Resolvers Revealed: Characterizing DNS Resolvers and their Clients

CRAIG A. SHUE, Worcester Polytechnic Institute
ANDREW J. KALAFUT, Grand Valley State University

1. INTRODUCTION

The Domain Name System (DNS) performs an essential Internet duty: the translation of host names, which are convenient for humans, into IP addresses, which are used to route packets. To do so, an application on an end-user's system must contact a DNS resolver to perform these translations. While the user's system may run a DNS resolver locally, many use an ISP resolver (sometimes called a *DNS cache*) to perform the resolution on the client's behalf. This resolver then must proceed through a series of queries to locate the DNS server responsible for the relevant DNS records, called the *authoritative server*, which it then queries to retrieve the host to IP address mapping.

This DNS process can be leveraged to detect attackers. Botnets regularly make use of the DNS for command and control and to perform reconnaissance on a destination organization [Choi et al. 2007; Oberheide et al. 2007]. These attack applications have characteristics that deviate from legitimate users. As an example, the recent Feederbot botnet issued customized DNS queries and bypassed its local ISP DNS resolvers to issue queries, likely to evade detection [Dietrich et al. 2011]. This provides opportunities to detect bots by profiling their queries and associations with DNS resolvers. However, no prior work has systematically determined the resolvers used by clients or the query patterns used by these resolvers, preventing such opportunities from being realized.

At the same time, the DNS has received significant attention from researchers. Some prior work has studied DNS query performance, caching effectiveness, resilience of DNS servers, and even the contents of DNS servers. Other prior work has sought to leverage DNS in novel ways. In previous work, we proposed using the authoritative DNS server for access control, allowing it to provide accurate IP mappings as "keys" to reach protected servers, while churning server IP address to prevent access without the proper mapping [Shue et al. 2012]. These novel techniques require a detailed understanding of the DNS, both from an authoritative server and from a resolver standpoint.

While our approach required cooperation from DNS resolvers, these very DNS resolvers appeared to be an understudied topic. Prior work did not address several key questions for us. In particular, we wanted to know if we could 1) distinguish a resolver on a (possibly malicious) end-user system from an ISP-class resolver, 2) associate a client with a particular resolver to create a narrow (and thus more secure) capability, and 3) build useful historical information about a particular resolver and the prior behavior of its clients.

In this work, we broadly explore DNS resolvers to answer these questions. In doing so, we make the following contributions:

(1) **An Approach to Distinguish Resolver Classes:** Simply by passively examining the DNS queries to our resolvers, we can distinguish the resolver software and underlying operating systems in use by many resolver types. With carefully crafted responses, we can further distinguish the most popular resolver implementations (BIND and Microsoft DNS server).
(2) **Detection of Anomalous Query Patterns:** While DNS is highly structured, we can find interesting behavior in the temporal querying patterns of resolvers, with some repeat queries being issued before the expiration of the records in a prior response.

Some ISP resolvers are prolific premature queriers and regularly re-request resolutions for the same records from our servers before the TTL expires.

(3) **Creation of Passive Association Approach for Clients and Resolvers:** While prior work by Mao *et al.* [Mao et al. 2002] developed active probing techniques to link a client and resolver, these techniques only work for established Web communication and cannot be used to screen arbitrary incoming connections. We create a passive, application-agnostic association approach for clients and routers and articulate the challenges in such associations. This technique can augment passive DNS measurements [Zdrnja et al. 2007], which are used in a variety of domains, including malware detection [Grier et al. 2012].

The rest of this paper is structured as follows. In Section 2, we survey related work. In Section 3, we analyze the DNS resolvers that visit our organization. In Section 4, we provide approaches to associate clients and resolvers. In Section 5, we discuss implications of our work for security. Finally, we conclude with discussion in Section 6.

2. RELATED WORK

There have been a large range of studies on the DNS, from studies on DNS caching [Jung et al. 2002; Cohen and Kaplan 2003] to DNS servers and their behavior [Sisson 2010] to the security of DNS resolvers [Dagon et al. 2008]. Each of these studies focuses on a different aspect of the domain name system.

Our own prior work related to DNS [Kalafut et al. 2011] investigated the contents of DNS zones and found some errors in their configurations. Prior work by Pappas *et al.* [Pappas et al. 2009] investigated three specific configuration errors with potential impacts on DNS availability in greater detail. Each of these concentrates on the configurations present in DNS zones or at authoritative DNS servers. Although our current study is not specifically looking for misconfigurations, as these prior two works did, this study does briefly touch on the configuration of DNS resolvers.

The work most closely related to this one is from Mao *et al.* [Mao et al. 2002]. In this work, the authors seek to determine how well a content distribution network (CDN) can predict the optimal server for a client based on the DNS resolver associated with the client. To do so, the author use a clever active probing approach: when a client visits the Web server, the HTML page will include an `img` tag with a source host name that encodes the client's IP address (such as `http://client-1.2.3.4.example.com/1.jpg`). When the client's browser prepares to automatically retrieve the image, it will issue a DNS request for the host name through its local DNS resolver. The `example.com` DNS server can then link the client's IP address with the resolver querying on its behalf. While powerful, this approach has two important drawbacks. First, this method works after the client and server have had considerable interaction; however, if the associations are to be leveraged for traffic filtering, the decision must be made upon the client's first packet, not after a connection has been established. Additionally, this method relies on the behavior of Web browsers. We would like to associate clients and resolvers in an application-independent manner.

3. UNDERSTANDING DNS RESOLVERS

We begin this section by providing a short background on DNS resolvers and servers. Then we begin our analysis by examining the queries issued by the DNS resolvers that access our site. Specifically, we collect a month of DNS interactions with our servers and analyze each of the flag bits in the DNS query headers. We next explore the responses of DNS resolver implementations to some specific configurations of records designed to be special cases. Finally, we examine the frequency at which resolvers query our site, and whether they appear to properly cache our responses.

