DNS Resolvers Considered Harmful*

Kyle Schomp†, Mark Allman‡, and Michael Rabinovich†
†Case Western Reserve University, Cleveland, Ohio, USA
‡International Computer Science Institute, Berkeley, California, USA
kyle.schomp@case.edu, mallman@icir.org, michael.rabinovich@case.edu

Abstract— The Domain Name System (DNS) is a critical component of the Internet infrastructure that has many security vulnerabilities. In particular, shared DNS resolvers are a notorious security weak spot in the system. We propose an unorthodox approach for tackling vulnerabilities in shared DNS resolvers: removing shared DNS resolvers entirely and leaving recursive resolution to the clients. We show that the two primary costs of this approach—loss of performance and an increase in system load—are modest and therefore conclude that this approach is beneficial for strengthening the DNS by reducing the attack surface.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network communications

General Terms

Design; Security

1. INTRODUCTION

The Domain Name System (DNS) is a key component of the Internet infrastructure. It provides mapping between human-readable names of Internet hosts and their network-interpreted numerical IP addresses. Client devices generally leverage DNS *resolvers* to discover IP addresses. These resolvers in turn obtain the mappings from *authoritative* DNS servers, which maintain these mappings.[1] DNS resolvers abstract the multi-step iterative DNS resolution procedure from clients and provide a shared cache across clients—thus offering the possibility of better performance to clients and scalability to DNS itself. While DNS resolvers follow the classic architectural approach of modularity, we question whether this factorization is still useful in the modern Internet, or whether we should eliminate resolvers and instead have clients perform their own recursive resolutions. Eliminating DNS resolvers promises a number of benefits.

First, removing resolvers simplifies the overall system. Modern DNS resolvers constitute a complex infrastructure with many distinct components [23], making the system difficult to manage and troubleshoot. Pushing the functionality to clients makes the clients themselves more complex, but in a way that is easier to manage and more transparent than our current nebulous situation where a resolution fails because of "something out there".

Second, DNS resolvers are vulnerable to multiple forms of attack [14, 17, 24] such as fraudulent record injection, which expose end users to critical security threats. Further, attackers can launder requests through the upwards of 30M open resolvers [2, 23] that will answer arbitrary queries from arbitrary clients. These open resolvers can be used by an attacker to hide their tracks (e.g., as part of a wider DoS campaign) or circumvent firewalls to expose closed portions of the resolution ecosystem—i.e., that only answer queries for "internal" hosts—to indirect attacks. By removing DNS resolvers, we (i) eliminate the threat of resolver cache poisoning attacks on clients conducting their own resolutions and (ii) we reduce the overall attack surface of the DNS ecosystem as shared resolvers gradually disappear.

Third, shared DNS resolvers handicap the operation of replicated services, notably content delivery networks (CDNs)—which carry 39–55% of Internet traffic [12]. Since a DNS request generally precedes content requests, CDNs often use the origin of the DNS request as a hint about the location of the client. However, previous work shows that clients and their DNS resolvers may in fact be far apart and therefore the replica chosen based on the DNS resolver's IP address will offer suboptimal performance [4, 16]. Our proposal of simply removing shared DNS resolvers directly tackles this issue—without additional mechanisms such as

*This work was partially supported by NSF grants CNS-1237265, CNS-0831535, and CNS-0831821.

[1]The DNS ecosystem consists of many different components that play a part in the name resolution process. In this paper we consider all resolution components that lie between clients and authoritative name servers as "resolvers" no matter their specific function.

Hotnets-XIII, October 27–28, 2014, Los Angeles, CA, USA.

http://dx.doi.org/10.1145/2670518.2673881

[9]—by exposing the client's IP address to the authoritative DNS servers that direct clients to specific replicas.

In addition, we note that clients may independently choose to resolve hostnames themselves without changes anywhere else in the system—there are no barriers to transition to our approach. While this does not eliminate the security issues surrounding shared resolvers, it makes them moot for clients that have chosen to conduct their own lookups.

