DualDNSMiner: A Dual-Stack Resolver Discovery Method Based on Alias Resolution

Dingkang Han[1,2,3], Yujia Zhu[1,2,3], Liang Jiao[1,2,3], Dikai Mo[1,2,3], Yong Sun[1,2(✉)], Yuedong Zhang[4], and Qingyun Liu[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{handingkang,zhuyujia,modikai,sunyong,liuqingyun}@iie.ac.cn, jiaol@cert.org.cn

[2] National Engineering Laboratory of Information Security Technologies, Beijing, China

[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

[4] CNCERT, Beijing, China
zyd@cert.org.cn

Abstract. With the rapid development of IPv6 network applications, the transition to IPv6 dns has accelerated. In this process, dual-stack resolvers take on the crucial role that ensures the resolution of domains under hybrid network conditions. However, the lagging deployment of IPv6 defence measures may undermine the overall security of resolvers, making the discovery of dual-stack resolvers vital for DNS security analysis. Previous methods for discovering dual-stack resolvers are built on strong but impractical assumptions, ignoring resolvers with multiple alias IP addresses. In this article, we propose a new dual-stack resolvers discovery model based on alias resolution - DualDNSMiner. DualDNSMiner involves address alias resolution technology in order to recognize hosts with multiple alias addresses and identify dual-stack resolvers. Large-scale measurement experiments show that, DualDNSMiner can reliably discover over 80% more new dual-stack resolvers compared to previous judgment rules. In addition, we put forth a novel approach to validate the accuracy of our findings. The results demonstrate that the precision of DualDNSMiner can exceed over 90%. Finally, the results of DualDNSMiner provide the first proof of the widespread use of alias addresses in DNS resolvers, which is crucial for analyzing the process of DNS's IPv6 evolution.

Keywords: DNS · IPv6 · dual-stack resolver · alias resolution

1 Introduction

With the rapid development of IPv6 networks, the adoption rate of IPv6 applications has been increasing year by year [2,3]. More and more users are now choosing to access websites and applications through IPv6 networks [14]. This

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 563, pp. 347–364, 2024.
https://doi.org/10.1007/978-3-031-54531-3_19

has resulted in a significant increase in the demand for IPv6 domain name resolution. This implies that, for a foreseeable amount of time, with the continued existence of IPv4 network applications, both IPv4 and IPv6 will have strong domain name resolution demands.

As the actual executor of domain name resolution, the Domain Name System (DNS) plays an important role in the transition from IPv4 to IPv6. DNS mainly consists of three components [23], namely: the client that sends requests, the authoritative name server that holds domain name records, and the resolver, which is the most important component in DNS resolution. The resolver can be divided into the forwarder, which forwards DNS requests, and the recursive resolver, which asks authoritative name servers directly. Due to the vital role played by recursive resolvers, various modifications have been proposed for recursive resolvers to handle the IPv4/IPv6 transition [4,5]. One of the proposed solutions is to modify existing IPv4 resolvers, so that they can also perform IPv6 resolution, which is referred to as **dual-stack resolvers** [12].

However, the interdependence between IPv4 and IPv6 is prone to unknown security issues. The lagging deployment of firewall, filtering, and intrusion detection systems for IPv6 provides an alternative path for application layer attacks [17]. Dual-stack resolvers could be at serious security risk from DDoS attacks because IPv6 protocol requires larger packets than IPv4 [16]. Whether an attack against the IPv6 address of a DNS server impacts an organization's corresponding service for IPv4 depends on whether it is dual-stacked. Therefore, discovering the deployment of dual-stack resolvers is a highly meaningful task for both network operators and researchers. By having a comprehensive understanding of the status of dual-stack resolver deployments, network security professionals can recognize potential risks.

To discover dual-stack resolvers, some previous work [1,7] based on the assumption that *dual-stack resolvers have a pair of* $\{IPv4, IPv6\}$ *addresses* to discover dual-stack resolvers from the relationship between IPv4 and IPv6 resolvers. Obviously, this assumption holds true if there is only one IPv4 or IPv6 address pointing to the host. However, because some hosts have multiple network interfaces, there can be multiple different IPv4 or IPv6 addresses pointing to the same host. These addresses are called *alias addresses*. Some work has shown that alias addresses are abundant in networks, especially IPv6 addresses [20,24]. This means that some resolvers could have multiple pairs of $\{IPv4, IPv6\}$ addresses at the same time, but will never be discovered by previous methods. This creates a challenge for the discovery of dual-stack resolvers.

In this paper, we focus on recursive resolvers and propose a new dual-stack resolver discovery algorithm, DualDNSMiner. Unlike previous work, DualDNSMiner will map multiple IPv4 (IPv6) alias addresses to the same IPv4 (IPv6) host by means of alias resolution. This makes the address relationship will be transformed into a host relationship. Thus, the criterion for a dual-stack resolver in DualDNSMiner is that **the resolver with a pair of** $\{IPv4, IPv6\}$ **hosts** is dual-stacked. Our work makes three main contributions:

1 This is the first application of alias resolution method to solve the problem of dual-stack resolver discovery. The proposed discovery algorithm can

increase the resolver discovery rate by **more than 80%** compared to previous methods with relatively low cost. It helps to advance the analysis of DNS deployments during the IPv6 transition process.
2 We present a novel method for verifying whether a resolver is dual-stacked. In comparison to previous works that relied on verifying through IPv4&IPv6 addresses pattern similarity, our method offers a more reliable and data-supported approach for verification. This will facilitate the validation of the effectiveness of future works for the dual-stack resolver discovery.
3 Through our global measurements from two vantage points in different countries, our data suggests that close to 50% of known dual-stack resolvers have alias addresses. Owning more addresses indicates that dual-stack resolvers have more selectable data paths than previously thought, which increases the risk of being vulnerable to application layer attacks.