3.1. Background

Hosts on the Internet are often configured with an ISP-provided DNS resolver that will assist in resolving host names to IP addresses. This configuration is often provided to end hosts by ISP DHCP servers. The consumer-grade variants of operating systems, such as Microsoft Windows, Apple's OS X, and Linux desktop distributions, contain resolver software, but in practice, this resolver software is only used to send recursive queries to the third-party (ISP-provided) resolver (and often the built-in resolver is only capable of this function). These recursive queries essentially request that the third-party resolver issue all the necessary queries across the DNS hierarchy, all the way from the root DNS servers to the authoritative server for the requested host, and simply provide the result to the requestor. DNS server software is used both as the ISP provided DNS resolver and as the authoritative servers. The most popular DNS servers, BIND and Microsoft DNS server, are often used on different operating systems and may have different support for DNS features.

While each of these resolvers, whether a limited built-in resolver or a full-feature application, follow the DNS standards, the standards do not specify every detail of every aspect of the query packet. The values in some fields may be left to the specific implementation. Different resolvers therefore may be implemented differently and therefore have slightly different ways for querying for the same host. Similarly, the DNS specifications do not specify what a server should do for all possible inputs. There are possible queries or responses that are not meaningful or allowed, but the appropriate response to such messages is not specified by the standard. Therefore, different implementations may handle these cases differently. In this section, we focus broadly on the differences in the behavior of different implementations.

3.2. Data Collection

To perform this study, we collected data from the network at the Oak Ridge National Laboratory (ORNL), the largest US Department of Energy laboratory. The ORNL network is used by around 5,000 staff members for general use, with its own enterprise network that is typical for an organization of its size. However, ORNL also provides access to its supercomputers and transfers data from unique science facilities, such as a graphite reactor and neutron accelerator. The network also hosts various government Web sites, such as `www.fueleconomy.gov`, a site that was particularly popular during the 2009 Car Allowance Rebate System program (also known as "Cash for Clunkers").

We performed DNS packet captures (which we label our *DNS packets* data set) at each of our authoritative DNS servers. This allowed us to see each of the DNS requests we received, including the full query. We collected this data from August 1 to August 31, 2010.

3.3. DNS Query Diversity

We profile DNS resolvers based on the way they interact with our authoritative DNS servers. We have only one packet that we can observe from the resolver: the DNS query. Since this query packet is specified in detail in RFC 1035 [Mockapetris 1987], there is not much flexibility in forming a query for an IPv4 address of a given host. However, there is still enough flexibility in these queries that they may provide a hint at the resolver in use. The fields for EDNS0 and DNSSEC, along with the use of recursion flags, can provide some insight on resolvers.

DNS resolvers supporting EDNS0 [Vixie 1999] can use optional enhancements for the protocol. For example, these extensions can allow a resolver and DNS server to agree on a larger packet sizes than the original 512 byte limit, while still supporting backwards compatibility. By studying these extensions, organizations can gauge the amount of resolver support for newer features.

Table I. Top 10 Variants of A record DNS queries for www.ornl.gov from a set of 2,569,434 queries and 127,499 unique DNS resolvers

Request Count	Unique Resolvers	Recursion Desired	Checking Disabled	Extensions for DNS Present	Acceptable Reply Packet Size	DNSSEC OK Bit
1,094,430	49,102			X	4096	X
721,421	53,051					
565,831	18,581		X	X	4096	X
59,204	3,682		X	X	4096	
25,155	1,965			X	512	X
19,420	2,893		X	X	2048	
18,552	1,815		X	X	2048	X
16,991	1,226		X	X	512	X
10,966	62		X	X	1460	X
5,470	1,302			X	4096	

Resolvers can also signal support for DNS security extensions (DNSSEC), defined in RFC 4035 [Arends et al. 2005]. This RFC defines two bits in the base DNS header that were previously reserved. One of these, the checking disabled bit is set by a resolver to indicate that the resolver takes responsibility for verifying the integrity of resource records it receives using DNSSEC. The other bit, the authenticated records bit, is only set by DNS servers and unlikely to be useful in analyzing DNS queries. DNSSEC-aware resolvers must use EDNS0 and these resolvers must also set the DO (DNSSEC OK) bit in the EDNS0 resource record.

The final field we examine is the recursion desired bit. Resolvers can perform queries iteratively, in which the resolver itself contacts each intermediary server to perform the resolution, or request that another server perform such queries recursively on the resolver's behalf. The server receiving this recursion desired request need not honor it and, depending on configuration, may simply deny the recursive request. If a resolver issues a query to an authoritative DNS server with the recursion desired bit set, it may be a sign of misconfiguration since there is no need for recursion at the authoritative server.

With these fields in mind, we analyze the incoming DNS queries from the resolvers that communicate with our authoritative DNS servers. We selected a single specific popular query, the A record query for the www.ornl.gov host name. This query appears in our data set 2,569,434 times with 92 variants of the features discussed above. In Table I, we show the top 10 variants for this query, with the remaining 82 query types being issued a combined total of only 31,994 times. From this table, we see that the most common query type we encounter has full DNSSEC support with the DNS header extensions available. The next most common query, which is actually issued by more unique resolvers than the first, lacks support for both the DNS header extensions and DNSSEC. Some resolvers queried for the www.ornl.gov host name using multiple variants. Of the 127,499 resolvers that queried for the record, 8,453 issued the query using multiple variants.

One insight from this data is that the resolver software used can yield significant variation for the same query. The support for DNSSec, for example, shows up even in simple queries to allow the authoritative server to respond with the additional records for cryptographically verifying the mappings. Consumer-grade resolvers may also ask for recursive resolutions, despite communicating with an authoritative server. These characteristics can be used to distinguish the resolver software.

To link observed query patterns with DNS resolver software, we examined several DNS resolvers: the default resolvers in Macintosh OS X 10.6.5, OpenSUSE Linux 11.2, Windows XP Professional and Windows 7 Home Premium, an installation of BIND 9 on an Ubuntu 10.04 machine, and Microsoft DNS Server running on Windows Server 2008. This collection of machines provides a mix of client and server-grade resolvers.