In the remainder of this paper we empirically establish that in terms of performance and scalability the benefits of shared DNS resolvers are at best modest. Through trace-driven simulation, we show that direct client resolution provides similar performance to the end user when compared to using a shared resolver. Further, we show that the overall load increase on the rest of the system is modest. While these are not the only two issues to tackle when considering the removal of DNS resolvers—we briefly sketch others in § 6—we believe these are the two largest initial questions to consider. We believe our initial investigation shows eliding DNS resolvers to be a promising approach for strengthening the overall name resolution process.

2. RELATED WORK

Many previous studies discuss specific DNS security vulnerabilities (e.g., [7, 10, 24, 25]) and offer point solutions (e.g., [5, 8, 27, 29]). We do not address any specific issue, but rather observe that many problems arise in shared resolvers and therefore we simply *eliminate the target*, which mitigates both known and unknown vulnerabilities. Unlike previous work to mitigate resolver vulnerabilities through modifications to the DNS ecosystem—which naturally adds complexity—our approach requires no changes to the protocol and results in an overall less complex ecosystem.

DNSSEC [6] is a general approach that strives to tackle security issues not by point solutions that aim to fix parts of the infrastructure, but by cryptographically securing the information in DNS transactions. Currently DNSSEC deployment is low—with only roughly 1% of resolvers validating DNSSEC records [11, 13] despite DNSSEC approaching its 10^{th} anniversary. Our approach is orthogonal to DNSSEC.

In addition to security concerns, shared resolvers pose challenges to CDNs as we discuss in § 1. In particular, CDNs frequently assume DNS resolvers and clients are close, which turns out to be wrong in some cases [4, 18, 21, 26]. Several proposals develop ways to convey clients' network location to CDNs within DNS requests [9, 15, 19]. Our proposal simply provides this information to CDNs as the source address of the DNS request.

3. DATASETS AND METHODOLOGY

We leverage three datasets in our study. Our first dataset is a 4 month long set of traffic logs from the Case Connection Zone [1]—a fiber-to-the-home network connecting roughly 100 residences to the Internet with 1 Gbps fiber. Our logs are collected using Bro [3]. For each DNS transaction, we record the request and response and corresponding timestamps. For each TCP connection, we record a summary that includes the initiation time, duration, IP addresses, port numbers, bytes transferred and some ancillary information. We collect data between April 1 and July 31, 2012 from a vantage point between the houses and ISP's network—which also places the monitor between the houses and the ISP's two shared DNS resolvers—as illustrated in Figure 1. This vantage point has two implications: (i) we cannot observe which device within a house is responsible for specific traffic as the residences are NAT'ed and previous work shows multiple devices per house exist in our network [22], and (ii) we cannot observe the traffic between the ISP's shared resolvers and the authoritative DNS servers (ADNS). The delay between our vantage point and both the users' end hosts and the shared resolvers is typically less than 1 msec.

Our passive monitor records 58.8M DNS resolutions. Of these, we find 41M—475K unique domain names—to have valid DNS requests and responses in the trace. We exclude 17.8M transactions for one of three basic reasons: (i) Bro glitches that cause bad timestamping[2] (180K), (ii) no response to DNS queries (8M), (iii) requests with no valid question (90K) and (iv) transactions with responses that have no resource records nor any indication of an error (9.5M). We additionally link DNS transactions with subsequent TCP connections[3]: we link a connection with the nearest preceding DNS query from the same IP address whose response includes the remote IP address used in the TCP connection. We find TCP traffic that leverages 20.4M (or 50%) of the DNS resolutions. Our dataset includes 242M TCP connections and we use the filtering techniques described in [22] to remove the invalid connections such as those never completing the handshake. Of the remaining 108M valid TCP connections, we find that 39% do not use a remote IP address found in a previous DNS response (e.g., BitTorrent connections). This leaves 66.3M (61%) TCP connections that leverage the DNS.

As a baseline, Figure 2 shows the distribution of lookup duration found in our logs on the "DNS resolution time" line. The step at less than 1 msec represents names in the shared resolver cache, while the step at 10 msec is due to responses from nearby ADNS servers. Significantly, we find that hosts do not always create TCP connections immediately after DNS responses. The "Delay before use" line in the figure shows the distribution of the time between a DNS response and the initiation of the first TCP connection based on that response. Note, 20.6M DNS resolutions do not trigger subsequent TCP activity and therefore show in the distribution as having infinite delay. We suspect these unused and delayed-use resolutions indicate DNS prefetching, which is

[2]We find this to be a general Bro issue not triggered by DNS traffic. The bad timestamps do not seem to disproportionately impact a certain kind of DNS transaction and therefore we believe there is no systematic measurement bias.

