Quality of Name Resolution in the Domain Name System

Casey Deccio
Sandia National Laboratories
ctdecci@sandia.gov

Chao-Chih Chen
University of California, Davis
cchchen@ucdavis.edu

Jeff Sedayao
Intel Corporation
jeff.sedayao@intel.com

Krishna Kant
National Science Foundation
krishna.kant@intel.com

Prasant Mohapatra
University of California, Davis
pmohapatra@ucdavis.edu

***Abstract*—The Domain Name System (DNS) is integral to today's Internet. Name resolution for a domain is often dependent on servers well outside the control of the domain's owner. In this paper we propose a formal model for analyzing the name dependencies inherent in DNS, based on protocol specification and actual implementations. We derive metrics to quantify the extent to which domain names affect other domain names. It is found that under certain conditions, the name resolution for over one-half of the queries exhibits influence of domains not expressly configured by administrators. This result serves to quantify the degree of vulnerability of DNS due to dependencies that administrators are unaware of. The model presented in the paper also shows that the trusted computing base of a domain name is much smaller than previously thought. The model also shows that with caching of NS target addresses, the trusted computing base expands greatly, thereby making the DNS infrastructure more vulnerable.**

I. Introduction

Nearly all of today's Internet applications rely on the Domain Name System (DNS) for proper function. Its major role of name-to-address translation is especially key to users, who are largely accustomed to recognizing Internet "locations" by human-friendly words, titles, and abbreviations, rather than numeric IP address. DNS is also necessary for email delivery, service discovery, and host identification. Since DNS details are often left to the client resolver and abstracted at the application level, its integrity and security are critical. While temporary failures due to misconfiguration may cause inconvenience, targeted attack by malicious parties could be much less discernible, and the repercussions more severe. Malicious parties seek to taint DNS responses, redirecting applications to servers within their control, where sensitive information can be stolen.

While the concept of name resolution is relatively simple, the overall system is complex and its effects far-reaching. Name resolution for a domain is often dependent on servers well outside the control of the domain's owner and managed by third parties. A network of inter-organizational relationships overlays the DNS infrastructure, and configurations that create a dependency on peer organizations are in turn affected by the security and accuracy of name spaces linked through this network. An understanding of a domain's context in the entire system is integral for reliability, integrity, and security of DNS.

In this work we analyze the network of inter-organization dependencies comprising DNS. We derive a model to represent this network, based on DNS behavior in specification and implementation. Metrics are derived from the model to analyze the quality of name resolution for a domain name, based on the other names that affect its resolution. A large sample of recent DNS name dependency data was collected and analyzed based on these metrics. The results show how configurable caching behaviors of name servers affect the size of the namespace that influences a domain. The amount of influence coming from namespace not explicitly configured by DNS administrators is also analyzed.

The primary contributions presented in this research are:

- A formal model for analysis of DNS name dependencies, based on specification and actual implementations
- Metrics for quantifying the influence domain names have on other domain names

Previous work in this area is described in Section II. In Section III we introduce the concept of DNS name dependencies and review pertinent fundamentals of name resolution. In Section IV we formalize a graph model for analyzing DNS name dependencies and derive methods for quantifying influence. We describe methodologies employed for data collection and an evaluation of the observed quality of name resolution in Section V. Future work is discussed in Section VI, and we conclude in Section VII.

II. Previous Work

The concept of name dependencies was most recently analyzed by Ramasubramanian, et al. [1]. Their research identifies a set of name servers that affect the resolution of a given domain name and which collectively comprise its *trusted computing base* (TCB).

We build on the work presented in [1], performing further examination of several areas to create a model of name

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

dependencies in DNS. The metric largely referred to in [1] is the number of distinct name servers in the TCB—identified both by IP address and name. In practice, redundant servers are typically deployed by an organization to provide diversity and high availability. In such cases, it is likely that versions and configurations are consistent across the servers maintained by a single organization. In this research we examine diversity of the namespace in the TCB. We also consider the role of glue records and caching.

Pappas, et al. [2] surveyed the DNS infrastructure for configuration errors that negatively impact DNS robustness. The authors examined subtle misconfigurations that could bring about behaviors such as diminished server redundancy, lame delegation, and cyclic dependency. This research presents a model that may be used to methodically identify DNS configuration errors and security vulnerabilities.

