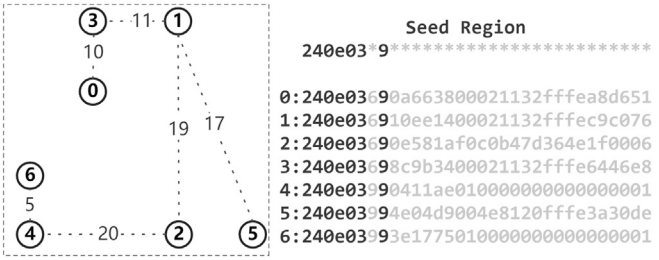
$PS = \emptyset, OS = \emptyset$
$dis = []$
for $\forall i, j \in X.seeds, i \neq j$ **do**
  if distance(i, j) $\leq \alpha$ **then**
    dis.append((distance(i, j), i, j))
  end if
end for
sort(dis)
G = Graph()
$G.V$.from($X.seeds$)
for $d, i, j \in dis$ **do**
  if i is reachable for j in G **then**
    continue
  end if
  $sg1 = G.ConnectedComponent(i)$
  $sg2 = G.ConnectedComponent(j)$
  $sgDensity1 =$ DENSITY($sg1$)
  $sgDensity2 =$ DENSITY($sg2$)
  $\rho =$ DENSITY($sg1 \cup sg2$)
  if $sgDensity1 < \rho$ **and** $sgDensity2 < \rho$ **then**
    $G.E$.add(i, j, length=d)
  end if
end for ▷ The minimum spanning tree clustering based on Kruskal algorithm.
for $CC \in G.ConnectedComponents()$ **do**
  if len($CC.seeds$)>1 **then**
    $PS.add(CC.seeds)$
  else
    $OS.add(CC.seeds)$
  end if
end for
return PS, OS

function DENSITY(G)
  if G.seeds.size()==1 **then**
    return 0
  else
    return $\frac{G.seeds.size()}{G.freedimensions.size()}$
  end if
end function

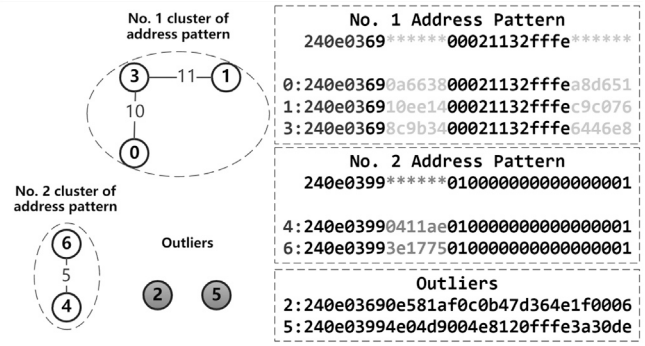
During the tree spanning above, 6Graph would evaluate whether those edges could be added to the graph. In short, graph-theoretic pattern mining aims to maximize the density of a vertex set where the vertices can reach each other, a.k.a, connected components.

For two vertices (i and j) of the current shortest edge, 6Graph calculates the density (see Section 3.1) of the vertices i, j corresponding connected components and their union cluster. Then, 6Graph recursively traverses all the candidate edges for the graph spanning and connects the vertices only when the density after the seed aggregation above increases.

After the graph spanning above, the entire undirected graph will be divided into several subgraphs according to its connected components, i.e., graph cutting. After graph cutting, there are some subgraphs including only single seeds which are the detected outliers because those seeds are far away from others (see Section 3.1) and a single seed without free dimension cannot represent an IPv6 address space



(a) The graph (left) and corresponding seeds (right) before cutting



(b) The graph (left) and corresponding seeds (right) after cutting

Fig. 7. The instance of density-based graph cutting for address pattern mining from the given seed region, where the edge distance is the number of free dimensions between two nodes.

for the new target generation. On the other hand, those non-isolated seed clusters in other subgraphs will be regarded as the IPv6 address patterns.

As shown in Fig. 7, the algorithm can dig out the real address patterns and detect two outliers from the given seed region. Note that the edge is the number of free dimensions between two nodes, i.e., the *distance* defined in Section 3.1. Obviously, the address pattern can meet the requirements in Section 2.2 and the entire scanning space of all of them reduce to 2.0×10^{-7} of that of the given seed region.

After pattern mining, the outliers will be used as the input for the new round of space partition.

Complexity analysis: Let n be the number of seeds in a given seed region. The algorithm first calculates the distances between all seeds and sorts the distances list, which has the worst-case time complexity of $O(n^2 log n)$. Then, it constructs a "minimum spanning tree" and the maximum number of edges is below $O(n)$ and time consumption is not beyond $O(n^2)$. Note that the number of seeds n of the given seed region is much smaller than the entire seeds' m. We assume that there are $\frac{m}{\overline{n}}$ regions from the entire set of m seeds whose average number of seeds is $\overline{n}$. The total pattern mining time complexity of the entire seeds is $O(m\overline{n}log\overline{n}), \overline{n} \ll m$.

Remark 1. The graph-theoretic pattern mining algorithm is an enhanced minimum spanning trees (MST) clustering, whose correctness has been declared in related literature [33,34]. Compared with the MST clustering where the dissimilarity threshold value needs to be defined apriori, 6Graph circumvents this disadvantage and generates the new cluster based on the density increase of seed region instead of empirical and weakly interpretable dissimilarity threshold. Similarly, K-means or DBSCAN clustering methods which require the pre-set hyperparameters are also not convincing and able to be adopted for outlier seed detection and pattern mining.

Fig. 8. The instance of distance-based target generation where the gray nibbles represent one Hamming distance.

4.4. Distance-based target generation

Benefiting from the more fine-grained address patterns than the existing methods and considering the efficiency of IPv6 target generation, we exploit a simple distance-based target generation algorithm.

The basic institution is that the distances between active IPv6 addresses should be tiny in the same address patterns. In other words, 6Graph only considers the targets which are only one Hamming distance from the given seeds.

As shown in Fig. 8, the seed 20010288220110000000000-000000068a with three free dimensions could generate about 45 IPv6 targets after filtering out the known seeds. Additionally, the set of known seeds includes not only the active seeds but also the known inactive IPv6 addresses from scanning results to avoid repeated scanning and budget waste. After the target generation, the candidate targets will be randomly sampled as targets for IPv6 scanning rather than all the candidates according to the allocated budgets of this pattern on this round. Considering the deepening of the seed space cognition of an address pattern, we dynamically increase the budget allocated for this pattern on the next round. Assuming that the number of known addresses including the active and the scanned inactive addresses on the pattern P_j in t_{th} iteration is k and the $C_j \subseteq C$ is the set of initial seeds of P_j, the number of allocated budgets in next iteration are equal to k_{t+1}:

$$k_{t+1} = \sum_{i=1}^{t} k_i, \quad k_1 = C_j.size() \tag{4}$$

We cannot deny that the target generation is rough and simple compared with the existing methods [19,20]. However, 6Graph focuses on address pattern mining and the high hit rate of IPv6 scanning is just the result of appropriate space partition. Thus, to avoid wrongly attributed to our pattern mining algorithm, we exploit the simple distance-based target generation method for fairness.