[3]We focus on TCP traffic because less than 0.1% of UDP traffic in our network uses IP addresses from DNS responses.

common in modern browsers. We find a TCP connection using a DNS response within 50 msec in 36% of the cases. Observing that hosts do not immediately use DNS responses is significant because this indicates there is slack in the process which may allow for longer DNS transactions without impacting the connections that depend on the results.

While we cannot observe the shared resolver's iterative resolution process from our vantage point, we need the timing information about each iterative step to drive our simulations. Therefore, we collect a second dataset by using *dig* to iteratively resolve the names from our passive data collection and record the durations of all iterative steps of the lookup process from a machine within the Case Connection Zone. We perform each iterative step five times and use the average transaction time in our simulations.

We lookup the 475K unique domain names in our trace in two steps. Between November 26 and December 12, 2013 we resolved the 197K names that we find in subsequent TCP traffic. Of these, 5K could not be resolved either due to ADNS server misconfiguration or because the name no longer existed. The 192K unique names we successfully resolve account for nearly 99% of the 20.4M used resolutions in our traces. Further, the successful lookups also cover 98% of the over 66M TCP connections that utilize DNS in our traces. We conduct a second set of active probes for the unused names on April 10, 2014. This second round of probing was conducted at much higher rate than the first, which caused queuing delays and hence we consider the timing information to be inaccurate. However, we never use the timing information from these lookups as they represent names without subsequent TCP connections. Rather, our objective in this probing is to obtain the time-to-live (TTL) of each record as this will impact our assessment of load (§ 5). In total we have probe data for 459K unique domain names, covering 97% of the resolutions in our trace.

Given these two sets of data, we conduct trace-driven simulations of end-host resolutions that use the natural traffic load we observe in our traces, as well as the timing and TTL information from the active probing to simulate the needed steps of the iterative process for each lookup. We derive the following variables from our traces: T_s is the time we observe a given DNS request that starts a resolution, T_f is the time we observe the corresponding response that finishes a resolution and T_c is the time we observe a TCP connection using the given DNS response to initiate a connection.[4] Further, when simulating DNS resolutions from the client, we use the DNS transaction start times from the traces, but the DNS responses will come back at T'_f—which depends on the state of the simulated client's cache and the timing of the required iterative DNS transactions. Thus, while in the traces $T_s < T_f < T_c$ holds, in our simulations, T'_f can fall at any point after T_s. We assume that processing time between a DNS response and subsequent TCP connection is minimal. Therefore, when $T'_f <= T_c$, our simulated DNS transaction does not interfere with follow-on TCP activity. However, when $T'_f > T_c$ our simulated DNS transaction actively impedes follow-on TCP activity, which would otherwise be ready to proceed at T_c, but would be forced to wait until T'_f to commence.

Finally, during one week of the passive monitoring of the network we sketch above, we collect TCP SYN/FIN/RST packet traces (June 11-17, 2012). We use *p0f* [30] to determine a signature for each TCP connection in our corpus which contains inferences about the hosts based on operating system fingerprint, MSS and IP TTL. Since each home in the network is NAT'ed these signatures allow us to gain some visibility into per-device activity within the house. While not perfect—as two like systems will have the same signature—we find 294 unique signatures across the 100 residences during the week of our trace. Unfortunately we cannot fingerprint UDP to then correlate the DNS transactions with specific TCP connections. We therefore assign DNS transactions the signature of the closest TCP connection that leverages the binding in the DNS response. We consider this the worst case since it is the soonest the binding will be needed.