In Sect. 2, we will introduce some related works on the dual-stack resolver and alias address. In Sect. 3, we will present our innovative method, and in Sect. 4, we will compare the performance of our method and previous methods in practical detection. Section 5 & 6 provide our future work outlook and conclusion, respectively.

2 Related Work

2.1 Dual-Stack Resolver Discovery

To our knowledge, we are first one to achieve targeted discovery for dual-stack resolvers by leveraging their characteristics. However, some studies have observed the existence of associations between IPv4-IPv6 addresses of resolvers through CNAME redirection [7]. Al-Dalky *et al.* [1] referred to these addresses that have mutual association as a resolver collaborative pool and regarded pools that have only one IPv4 and one IPv6 address as dual-stack resolvers. They named this pool behavior as multi-port behavior and interpreted it as either a single multi-port machine or load balancing across resolver fields.

Some works have also made their own contributions to the discovery of dual-stack machines. Geoff *et al.* have tested 45 million clients by deploying dual-stack authoritative name servers [19]. The results show that 18% of clients use resolvers that can handle both IPv4 and IPv6 requests. Beverly *et al.* [8] focus on using TCP options and timestamps to identify IPv4 and IPv6 dual-stack machines, but this method have limitations when applied to DNS recursive resolvers. This techniques require TCP, which is a backup transmission protocol for DNS and not all TCP implementations support TCP timestamp options. Notably, our technique requires minimal support from the target resolver. Moreover, this method is primarily designed to determine whether a pair of IPv4 and IPv6 addresses are pointing to the same host. For situations that involve a larger number of addresses, the required probing cost for this method will increase exponentially. Our new algorithm, on the other hand, will complete the discrimination task in a more lightweight manner.

2.2 Alias Address

Alias addresses refer to a group of different addresses that point to the same host. Generally, an IP address represents an interface, and a machine may have multiple interfaces configured, and thus multiple addresses for the same host can be derived based on the interface. There has been a considerable amount of previous works on identifying IPv4 alias addresses [6,20], and the work by Murdock *et al.* [24] is the first to demonstrate the widespread existence of IPv6 alias addresses. Their data indicates that alias addresses are distributed across a considerable number of ASs. Some of these ASs belong to large network service providers such as Google, Cloudflare, and Akamai which are also major public DNS service providers.

3 Methodology

Taking into consideration the widespread existence of alias addresses, we confidently posit that a dual-stack resolver does not necessarily have only one IPv6 and one IPv4 address. In other words, the address relationship of a dual-stack resolver may be 1-N or even M-N. Based on this assumption, we have designed a novel dual-stack resolver discovery model called **DualDNSMiner**. The model's structure is illustrated as Fig. 1.

DualDNSMinner principally comprises of two steps. The first stage involves identifying the relationship between IPv4&IPv6 resolvers, and extracting address clusters that fulfill the requirements for determining the dual-stack resolvers, as well as their inter-relationships. The second step involves performing alias resolution on IPv4 and IPv6 addresses separately. Consequently, the address-address relationships can be converted into host-host relationships. Finally, the 1-1 host-host relationship will be identified, which represents the dual-stack resolvers recognized by DualDNSMiner.

3.1 Active DNS Measurements

It is essential to know the IPv4/IPv6 addresses owned by the dual-stack resolver first and confirm their correlation, so as to further confirm the possibility that they belong to the same host. We thus speculate on the relationship between IPv4 and IPv6 resolvers through active DNS measurement [7].

In DualDNSMiner, a prober actively probes the resolvers issuing specially crafted queries for DNS records for which we are authoritative. In this way, we control both the DNS probes and the authority of the DNS records being probed, thereby permitting testing of open DNS recursive resolvers in our dataset. Our authoritative domains are served by a custom DNS server that is standards compliant [13]. Our authoritative server listens on both IPv4 and IPv6, but returns different results depending upon the incoming request. The server handles multiple domains that support either IPv4 or IPv6 requests, where the choice of domain impacts the IP protocol used by a recursive resolver. We initiate queries