Other behavioral studies for DNS robustness and security have been performed in [3], [4]. Design of next-generation DNS systems using peer-to-peer overlay networks have been suggested in [5]–[7] both for security and performance enhancement.

III. Name Dependencies in DNS

Resolution of a domain name is often dependent on resolution of intermediate names, which in turn depend on others. Three specific components in the DNS protocol lead to such name dependencies:

- *Parent zones*: Because name resolution is performed by traversing the name hierarchy from the top down, a name is always dependent on its parent zone.
- *NS targets*: The NS (name server) resource record (RR) type uses names as targets, rather than addresses, so a resolver must resolve the names of NS targets before it can query the corresponding authoritative servers.
- *Aliases*: If a name resolves to an alias (i.e., CNAME RR type), then to obtain an address, the alias target must also be resolved.

This research focuses on the diversity of the namespace rather than the number of servers. Domain name u *depends on* domain name v if resolution of v may *influence* resolution of u. Dependence is transitive: if u depends on v and v depends on w, then u depends on w. The term *trusted computing base*, as used in this work, refers to *zones*, which typically correspond to governing organizations or configurations.

The raw size of the TCB is not enough to measure the effects of third-party namespace on resolution of a domain name, as in [1]. In some cases policy or preference may dictate whether or not the existence of a zone is acceptable in the TCB (e.g., a government zone that prohibits zones operated by foreign governments in its TCB). However, a thorough analysis will show that not all names have equal influence. In this research we introduce *level of influence* $I_u(v)$ as a quantitative measure of v's influence on u. Level of influence is formally defined in Section IV.

Influence is categorized into two classes: *active* and *passive* influence. If domain name u is actively influenced by domain name v, then with some non-zero probability resolution of v will be *required for* resolution of name u. If domain name u is passively influenced by domain name v, then although v may not be required for resolution of u, resolution of v may *affect* resolution of u with some probability—determined by cache contents of resolvers and authoritative servers at the time of query. The conditions for active and passive influence are described later in this section.

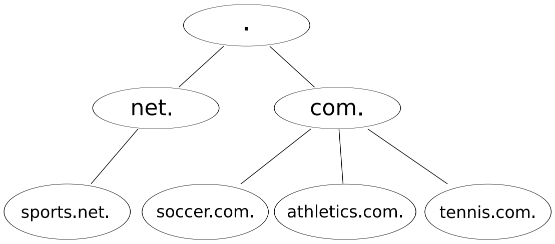

Fig. 1. The zone hierarchy for the the zone data shown in Table I.

Some discussion of specific aspects of DNS behavior is required to properly create a well-formed dependency model. The role of glue and additional records in delegation, the selection of authoritative name servers, and the trust ranking of data are discussed in the remainder of this section. Table I is provided as a reference for this discussion. It contains the data for several fictitious zones, shown hierarchically in Fig. 1. The behaviors of two popular DNS server implementations are also referenced: the Berkeley Internet Name Daemon version 9.5 (BIND) [8] and djbdns [9].

A. Glue and additional records

When a query for a name in zone z reaches name server s, which is authoritative for $Parent(z)$, s returns the set of NS RRs corresponding to the name servers authoritative for z, as a “referral”. The set of NS target names for this set is denoted NS_z. Addresses of the NS targets in NS_z are required for the resolver to subsequently query the servers. If any NS targets are subdomains of z, then s must also include *glue records* for those targets in the response's *additional* section to “bootstrap” the resolution process, so there isn't a cyclic dependency between a zone and its descendants [10]. The glue records are generally A (address) RRs corresponding to the target names of the NS RRs for z but maintained in the $Parent(z)$ zone. The NS RRs and associated glue records for *tennis.com* are found on lines 7–11 of the *com* zone in Table I.

Server s may also send pertinent non-glue A RRs in the additional section of its response to expedite the resolution process for the resolver, if the records are available locally (e.g., if s is also authoritative for the zones to which the targets belong or if s has an answer cached from an authoritative response) [10]. However, any such RRs included in the response for which $Parent(z)$ is not a superdomain are considered *out-of-bailiwick* (i.e., outside its jurisdiction). Thus resolver implementations should independently obtain an authoritative answer for the out-of-bailiwick target names before querying such servers.