For each resolver, we used the default settings and configuration and used packet captures to observe the DNS packet headers in the resolver's query. Each of the operating system

default resolvers set the recursion desired flag, causing them to not match any of the top 10 variants. The BIND resolver exactly matched the top query variant we observed in Table I. Similarly, the Microsoft DNS Server by default did not set the recursion desired flag and matched the second entry in Table I. Each of the other systems required us to specify a forwarding name server, likely resulting in the name servers setting the recursion desired flag. The seventeenth most common query type, used by 363 unique resolvers, was the most popular entry with the recursion desired flag set. As previously noted, those resolvers may be misconfigured, since the authoritative server for a domain has no need to support recursion and may, depending on configuration, simply deny requests with the recursion bit set.

These results show that organizations can passively distinguish many resolvers' types simply based on how they construct their DNS queries. While this can be easily disguised with a protocol normalizer, there currently is little incentive for normalizing.

3.4. Interactive DNS Resolver Profiling

While passive analysis of DNS resolvers can show some differences in queries, we can learn more about DNS resolvers by issuing non-standard replies to interact with the resolvers. This more extensive interrogation of the DNS resolvers can lead to unspecified behavior and reveal implementation details of the resolver, allowing us to discover the version of resolver software used by a client even if the resolver uses a customized configuration.

To determine how resolvers respond to unspecified behavior, we implement a DNS server to provide arbitrarily crafted responses. We create a server that mocks authority for `example.com`; for each response, regardless of the client's query, this server returns a `NS` record (`example.com NS ns1.example.com`) in the authority section of the packet and includes an `A` record for `ns1.example.com` in the additional records section of the reply. We also added a proper response record for any `PTR` record associated with the name server, since some resolvers actively query for the `PTR` record of the name server.

We then tested the same variety of DNS resolvers as was used in Section 3.3 We configured BIND9 and the Windows Server system to forward to our customized authoritative DNS server, and configured the default DNS server on the other systems to be our customized server. To issue DNS requests, we used the `nslookup` utility and the appropriate `gethostbyname` library calls, which uses the operating system's DNS resolution library, on each of the systems.

To test the resolvers, we implement the following three non-standard zone configurations:

— **Case 1:** We create zone records that form a `CNAME` chain that is 15 `CNAME` records long, followed by an `A` record for the 16th host name. As an example, one of the `CNAME`s in this chain is `case1-1.example.com CNAME case1-2.example.com`,
with `case1-2.example.com` yielding another `CNAME` for `case1-3.example.com`. We set the TTLs for each record to be long enough to avoid any timeouts. We return only one answer record per packet so the resolver would have to issue subsequent queries to see the entire chain.
— **Case 2:** We return multiple answer records in the same packet. Both a `CNAME` and an `A` record are returned for the same host name, with the `CNAME` pointing to a host record that does not exist in the zone.
— **Case 3:** We return a single response, a `CNAME`, but rather than provide a host name in the response, we incorrectly provided an IP address. Such mistakes were found in a number of zones in previous work [Kalafut et al. 2011].

The first case allowed us to see major differences in the resolver behavior. Of the resolvers, only BIND9 and the Microsoft DNS server would follow `CNAME` queries to assemble answers across packets. BIND9 did successfully traverse the `CNAME` chain and obtained the terminating `A` record, providing the full response to the `nslookup` utility. The Microsoft DNS server

gave up following the CNAME chain after 9 queries. The remaining resolvers refused to follow the CNAME chain at all and instead issued only the CNAME response to nslookup while the gethostbyname libraries did not report any host names.

The second case also allowed us to observe differences in resolvers. The Microsoft DNS server, Windows XP, and Windows 7 resolvers returned the IP address in the A record we provided. However, they improperly handled the CNAME response: rather than returning the correct host name indicated in the CNAME, the "alias" line returned the originally queried host name, preventing the user from seeing the correct alias portion of the CNAME. However, this strange alias behavior was not present in the gethostbyname output and may be an artifact of the nslookup utility. The Mac OS X and Linux machines correctly provided both the CNAME and A record replies to nslookup. Interestingly, BIND9 did not provide both responses. Instead, BIND9 would return the A record response if it was supplied first in our reply packet. However, if the CNAME was supplied first in the packet, BIND9 would issue the follow-up query for the aliased host name, discarding the A record response from the first packet. Further, if the CNAME query failed, such as when our resolver indicated the record did not exist, BIND9 reported the failure to nslookup rather than provide the original A record response.

Each of the resolvers handled the final case correctly: they provided the CNAME response as an alias rather than interpreting it as an A record response. However, BIND9 appeared to recognize the error: while BIND9 would normally issue follow-up queries for a CNAME record, it simply aborted in this case rather than try to issue the follow-up queries. The Microsoft DNS server did not detect this and did issue the follow-up A record queries for the CNAME alias.

We tested BIND9 and Microsoft DNS server with additional cases. We found that the BIND9 resolver is fairly robust in its resolution efforts. It would not follow CNAME chains that were 17 elements or longer. As we previously noted, the Microsoft DNS server aborted after 9 elements in the chain. We also experimented with CNAME chains in which we forced records to expire during the resolution process by setting short TTLs and inserting delay in our server. The BIND9 resolver noticed the expiration and attempted to renew the records. We then altered our records so that attempts to renew records would cause BIND9 to follow both the original and the new CNAME chains, reporting the result of whichever chain terminated (or reached 17 records) first. The Microsoft DNS server did not attempt to renew any records, but it did appear to detect the expirations: it did not traverse as far into the CNAME chain as it did in Case 1, aborting after 6 steps. Both BIND9 and the Microsoft DNS server detected CNAME loops when we experimented with them. These additional features make detecting BIND9 and the Microsoft DNS resolvers particularly easy and unlikely that other resolvers would behave identically.

These results show that we can distinguish server-class DNS resolvers, such BIND9 and Microsoft DNS server, from client-class resolvers, such as those in Linux, Mac OS X, and Windows machines. Further, we were able to distinguish the server resolvers within the classes: BIND9 and Microsoft DNS server would abort after traversing different length CNAME chains. While we could distinguish the client resolvers from the servers, we were not able to distinguish amongst the client resolvers.