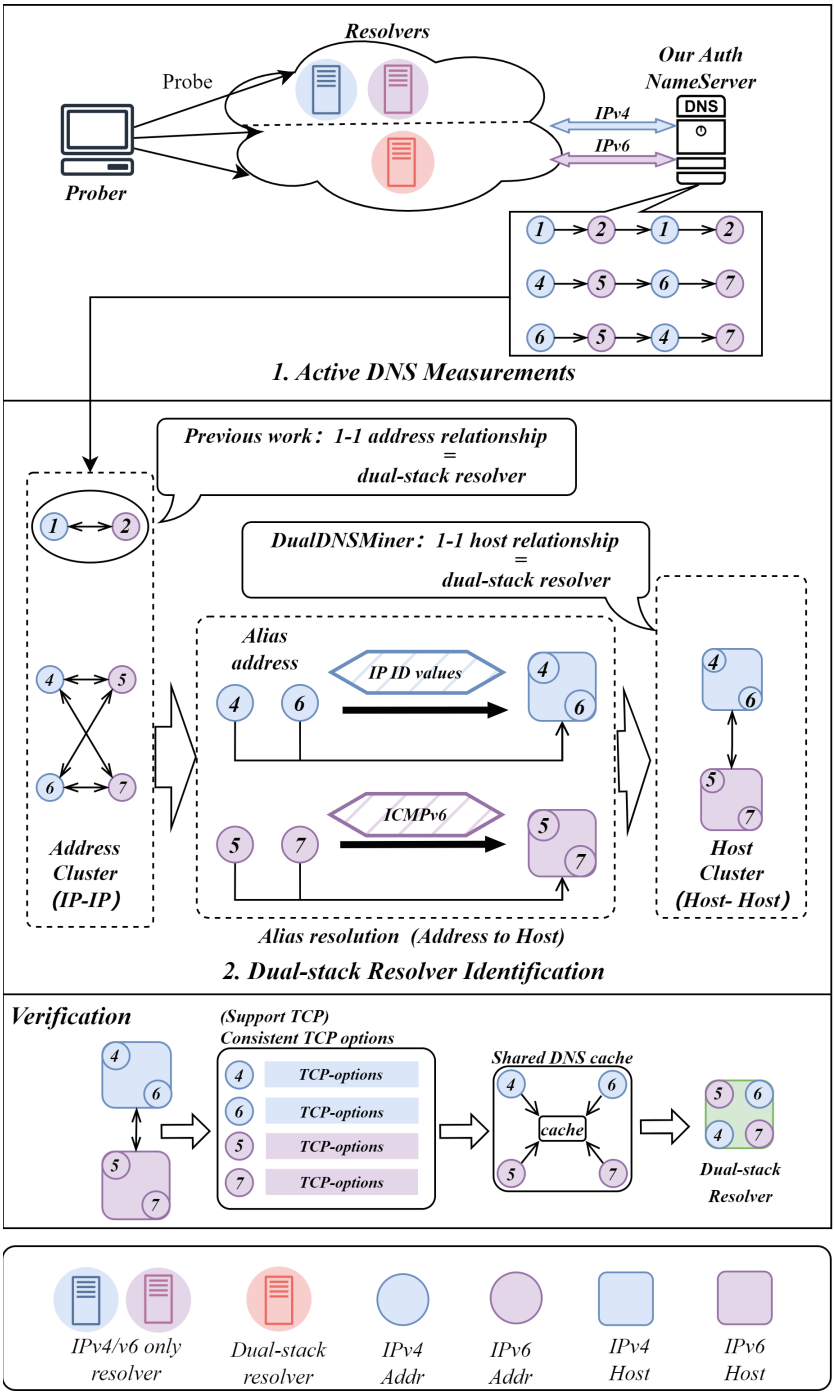

Fig. 1. The workflow of DualDNSMiner. The circle represents an address, and the rounded rectangle represents a machine with corresponding addresses.

to the open and forwarding resolvers. The results from our DNS server for the queried object induce the resolver under test to issue a series of queries that alternate between IPv4 and IPv6 for network transport. We maintain state between requests by specially encoding the returned results such that the final response to the recursive query is a "chain" of IPv4 and IPv6 addresses used by the resolver under test. Figure 2 provides an example timing diagram of the interaction of our prober and an authoritative DNS server with an open resolver whose addresses we wish to infer.

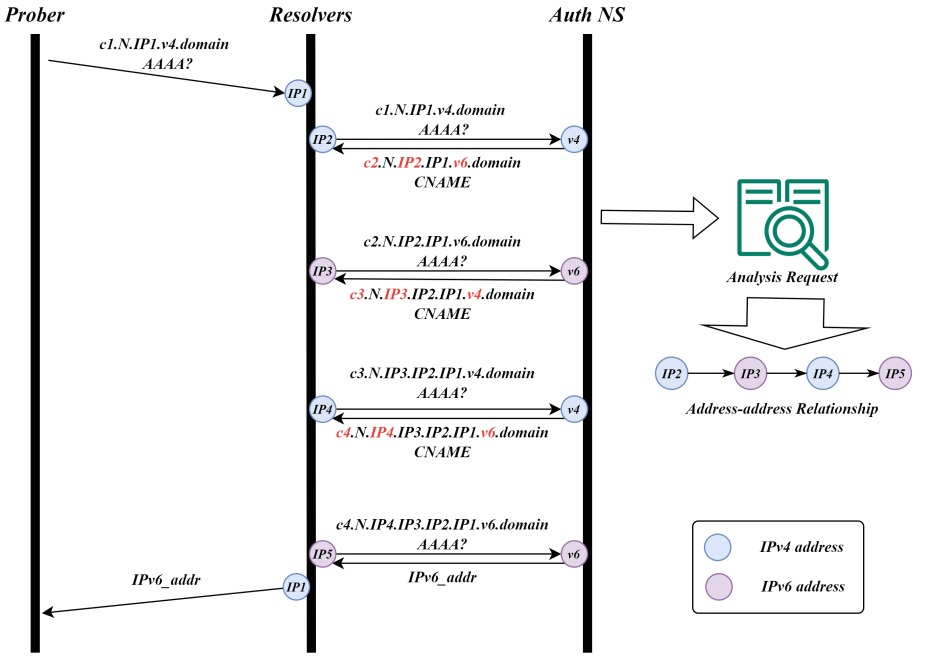

Fig. 2. Example of discovering the association between IPv4 and IPv6 resolvers through active probing. We probe the resolver for our particular domain name, including a nonce N. The participating IPs(*IP2-4*) are encoded in the CNAME response. The `AAAA` response (*IPv6_addr*) from resolver helps us determine whether the request has been resolved correctly. The final result is the sequence of IPv4, IPv6 addresses used by the resolver (here *IP2,IP3,IP4,IP5*). Note that we discard *IP1*, as it may be a forwarder rather than the recursive resolver we are looking for.