5. Performance evaluation

We compare the performance of 6Graph and other existing methods, including 6Hit [19], 6Tree [20], 6Gen [27] and Entropy/IP [26], with real-world test. Excluding 6Gan [25] from the baselines is mainly due to the unavailability of its source code.

5.1. Dataset description

Gasser et al. [35] have collected active IPv6 addresses from multiple sources, as shown in Table 1. We used the data published on Jan. 4th, 2021 as the sorted active address set C, including 4.9 M detected IPv6 addresses.

To investigate the impact of initial seed on the performance of different methods in a real-world test, we adopt three strategies to form seed sets from C: downsampling, biased sampling, and prefix-based

Table 1
Main sources of public active address set.

Main sources	Nature
Alexa top websites [36]	Servers
Statvoo websites [37]	Servers
Cisco Umbrella websites [38]	Servers
Zone files for several top-level domains [39]	Servers
CAIDA IPv6 DNS names data [40]	Servers
TLS certificates in certificate transparency logs	Servers
Rapid7 FDNS ANY dataset [41]	Servers
RIPE Atlas dataset [42]	Routers

Address range : 2001:200::1 – 2c0f:ffc8:4001:4::2
Total number of active addresses $\approx 4.9M$.

Table 2
Characteristics of seed sets.

Seed set	Number	Selection strategy	Covering	Range
C_1	$5K$	Down sampling	$4.9M$	2001:200:16a:24::102 ~ 2c0f:feb0:1:2::49d
C_2	$30K$	Down sampling	$4.9M$	2001:200:0:1cd1::13 ~ 2c0f:feb0:1:2::7f1
C_3	$0.1M$	Down sampling	$4.9M$	2001:200:0:1::3 ~ 2c0f:ff40:1:1012
C_4	$5K$	Biased sampling	$5K$	2001:200:0:1::1 ~ 2001:12ff:0:7000::6
C_5	$30K$	Biased sampling	$30K$	2001:200:0:1::1 ~ 2001:1348:1:7::6
C_6	$0.1M$	Biased sampling	$0.1M$	2001:200:0:1::1 ~ 2c0f:fef8:0:102f::2
C_7	$2.5M$	Biased sampling	$2.5M$	2001:200:0:1::1 ~ 240e:38b:86c4:6d01::
C_8	$2.5M$	Biased sampling	$2.5M$	240e:38b:86ff:1f01:: ~ 2c0f:ffc8:4001:4::2
C_9	$5K$	Prefix-based sampling	$4.9M$	2001:12f0:700:20::67 ~ 2c0f:fef8::e2e:1b
C_{10}	$30K$	Prefix-based sampling	$4.9M$	2001:1218:6009::1 ~ 2c0f:ff48::8
C_{11}	$0.1M$	Prefix-based sampling	$4.9M$	2001:1218:6009::1 ~ 2c0f:ff98::1

sampling. In downsampling, we randomly sample a certain number of addresses from C as 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 $C_i, i \in \{4,5,6\}$. And, we utilize the entire IPv6 dataset for evaluation and split it into two 2.5M seed sets $C_i, i \in \{7,8\}$. In prefix-based sampling, we extract 278K IPv6 /48 prefixes totally from the entire dataset and respectively sample the seeds of 2000, 12000, 40000 random prefixes as seed set $C_i, i \in \{9,10,11\}$. More details of these seed sets are shown in Table 2.

5.2. Real-world test results

We conducted the Internet-wide scanning experiments in January 2021 and limited the probing rate to 20 million bps for strictly ensuring Internet citizenship as suggested by Partridge and Allman [43]. The experiments including performance evaluation of 6Graph and existing methods were performed at the Hangzhou data center of Alibaba Cloud (AS37963). An IPv6 single-threaded address scanner using ICMPv6, TCP, and UDP was deployed on a Linux platform with an Intel Platinum 8260 Processor (2.50 GHz, 4 core) and 8 GB RAM.

In the real-world tests, we mainly adopt the scanner using ICMPv6 probes for performance evaluation of 6Graph and the baselines. Additionally, we also exploit the scanner using TCP/UDP probes to scan some common ports on the Internet (e.g., 22, 53, 80, and 443) shown in Table 6, because it is impossible to probe every port of an IPv6 address. More details of ICMPv6 and TCP/UDP scanning are shown in Section 5.3.2

To ensure fairness, we conducted the preprocessing step (i.e., pattern mining of 6Graph, 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 and 6Hit) require huge computing resources and the main challenge of IPv6 target generation is the hit rate of large-scale scanning. Even with the offline preparation, some methods still failed to cope with the large size seed sets within the reasonable time limit. Therefore, only 6Tree [20] and Entropy/IP [26] could be adopted as the baseline on the seed sets $C_i, i \in \{7,8\}$.

5.2.1. Size of scanning space

As shown in Fig. 12, 6Graph successfully reduced the entire scanning space by several orders of magnitude and could generate a larger number of fine-grained address patterns with high density, which can account for the high performance of 6Graph. Additionally, the experiments and related work [19] show that the numbers of address patterns (6Graph) on downsampling seed sets $C_i, i \in [1,3]$ and prefix-based sampling seed sets $C_i, i \in [9,11]$ are less than that on the same size biased sampling datasets $C_i, i \in [4,8]$, which means the patterns from downsampling datasets could be more effective.

5.2.2. Number of active addresses

On seed set $C_i, i \in [1,11]$ introduced above, we used the active address numbers detected by ICMPv6 probes to evaluate 6Graph and the baselines. The results are shown in Figs. 13 and 15 where we can see that the methods (i.e., 6Graph, 6Hit, and 6Tree) based on space partition are usually higher than others. It is worth highlighting that 6Graph outperforms the state-of-the-art methods on all the datasets, which demonstrates the powerful pattern mining ability of 6Graph.