$ORIGIN soccer.com.

	Name	Type	Value
1	soccer.com.	NS	ball.soccer.com.
2	soccer.com.	NS	racket.tennis.com.
3	soccer.com.	NS	ns1.sports.net.
4	ball.soccer.com.	A	10.0.1.1
5	www.soccer.com.	CNAME	www.tennis.com.

$ORIGIN tennis.com.

	Name	Type	Value
1	tennis.com.	NS	ns1.tennis.com.
2	tennis.com.	NS	ball.soccer.com.
3	tennis.com.	NS	ns1.sports.net.
4	ns1.tennis.com.	A	10.0.2.1
5	www.tennis.com.	A	10.0.2.2
6	racket	A	10.0.2.3

$ORIGIN athletics.com.

	Name	Type	Value
1	athletics.com.	NS	ns1.athletics.com.
2	ns1.athletics.com.	A	10.0.6.1

$ORIGIN com.

	Name	Type	Value
1	com.	NS	ns1.com.
2	ns1.com.	A	10.0.3.1
3	athletics.com.	NS	ns1.athletics.com.
4	soccer.com.	NS	ball.soccer.com.
5	soccer.com.	NS	racket.tennis.com.
6	soccer.com.	NS	ns1.sports.net.
7	tennis.com.	NS	ball.soccer.com.
8	tennis.com.	NS	ns1.tennis.com.
9	tennis.com.	NS	ns1.sports.net.
10	ball.soccer.com.	A	10.0.1.1
11	ns1.tennis.com.	A	10.0.2.1
12	ns1.athletics.com.	A	10.0.6.1

$ORIGIN sports.net.

	Name	Type	Value
1	sports.net.	NS	ns1.sports.net.
2	sports.net.	NS	ns1.athletics.com.
3	ns1.sports.net.	A	10.0.4.1

$ORIGIN net.

	Name	Type	Value
1	net.	NS	ns1.net.
2	ns1.net.	A	10.0.5.1
3	sports.net.	NS	ns1.sports.net.
4	sports.net.	NS	ns1.athletics.com.
5	ns1.sports.net.	A	10.0.4.1

TABLE I
THE ZONE DATA FROM SEVERAL FICTITIOUS ZONES, WHOSE HIERARCHY IS SHOWN IN FIG. 1. ANY COINCIDENCE WITH ACTUAL ZONES OF THE SAME NAME IS UNINTENTIONAL.

The resolver is responsible for resolving any names from NS_z which are out-of-bailiwick or not included in the additional section of a response from s. Such induced queries indicate active influence of the resolved names on z, since it is directly dependent on their resolution.

B. Name server selection

RFC 1035 [11] describes the process by which servers are selected by a resolver for querying a zone z as part of the resolution process. The resolver begins with the list of all server names NS_z. The addresses known by the resolver for target names in NS_z initially populate the set of corresponding addresses, and it initiates requests in parallel to acquire addresses for the remnant. The resolver also associates historical statistics, such as response time and success rate, to each address. The complete set of addresses corresponding to NS target names in NS_z is denoted NSA_z. A resolver will avoid using an address from NSA_z twice until all addresses have been tried at least once. After that, it prefers the server with the best performance record, thus fine-tuning the performance for lookups of z [11].

This behavior is not consistent across implementations. The djbdns name server selects a server from NSA_z uniformly at random. However, a resolver using BIND, which follows the performance-based selection guideline, will gravitate toward preferring a single server or set of servers in NSA_z. We make the assumption that requests for subdomains of z arrive from resolvers in diverse network and geographic locations, such that the preference to servers in NSA_z is distributed uniformly among such resolvers. This leads to an equal probability that any server in NSA_z receives a query for subdomains of z.