Note that our methodology includes several techniques to ensure accuracy. First, each query from the prober includes a nonce that prevents effects due to DNS caching. Second, all state is maintained in the queries themselves, thereby removing the potential for miscorrelation of IPv4 and IPv6 addresses. The prober queries the open resolver or forwarder for a single `AAAA` record. The resolver can only fetch this name using IPv4, but instead of returning the record's value,

our server returns a canonical name (`CNAME`) alias. This `CNAME` encodes the IPv4 address which contacted our server; for example an IPv4 address 1.2.3.4 is encoded into the `CNAME`:

`1-2-3-4.v6.domain.`

This returned `CNAME` exists within the IPv6-only domain. The next CNAME redirects back to IPv4, encoding both IPs. After following another `CNAME` back to the IPv6 domain, our server finally returns a preset `AAAA` record. Note that while DNS authority servers may typically include multiple records in a single returned result, our server only returns one result at a time in order to force multiple lookups and infer the chain. The `CNAME` encoding scheme ensures that, even in the worst case ASCII IPv4 and IPv6 encoding expansion, the chains of length 4 are less than 512 bytes. As 512B is the limit for DNS over UDP, we ensure that the chains rely on neither truncation nor EDNS0 [26]. At the end of the domain name resolution, the authoritative DNS, notes the addresses contained in the requested domain (*IP2-IP4*), and notes the source address of the incoming query (*IP5*). The authoritative DNS then records this chain and returns the `AAAA` record that we set up in advance to allow the prober to check whether domain name resolution was successful or not.

As we will show in Sect. 4.2, many large-scale resolvers are actually clusters, not individual systems. A cluster might be behind a single publicly facing IP address with load distributed among multiple backend machines, or might encompass multiple publicly visible IP addresses. Thus, one can repeat the active DNS probes multiple times in order to gain a more complete picture of cluster structure when present. Since the DNS specification [13] requires that the recursive resolver process the entire CNAME chain, these four IP addresses should represent the same "system" responsible for completing the DNS resolution. The replies themselves have a 0 s TTL and the request contains a counter, thus a resolver should never cache the result.

Subsequently, DualDNSMiner merges the probing results (i.e. associations between addresses) to form a relational graph by merging same-IP nodes, as shown in Fig. 1. Strongly connected subgraphs are discovered from the relational graph in order to transform all the results into address clusters composed of different addresses. Since our objective is to discover dual-stack resolvers, DualDNSMiner will extracts address clusters satisfying specific conditions: **1) all addresses in the cluster belong to the same country; 2) all addresses in the cluster are managed by the same ISP.** Next, DualDNSMiner will perform alias resolution on these address clusters.

The active measurement forces the resolver to use IPv6 (instead of relying on a resolver's preference for IPv6 over IPv4). Since the measurements all occur within a short time window, this measurement is not affected by network changes. It also produces a set of up to four associations, allowing it to more effectively and precisely map the structure of a cluster resolver.

3.2 Dual-Stack Resolver Identification

This step is the core part of DualDNSMiner. Prior studies [1,7] regarded the addresses assigned to the same organization or company with 1-1 relationships as potential dual-stack resolvers, and attempted to evaluate their existence by assessing the similarity of address patterns between IPv4 and IPv6 addresses. Although this standard is simple, it has a flaw. As mentioned before, the relationships between v4-v6 resolvers discovered through active probing are address-address (or interface-interface) relationships that do not represent the host-host relationships. Due to the existence of alias addresses, different addresses can actually point to the same host, as Fig. 3 shows.

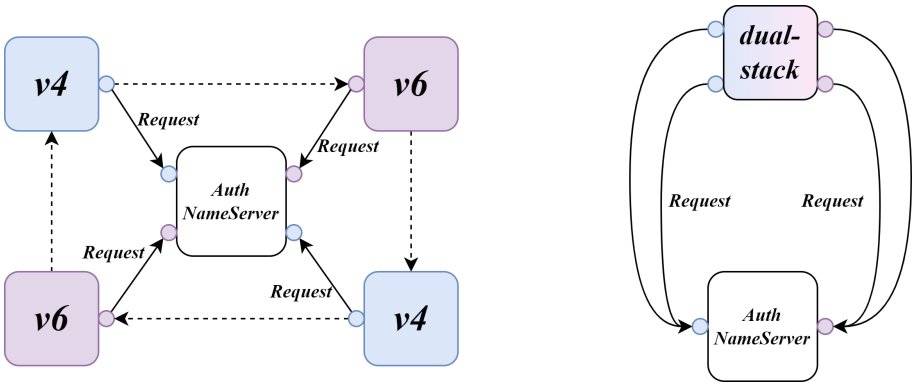

Fig. 3. The circle represents an address (interface), and the rounded rectangle represents a machine with corresponding interfaces. Active probing can reveal the relationship between dispersed address interfaces(left), but in reality, they may belong to the same dual-stack resolver(right).

Therefore, in order to determine whether a dual-stack resolver has these addresses, the addresses need to be took alias resolution. This will enable DualDNSMiner to convert the address-address relationships into host-host relationships. For any address cluster obtained in the first stage, DualDNSMiner employs different discrimination methods for the corresponding IPv4 and IPv6 addresses.

Alias Resolution for IPv4. The primary idea for alias resolution of IPv4 addresses utilizes IP identification (ID) values information within IPv4. Many routers generate IP ID values using a simple counter that is shared across interfaces on the router. If probe packets are simultaneously sent to two interface addresses of the same machine, it is possible to determine if they are generated by the same machine by inspecting the IP ID values in the response packets. This information can be used to confirm whether these IPv4 addresses are alias.

Ally [15], RadarGun [6], and MIDAR [20] all use this basic idea to detect aliases, but differ in how they detect shared counters. Since MIDAR is the first among the three to develop viable tools for conducting large-scale network topological analysis, we utilized MIDAR to perform alias resolution of IPv4 addresses within the address cluster.

Alias Resolution for IPv6. The protocol structure of IPv6 and IPv4 data packets is entirely dissimilar. The IPv6 protocol neither allows in-network packet fragmentation, nor the IPv6 header contains an identifier field similar to the IPv4. Thus, the method that relies on IP ID values cannot be applied to IPv6. Fortunately, the IPv6 protocol provides support for end-host fragmentation, and the end-host has the responsibility to maintain the PMTU status. If a prober tricks fragmentation for one address of a host, when the prober sends ICMPv6 Echo Requests of the same size to other addresses under the same host within a short time, those addresses will also respond to the fragmented packets. Building upon this concept, Beverly [9] *et al.* proposed to obtain the fragment identification of the interface by sending the ICMPv6 Packet Too Big message. Similarly, we have optimized the original probe steps to enable DualDNSMiner to handle large-scale IPv6 alias resolution tasks. Algorithm 1 represents our IPv6 alias resolution process.