5.2.3. Hit rate

Figs. 14 and 16 show the hit rate performance of 6Graph per iteration using ICMPv6 probes on seed sets. We can see that in general the hit rate of 6Graph is maintained at a higher level even though there are some fluctuations. The orange dashed lines indicate the final results of the 10M probes on the candidate datasets which achieve the highest average hit rate beyond 35% in large-scale scanning ($C_i, i \in \{1,2,3,9,10,11\}$) and also outperforms the existing methods on other datasets ($C_i, i \in [4,8]$) as shown in Table 4.

5.3. Address analysis

5.3.1. Seed biases and sampling strategy

Intuitively, 6Graph performs better on downsampling and prefix-based sampling seed sets than on the biased sampling seed sets. The underlying reason is that the downsampling and prefix-based sampling seed sets contain the address patterns from the entire 4.9M seeds, while the address patterns of the biased sampling seed sets are limited to a significantly narrower range (e.g., 5k). Thus, the generated targets of the poor "quality" seed sets (i.e., biased sampling) can only manage to provide the low-level hit rates stably, while targets of the high "quality" seed sets (i.e., downsampling and prefix-based sampling) can achieve outstanding performance and cause the hit-rate oscillation as shown in Figs. 14 and 16.

Furthermore, we state that the sizes ($> 5K$) of large seed sets have no significant impact on the hit rates because only two seeds are enough to determine a seed region. In other words, apart from a few omissions of the potential address patterns, the sampling ranges (covering) are more representative of the seed sets. Therefore, the performance indeed depends more on the covering range (sampling) rather than on the size of the seed set, which is consistent with the results.

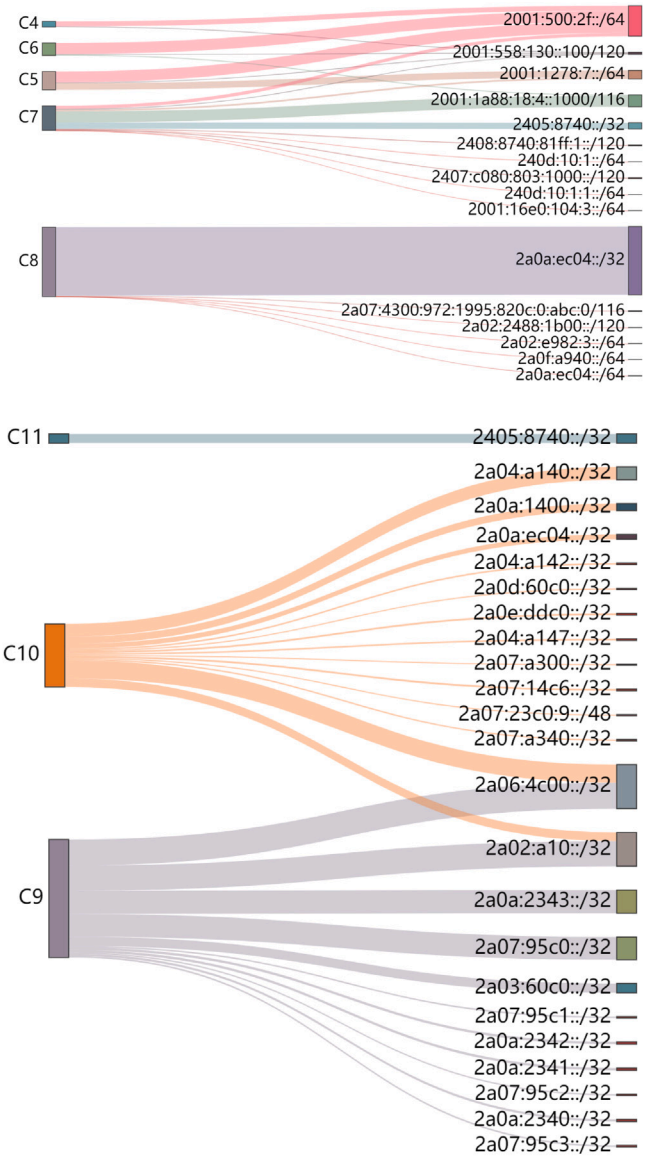


Fig. 9. The aliases of the discovered active addresses on the candidate datasets $C_i, i \in [1, 11]$ and their aliased prefixes (partly plotted).

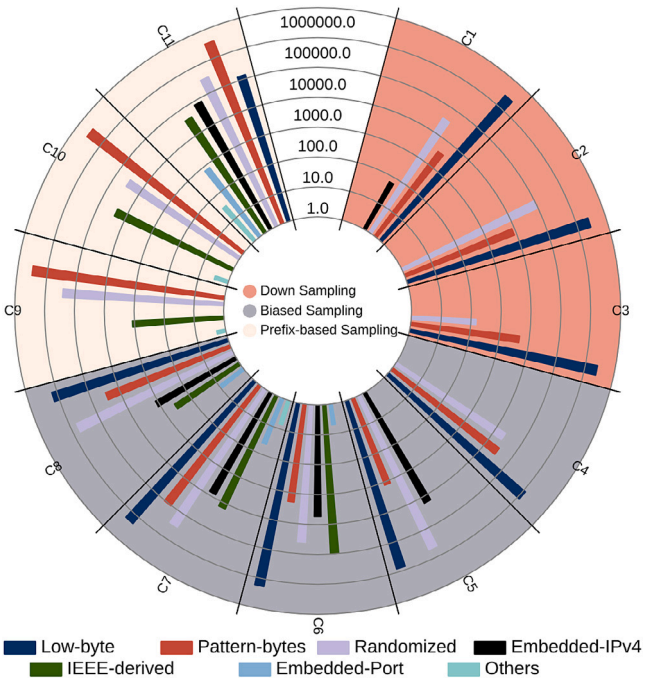


Fig. 10. The numbers of the discovered active addresses and their categories on the candidate datasets $C_i, i \in [1, 11]$.

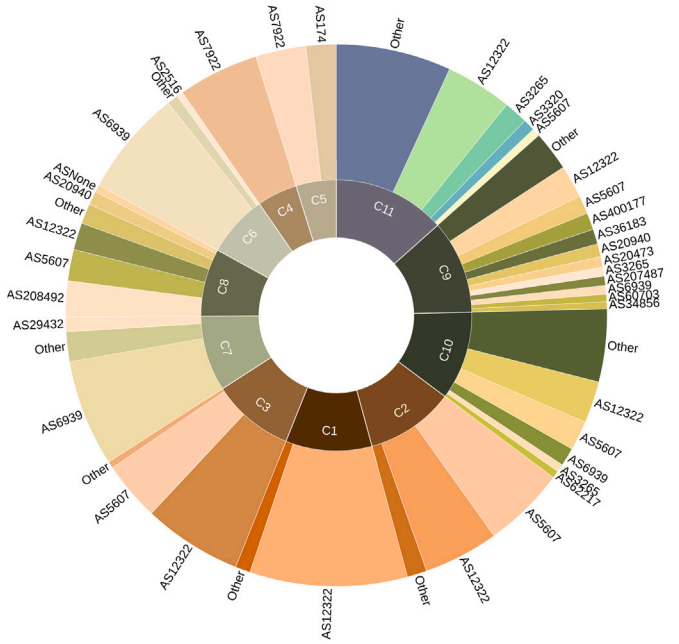


Fig. 11. The ASes of the discovered active addresses on candidate seed sets $C_i, i \in [1, 11]$ (partly plotted).