C. Trust ranking

RFC 2181 [12] outlines a relative ranking of trustworthiness of data for name servers to consider as part of operation. Among the total ranking are the following (in decreasing order of trustworthiness):

- Data from a zone for which the server is authoritative, other than glue data
- The authoritative data include in the *answer* section of an authoritative reply
- The data in the *authority* section of an authoritative reply
- Glue from a zone for which the server is authoritative
- Data from *additional* section of a response

This trust ranking has effects on name dependencies with regard to both the resolver and the authoritative server. The authoritative set of NS target names for z, NS_z, may differ from those stored in $Parent(z)$, NS'_z. While a resolver must initially use the set NS'_z provided by a server authoritative for $Parent(z)$, once it receives an answer in z from a server authoritative for z, it will use the target names in NS_z (provided in the authority section) in preference to those in NS'_z. This behavior is consistent with both BIND and djbdns. Server selection therefore depends not only on the NS targets in NS_z but also on the probability that the set of NS RRs for z has been cached by the resolver—either from the answer or authority section of an authoritative reply. This probability is denoted $P_{NS}(z)$.

If authoritative server $s \in NSA_{Parent(z)}$ has caching functionality enabled and has stored the A RR for an NS target $v \in NS_z$ from the answer section of an authoritative response, according to the RFC, it will trust this RR more than a glue in its own configuration. $P_s(v)$ denotes the probability that s has in cache and provides such authoritative data for v. This behavior is configurable in BIND, but it is enabled by default.

If resolver c has cached the address for $v \in NS_z$, as the result of an answer from an authoritative source from a prior transaction, then c deems the cached data more trustworthy than any data received in the additional section of a response. Thus, it will use the previously cached data in preference to data—whether from glue or s's cache—returned in the additional section by $s \in NSA_{Parent(z)}$. $P_c(v)$ denotes the probability that c has and uses such authoritative data for v in its cache. BIND adheres strictly to this, as it will direct queries to an address received by a more "trustworthy" source, even across a large delay, over a server returned in an additional section within close proximity—unless the authoritative data is an alias (i.e., a CNAME RR). The djbdns name server treats the A RRs with equal precedence, but will always use an authoritative CNAME RR over an additional A RR of the same name.

The combination of $P_s(v)$ and $P_c(v)$ determine the likelihood that either s or c has and uses a cached authoritative answer for v. Since the probabilities are independent of one another, the combined probability $P_{\{s,c\}}(v)$ is calculated:

$$P_{\{s,c\}}(v) = P_s(v) \vee P_c(v) = 1 - (1 - P_s(v))(1 - P_c(v))$$

Suppose $v \in NS_z$ is a subdomain of $Parent(z)$, $Parent(v) \neq z$, and $Parent(z)$ is properly configured with a glue record for v. If $s \in NSA_{Parent(z)}$ or resolver c has previously obtained the address for v through the resolution process, then z is affected by v and its name dependencies. Passive influence of v on z occurs when $P_{\{s,c\}}(v) > 0$.

IV. DNS DEPENDENCY MODEL

Name dependencies are quantified using level of influence, which is the probability that one name will be utilized for resolving another. Thus $I_u(v) = [0, 1]$—v's level of influence on u—represents the probability that domain v will be used in the resolution process for u. Dependencies may be reciprocated (i.e., $I_u(v) > 0$ and $I_v(u) > 0$), though the level of influence in each direction may differ. The level of influence of a domain does not necessarily indicate the trustworthiness of that domain. It will be shown that dependencies of a domain propagate along dependency paths to domains outside of its control. In the remainder of this section, a model is defined for analysis and quantification of DNS name dependencies.

A. Name dependency graph

To derive the values for influence of domain name d a directed, connected graph, $G_d = (V_d, A_d)$, is used to model name dependencies. The graph G_d contains a single sink, r, which is the root zone. Each node in the graph $v \in V_d$ represents a domain name, and each edge, $(u, v) \in A_d$, signifies that u is directly dependent on v for proper resolution of itself and any descendant names. Each edge, $(u, v) \in A_d$, carries a weight, $w(u, v)$, indicative of the probability that it will be followed for resolving u. A name dependency graph for domain name *www.soccer.com* is shown in Fig. 2, built from the data in Table I.