Algorithm 1. isv6aliases(C): Determine whether the IPv6 addresses in the cluster C corresponds to the same machine

Require: Active IPv6 addresses in the cluster C
Ensure: Collection of addresses for the same machine M

```
while C is not null do
  M+=1
  sendTooBig(C_0)
  if IsFrag(C_0) then
    for i = 1 to length(C) do
      sendEcho(C_i)
      if IsFrag(C_i) then
        {IP_M} ← C_i
        DELETE(C,C_i)
      end if
    end for
  end if
  DELETE(C,C_0)
end while
return {{IP_M},{IP_M+1},...}
```

Specifically, the IPv6 addresses in an address cluster will be treated as a single set. First, DualDNSMiner collects the status of all addresses in the set (whether they respond, whether they are fragmented, and the size of their PMTU) by

sending ICMPv6 echo requests. Then it performs alias resolution in a loop. During each iteration, we send a Packet Too Big packet to the first address in the set and check if its response characteristics change (whether it responds, is fragmented, and the size of fragmentation). It sends echo request packets to other addresses and checks if their responses also change simultaneously. All IPv6 addresses that change simultaneously will be extracted as addresses of the same host and removed from the set. This process continues until the set is empty, completing the IPv6 alias resolution of the address cluster.

Identification. The host-to-host relationship is a crucial criterion for DualDNS-Miner to determine whether a set of addresses belongs to the same dual-stack resolver. After conducting alias resolution of the IPv4 and IPv6 addresses in different address clusters, DualDNSMiner will be able to group one or more addresses into a single host. This allows us to convert one or more IP nodes into a single host node in the original association graph obtained from active measurement. As a result, DualDNSMiner can obtain cooperative relationships between IPv4 and IPv6 hosts, yielding a cooperative association graph at the host level.

We believe that for a dual-stack resolver, it can only be abstracted as a $\{IPv4, IPv6\}$ host pair at the IP protocol level, and it cannot form multiple pairs (which means there are at least two different machines in it). Based on this concept, a dual-stack resolver's feature in the graph should exhibit **a stable association between an IPv4 host and an IPv6 host without a third one**. Therefore, DualDNSMiner searches for such host relationships to identify dual-stack resolvers and complete the discovery task. The addresses corresponding to the IPv4 and IPv6 hosts included in the relationship are the ones that point to the same dual-stack resolver.

4 Result

4.1 Data Sets

We first need to discover and collect the relationship between IPv4/IPv6 resolvers as our dataset by actively sending DNS query requests. In order to ensure the accuracy of the experiment, we rent two virtual private servers (VPS) in the America and China as our vantage points. All IPv4 open resolvers are tested by sending DNS query request packets to all globally reachable IPv4 addresses through these points. The observation time and collection results of each vantage point are shown in the Table 1.

Through active measurement experiments conducted across the entire IPv4 address space, we discover 77,709 unique recursive resolvers (addresses that send DNS requests to our authoritative servers) based on 895,702 open resolvers (addresses that could return DNS responses). Of these resolvers, 12,602 are IPv6 addresses and 65,107 are IPv4 addresses. After conducting strong connected subgraph detection, we identify 6,987 address clusters. By filtering the

Table 1. Configuration of the measurement and the result of the collected IPv4&IPv6 resolver address from name server's log. Note that the collected addresses all belong to recursive resolvers.

Vantage Point	Collection Period	IPv4	IPv6	Notes
Hong Kong, China	Apr. 19 to May. 27, 2023	64,421	12,059	10 repeated tests to global IPv4 open resolvers.
California, USA	Apr. 1 to 5, 2023	64,839	12,206	5 repeated tests to global IPv4 open resolvers.
Sum		65,107	12,602	Removing duplicate values from the results obtained at different vantage points

clusters according to the rule that all addresses in the cluster are located in the same country and managed by the same ISP, we narrow down the clusters to 2,184. All subsequent data analysis and DualDNSMiner's dual-stack resolver discovery results are based on these address clusters.

Based on the exploration results, we conduct comparative experiments to demonstrate the highlights of DualDNSMiner. In this section, we will introduce the experimental results and explain some of our findings.

4.2 Relationship Between IPv4-IPv6 Resolvers

1-1. The 1-1 IPv4-IPv6 relationship has always been considered as the primary feature of dual-stack resolvers. Our data reveals 741 address clusters that consisted of an IPv4 and IPv6 address. By comparing IPv4 and IPv6 address strings, we found that 279 address clusters had IPv6 addresses with the IPv4 address **directly** embedded in (e.g., 1.2.3.4 & abcd::1:2:3:4), while 84 IPv6 addresses with IPv4 **hex encoded** and embedded (e.g., 1.2.3.4 & abcd::0102:0304). This discovery led us to a conclusion similar to that of Al-Dalky's work [1], which is that there is a certain similarity between IPv4 and IPv6 addresses in the 1-1 relationship. This similarity may be due to special customization by network or resolver administrators to facilitate addressing.

1-N. The 1-N relationship is the primary focus of our DualDNSMiner. This relationship only requires alias resolution of N addresses to determine whether it is a dual-stack resolver. This is simpler than determining the M-N relationship (which is more likely to be caused by load balance) and is therefore more likely to discover new dual-stack resolvers. Our data reveals 938 address clusters that consisted of multiple addresses and are divided according to the IP protocol at a ratio of 1:N.