5.3.2. ICMPv6 vs TCP/UDP

The state-of-the-art methods [19,20] both exploit the ICMPv6 scanner for performance evaluation and achieved good results. However, there is still a technological gap in active IPv6 address exploration using other protocols. To this end, we experimentally leverage the ICMPv6 and TCP/

UDP probes to scan the targets generated by 6Graph. Given the same seeds and probing budgets, the 6Graph-based scanner using ICMPv6 probes could find far more active addresses than it using TCP/UDP probes. In other words, the employment of a TCP/UDP scanner might cause more false-negative results, which means the alive IPv6 hosts are regarded as inactive.

As shown in Table 6, the TCP probes on the 80,443 port could get the response than other TCP/UDP probes, because an IPv6 host will usually respond to an ICMPv6 probe but only respond to the TCP/UDP requests when it provides specific services on those ports (e.g., 80 for HTTP, 443 for HTTPS and 53 for DNS). Furthermore, we survey the response rates on the seed set $C_i, i \in [1, 11]$ using different protocols. As shown in Table 7, ICMPv6 probes have the highest response rates, which means more ICMPv6 active addresses will be fed back to 6Graph's next iteration and 6Graph can achieve better performance using ICMPv6 probes, i.e., protocol biases. Therefore, it takes a comprehensive method to render the status of IPv6 address space accurately.

5.3.3. Number of seeds

As mentioned above, the number of seed sets ($> 5K$) is negligible for the performance evaluation. However, how low the number of the seed sets can be used for target generation without performance loss is

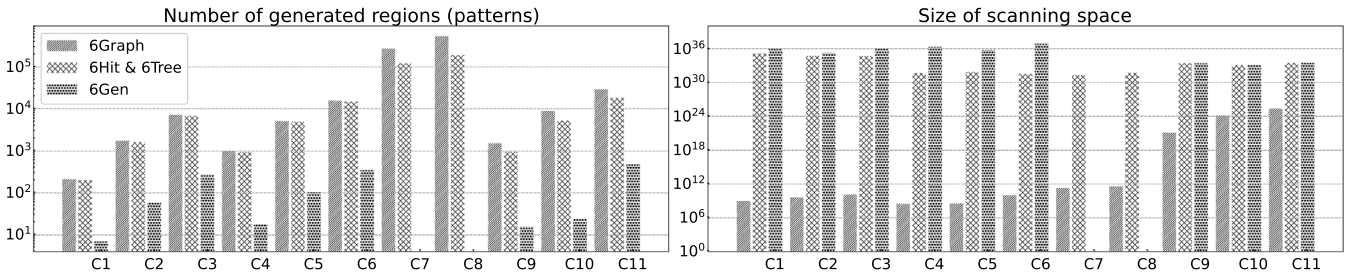


Fig. 12. The numbers of generated seed regions (patterns) and their entire scanning space (logarithmic scale).

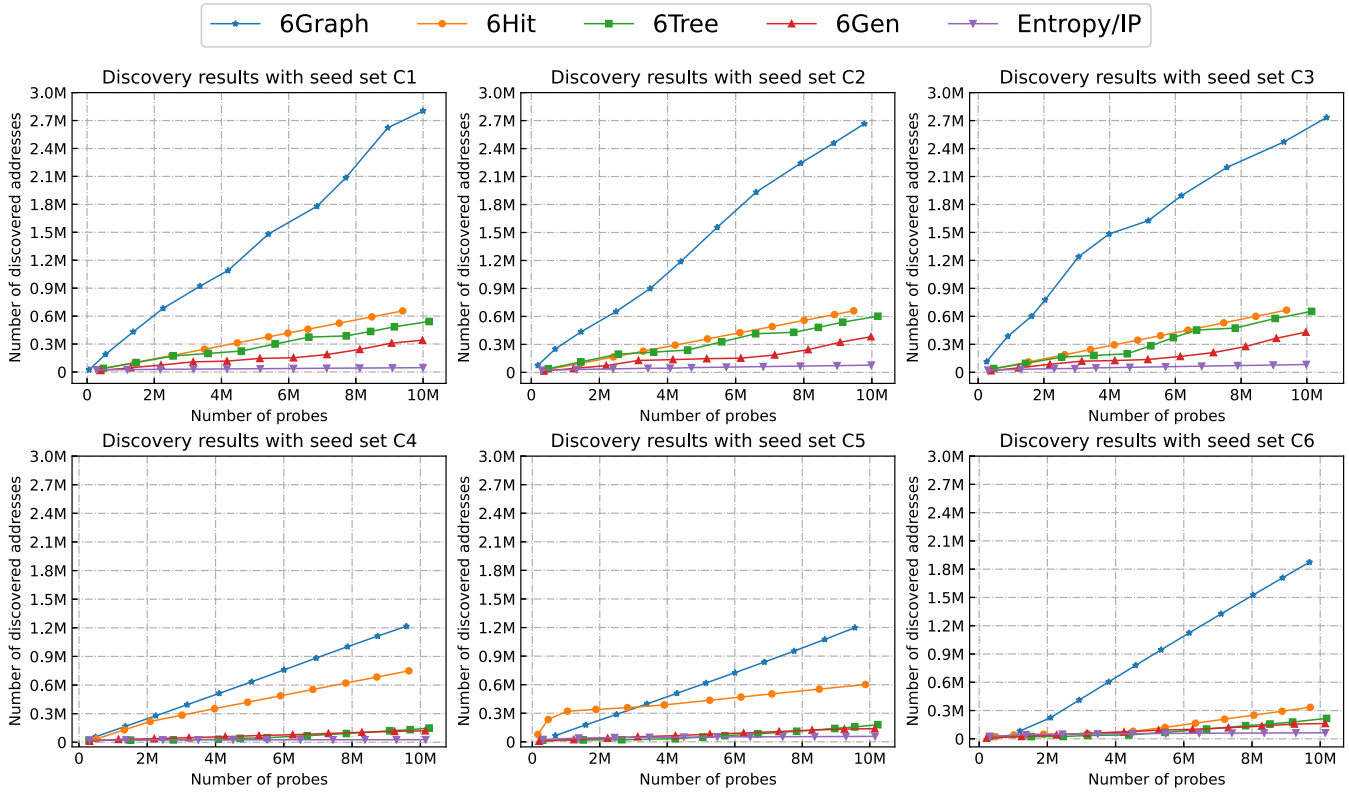


Fig. 13. Discovery of active addresses using ICMPv6 probes with a budget of 10M on seed sets $C_i, i \in [1, 6]$.