Edges are placed on the graph from each domain name $u, u \neq r$ to its parent $Parent(u)$ with $w(u, Parent(u)) = 1$; a domain name is always dependent on its parent. If resolution of domain name u yields a CNAME RR, then an edge is placed between u and its target name, v, with $w(u, v) = 1$; the resolution of an alias is always dependent on the resolution of its target. Such edges in Fig. 2 are those between *www.soccer.com* and its parent, *soccer.com*, and between *www.soccer.com* and its alias, *www.tennis.com*.

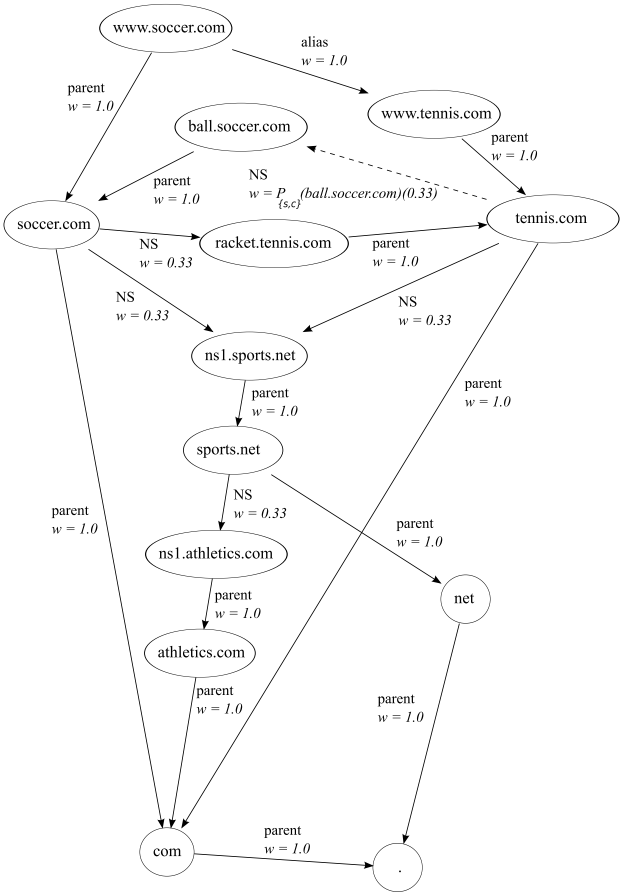

Fig. 2. The dependency graph for the domain name *www.soccer.com*, derived from the zone data in Table I. The solid lines represent active influence, and the dashed lines represent passive influence.

Placement of edges and weights corresponding to NS target dependencies is somewhat involved and draws from the discussion in Section III. The considerations are summarized in Table III.

The consideration of NS targets for use in resolving z is subject to $P_{NS}(z)$—the probability that the set of NS RRs for z are cached from authoritative source at the resolver. The probability that NS target v is considered for use is:

$$\rho(v) = P_{NS}(z)P(v \in NS_z) + (1 - P_{NS}(z))P(v \in NS'_z)$$

This only affects the weight of an NS target edge if $NS_z \neq NS'_z$. For simplicity we assume that $NS_z = NS'_z$ unless specified otherwise.

Let S_v represent the set of addresses for $v \in NS_z$. If NS target $v \in NS_z$ is not a subdomain of $Parent(z)$, edge (z, v) is added to G_d with $w(z, v) = \frac{|S_v|}{|NSA_z|}$—the fraction of total authoritative server addresses for z corresponding to

Term	Definition
r	The root domain name "."
$I_u(v)$	The measure of domain name v's influence on domain name u
$I_u(D)$	The aggregate influence of names in set D on domain name u
$Parent(d)$	The nearest ancestor zone of domain name d
$Cname(d)$	The alias target of domain name alias d
NS_z, NS'_z	The set of NS target names authoritative for zone z, as configured in zone z itself and zone $Parent(z)$, respectively
NSA_z	The set of addresses corresponding to the names in NS_z
NSA_z^y	The set of servers authoritative for zone z but not for zone y
$P_{NS}(z)$	The probability that the resolver has the authoritative set of NS RRs for z cached from an authoritative source
$\rho(v)$	The probability that $v \in NS_z$ will be considered for resolution of z, based on $P_{NS}(z)$
$P_{\{s,c\}}(v)$	The probability that either s or c has in cache and uses target name v from an authoritative source
$G_d = (V_d, A_d)$	Name dependency graph for domain name d
$G'_d = (V'_d, A'_d)$	Active influence dependency graph for domain name d
$w(u,v)$	The weight of edge (u,v) in A_d—the probability that v will be used in resolving u
S_u	The set of addresses corresponding to name server u
Z_d	The set of influential zones in V_d
U_d	The set of non-trivial zones in Z_d
U'_d	The set of first-order dependencies in U_d