In addition, we also compare the distribution of addresses in each cluster. Out of the 938 address clusters, 644 are of the 1[$IPv6$]-N[$IPv4$] type, while 294 are of the 1[$IPv4$]-N[$IPv6$] type. Furthermore, the maximum N value in the former type is larger than that in the latter type (195 vs. 13). Obviously, compared

with IPv6 resolvers, IPv4 resolvers are more concentrated and tend to choose the same IPv6 resolvers to complete domain name resolution tasks. It is more likely due to forwarding upstream by the resolver. For example, RFC3901 [11] provides a solution in which IPv4-only resolvers forward requests for IPv6-only domain names to dual-stack resolvers to complete the domain name resolution. Fortunately, only 47 address clusters had an N value greater than 10, which did not significantly affect the effectiveness of DualDNSMiner.

M-N. The association between IPv4 and IPv6 resolvers is extremely complex in some address clusters. Our data has revealed that 505 address clusters are composed of no less than one IPv4 and one IPv6 address, with the most common M-N combination being 2-2, as shown in the Table 2. The IPv4 and IPv6 addresses in these combinations exhibit similarity in address string patterns similar to the 1-1 relationship.

Table 2. The top 5 most numerous association and clusters with the most address in M-N

No.	$M-N$	*Count*	*ISP*	$M-N$
1	2-2	203	Chunghwa Telecom	317-623
2	3-2	47	Telmex Colombia	559-2
3	4-2	46	CAT Telecom	537-5
4	2-3	39	BIZNET NETWORKS	527-10
5	5-2	37	Chunghwa Telecom	109-398

Furthermore, we also have observed that there are some large address clusters, although they are highly unlikely to be dual-stack resolvers (as it is unusual to configure hundreds of IP addresses on a single machine). In these clusters, the proportion of IPv4 and IPv6 addresses is highly uneven. In the top five large clusters, three ones have over 50 IPv4 addresses for every IPv6 address. Such a large difference demonstrates the imbalance in DNS resolution. On the one hand, this may be due to IPv6 addresses pointing to resolver clusters much larger than those for IPv4. Thereby, the two are balanced in terms of hardware resources. On the other hand, it may be caused by defective IPv6 adaptation configurations.

4.3 Dual-Stack Resolver

Due to the current lack of publicly available datasets for dual-stack DNS resolvers, it is essential to employ appropriate verification methods to ascertain the accuracy of the results. Previous work [1,7] cannot prove whether the resolvers are dual-stacked by using the IPv4&IPv6 addresses pattern similarity method. To accurately discover dual-stack resolvers, we propose an active verification method by using the TCP options and cache sharing situations.

On one hand, the efficacy of TCP option-based techniques in detecting dual-stack servers has been demonstrated [8]. We believe that it can also accurately validate DualDNSMiner's dual-stack resolver discovery results. However, this method is only applicable to TCP, not to UDP, which is mainly used by DNS. This makes it impossible for us to accurately verify that resolvers are really dual-stack if they don't support the TCP protocol.

On the other hand, since dual-stack resolvers typically use a single component to simultaneously listen for requests on IPv4/IPv6 addresses, they could use the same cache to speed up resolution unless there is special customization [10]. Therefore, we can also verify whether it is a dual-stack resolver by whether the cache is shared between IPv4/IPv6 resolvers. For each pair of IPv4 and IPv6 addresses, both addresses are queried for the same qname, based on a hash of both addresses and the measurement timestamp:

`h($ipv4$ipv6$timestamp).cachecheck.ourdomain.`

This query is performed twice over IPv4, and twice over IPv6. All the four queries are 5 s apart. Based on the TTL values in the answers, we can determine whether the resolver is actually caching on any or both of the protocols, and whether that cache is shared. However, it should be pointed out that some works have shown that about 10% to 20% of resolvers will share the same cache server [21]. This means that this method has a certain percentage of false positives, so it cannot completely replace the TCP option-based method. Therefore, we only verify whether the parser that does not support TCP is dual-stack by judging whether IPv4/IPv6 share the cache.

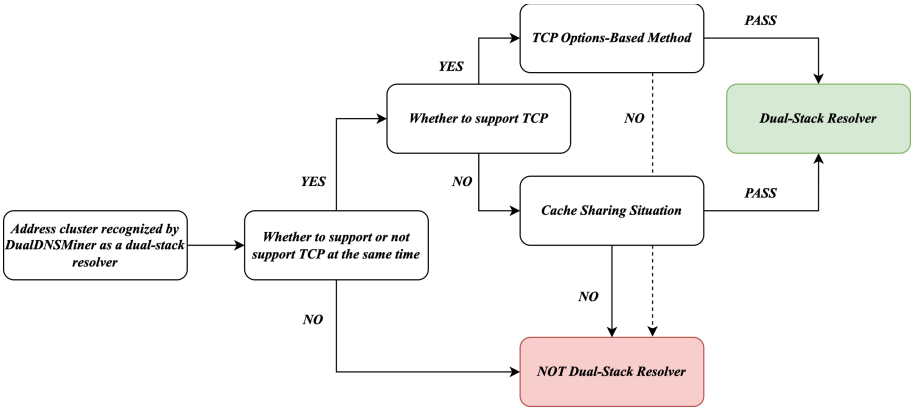

Fig. 4. The process of verifying whether the recognition result of DualDNSMiner is correct.

In summary, our verification method is as follows: First, we will check whether the resolvers (IPv4&IPv6) in a cluster support TCP consistently (all support or none). For clusters that all resolvers support TCP, verify whether they are dual-stack based on TCP options; and for clusters that none of the resolvers

support TCP, verify whether they are dual-stack by checking whether resolvers in the cluster use the same cache; and for clusters that support inconsistent (Some resolvers are supported while others are not), which is obviously caused by different operating systems or different configurations of DNS services, so they can be considered as non-dual-stack. The entire verification process is shown in the Fig. 4.