We note that the cases we create would cause some resolvers to fail to resolve the intended destination. Organizations may be unwilling to use these tests for their primary server records, such as that for their public-facing Web server. However, these organizations can use these approaches for non-primary records. For example, the primary Web server at these organizations could embed a single pixel HTML IMG tag that would cause the client to attempt to resolve a test host name that would determine the resolver used by the client. In other applications, such as mail, a temporary outage may be acceptable, assuming mail servers correctly re-attempt a failed mail delivery. While our test approaches might cause some resolvers to fail to resolve the record, the TTL on these records can be short. When

the mail server attempts to resolve the mail server's host name again, the DNS server can instead provide the correct response rather than the test.

These results show that non-standard responses can be used to learn more about the DNS resolvers used by organizations.

3.5. Resolver Patterns and Premature Querying

Some DNS resolvers contact our authoritative servers regularly while others visit rarely. We examined the inter-arrival time of our queries and the frequency at which a resolver visits. When we looked at just `MX` or `A` records DNS queries for our mail and Web servers, we found 236,532 unique DNS resolvers visit our site. We found that 113,238 resolvers queried our DNS servers just once for the entire month. Another 35,046 resolvers queried us twice, with 17,972 querying us three times. At the other end of the spectrum, 8,135 resolvers queried us 100 times or more, 524 queried us 1,000 times or more, and 3 resolvers queried us at least 10,000 times.

We next divided the queries we received from resolvers into unique 60 minute time windows and looked at how many windows the resolvers visited us during. There were 134,689 resolvers that visited us during only one hour window, which naturally includes all resolvers that queried us just once. Another 27,598 queried us during just two unique windows while 13,433 queried us during just three unique windows. At the opposite end, 6,642 resolvers queried us during 100 or more hour blocks while 105 queried us during at least 700 of the 744 hours in August. The TTL for the mail servers were about 3 hours while the TTL for the Web server was 5 minutes.

We also noticed a pattern where resolvers re-issued queries before the prior records expired. When an authoritative DNS server sets a time-to-live (TTL) in a reply to a resolver, the resolver caches it to avoid having to issue the query again. However, resolvers may not wait the full TTL before issuing a DNS request again. Some resolvers may choose to lower the TTL for records if they exceed a certain limit. This may protect a caching resolver from having to store many long-lived records that are rarely used. Other resolvers may simply be misconfigured or resource deprived, causing records to be purged prematurely. Other resolvers may be using proactive caching [Cohen and Kaplan 2003] to prevent popular DNS records from leaving the cache by retrieving a new version of the records before the entry expires.

To scope our examination, we focused on requests for our mail and Web servers. We found 26,352 resolvers that re-queried our site for an `A` or a `MX` record that had not already expired. Upon examining the requests, it appeared some queries were likely issued because the resolver had not received the initial response from our authoritative DNS servers. Since most DNS queries are issued through UDP, there are no reliability guarantees; resolvers must simply issue another request if a packet is dropped. To minimize this effect, we exclude any duplicate queries issued within five seconds of the original query. This reduced the count to 17,144 unique resolvers that prematurely queried our site.

During our examination period, our organization used a five minute (300 second) TTL for queries about our Web servers and a three hour (10800 second) TTL value for queries about our mail servers. Given the different time intervals, we examined the queries to these servers separately. There were 376,575 premature queries about our mail servers while there were 101,382 premature queries about our Web servers. In both cases, we calculated the difference between the original query and the repeated query, again excluding any results with a difference of less than five seconds to avoid examining retransmissions due to network packet loss. In Figure 1, we grouped queries into 10 second-wide bins. Based on this figure, we can see that about 35% of premature queries are issued within ten seconds of the original query. This is probably the result of DNS response packet loss. Most bins range between 1.7% and 2% of the premature requests. However, within 30 seconds of the record expiring, we see a relative increase in the percentage of premature queries, which may indicate the

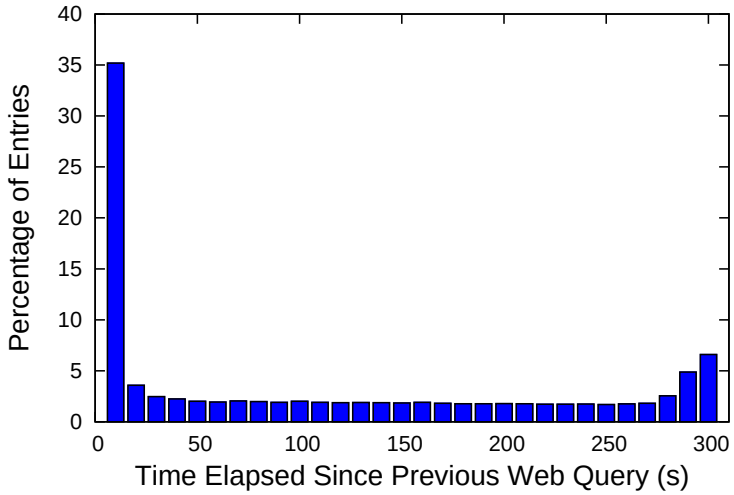

Fig. 1. Time interval histogram for premature Web server queries

use of proactive caching. Other resolvers could be using fast clocks and expiring records a few seconds early. However, in this analysis, we do not see any evidence of DNS resolvers shortening our TTL to a local limit value.

When we examined the mail server DNS patterns, we noticed some irregular caching behavior. In Figure 2, we show the premature queries grouped into 300 second-wide bins. We again see a relatively high percentage of premature queries, about 10%, are issued in the first 300 seconds (with 7% in the first 10 seconds), which again is likely due to query response loss. However, we see some relative increases in premature querying in the middle of the window. We see relative increases in the number of repeated queries around the one hour (3600 seconds) and two hour (7200 seconds) marks, which may be the result of preconfigured limits and query shortening. Interestingly, we did not see a significant relative increase towards the end of the end of the TTL validity period as we did with the Web queries. It appears that proactive caching for mail server records is not used as regularly as Web requests.