4. IMPACT ON PERFORMANCE

We now turn to examining the impact of removing shared resolvers on TCP connections. Largely the impact manifests as changes in the duration of the resolution process which is a prerequisite for TCP connections. As a baseline we plot the distributions of the difference between the actual (via a shared resolver) and simulated (directly by clients) resolution times for each DNS resolution in our 4 month trace in Figure 3. A positive value indicates that the simulated resolution took longer than the natural duration (i.e., $T'_f > T_f$). The simulated direct client resolutions take less time for 19% of the resolutions, roughly the same amount of time in 26% of resolutions, and more time in 55% of resolutions. The figure shows that direct client resolution adds no more than 50 msec to the resolution process in 84% of the cases.

Also, relying on client resolution can impose delay on a TCP connection only if the DNS response comes after TCP is otherwise ready to begin (i.e., $T'_f > T_c$). First, 39% of the 108M TCP connections in our trace do not require a DNS lookup—making DNS changes moot.

Next, we concentrate on the 61% of TCP connections that do utilize DNS responses. These connections pose a problem because they may come from multiple devices within a residence. Since we cannot distinguish between the devices in our full 4 month dataset we conflate multiple devices' caches together. To cope with our suboptimal vantage point we first derive bounds for the impact experienced by these connections. Recall that the impact from the removal of a shared DNS resolver is ameliorated by two factors: (i) the delayed use of DNS resolutions and (ii) device-level DNS

[4]Given that multiple TCP connections can leverage a single DNS lookup, T_c is actually a set of values and we perform our computations on each value. However, for ease of exposition we often discuss it as a single value.

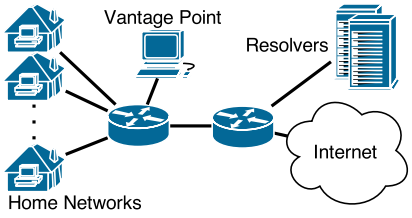

Figure 1: Monitoring vantage point

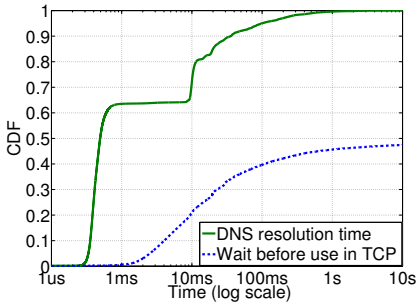

Figure 2: Dist. of DNS trans. time and time between DNS response and TCP connection.

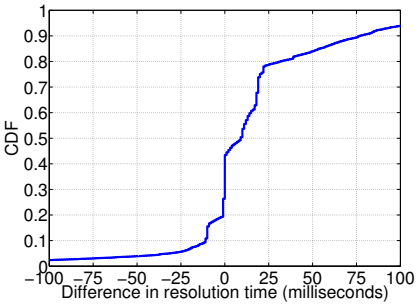

Figure 3: Dist. of diff. between simulated client resolution and resolution in the trace with a shared resolver.

caches. We first assume an optimistic case where all traffic within the residence involves a single device with a single cache. The distribution of the added delay imposed on the TCP connections is shown in the "Unified home caches" line on Figure 4. In this simulation, only 12% of TCP connections using DNS experience an added delay under direct client resolution, and 4% experience 50 ms or more of delay. Second, we assume no client DNS caching at all and present the distribution of added time for each TCP connection requiring a DNS lookup on the "No cache" line. The line shows that nearly 60% of TCP connections using DNS are not impacted because their DNS resolutions complete before the use despite any added delay direct client lookup may impose. Taking Figure 4 together with our finding that 39% of TCP connections do not rely on DNS, indicates that 75–93% of all TCP connections will feel no impact from direct client DNS resolution.

While these bounds show the impact of direct client lookup is modest at most, we aim to more accurately determine where the performance may fall. We leverage the *p0f* signatures that annotate one week of our data (see § 3) to develop a refined—even if not fully accurate—view of device-level caches and re-run our simulation for the given week. The distribution of the amount of time direct client lookup adds to TCP connections is given by the "p0f caches" line on Figure 4. The results are similar to those under the assumption of one unified cache for the entire house, which shows that reality is likely closer to the lower-cost bound.

5. IMPACT ON SCALABILITY

In addition to performance issues, our proposal for eliminating shared resolvers also has potential scalability issues. By caching records, shared resolvers shield authoritative servers from the full workload imposed by clients. Further, by performing iterative lookups on clients' behalf, the resolvers relieve the end hosts from the need to perform these steps. However, under our proposal we force each client to individually consult the authoritative infrastructure and hence we increase the work for the network, the clients and the authoritative servers.