TABLE II
TERMS AND DEFINITIONS USED IN THIS RESEARCH.

v subdomain of $Parent(z)$	Glue exists	$Parent(v) = z$	$w(z,v)$	Example (Table I and Fig. 2)
no			$\frac{\lvert S_v\rvert}{\lvert NSA_z\rvert}$	*soccer.com* → *ns1.sports.net*
yes	no		$\frac{\lvert S_v\rvert}{\lvert NSA_z\rvert}$	*soccer.com* → *racket.tennis.com*
yes	yes	no	$\frac{\lvert S_v\rvert P_{\{s,c\}}(v)}{\lvert NSA_z\rvert}$	*tennis.com* → *ball.soccer.com*
yes	yes	yes	0	*soccer.com* → *ball.soccer.com*

TABLE III
RULES FOR DETERMINING WHETHER OR NOT AND WITH WHAT WEIGHT $w(z,v)$ A EDGE IS PLACED BETWEEN A ZONE z AND AN NS TARGET $v \in NSA_z$.

v. Resolution of v is required for (i.e., actively influences) resolution of z.

If target name $v \in NS_z$ is a subdomain of z, the $Parent(z)$ zone should include a glue record for v. If no glue record exists for v in the $Parent(z)$ zone, then resolution of v is required for (i.e., actively influences) resolution of z, and an edge (z,v) is added to G_d with $w(z,v) = \frac{|S_v|}{|NSA_z|}$.

If a glue record for v exists in bailiwick, then resolution of v is not required for resolving z because the resolver will use the address provided in glue from the $Parent(z)$ authoritative server. When $Parent(v) = z$, there is no edge (z,v) in G_d; all servers authoritative for z have the authoritative data for v. However, when $Parent(v) \neq z$ an edge (z,v) added with $w(z,v) = \frac{|S_v| P_{\{s,c\}}(v)}{|NSA_z|}$; the name v passively influences z, dependent on the probability that either the resolver or the authoritative server has the address for v cached from an authoritative source.

The active influence dependency graph, G'_d, of domain name d is the subgraph of G_d produced when $P_{\{s,c\}}(v) = 0, \forall v \in V_d$ and nodes with only zero-weight in-edges are removed from the graph. The active influence dependency graph for *www.soccer.com* would be created by eliminating the *ball.soccer.com* node in Fig. 2.

B. Level of influence

An analysis of the *dependency paths* in G_d is necessary to determine the level of influence of the domain names $v \in V_d$ on d. The dependency paths in G_d are modeled by performing a depth-first traversal of G_d, beginning with d. This depth-first traversal produces the exhaustive set of intermediate paths of name dependencies for resolving d. The path $d, n_1, \ldots, n_i$ represents a single dependency path.

For a given domain name $u \in V_d$, resolution of u often requires following multiple branches at an intermediate node, depending on the relationship between the dependency types. For NS target dependencies of u at most one address from NSA_u is followed (assuming no server failure). However, alias and parent dependencies exist independently of the NS target dependencies. For example, when resolving names in *tennis.com* using the zone data from Table I, either *ns1.tennis.com*, *ball.soccer.com*, or *ns1.sports.net* will be selected, each with equal probability. However, its resolution remains entirely dependent on its parent, *com*, regardless of which server in $NSA_{\textit{tennis.com}}$ is selected for query.