Table 3
The hit rates of 6Graph using ICMPv6 probes on different numbers of seed sets (Including aliases).

Seed Number	1K	2K	3K	4K	5K	30K	0.1M
Down Sampling	5.66%	9.69%	14.16%	19.30%	27.03%	27.73%	25.78%
Biased Sampling	2.07%	4.04%	6.11%	7.71%	12.66%	12.67%	19.47%
Prefix-based Sampling	6.97%	11.25%	18.87%	23.78%	32.67%	34.05%	36.94%

still an open problem. We additionally adopt 12 small-scale seed sets (i.e., 1K, 2K, 3K, and 4K) for 6Graph's performance evaluation based on downsampling, biased sampling, and prefix-based sampling strategies. As shown in Table 3, the decremental number of seeds causes 6Graph's hit rates to decrease and 6Graph's performance on the small seed sets (1K) has decayed by about 80%. Therefore, the lowest number of seed sets we exploit for tests is 5K.

5.3.4. Aliases

Prior works [16,27] propose that the alias is the non-negligible challenge for Internet-wide scanning and could result in the false positives of active addresses. Following the existing methods [19,20], 6Graph has removed the IPv6 aliases from results according to the known alias prefixes [35] but does not ensure that there will be no unknown aliases because IPv6 alias resolution is not within our work.

Furthermore, Table 5 and Fig. 9 demonstrate the discovered aliases of the active addresses and their aliased prefixes[2] on the candidate seed sets $C_i, i \in [1, 11]$. Note that there are a few aliases in the results, which illustrates that IPv6 aliases will not have a significant impact on the (hit rate) evaluation because Gasser et al. [35] have discarded the aliases when the IPv6 seeds were collected. Additionally, the aliases are concentrated in several aliased prefixes and only exist on the results of the seed sets $C_i, i \in [4, 11]$ because the adjacent seeds from biased sampling and prefix-based sampling usually have a more narrow address range (covering) and are more similar to each other. Therefore, 6Graph on biased sampling and prefix-based sampling seed sets could generate the address regions that are located in the aliased prefixes and generate the IPv6 targets with a high aliased rate. Even so, 6Graph still achieved the highest hit rates on all seed sets $C_i, i \in [1, 11]$ after the alias removal.

[2] Fully plotted at https://lab-ant.github.io/6GraphDataPlot/Aliases.html

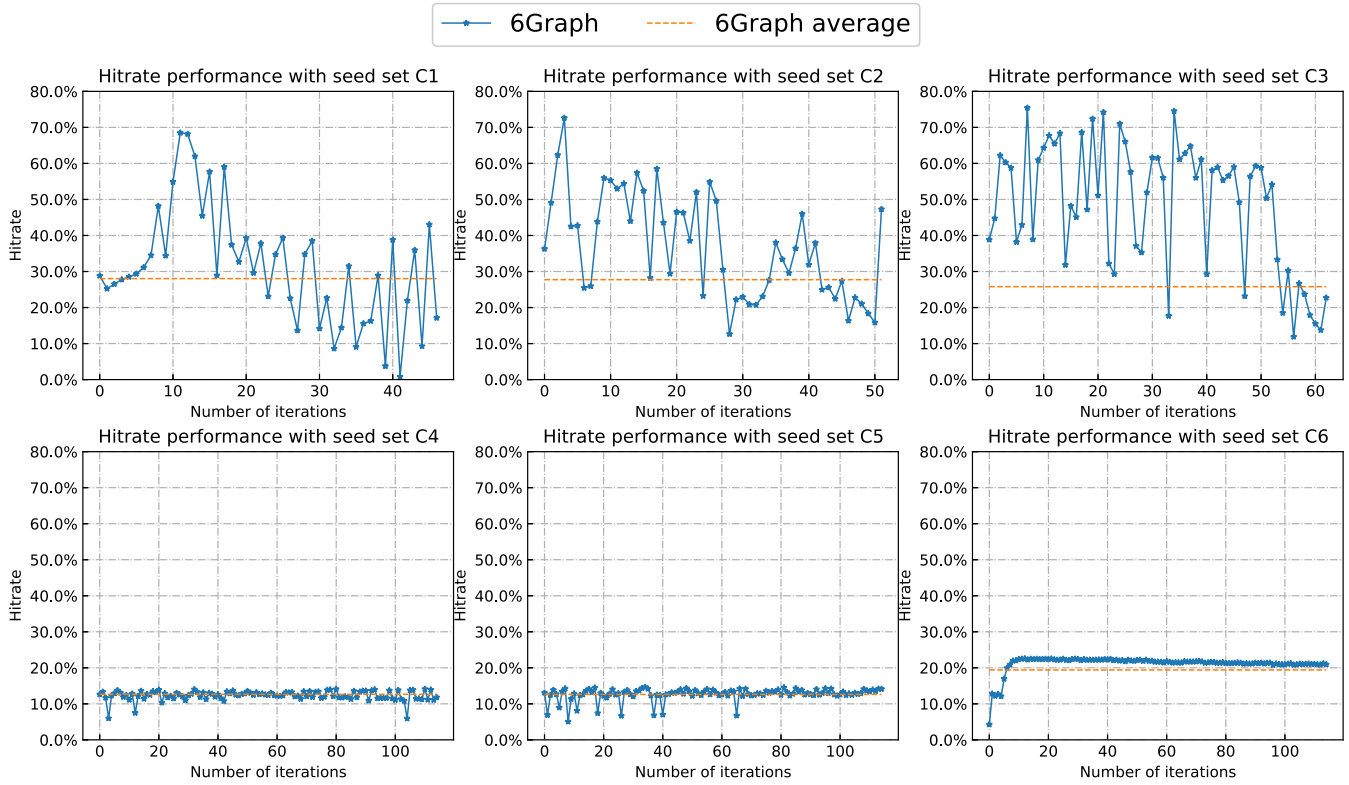


Fig. 14. The hit rate of 6Graph per iteration using ICMPv6 probes with a total budget of 10M. The orange dashed lines indicate the average hit rates on the given seed sets $C_i, i \in [1, 6]$.