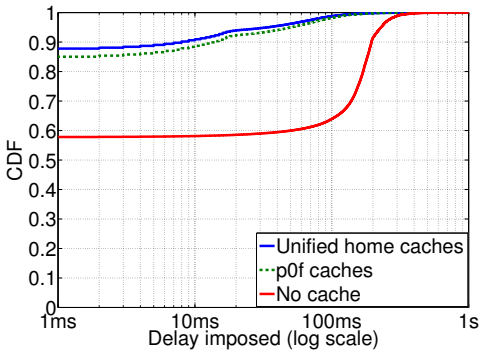

Figure 4: Delay added by client resolution to the TCP connections that require DNS.

In terms of network load, we find DNS to be less than 0.1% of the total traffic volume regardless of whether clients rely on a shared resolver or directly resolve names themselves. This indicates that network load is not a concern regardless of approach taken. Next we divide our four month dataset into ten seconds bins for each host within the network and record the number of DNS lookups for each bin. With a shared resolver, we find the per bin average, 99.9 percentile and peak to be 0.267, 80, and 438 transactions, respectively. With direct client resolutions, these numbers increase by 12%, 20% and 53%, respectively, but the absolute levels remain manageable—e.g. the 99.9 percentile remains under 100 transactions, or 10 transactions per second. We therefore conclude that neither network nor client load present a barrier to our approach.

We next assess the load increase on the authoritative infrastructure. We simulate both shared resolver and direct client resolution behavior using the workload in our trace. We use relative change in load as an approximation of the global load increase for authoritative domains. First, we find that roughly 93% of the authoritative domains do not experience an increase in the average load (over our ten second windows) while over 99% of authoritative domains show no increase in the peak load. This is due to sparse use of these domains—across both device and time—and therefore

Simulation	Load (DNS transactions / 10 secs)		
	Average	99.9^{th}	Peak
Shared resolver	0.54	17	127
Client resolution	1.84	36	145
1 week TTL	1.15	28	131
2 week TTL	0.89	25	131
3 week TTL	0.77	23	131
2 questions	0.87	33	135
10 questions	0.64	29	135
1 wk / 2 ques	0.72	31	135
3 wk / 10 ques	0.55	28	135
5-day per-p0f signature caches			
Shared resolver	0.32	4	30
Client resolution	1.60	14	62

Table 1: Load on the ".com" TLD.

low likelihood of benefiting from a shared cache. However, the domains that do experience an increase in load are already popular domains, which we may exacerbate. The "google.com" second-level domain (SLD) and the ".com" top-level domain (TLD) are the most popular domains in our trace and would experience average load increases of 2.6 and 3.4 times, respectively, without shared resolvers. Given the popularity of these zones we may have expected an increase on the order of the number of houses we monitor. The increases are two orders of magnitude less, indicating that clients' caches are crucial to dampening demand.

In addition, we note that whether or not to use a shared resolver is a decision made by clients and edge networks and not the authoritative domains. Clients could organically choose to directly contact ADNS servers without involving a shared resolver. Thus, ADNS servers cannot avoid dealing with the additional load such decisions would yield.[5] We are not interested in setting up a situation where clients and domains are at odds and therefore next investigate two ways to mitigate the additional load stemming from client resolution.

Domains like "google.com" can directly manage the load increase by dynamically tuning the TTL of their records to trade load for flexibility. However, the TLDs have less flexibility over the TTL of their records given that these records affect another party (their client SLDs). We thus focus on the ".com" TLD, which is the most popular TLD with 54% of all transactions across all TLDs in our dataset. The first two rows of Table 1 provide a baseline for the average, 99.9^{th} percentile, and peak load that the ".com" TLD experiences per ten second bin when using a shared resolver and client resolution. With client resolution, the load increases by factors of 3.4, 2.1 and 1.1 at the average, 99.9^{th} percentile and peak, respectively. Below we consider two load mitigation techniques: (i) a static increase in the records' TTL by ADNS servers and (ii) opportunistic use of extra DNS questions by the clients.