Ground Truth Validation. Before verifying whether the result of DualDNS-Miner is correct, we need to test whether our verification method is reliable. We build 12 different resolvers as the ground truth in the docker environment using `bind9` [25] and `unbound` [22]. The specifics of each resolver (use `bind9`) are shown in the Table 3. There are also 6 resolvers deployed with `unbound` using similar configurations. All resolvers enable caching and are built through the docker image of `ubuntu` [18].

Table 3. Configuration of our resolvers in docker.

ID	*TCP Support*	*Type*	*Note*
R1	✔	dual-stack	2 IPv4 & 2 IPv6 addresses
R2	✔	IPv4-only	Configure *R1* as the upstream resolver
R3	✔	IPv6-only	Configure *R1* as the upstream resolver
R4	✗	dual-stack	2 IPv4 & 2 IPv6 addresses
R5	✗	IPv4-only	Configure *R4* as the upstream resolver
R6	✗	IPv6-only	Configure *R4* as the upstream resolver

In particular, *R1* and *R4* are the dual-stack resolvers we are actually looking for. They both have two IPv4 addresses and two IPv6 addresses, thus simulating the actual alias address situation. The main difference between the two lies in whether they support TCP. Other resolvers only support IPv4 or IPv6, and these two dual-stack resolvers are configured as their upstreams to achieve cross-stack resolution [11].

We perform 10 replicate tests on these resolvers. According to the evaluation process shown in Fig. 4, *R1*, *R2* and *R3* will be tested by the TCP option-based method, while *R4*, *R5* and *R6* will evaluate by judging whether all addresses in cluster share the same cache due to the lack of TCP support. The results show that our verification method can successfully identify the address cluster (2 IPv4 + 2 IPv6) of *R1* or *R3* as the dual-stack resolver every time, while other resolvers be judged that their address cluster do not belong to one machine. This shows that the addresses of these resolvers do not belong to a dual-stack resolver, which also shows that our verification method is reliable.

However, it needs to be pointed out that although our verification method can accurately identify the dual-stack resolver, the TCP option-based method itself needs to send a large number of probe packets to confirm whether a pair

of IPv4 and IPv6 addresses belong to the same host. As we will show later, this method is much less efficient for large-scale dual-stack resolver discovery than our DualDNSMiner. Therefore, we only use this method as the result verification. In fact, our results also show that DualDNSMiner can also quite accurately discover dual-stack resolvers.

Real-World Results. We compare the practical effectiveness of DualDNSMiner with the 1-1 address relationship judgment rule used in previous works, and verify the results, which are shown in Table 5 in the Appendix A.

As shown in Table 5, DualDNSMiner further improves the discovery efficiency of dual-stack resolvers. Compared to previous methods, it can discover **over 80%** more possible dual-stack resolvers from 1-N and M-N address relationships. Moreover, the validation experiments indicate that DualDNSMiner also achieves a high accuracy rate (**over 90%**) for dual-stack resolver determination results. However, it is worth noting that DualDNSMiner and the new validation method are still unable to confirm some address clusters whether they are dual-stack resolvers. This is caused by the response or deployment characteristics of hosts in these clusters, such as the lack of TCP timestamp options or the lack of expected responses or changes after sending ICMPv6 Too Big packets.

We also analyze the distribution of IPv4 and IPv6 addresses in the 1,237 address clusters that are verified as dual-stack resolvers(**TCP option & share cache**), as shown in Table 4. A significant number of dual-stack resolvers with multiple IPv4/IPv6 addresses have been overlooked in previous work. Quite a few dual-stack resolvers have the same number of IPv4 and IPv6 addresses or more IPv4 addresses. We believe this is caused by all or some of the interfaces owned by the IPv4 resolver being assigned corresponding IPv6 addresses.

Table 4. Statistics on the address associations in different dual-stack resolvers.

No.	$M-N$	*Count*	*No.*	$M-N$	*Count*
1	1-1	735	6	1-3	39
2	2-1	134	7	3-2	30
3	1-2	127	8	3-1	24
4	2-2	49	9	2-3	23
5	3-3	47	10	4-2	19

Furthermore, in terms of discovery efficiency, the TCP option requires sending TCP packets to judge the **63,907** pairs of IPv4 and IPv6 address relationships in all address clusters. In comparison, in our experiments, DNSMiner only sent **49,823** packets to complete the alias resolution of the same address clusters. The discrimination process is based on the results of alias resolution, no request packets need to be sent. Therefore, DualDNSMiner can discover dual-stack resolvers from large-scale address relationships at a **lower cost**, improving actual discovery efficiency.

5 Future Work

DualDNSMiner has successfully excavated more dual-stack resolvers in DNS with traditional methods based on 1-1 addresses relationship, thereby advancing IPv6 DNS measurement efforts. However, due to a few addresses retaining their original response packet characteristics even after receiving a Too Big packet, DualDNSMiner cannot perform alias resolution on them. These addresses cannot be determined as a separate address or a host shared with other addresses. Consequently, a considerable portion of the address clusters in our experimental results remain *Unknown* (Table 5). We will endeavor to explore alternative methods to address this issue in our future work.

6 Conclusion

In this paper, our focus is on the recognition task of dual-stack DNS resolvers. Unlike previous works, we aim to discover dual-stack resolvers from the more complex relationship of IPv4-IPv6 name servers. We introduce a novel technology named DualDNSMiner, and apply it to practical measurements. This technique primarily utilizes alias resolution to transform address-address relationships obtained through active probing into host-host relationships, thus allowing the detection of dual-stack resolvers. We have also proposed a validation method to examine the accuracy of DualDNSMiner's results through data-driven means. The results indicate that compared with traditional dual-stack resolver discovery algorithms, DualDNSMiner can improve the discovery rate by over 80%, and the accuracy of DualDNSMiner's detection results exceeds 90%. Furthermore, DualDNSMiner's recognition results reveal that a considerable number of dual-stack resolvers have multiple IPv4 and IPv6 addresses. In the future, we will continue our research to find a more effective dual-stack DNS resolver discovery method.