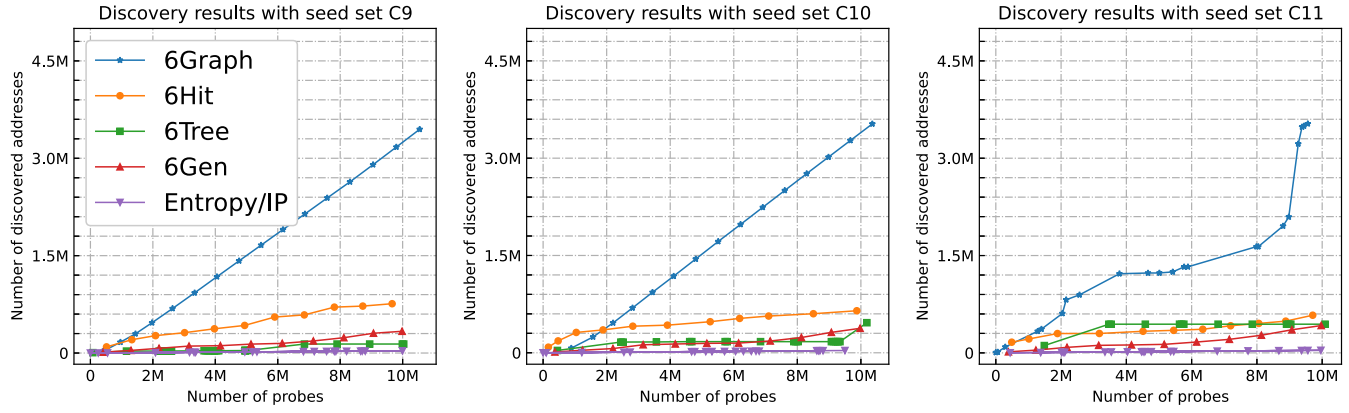


Fig. 15. Discovery of active addresses using ICMPv6 probes with a budget of 10M on seed sets $C_i, i \in [9, 11]$.

5.3.5. Categories

Fig. 10 shows the categories of discovered active addresses. The Low-byte, Pattern-bytes, and Randomized addresses are the main part of test results due to two factors: 1) Those address patterns are widely used for IPv6 address management on most domains. 2) Those address patterns compared with other address patterns, are easier to mine and explore. Besides, the EUI64 (IEEE-derived) addresses are relatively rare and the efficient discovery of IPv6 addresses with specific patterns (e.g., IEEE-derived) is still an open problem.

5.3.6. Ases

Fig. 11 demonstrates the autonomous systems (AS),[3] into which we map the discovered active addresses [44,45]. Intuitively, the diversity of autonomous systems is related to the size of the discovered IPv6 addresses with the same scanning budgets. For example, those results of the seed set $C_i, i \in [6, 11]$ with higher hit rates are from more ASes. As mentioned in Section 5.3.1, the wide sampling range of seeds is significant for target generation. Empirically, those seed sets $C_i, i \in [6, 11]$ including diverse ASes and prefixes would outperform the homogeneous seed sets (C_4, C_5), which is consistent with the test results.

In summary, large-scale scanning on the IPv6 Internet shows that 6Graph achieved 12.6%–35.0% hit rates for eight candidate datasets, which is an 8.8%–275.0% improvement over the state-of-the-art methods.

6. Conclusion

In this work, we proposed 6Graph, an efficient address pattern mining method for discovering active IPv6 addresses. 6Graph adopts

[3] Fully plotted at https://lab-ant.github.io/6GraphDataPlot/ASes.html

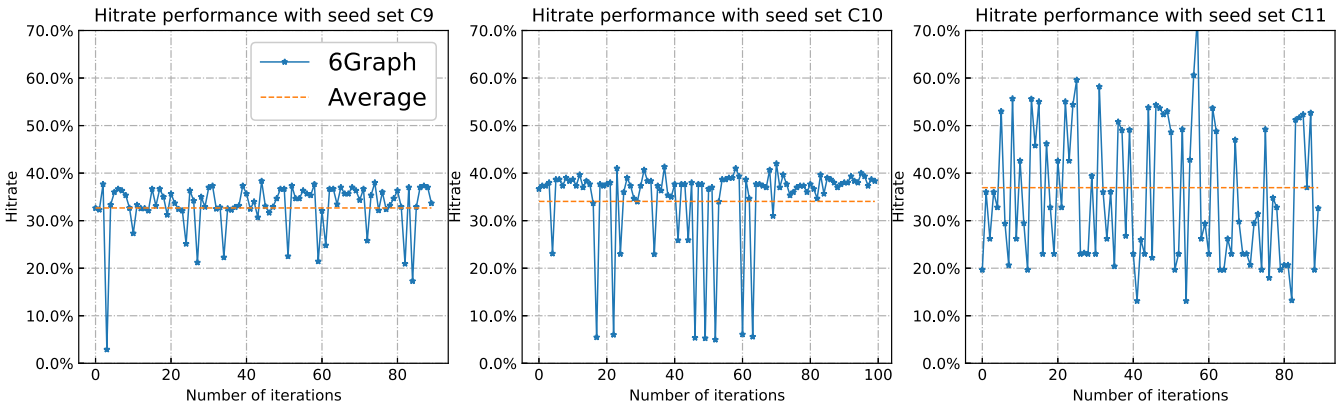


Fig. 16. The hit rate of 6Graph per iteration using ICMPv6 probes with a total budget of 10M. The orange dashed lines indicate the average hit rates on the given seed sets $C_i, i \in [9, 11]$.

Table 4
Average hit rates of target generation algorithm using ICMPv6 probes with 10M budgets (Aliases removed).

Approach	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}
Entropy/IP [26]	0.93%	1.56%	1.67%	0.54%	1.13%	1.27%	1.25%	6.57%	0.34%	0.36%	0.40%
6Gen [27]	5.13%	5.75%	6.48%	1.79%	2.03%	2.40%	–	–	3.35%	3.78%	4.22%
6Tree [20]	8.99%	9.88%	11.66%	2.18%	2.63%	3.19%	9.54%	9.49%	1.38%	4.55%	4.38%
6Hit [19]	10.47%	10.46%	12.66%	11.62%	9.14%	5.17%	–	–	7.86%	6.58%	5.98%
6Graph[a]	**27.03%**	**27.73%**	**25.78%**	**12.64%**	**12.61%**	**19.41%**	**23.63%**	**21.16%**	**29.31%**	**27.75%**	**35.84%**

[a]Our work.