[5]While domains could take steps to incentivize use of shared resolvers through various load management techniques (e.g., preferentially dropping incoming requests not from a known shared resolver), these techniques would make accessing the domains more brittle and run contrary to the goal of most domain operators to make their domains as accessible as possible.

Increase TTLs: The first way for domains to shed load is to increase TTLs such that clients cache their records longer. This has been previously proposed as a way to improve availability of SLD mappings [20]. The cost of increasing the TTL is reduced flexibility in changing the name-to-address bindings. To see the significance of this issue, we actively request all SLD delegation records from all TLDs we observe in our traces and find 82% have TTLs of two days. Furthermore, more than 99% of resolutions in our traces are for records under SLDs that have TTLs of two days. At the same time, we find that SLD delegations do not change often. We actively resolve these records every day for 67 days and find that an average of 1.1% change within one week with linear growth over longer time periods (e.g., 2.3% and 3.4% change after two and three weeks, respectively).

To understand how increasing TTLs would impact the load on the TLD servers we vary the TTL of the ".com" delegation records in our simulation. The second group of results in Table 1 shows the impact of TTLs from 1–3 weeks. As expected, the load drops as the TTL increases. Still, while the peak load falls to within 3% of the peak load when using a shared resolver, the average load is twice the current load for the one-week TTL and 43% more for the three-week TTL. While we find that over 96% of these records do not change within three weeks, we believe it is unlikely that the community will deem a TTL of three weeks practical due to the lack of flexibility when changes are in fact needed.

Multiple DNS Questions: A second method to reduce the load on authoritative servers is for clients to piggyback DNS prefetching questions on naturally occurring lookups. In current usage, all DNS transactions involve asking one question. However, the DNS protocol supports multiple questions per request. By opportunistically appending questions for SLD records that the client is likely to use, the client can populate its cache and avoid a later specific query for the piggybacked record. For instance, if the client suffers a cache miss for the "google.com" delegation record and also notices that its cached copy of "amazon.com" will expire soon, the client could ask for both records in a single request (which is required anyway). This technique could potentially reduce the number of DNS transactions arriving at the TLDs, but not the total number of questions.[6]

To explore opportunistic prefetching of DNS records we simulate clients that track client accesses for each SLD. For each DNS resolution, we increment a counter for the corresponding SLD and at the end of each day all counters are halved (to decay historical popularity).[7] When making a necessary DNS request to a TLD, the client adds questions to the request for the most popular SLD delegation records

[6]In fact, the total number of questions could increase, as well, since clients may opportunistically request records that are never subsequently used.

[7]This is a simple algorithm that could be refined in many ways but suffices to get an initial understanding of the efficacy of the general technique.

that are either not in the cache or are close to expiring. We explore requests with 2–10 questions in our simulations.[8]

The third group of results in Table 1 shows the load client resolution places on the ".com" TLD server with 2 and 10 questions per DNS transaction. Including a second question in each request decreases the average load to less than half that of as-needed client resolution. The savings at the 99.9^{th} percentile and peak are more modest at 9% and 7%, respectively. Increasing the number of questions to 10 cuts the load of direct resolutions by almost two-third and yields the average load within 20% of using a shared resolver.

One issue with prefetching is that—unlike everything else we propose—leveraging multiple questions per DNS transaction will require changes to authoritative servers. While posing multiple questions is consistent with the DNS protocol and hence no specification changes are needed, we find that authoritative servers generally ignore all but the first question in a request. Further, answering multiple questions per DNS request naturally will increase the processing cost of completely answering the request. An attacker could leverage this feature to coax a busy TLD server becoming even busier—and ultimately overloading the server to the point of impacting normal requests. TLD servers can mitigate this in a number of ways, including prioritizing resources to clients making only a small number of requests [28] and declining to answer multiple questions per query when the load is high.

Combining Methods: Finally, the two mitigations we investigate above are not mutually exclusive and therefore we next study the efficacy of both extending the TTL and opportunistically prefetching delegation records, as we show in the fourth group of results in Table 1. The first combination involves setting the TTL to one week and using two questions per DNS request. In this case we increase the average load by one-third, the 99.9^{th} percentile load by 82% and the peak load by 6% compared with using a shared cache. On the other end of our parameter space, a three-week TTL with ten questions per transaction produces an average load that is nearly the same as the load with a shared resolver. However, the 99.9^{th} percentile and peak load show a 65% and 6% increase, respectively. Note that extra questions can actually increase peak rates due to changing the timing of the query flow and sending unnecessary requests.