The level of influence $I_u(v)$ is calculated by determining the probability that domain name v is utilized for resolving u, as shown in the `Influence` algorithm (Algorithm 1). The probabilities of encountering v in the dependency paths beginning with each of u's direct dependencies must first be recursively calculated and aggregated. The probability of encountering v in a path beginning with edge $(u,j) \in A_d$ is calculated by multiplying the probability, $w(u,j)$, of following edge (u,j) by the probability of encountering v in the path

Algorithm 1 `Influence`(u, v, p, H)

```
Input: Domain names $u, v \in V_d$
Input: Current path probability $p$
Input: Set of names in preceding path $H$
Output: Influence of $v$ on $u$, given $p$
 1: if $u = v$ then /* $u$ is the name being sought */
 2:     return $p$
 3: else if $u = r$ then /* root name $r$ reached before $v$ */
 4:     return 0
 5: else if $u \in H$ then /* a cycle is detected */
 6:     return 0
 7: else
 8:     /* Add $u$ to the history of names visited */
 9:     $H \leftarrow H \bigcup \{u\}$
10:     /* Influence of $v$ on $u$ through NS targets */
11:     $P_{NS} \leftarrow$
            $\sum_{c \in NS_u}$ Influence$(c, v, w(u, c)p, H)$
12:     /* Influence of $v$ on $u$ through $Parent(u)$ */
13:     $P_P \leftarrow$ Influence$(Parent(u), v, p, H)$
14:     /* Influence of $v$ on $u$ through an alias */
15:     if $u$ is an alias then
16:         $P_A \leftarrow$ Influence$(Cname(u), v, p, H)$
17:     else
18:         $P_A \leftarrow 0$
19:     end if
20:     /* Aggregate influence of all name dependencies */
21:     return $1 - (1 - P_{NS})(1 - P_P)(1 - P_A)$
22: end if
```

beginning with j (lines 11, 13, and 16).

The formula used for aggregating the probability of encountering v in paths beginning with each of u's direct dependencies is as follows. First the probability of encountering v through any NS-type dependencies is determined by calculating the sum of encountering it in each of the NS-type dependency edges—the probabilities are dependent on one another (line 11). This probability is then combined independently with the probability of encountering v in paths beginning with any alias- or parent-type dependencies (line 21).

The probability of reaching node n_i through the dependency path $d, n_1, \ldots, n_i$ (line 1) is the described using the following recurrence:

$$P(d, n_1, \ldots, n_i) = \begin{cases} w(d, n_1) & \text{if } i = 1 \\ P(d, n_1, \ldots, n_{i-1}) w(n_{i-1}, n_i) & \text{otherwise} \end{cases}$$

Calling `Influence`$(\textit{www.soccer.com}, \textit{sports.net}, 1.0, \emptyset)$ yields $0.62 + 0.06 P_{\{s,c\}}(\textit{ball.soccer.com})$.

C. Graph properties

Since finding the level of influence of a single name on d requires finding all paths between d and r, the time complexity for this operation is exponential. However, often it may suffice to simply to know the set of names influencing d, or other representative properties of G_d. This section describes some properties from which metrics can be derived for quantifying the TCB of d and measuring the extent to which its resolution is affected by third parties.

Algorithm 2 `NonTrivialZones`(d)

```
Input: Domain name $d$
Output: Set of non-trivial zones in $V_d$
$D \leftarrow \{Parent(d)\}$
for all $(u, v) \in A_d$ do
    if $(u, v)$ is an NS target or alias dependency then
        $D \leftarrow D \bigcup \{Parent(v)\}$
    end if
end for
return $D$
```

1) Influential zones: The set of influential zones $Z_d \subseteq V_d$ is a measure of the TCB of d. Although a single organization may maintain several zones in Z_d, it is generally representative of the diversity of organizations that influence resolution of d. In Fig. 2 $Z_{\textit{www.soccer.com}} = \{\textit{soccer.com}, \textit{tennis.com}, \textit{sports.net}, \textit{athletics.com}, \textit{com}, \textit{net}, .\}$.

2) Non-trivial zones: Non-trivial zones are the result of explicitly configured inter-zone dependencies. Included in this set are the parent zones of any NS or alias targets in A_d: $U \subseteq Z_d$. A non-trivial zone *foo.bar.com* that influences d may contribute up to four zones to Z_d. However, if no in-edges resulting from alias- or NS-type dependencies exist for any of its ancestor zones (*bar.com*, *com*, and "*.*"), then they exist in Z_d only because *foo.bar.com* is explicitly configured as a dependent zone and are thus trivial. Algorithm 2 identifies non-trivial zones by iterating the set of edges A_d and adding the parent zones of NS and alias targets. In Fig. 2 $U_{\textit{www.soccer.com}} = \{\textit{soccer.com}, \textit{tennis.com}, \textit{sports.net}, \textit{athletics.com}\}$.