Table 5
The discovered aliases of the active addresses using ICMPv6 probes.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}
Non-aliased	2.70M	2.77M	2.58M	1.26M	1.26M	1.94M	2.36M	2.12M	3.53M	3.45M	4.12M
Aliased	–	–	–	1.74K	5.61K	5.94K	7.34K	21.38K	0.36M	0.64M	0.12M
Aliased rate	–	–	–	0.13%	0.45%	0.31%	0.31%	0.99%	10.31%	18.52%	2.97%

Table 6
The hit rates of 6Graph using different protocols (Including aliases).

Protocols (Ports)	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}
ICMPv6	27.03%	27.73%	25.78%	12.66%	12.67%	19.47%	23.70%	21.37%	32.67%	34.05%	36.94%
TCP(21)	0.33%	0.69%	< 0.01%	< 0.01%	0.04%	0.05%	0.16%	0.30%	0.14%	0.09%	0.79%
TCP(22)	1.46%	0.74%	< 0.01%	1.99%	2.29%	6.16%	3.15%	0.11%	1.47%	1.28%	3.30%
TCP(80,443)	4.04%	5.34%	2.22%	0.02%	0.27%	7.18%	3.64%	2.02%	2.47%	1.66%	4.35%
UDP(25)	0	< 0.01%	0	0	< 0.01%	< 0.01%	< 0.01%	0	0	< 0.01%	0
UDP(53)	0.05%	0.26%	0.10%	< 0.01%	0.02%	5.30%	2.54%	0.13%	1.03%	0.57%	1.56%
ALL\ICMPv6[a]	3.28%	2.96%	1.43%	0.06%	0.40%	5.65%	5.46%	1.55%	2.80%	1.48%	3.24%

[a]TCP(20) ∪ TCP(22) ∪ TCP(80,443) ∪ UDP(25) ∪ UDP(53) \ ICMPv6.

Table 7
The response rates on seed sets $C_i, i \in [1, 11]$ using different protocols.

Protocols (Ports)	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}
ICMPv6	72.72%	73.26%	73.11%	89.04%	84.50%	74.08%	65.26%	74.98%	63.54%	64.08%	62.40%
TCP(21)	14.26%	14.37%	13.67%	0	0.13%	1.04%	5.17%	20.58%	2.02%	2.18%	2.13%
TCP(22)	18.98%	19.48%	19.60%	0.48%	3.01%	9.70%	17.42%	16.63%	6.64%	7.71%	7.82%
TCP(80,443)	28.98%	30.18%	30.00%	0	0.36%	15.86%	19.47%	36.99%	10.45%	10.85%	10.57%
UDP(25)	0	0	0	0	0	0	< 0.01%	0	0	< 0.01%	0
UDP(53)	4.46%	4.67%	4.53%	0	0.14%	5.16%	4.05%	4.19%	3.88%	3.43%	3.52%

a novel graph-theoretic pattern mining algorithm to automatically find the address patterns and filter out the "outlier" seeds, which results in fine-grained and reasonable space partition and effectively reduces the size of the scanning space. This unique feature solves the "black sheep" (outlier seeds) problem and low hit rate challenges caused by the space partition based on only one single split indicator. With the real-world test, 6Graph has achieved much better performance on hit rate than the state-of-the-art solutions, 6Hit and 6Tree, 6Gen, and Entropy/IP.

CRediT authorship contribution statement

Tao Yang: Methodology, Formal analysis, Writing – original draft. **Bingnan Hou:** Methodology, Writing – review & editing. **Zhiping Cai:** Conceptualization, Resources, Writing – review & editing, Funding acquisition. **Kui Wu:** Writing – review & editing. **Tongqing Zhou:** Writing – review & editing. **Chengyu Wang:** Writing – review & editing.

Declaration of competing interest

One or more of the authors of this paper have disclosed potential or pertinent conflicts of interest, which may include receipt of payment, either direct or indirect, institutional support, or association with an entity in the biomedical field which may be perceived to have potential conflict of interest with this work. For full disclosure statements refer to https://doi.org/10.1016/j.comnet.2021.108666. 1. Tianyu Cui, cuitianyu@iie.ac.cn, School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China. 2. Gaopeng Gou, guogaopeng@iie.ac.cn, School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China.

Acknowledgment

This work is supported by the National Key Research and Development Program of China (2018YFB1800202).