We focused on the top five resolvers issuing premature queries. In each case, the host name associated with these IP addresses indicated that the machine was associated with the ISP's DNS infrastructure. The top resolvers included one machine from a Ukrainian ISP, one from a US residential broadband ISP, two from another US residential broadband provider, and one from a Chilean ISP. For each of these resolvers, the premature request length ranged from immediate to almost waiting for the record to expire. Some resolvers, such as the Ukrainian and Chilean resolvers, focused on mail server records, while the US ISP with only one prominent resolver focused on queries concerning our Web server. The final ISP, the US ISP with two prominent resolvers, seemed to query both our Web and mail servers equally. Interestingly, this last ISP's resolvers had adjacent IP addresses and the host names suggested the resolvers were serving the same geographical region of our organization. This ISP may be querying our site more regularly than others simply because our organization's staff may be using this ISP at home and accessing our servers, causing the DNS lookups.

We manually inspected the resolvers with the most premature queries, looking for signs of renewals just prior to expiration of records, which might be evidence of pro-active caching, or requerying only after a specific amount of time, which may be indicative of administrative

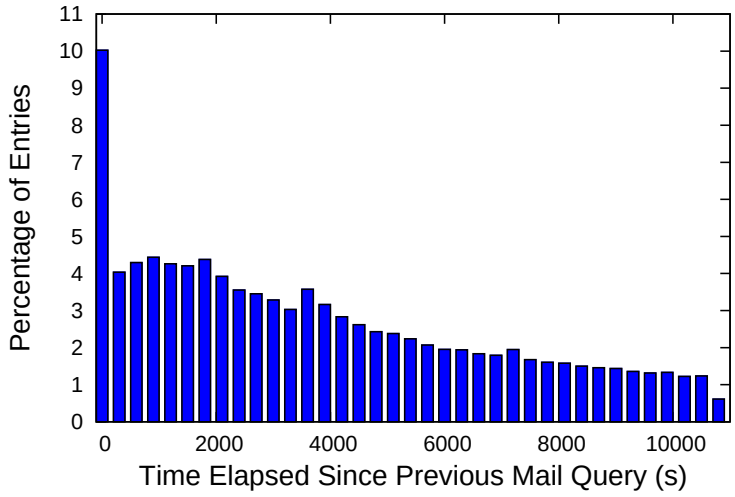

Fig. 2. Time interval histogram for premature mail server queries

cache storage limits. However, the querying behavior of these resolvers was more uniform, with repeated queries after only a short time to those that happened just before the record would expire. As a result, we speculate that these sites may simply be discarding our DNS records early due to resource constraints or misconfiguration.

4. ASSOCIATING CLIENTS WITH THEIR DNS RESOLVERS

Associating DNS resolvers with their clients provides opportunities ranging from optimizing content distribution networks to organizational security. For example, an organization may keep records of prior attacks by clients. If the organization can associate these requests with the DNS resolver that the client used, it may use history to customize the response its authoritative DNS server issues. The organization may choose to provide an accurate response, divert the clients to a honeypot server, or simply withhold a response to resolvers with pathologically malicious clients. By leveraging this association, an organization can make decisions based on the source network, without concerns about DHCP, subnetting, or the scalability of blacklisting in an IPv6 world.

While a powerful approach, the destination organization must be able to associate clients with their resolvers. A perfect association may be needed in some cases, but in many others, including security systems for capabilities or historical policy decisions, security, a list of most probable associations may suffice.

We now describe the data we studied, the challenges in associating clients and resolvers passively, and then describe our association strategy.

4.1. Preparing the Data

To augment the *DNS Packets* data set described in Section 3.2, we collected additional data from the Oak Ridge National Laboratory network, during the same August 1 to August 31st time frame. Specifically, we captured packet header data from each UDP Packet, ICMP packet, and TCP SYN packet that traversed our network border. From each packet, we recorded the source and destination IP addresses, port numbers, packet lengths, and in the case of TCP, the flags set in the packet header. We additionally used an August 15, 2010 routing table from the Route Views Project [University of Oregon Advanced Network Technology Center 2010] to perform longest prefix matching on the IP addresses of the

clients to determine their autonomous system (AS). The AS was with each client connection, allowing us to perform network-grain analysis. We label this data set the *network headers* data set.

The network headers data set gives us a connection history of all interactions from external systems to our own. However, neither this data set nor the DNS packet captures on their own provide enough information to associate clients with their DNS resolvers. This can only be done by combining the two types of data. By merging the two data sets, we can determine when each resolver contacted our site, what the resolver queried for, and the time and system that each client contacted.

Although we had used NTP to synchronize both the clocks on the network perimeter logging system and the DNS packet capture server, the timestamps of the same DNS request packet being seen at each differ due to an approximately 8 millisecond average latency between the perimeter and the DNS server. Accordingly, we subtract 8 milliseconds from the time recorded in the *DNS Packets* data set. Because the latency varied, even after this correction, matches between the two sets are not exact. To compensate for this when combining the two data sets, for each DNS lookup seen in the *network headers* data set, we search for the record in the *DNS packets* data set with the same resolver and destination IP address which is the closest temporal match within one second. If no appropriate match is found, we discard the entry. The cases with no match were concerning, so we consulted the ORNL IT operations team who indicated that many of the unmatched DNS queries were due to a firewall between the collection points that was blocking access to abusive DNS resolvers (e.g., those engaged in flooding associated with denial-of-service attacks), causing them to not appear in the DNS packet captures.