The above analysis assumes a per-house DNS cache as our long-term data does not provide device-level visibility. We now turn to our p0f-augmented dataset to gain an initial understanding of load when residences are more appropriately divided. Since 99% of SLD records have TTLs of two days, we simulate only the last five days of the week-long dataset—leaving the first two days to warm the cache.[9] The last two lines in Table 1 show the load increases when we simulate with the p0f signatures. We find increase factors of 5, 3.5, and 2.1 respectively due to direct client resolution. While this trace is too short to evaluate our mitigation techniques, one could expect similar relative effect to our results with per-house caches.

[8] Answers for 10 questions generally fit within a 1500B packet.

[9] We could neglect cache warming in the previous section because (i) the DNS queries are dominated by resolutions of names below SLDs while now we are focusing on the SLD queries and (ii) the longevity of the data makes any warm up period insignificant.

6. ADDITIONAL CONSIDERATIONS

We now discuss two additional calculations which we cannot directly quantify, but are part of the tradeoff of eliding shared resolution infrastructure.

Privacy Concerns: Direct client resolution can reduce users' privacy. When using a shared resolver, clients gain a measure of anonymity as outside their edge network lookups cannot be directly attributed to a specific client. Therefore, a downside to removing the shared resolver is the loss of this measure of privacy. This may be viewed by some users as too revealing to eavesdroppers or simply ADNS servers that log individuals' activities. On the other hand, as we discuss in § 1 and § 2 the ability to directly locate clients is useful for CDNs. As a comment on this tussle we note that many users are willing to use open shared resolvers (e.g., Google DNS) and are therefore comfortable with directly attributable DNS requests arriving at a large third-party network.

Policy Issues: Additionally, shared resolvers also allow network operators to implement edge network policy (e.g., not allowing resolution of some site a company does not wish employees to use while working). Using our approach of direct client resolution removes the DNS resolver as a control point in the network. However, our proposal does not preclude the use of a shared resolver in such cases. We simply view this as akin to web downloads where the expectation is that clients and web servers directly communicate, but in some cases a proxy is placed in the path to implement policy.

7. CONCLUSION

Traditionally, our community's response to security problems is to harden a protocol or its implementation. In this paper we take an alternate approach to DNS security, suggesting a different factorization of the work that eliminates shared DNS resolvers. The benefit of this approach is to reduce DNS' attack surface. Through an initial study of a single network, we show that while there are costs, those costs are modest and manageable. For instance, less than 10% of TCP connections will be delayed by direct client resolution. Further, the 99.9^{th} percentile load does not increase at all for 90% of the ADNS servers and by a factor of two at the .com TLD server–with no effort to mitigate the additional load. There are policy and privacy concerns, as well, but we believe this initial investigation shows that leaning on clients to do their own lookups deserves serious consideration. Further, we believe this effort illustrates that revisiting the fundamental way we arrange networks in the context of modern network realities may well be useful across other components of the system, as well.

8. REFERENCES

[1] Case Connection Zone. http://www.caseconnectionzone.org/.

[2] Open Resolver Project. http://openresolverproject.org/.

[3] The Bro Network Security Monitor. https://www.bro.org/.

[4] H. A. Alzoubi, M. Rabinovich, and O. Spatscheck. The Anatomy of LDNS Clusters: Findings and Implications for Web Content Delivery. In *International Conference on World Wide Web*, 2013.

[5] M. Antonakakis, D. Dagon, X. Luo, R. Perdisci, W. Lee, and J. Bellmor. A Centralized Monitoring Infrastructure for Improving DNS Security. In *Recent Advances in Intrusion Detection*, 2010.

[6] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. *RFC 4033*, 2005.

[7] S. Ariyapperuma and C. Mitchell. Security Vulnerabilities in DNS and DNSSEC. In *IEEE International Conference on Availability, Reliability and Security*, 2007.