References

[1] I. Livadariu, K. Benson, A. Elmokashfi, A. Dhamdhere, A. Dainotti, Inferring carrier-grade NAT deployment in the wild, in: Proc. INFOCOM 2018, IEEE, USA, 2018, pp. 2249–2257.
[2] Google, Ipv6 adoption statistics, 2021, URL https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption.
[3] G. Huston, Ipv6 BGP table reports, 2021, URL https://bgp.potaroo.net/index-v6.html.
[4] B. Hou, C. Hou, T. Zhou, Z. Cai, F. Liu, Detection and characterization of network anomalies in large-scale RTT time series, IEEE Trans. Netw. Serv. Manag. 18 (1) (2021) 793–806, http://dx.doi.org/10.1109/TNSM.2021.3050495.
[5] Z. Durumeric, E. Wustrow, J.A. Halderman, Zmap: Fast internet-wide scanning and its security applications, in: Proc. {USENIX} Security 2013, USENIX Association, USA, 2013, pp. 605–620.
[6] D. Myers, E. Foo, K. Radke, Internet-wide scanning taxonomy and framework, in: Proc. AISC 2015, Vol. 27, Australian Computer Society, Inc, Australia, 2015, p. 30.
[7] R. Beverly, R. Durairajan, D. Plonka, J.P. Rohrer, In the IP of the beholder: Strategies for active IPv6 topology discovery, in: Proc. IMC 2018, ACM, USA, 2018, pp. 308–321.
[8] R. Beverly, Yarrp'ing the internet: Randomized high-speed active topology discovery, in: Proc. IMC 2016, ACM, USA, 2016, pp. 413–420.
[9] 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: Proc. NDSS 2016, The Internet Society, USA, 2016.
[10] D. Plonka, A. Berger, Kip: A measured approach to IPv6 address anonymization, 2017, arXiv preprint arXiv:1707.03900.
[11] Q. Scheitle, O. Gasser, P. Sattler, G. Carle, HLOC: Hints-based geolocation leveraging multiple measurement frameworks, in: Proc. TMA 2017, IEEE, Ireland, 2017, pp. 1–9.
[12] J. Matherly, Shodan: The search engine for internet-connected devices, 2021, URL https://www.shodan.io.
[13] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, J.A. Halderman, A search engine backed by internet-wide scanning, in: Proc. CCS 2015, ACM, USA, 2015, pp. 542–553.
[14] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey, et al., The matter of heartbleed, in: Proc. IMC 2014, ACM, Canada, 2014, pp. 475–488.
[15] K. Borgolte, S. Hao, T. Fiebig, G. Vigna, Enumerating active IPv6 hosts for large-scale security scans via DNSSEC-signed reverse zones, in: Proc. SP 2018, IEEE, USA, 2018, pp. 770–784.
[16] Y. Tao, G. Hu, B. Hou, Z. Cai, J. Xia, C.C. Fong, An alias resolution method based on delay sequence analysis, CMC-Comput. Mater. Continua 63 (3) (2020) 1433–1443.
[17] A.S.A.M.S. Ahmed, R. Hassan, N.E. Othman, Ipv6 neighbor discovery protocol specifications, threats and countermeasures: a survey, IEEE Access 5 (2017) 18187–18210.
[18] G. Song, L. He, Z. Wang, J. Yang, T. Jin, J. Liu, G. Li, Towards the construction of global IPv6 hitlist and efficient probing of IPv6 address space, in: Proc. IWQoS 2020, IEEE, Hangzhou, China, 2020, pp. 1–10.
[19] B. Hou, Z. Cai, K. Wu, J. Su, Y. Xiong, 6Hit: A reinforcement learning-based approach totarget generation for internet-wide IPv6 scanning, in: Proc. INFOCOM 2021, IEEE, Canada, 2021.
[20] Z. Liu, Y. Xiong, X. Liu, W. Xie, P. Zhu, 6Tree: Efficient dynamic discovery of active addresses in the IPv6 address space, Comput. Netw. 155 (2019) 31–46.
[21] A. Guénoche, P. Hansen, B. Jaumard, Efficient algorithms for divisive hierarchical clustering with the diameter criterion, J. Classification 8 (1) (1991) 5–30.
[22] R. Barnes, R. Altmann, D. Kerr, Mapping the great void: Smarter scanning for IPv6, 2012, URL http://www.caida.org/workshops/isma/1202/slides/aims1202_rbarnes.pdf.
[23] T. Cui, G. Gou, G. Xiong, 6GCVAE: Gated convolutional variational autoencoder for IPv6 target generation, in: Proc. PAKDD 2020, Springer, Singapore, 2020, pp. 609–622.
[24] T. Cui, G. Xiong, G. Gou, J. Shi, W. Xia, 6VecLM: LAnguage modeling in vector space for IPv6 target generation, in: Proc. ECML PKDD 2020, Springer, Belgium, 2021.
[25] T. Cui, G. Gou, G. Xiong, C. Liu, P. Fu, Z. Li, 6GAN: IPv6 multi-pattern target generation via generative adversarial nets with reinforcement learning, in: Proc. INFOCOM 2021, IEEE, Canada, 2021.
[26] P. Foremski, D. Plonka, A. Berger, Entropy/ip: Uncovering structure in ipv6 addresses, in: Proc. IMC 2016, ACM, USA, 2016, pp. 167–181.
[27] A. Murdock, F. Li, P. Bramsen, Z. Durumeric, V. Paxson, Target generation for internet-wide ipv6 scanning, in: Proc. IMC 2017, ACM, UK, 2017, pp. 242–253.
[28] R. Hinden, S. Deering, Ip version 6 addressing architecture, RFC 4291, RFC Editor, 2006, URL http://www.rfc-editor.org/rfc/rfc4291.txt.
[29] F. Gont, T. Chown, Network reconnaissance in IPv6 networks, RFC 7707, RFC Editor, 2016.
[30] S. Thomson, T. Narten, T. Jinmei, IPv6 stateless address autoconfiguration, RFC 4862, RFC Editor, 2007, URL http://www.rfc-editor.org/rfc/4862.txt.
[31] T. Narten, R. Draves, S. Krishnan, Privacy extensions for stateless address autoconfiguration in IPv6, RFC 4941, RFC Editor, 2007.
[32] R.W. Hamming, Error detecting and error correcting codes, Bell Syst. Tech. J. 29 (2) (1950) 147–160.
[33] J.C. Gower, G.J. Ross, Minimum spanning trees and single linkage cluster analysis, J. R. Stat. Soc. Ser. C. Appl. Stat. 18 (1) (1969) 54–64.
[34] C. Zhong, D. Miao, R. Wang, A graph-theoretical clustering method based on two rounds of minimum spanning trees, Pattern Recognit. 43 (3) (2010) 752–766.
[35] O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczyński, S.D. Strowes, L. Hendriks, G. Carle, Clusters in the expanse: Understanding and unbiasing IPv6 hitlists, in: Proc. IMC 2018, ACM, Boston, MA, USA, 2018, pp. 364–378.
[36] Alexa, The top sites on the web, 2020, URL https://www.alexa.com/topsites.
[37] Statvoo, Website analytics and reviews, 2020, URL https://statvoo.com.
[38] Cisco, Cisco umbrella, 2020, URL https://umbrella.cisco.com.
[39] PremiumDrops, Domain zone file and zone changes downloads, 2020, URL https://www.premiumdrops.com/zones.html.
[40] CAIDA, Ipv6 topology dataset, 2018, URL http://www.caida.org/data/active/ipv6allpreftopologydataset.xml.
[41] Rapid7, Forward DNS (FDNS), 2020, URL https://opendata.rapid7.com/sonar.fdns_v2/.
[42] RIPE, RIPE NCC atlas dataset, 2020, URL https://atlas.ripe.net.
[43] C. Partridge, M. Allman, Ethical considerations in network measurement papers, Commun. ACM 59 (10) (2016) 58–64.
[44] CAIDA, Routeviews Prefix to as mappings Dataset, URL https://www.caida.org/data/routing/routeviews-prefix2as.xml.
[45] H. Asghari, A. Noroozian, PyASN, URL https://github.com/hadiasghari/pyasn.

Tao Yang is a graduate student in Computer Science, School of Computer, National University of Defense Technology (NUDT) at Changsha, China. The objectives of his research are (i) to identify network traffic camouflage by AI, (ii) to detection Large-scale active IPv6 address and (iii) to explore AI-based IPv6 alias address resolution. He has degree B.Sc. from National University of Defense Technology in Network Engineering.

Zhiping Cai is a full Professor in Networking Engineering Department, School of Computer, National University of Defense Technology (NUDT) at Changsha, China. His doctoral dissertation has been rewarded with the Outstanding Dissertation Award of the Chinese PLA. Zhiping Cai's current research interests include Big Data, Network Security and Network Virtualization. He has served as a referee of paper review for the ToN, TPDS, ComNet. He serves on many conference program committees such as EUV, NAS.