For entries in which a close match is found, we merge information from the DNS request and response into the *network headers* data set, retaining the exact timestamp from the *network headers* data set. This allows us to have a consistent, high resolution time reference between packet arrivals. For all DNS responses, we recorded the IP addresses and TTL values in any `A` records present in the answer, authority, or additional records portions of the response. If a reply did not contain any `A` records, we still retained the record, but marked it to indicate the absence of `A` records. This approach allowed us to track exactly how much time each resolver should have cached any `A` record, providing an appropriate time window for use in associating resolvers with their clients. Finally, to prevent local traffic from affecting our results, we excluded all traffic from IP addresses in our own autonomous system. In our merged data set, we had 187,117,784 connections from clients, 38,632,137 DNS queries that matched both files, and 2,187,908 DNS queries that could not be matched in the two data sets. Of the client connections, there were $7,545,546$ Web connections from $421,489$ unique Web clients and $2,951,458$ mail connections from $738,320$ unique mail servers. In comparison, there were $449,834$ unique DNS servers that contacted our organization.

4.2. Challenges with Association

Associating DNS resolvers with their clients is challenging for several reasons:

— **Possible Lack of Network Proximity:** The recursive DNS caches for a client may not be co-located in the same network as the client. Some networks may rely on the DNS infrastructure of their Internet service providers. Services such as OpenDNS [OpenDNS 2011] and Google Public DNS [Google 2011] servers allow a client to query name servers in an entirely unrelated network. From a third-party perspective, it can be difficult to detect a relationship between the client and these remote DNS servers. Ager *et al.* first discovered and investigated this phenomenon [Ager et al. 2010].

— **High Traffic Volume:** Our organization received connections from millions of clients in our collection period, frequently with multiple clients connecting to the same server in

the same second. Even with short TTL values, due to DNS caching, a given Web client can have hundreds of candidate DNS resolvers and vice versa. The enormous number of possible candidates can make it difficult to determine which clients and resolvers are actually related.

— **Load Balancing:** A client could use multiple resolvers for its DNS requests. ISPs often employ load balancing server infrastructure to accommodate local client demand. Because of this, there may be multiple DNS resolvers associated with the same client machine at different times.

We initially attempted a straightforward method of associating clients and resolvers, which suffered from numerous challenges. In this initial association strategy, we examined each connecting client and searched for preceding DNS requests to associate the client with a plausible resolver. While this association approach sense for a security filtering system, it had several negative aspects from an analysis standpoint:

(1) The client may not have used DNS to connect. This may be caused by direct IP address connections or application-layer caching of DNS records.
(2) The valid time-to-live may be ambiguous if the server the client is accessing has multiple DNS records with differing host names and TTLs pointing to it, making it difficult to determine how far back to search for a match.
(3) The candidate resolver list quickly becomes unmanageable due to resolvers that frequently query our servers. At the time a client accesses a server, there are often hundreds of DNS resolvers who may still be caching a corresponding DNS record with a valid TTL.
(4) Some resolvers appear to be poorly configured and constantly query our DNS servers. These become the most frequent resolver associated with each client.
(5) Many clients visit only once, making it impossible to narrow down the resolver list by correlating the potentially used resolvers across visits.

The challenges of this approach proved too difficult to overcome, leading us to abandon it in favor of the association strategy discussed below.

4.3. Associating a Resolver's First Client

Given the limitations encountered in mapping from client accesses to DNS requests, we instead attempt to map from a DNS request to the clients likely to be using the response. While this may superficially appear to be a trivial change, it has significant advantages. First, a resolution request is typically the result of a client requiring information to establish a connection. Accordingly, a resolution providing an IP address is likely to be followed by a client leveraging that information to establish a connection. This eliminates the difficulties with clients caching records at the application layer or connecting directly to an IP address. Second, the TTL window is clear for these queries: each record has a set TTL associated with it, so we know how long to consider when searching for clients. This approach addresses the first two concerns of the previous approach, but it still faces issues with large candidate lists and challenges in correlating multiple interactions.

We begin by building a *potential association* list. This list is built by associating the first client that connects to one of our servers after a DNS query returning that server's IP address, with the resolver used to make the DNS request. For example, if a resolver at IP address `a.b.c.d` issues an `A` record query for our Web server and we next observe a client connection to that server from IP address `e.f.g.h`, we associate `a.b.c.d` and `e.f.g.h`.

This approach also has limitations. Specifically, it is unlikely to be successful for a busy machine since there may be unrelated clients that just happen to arrive between the resolver and the client. However, this limitation may be mitigated by making matches on low-volume machines, during slow periods for typically busy machines, or by leveraging additional

information. In our data sets, we must examine only periods of low activity. In situations where additional information is available, other approaches to mitigate this issue may be possible. For example, busy servers such as Web hosting providers may be able to leverage virtual hosting information (DNS lookups and HTTP requests for different host names on the same IP address) since these providers often have thousands of low-volume DNS domains hosted on the same infrastructure [Shue et al. 2007]. We applied a set of thresholds designed to reduce the potential for false associations:

(1) The elapsed time between the resolver and first client must be within one second. This heuristic is based on the thought that a DNS request is usually made because a client intends to visit immediately.
(2) There must be 30 or fewer clients within one minute of the resolution. This heuristic addresses the concern of associating unrelated clients and resolvers due to heavy traffic.
(3) During the month-long duration, the association must have occurred at least 10 times. This heuristic serves to reduce the chance that a potential association is made in error by only keeping those that occur frequently.
(4) For each client associated with multiple resolvers, we discarded associations that occurred fewer times than 50% of how often the most common association for that client occurred. This allows for clients that load balance between multiple resolvers, while discarding associations that are considered likely to be erroneous.

These criteria allow us to exclude spurious associations while allowing us to detect regular associations and the associations with load-balancing resolvers. After applying the thresholds, we obtained 13,268 potential associations between clients and resolvers. While these extremely conservative associations represent only a small percentage of unique clients (about 0.4%), they allow us to examine trends with lower risk of false associations. Of these associations, 10,473 (79%) were between clients and resolvers in the same autonomous system. In fact, 1,371 pairs were for the same IP address: a machine that regularly acts as a DNS resolver and a client for itself. At the same time, this association is not guaranteed: the remaining 21% of potential associations associate clients with resolvers outside their autonomous system.