Acknowledgment. This work is supported by the Strategic Priority Research Program of the Chinese Academy of Sciences with No. XDC02030400, the National Key Research and Development Program of China with No. 2021YFB3101001 and No. 2021YFB3101403.

A Discovery and Verification Results

Table 5. Comparison of DualDNSMiner with previous methods, and verification of results.

Experiment	*Method*	1-1	1-1 Unknown	1-N	1-N Unknown	M-N	M-N Unknown	Not Dual
Comparison	Previous method [1,7]	741	0	✘	✘	✘	✘	1443
	DualDNSMiner	**741**	0	**411**	227	**218**	106	481
Verification	TCP Option [8]	646/741	89/741	332/411	19/411	141/218	12/218	131
	Share Cache(Not Support TCP)	89/89	0	18/19	0	11/12	0	2
	TCP Option & Share Cache	**735/741**	0	**350/411**	0	**152/218**	0	133

References

1. Al-Dalky, R., Schomp, K.: Characterization of collaborative resolution in recursive DNS resolvers. In: Beverly, R., Smaragdakis, G., Feldmann, A. (eds.) Passive and Active Measurement. Lecture Notes in Computer Science(), vol. 10771, pp. 146–157. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76481-8_11
2. APNIC: Ipv6 capable rate by country. https://stats.labs.apnic.net/ipv6/
3. APNIC: Use of ipv6 for world (XA). https://stats.labs.apnic.net/IPv6/XA
4. Bagnulo, M., García-Martínez, A., Van Beijnum, I.: The NAT64/DNS64 tool suite for IPv6 transition. IEEE Commun. Mag. **50**(7), 177–183 (2012)
5. Bagnulo, M., Sullivan, A., Matthews, P., Van Beijnum, I.: DNS64: DNS extensions for network address translation from IPv6 clients to ipv4 servers. Technical report (2011)
6. Bender, A., Sherwood, R., Spring, N.: Fixing ally's growing pains with velocity modeling. In: Proceedings of the 8th ACM SIGCOMM Conference On Internet Measurement, pp. 337–342 (2008)
7. Berger, A., Weaver, N., Beverly, R.: Internet nameserver IPv4 and IPv6 address relationships, pp. 91–104. Association for Computing Machinery (ACM), New York (2013). 10(2504730.2504745)
8. Beverly, R., Berger, A.: Server siblings: Identifying shared IPv4/IPv6 infrastructure via active fingerprinting. In: Mirkovic, J., Liu, Y. (eds.) Passive and Active Measurement. Lecture Notes in Computer Science(), vol. 8995, pp. 149–161. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15509-8_12
9. Beverly, R., Brinkmeyer, W., Luckie, M., Rohrer, J.P.: IPv6 alias resolution via induced fragmentation. In: Roughan, M., Chang, R. (eds.) Passive and Active Measurement. Lecture Notes in Computer Science, vol. 7799, pp. 155–165. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36516-4_16
10. CoreDNS.io: CoreDNS-cache. https://coredns.io/plugins/cache/
11. Durand, A., Ihren, J.: DNS IPv6 transport operational guidelines. Technical report (2004)
12. Durand, A., Droms, R., Lee, Y., Woodyatt, J.: Dual-stack lite broadband deployments following IPv4 exhaustion. RFC 6333 (2011). https://doi.org/10.17487/RFC6333, https://www.rfc-editor.org/info/rfc6333

13. Elz, R., Bush, R.: RFC2181: clarifications to the DNS specification (1997)
14. Google: IPv6. https://www.google.com/intl/en/ipv6/statistics.html
15. Gunes, M.H., Sarac, K.: Analytical IP alias resolution. In: 2006 IEEE International Conference on Communications, vol. 1, pp. 459–464. IEEE (2006)
16. Hendriks, L., Oliveira Schmidt, R.D., Rijswijk-Deij, R.V., Pras, A.: On the potential of IPv6 open resolvers for DDoS attacks. In: Kaafar, M., Uhlig, S., Amann, J. (eds.) Passive and Active Measurement. Lecture Notes in Computer Science(), vol. 10176, pp. 17–29. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54328-4_2
17. Hu, Q., Asghar, M.R., Brownlee, N.: Measuring IPv6 DNS reconnaissance attacks and preventing them using DNS guard. In: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 350–361. IEEE (2018)
18. Hub, D.: ubuntu-official images. https://hub.docker.com/ubuntu
19. Huston, G.: IPv6 and the DNS. https://labs.apnic.net/?p=1343
20. Keys, K., Hyun, Y., Luckie, M., Claffy, K.: Internet-scale IPv4 alias resolution with MIDAR. IEEE/ACM Trans. Networking **21**(2), 383–399 (2012)
21. Klein, A., Shulman, H., Waidner, M.: Internet-wide study of DNS cache injections. In: IEEE INFOCOM 2017-IEEE Conference on Computer Communications, pp. 1–9. IEEE (2017)
22. Labs, N.: NLnet labs-unbound-about. https://nlnetlabs.nl/projects/unbound/about/
23. Mockapetris, P.V.: RFC1035: domain names-implementation and specification (1987)
24. Murdock, A., Li, F., Bramsen, P., Durumeric, Z., Paxson, V.: Target generation for internet-wide IPv6 scanning. In: Proceedings of the 2017 Internet Measurement Conference, pp. 242–253 (2017)
25. Okamoto, T., Tarao, M.: Implementation and evaluation of an immunity-enhancing module for ISC BIND9. Procedia Comput. Sci. **126**, 1405–1414 (2018)
26. Vixie, P.: Extension mechanisms for DNS (EDNS0). Technical report (1999)