[8] D. Bernstein. Introduction to DNSCurve. http://dnscurve.org/, 2008.

[9] C. Contavalli, W. van der Gaast, S. Leach, and D. Rodden. Client IP Information in DNS Requests. *IETF draft, work in progress*, 2010.

[10] D. Dagon, M. Antonakakis, K. Day, X. Luo, C. Lee, and W. Lee. Recursive dns architectures and vulnerability implications. In *Network and Distributed System Security Symposium*, 2009.

[11] K. Fujiwara. Number of Possible DNSSEC Validators Seen at jp. In *DNS-OARC Workshop*, 2012.

[12] A. Gerber and R. Doverspike. Traffic Types and Growth in Backbone Networks. In *Optical Fiber Communication Conference*, 2011.

[13] O. Gudmundsson and S. Crocker. Observing DNSSEC Validation in the Wild. In *Workshop on Securing and Trusting Internet Names*, 2011.

[14] A. Herzberg and H. Shulman. Fragmentation Considered Poisonous, or: One-domain-to-rule-them-all.org. In *IEEE Communications and Network Security*, 2013.

[15] C. Huang, I. Batanov, and J. Li. A Practical Solution to the Client-LDNS Mismatch Problem. *ACM SIGCOMM Computer Communication Review*, 42(2), 2012.

[16] C. Huang, D. A. Maltz, J. Li, and A. Greenberg. Public DNS system and Global Traffic Management. In *IEEE International Conference on Computer Communications*, 2011.

[17] D. Kaminsky. Black Ops 2008: It's the End of the Cache As We Know It. *Black Hat USA*, 2008.

[18] Z. M. Mao, C. D. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck, and J. Wang. A Precise and Efficient Evaluation of the Proximity Between Web Clients and Their Local DNS Servers. In *USENIX Annual Technical Conference, General Track*, 2002.

[19] J. S. Otto, M. A. Sánchez, J. P. Rula, and F. E. Bustamante. Content Delivery and the Natural Evolution of DNS: Remote DNS Trends, Performance Issues and Alternative Solutions. In *ACM Internet Measurement Conference*, 2012.

[20] V. Pappas and E. Osterweil. Improving DNS service availability by using long TTL values. IETF Draft. http://tools.ietf.org/id/draft-pappas-dnsop-long-ttl-04.txt, 2012.

[21] H. Qian, E. Miller, W. Zhang, M. Rabinovich, and C. E. Wills. Agility in Virtualized Utility Computing. In *IEEE Workshop on Virtualization Technology in Distributed Computing*, 2007.

[22] M. Sargent, B. Stack, T. Dooner, and M. Allman. A First Look at 1 Gbps Fiber-To-The-Home Traffic (TR-12-009). Technical report, 2012.

[23] K. Schomp, T. Callahan, M. Rabinovich, and M. Allman. On Measuring the Client-Side DNS Infrastructure. In *ACM Internet Measurement Conference*, 2013.

[24] K. Schomp, T. Callahan, M. Rabinovich, and M. Allman. Assessing DNS Vulnerability to Record Injection. In *Passive and Active Measurement Conference*, 2014.

[25] C. Schuba. *Addressing Weaknesses in the Domain Name System Protocol*. PhD thesis, Purdue University, 1993.

[26] A. Shaikh, R. Tewari, and M. Agrawal. On the Effectiveness of DNS-based Server Selection. In *IEEE International Conference on Computer Communications*, 2001.

[27] S. Tzur-David, K. Lashchiver, D. Dolev, and T. Anker. Delay Fast Packets (DFP): Prevention of DNS Cache Poisoning. *Security and Privacy in Communication Networks*, 2012.

[28] P. Vixie and V. Schryver. DNS Response Rate Limiting (DNS RRL). Technical Report ISC-TN-2012-1, Internet Systems Consortium, Apr. 2012.

[29] L. Yuan, K. Kant, P. Mohapatra, and C. Chuah. DoX: A Peer-to-Peer Antidote for DNS Cache Poisoning Attacks. In *IEEE International Conference on Communications*, 2006.

[30] M. Zalewski. p0f: Passive OS Fingerprinting tool. http://lcamtuf.coredump.cx/p0f.shtml.