The system as described so far does not nearly capture all actual associations between clients and resolvers. Infrequent clients will not be associated, as well as those that happened to not be the first using the results from a resolution, and those that only connect during busy periods. Therefore, we must generalize the potential associations. We aggregated these potential associations to the AS granularity. From these records, we found 3,233 associations between a client AS and a resolver AS. We then generalized from these associations by assuming that any host in the client AS could use any resolutions performed by any resolver host in the associated resolver AS. We acknowledge this assumption may not always hold: some resolvers may not share their results with other hosts. However, we used it to make a first pass at our data set.

We manually inspected the top 10 most popular non-matching potential ASN associations using the ARIN autonomous system number list [ARIN 2010] and show the entries in Table II. We can see that the top 3 most popular non-matching potential ASN associations are among different ASNs for the same organization. However, many of the remaining associations are unlikely. For example, the fourth and fifth most popular associations are unlikely. They associate various Google crawlers with Yahoo in the former case, and with Comcast residential resolvers in the latter case. We investigated these cases further and find the results happen by chance, indicating that our filtering strategy manages to occasionally make spurious potential associations, even with the thresholds designed to reduce the chances of this.

The potential associations, aggregated to the AS level, provide an idea of which clients visiting our organization can potentially use which resolvers. Once these potential asso-

Table II. Top 10 Most Popular Non-Matching ASN Associations

	AS Number		AS Name	
Count	Client	Resolver	Client	Resolver
292	7725	20214	Comcast	Comcast
118	22394	6167	Cellco	Cellco
100	3598	8075	Microsoft	Microsoft
75	15169	36647	Google	Yahoo
64	15169	7725	Google	Comcast
52	3786	4766	APNIC Res.	APNIC Res.
45	15169	33491	Google	Comcast
36	38631	23576	APNIC Res.	APNIC Res.
25	7132	6389	AT&T	BellSouth
16	8075	7725	Microsoft	Comcast

ciations are generated, we apply them to find the DNS resolution most likely associated with a given client visit. In this way, the information generated during slow periods can be leveraged to heuristically make associations during periods where it would otherwise not be possible. We apply the AS level potential association list to process our data sets and create actual associations between any resolvers and clients if:

(1) The resolver and client were in a potential AS association.
(2) The client connected to an IP address obtained by the resolver.
(3) The client connected before the TTL would have expired in the resolver's record.
(4) There was no more recent DNS resolution with an unexpired TTL from another potentially associated resolver for the same address.

Using only the 3,233 AS associations, we were able to associate 113 million (60%) of the 188 million client visits. In terms of distinct clients, 1.6 million (47%) of the 3.4 million distinct client IP addresses are associated every time they are seen. A further 2% can be associated some of the times they visit, but not for every visit.

Since the majority of potential associations were made in the same AS, we expanded the potential association list to include associations of all ASes with themselves, and then repeated the analysis. We could then associate 120 million (64%) of the results. With this latter standard, we were able to associate 2 million (60%) unique client IP addresses with resolvers every time they were seen, and a further 2% some of the time. We attempted a further heuristic approach, adding potential associations for all directly neighboring ASes, but this greatly increased the number of potential associations, while only adding associations for a few hundred thousand previously unassociated client visits. At the autonomous system granularity, 43% of ASes have only a single associated resolver and 81% of ASes have 5 resolvers or fewer.

While our manual verification shows that our passive association approach often makes plausible links between clients and resolvers, such as associations in the same ISP, we do not have independent validation of the approach. Future work could make simultaneous use of the active association approach proposed by Mao *et al.* [Mao et al. 2002] to determine the passive association accuracy rates for Web users.

5. IMPLICATIONS OF DNS ANALYSIS FOR SECURITY

By combining our analysis on DNS query behaviors and our associations between clients and their resolvers, we can develop new approaches for detecting and thwarting attackers.

Prior work efforts have tried to distinguish attackers from legitimate users based on their network characteristics. For example, Ramachandran and Feamster used the timing and network proximity of hosts to determine if machines were likely to be spam senders [Ramachandran and Feamster 2006]. The DNS fingerprinting we have developed provides opportunities for organizations to notice whether requests are coming from ISP infrastructure or from ad-hoc resolvers more likely to be associated with attacks.

Unfortunately, these fingerprints are not enough. Without a mechanism to link a resolver and its clients, a defending organization cannot determine the clients associated with potentially malicious DNS queries, nor can it tell if the client used a legitimate DNS server. Such mechanisms must work across protocols and should allow filtering at the connection negotiation stage, rather than after the client has gained access to a protected server. Our association strategy supports these uses, enabling new analysis.

As we previously discussed, the Feederbot example highlights the potential for using our work for new security goals. The botnet exhibited distinguishable DNS behaviors, matching our fingerprinting goals, and bypassed ISP DNS infrastructure, which is detectable by our association approaches. By linking our approaches, defending organizations can detect and mitigate (e.g., by blocking the attacker or directing the attacker to a honeypot) an attack.

One simple metric, a failure to be associated with a DNS resolver, could be a symptom of malicious activity. To evaluate this metric, we created a corpus of blacklisted IP addresses by merging multiple Internet blacklist data sources. We used IP addresses listed in the Spamhaus SBL and XBL blacklists [Spamhaus Project 2010a; 2010b] from August 2010, sender IP addresses from messages rejected by our organization's spam filters, and IP addresses of Web crawlers found to be harvesting email addresses from honey pot servers at our organization. With these data sets, we had a collection of 35,848,009 unique blacklisted IP addresses. We then compared the IP addresses of clients from both of our data sets. Of the 2,027,048 unique client IP addresses we were able to always associate with a resolver, we found that 412,759 (20.4%) appeared in these blacklists. Of the 1,353,864 unique client IP addresses that we were never able to associate, 239,091 (17.6%) appeared in these blacklists. For the 66,971 clients that we were able to associate at least once, but also failed to associate at least once, 16,689 (24.9%) appeared in these blacklists. Our results show that simply failing to associate with a resolver using our approach does not seem to be a good indicator for malicious intent.

In the cases where a malicious client is associated with a resolver, remediation may be possible. A destination organization can begin to aggregate client activity at the DNS resolver granularity. As we suggested in prior work [Shue et al. 2012], a destination organization may build a history of detected attacks and their associated resolvers. This empowers a destination to make policy decisions about resolvers associated with higher risk clients, including decisions about increased scrutiny, filtering, or even requirements that users authenticate or solve CAPTCHAs [Von Ahn et al. 2003] to distinguish automated clients.

Resolvers can also become a point-of-contact for destination organizations. In replying to queries from a DNS resolver, an authoritative server can provide quantitative feedback about the source organization and indicate problematic clients as additional records. ISP resolvers that choose to parse these additional records would learn about hosts on their network that may be compromised and launching attacks, even if these malicious clients choose to bypass the ISP's own infrastructure. This feedback could also offer incentives for addressing the behavior: the destination may declare that it will begin requiring authentication/CAPTCHA-solving on the ISP's users, which may be inconvenient, if the malicious activity continues. With such interactions, the source and destination organizations may be able to cooperate to reduce the overall level of attacks on the Internet.

6. CONCLUSION

We have introduced techniques to fingerprint DNS resolvers, revealing information about the specific server used and, in some cases, the resolver's underlying operating system. We find that the DNS resolvers can provide insight into an organization and its clients' systems. Roughly half of resolvers support DNSSec extensions, showing that regular usage of this protocol may hinge upon authoritative servers' DNSSec deployment. We have also identified patterns in resolvers and reported on some instances where resolvers are querying

our systems more than expected due to misconfiguration or resource limitations. We have also introduced passive approaches to associate DNS resolvers and their clients.

We believe this work shows the potential for using DNS resolvers as part of security decisions at a destination organization. Given the relatively smaller number of resolvers that client systems use, these resolvers may serve as useful aggregators for client history.

REFERENCES

Ager, B., Mühlbauer, W., Smaragdakis, G., and Uhlig, S. 2010. Comparing DNS resolvers in the wild. In *ACM Internet Measurement Conference*.

Arends, R., Austein, R., Larson, M., Massey, D., and Rose, S. 2005. Protocol modifications for the DNS security extensions. IETF RFC 4035.

ARIN. 2010. ASN listing.

Choi, H., Lee, H., Lee, H., and Kim, H. 2007. Botnet detection by monitoring group activities in DNS traffic. In *IEEE International Conference on Computer and Information Technology*. IEEE, 715–720.

Cohen, E. and Kaplan, H. 2003. Proactive caching of DNS records: addressing a performance bottleneck. *Computer Networks 41,* 6, 707–726.

Dagon, D., Provos, N., Lee, C., and Lee, W. 2008. Corrupted DNS resolution paths: The rise of a malicious resolution authority. In *Network and Distributed System Security Symposium*.

Dietrich, C. J., Rossow, C., Freiling, F. C., Bos, H., van Steen, M., and Pohlmann, N. 2011. On botnets that use dns for command and control. In *7th European Conference on Computer Network Defense (EC2ND)*.

Google. 2011. Google Public DNS. `http://code.google.com/speed/public-dns/`.

Grier, C., Ballard, L., Caballero, J., Chachra, N., Dietrich, C. J., Levchenko, K., Mavrommatis, P., McCoy, D., Nappa, A., Pitsillidis, A., et al. 2012. Manufacturing compromise: the emergence of exploit-as-a-service. In *ACM Conference on Computer and Communications Security*. ACM, 821–832.

Jung, J., Sit, E., Balakrishnan, H., and Morris, R. 2002. DNS performance and the effectiveness of caching. *IEEE/ACM Transactions on Networking 10,* 5, 589 – 603.

Kalafut, A., Shue, C., and Gupta, M. 2011. Touring DNS open houses for trends and configurations. *IEEE/ACM Transactions on Networking 19,* 6, 1666–1675.

Mao, Z. M., Cranor, C. D., Douglis, F., Rabinovich, M., Spatscheck, O., and Wang, J. 2002. A precise and efficient evaluation of the proximity between web clients and their local DNS servers. In *USENIX*.

Mockapetris, P. 1987. Domain implementation and specification. IETF RFC 1035.

Oberheide, J., Karir, M., and Mao, Z. 2007. Characterizing dark DNS behavior. *Detection of Intrusions and Malware, and Vulnerability Assessment*, 140–156.

OpenDNS. 2011. OpenDNS.

Pappas, V., Wessels, D., Massey, D., Lu, S., Terzis, A., and Zhang, L. 2009. Impact of configuration errors on DNS robustness. *IEEE Journal on Selected Areas in Communications 27,* 3, 275–290.

Ramachandran, A. and Feamster, N. 2006. Understanding the network-level behavior of spammers. In *ACM SIGCOMM*.

Shue, C., Kalafut, A., and Gupta, M. 2007. The Web is smaller than it seems. In *ACM Internet Measurement Conference*.

Shue, C. A., Kalafut, A. J., Allman, M., and Taylor, C. R. 2012. On building inexpensive network capabilities. *ACM SIGCOMM Computer Communication Review 42,* 2, 72–79.

Sisson, G. 2010. DNS survey: October 2010. Tech. rep., The Measurement Factory.

Spamhaus Project. 2010a. Exploits block list (XBL). `http://www.spamhaus.org/xbl/index.lasso`.

Spamhaus Project. 2010b. Spamhaus block list (SBL). `http://www.spamhaus.org/sbl/index.lasso`.

University of Oregon Advanced Network Technology Center. 2010. Route Views project. `http://www.routeviews.org/`.

Vixie, P. 1999. Extension mechanisms for DNS (EDNS0). IETF RFC 2671.

Von Ahn, L., Blum, M., Hopper, N., and Langford, J. 2003. Captcha: Using hard ai problems for security. *Advances in Cryptology—EUROCRYPT 2003*, 646–646.

Zdrnja, B., Brownlee, N., and Wessels, D. 2007. Passive monitoring of DNS anomalies. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 129–139.