3) First-order dependencies: A subset of non-trivial zones $U'_d \subseteq U_d$ are explicitly configured by d (or $Parent(d)$, if d is not a zone) and comprise *first-order* dependencies. U'_d also includes the non-trivial zones in the ancestry of each explicitly configured zone. Algorithm 3 finds all the alias (lines 6–8) and NS target (line 12) dependencies for a name d and then includes the parent zone for each target (line 16) and each non-trivial zone in its ancestry (lines 17–22). In Fig. 2 $U'_{\textit{www.soccer.com}} = \{\textit{soccer.com}, \textit{tennis.com}, \textit{sports.net}\}$.

4) Third-party influence: The computational complexity of calculating level of influence for all $u \in V_d$ renders it infeasible. A more useful and computationally feasible metric is determining how much domain d is influenced by names outside of U'_d, or its *third-party influence* (TPI): $I_d(U_d - U'_d)$. To do this, two helper algorithms are utilized: the `ControlledAlias` algorithm (Algorithm 4) analyzes a name to determine whether or not it aliases (directly or indirectly) another name outside of the set of U'_d; and the `ThirdPartyInfluence1` algorithm (Algorithm 5) determines the probability that resolution of u will utilize a name outside the set of U'_d. The latter is computed by aggregating the probabilities that u will utilize a name outside of U'_d from aliasing (lines 3–5) or from NS target dependencies in its

Algorithm 3 FirstOrderDeps(d)

```
Input: Domain name $d$
Output: Set of first-order dependencies in $V_d$
 1: $N \leftarrow$ NonTrivialZones($d$)
 2: /* $M$ is the set of explicitly configured names for $d$ */
 3: $M \leftarrow \{d\}$
 4: if $d$ is not a zone then
 5:    /* If $d$ is an alias, then add $Cname(d)$ to $M$ */
 6:    if $d$ is an alias then
 7:       $M \leftarrow M \bigcup \{Cname(d)\}$
 8:    end if
 9:    $d \leftarrow Parent(d)$
10: end if
11: /* Add NS target edges for zone $d$ to $M$ */
12: $M \leftarrow M \bigcup \{u \in V_d | \exists (d, u) \in A_d$, NS target dep.$\}$
13: $D \leftarrow \{d\}$
14: /* Add non-trivial zones in $M$'s ancestry to $D$ */
15: for all $u \in M$ do
16:    $v \leftarrow Parent(u)$
17:    while $v \neq r$ do
18:       if $v \in N$ then
19:          $D \leftarrow D \bigcup \{v\}$
20:       end if
21:       $v \leftarrow Parent(v)$
22:    end while
23: end for
24: return $D$
```

Algorithm 4 ControlledAlias(u, D)

```
Input: Domain name $u$
Input: Set of first-order dependencies $D$
Output: False if $u$ directly or indirectly aliases a name outside
     explicit dependency; True otherwise
 1: $H \leftarrow \{u\}$
 2: while $u$ is an alias do
 3:    if $Parent(Cname(u)) \notin D$ then
 4:       return False
 5:    else if $Cname(u) \in H$ then /* Loop detected */
 6:       return True
 7:    end if
 8:    $H \leftarrow H \bigcup \{u\}$
 9:    $u \leftarrow Cname(u)$
10: end while
11: return True
```

Algorithm 5 ThirdPartyInfluence1(u, D)

```
Input: Domain name $u$
Input: Set of first-order dependencies $D$
Output: Influence on $u$ by names outside of $D$
 1: if $u$ is not a zone then
 2:    /* $u$ aliases a name outside of $D$ */
 3:    if ControlledAlias($u, D$) $= False$ then
 4:       return 1.0
 5:    end if
 6:    $u \leftarrow Parent(u)$
 7: end if
 8: $P \leftarrow 0$
 9: /* Aggregate influence outside $D$ for $u$'s ancestors */
10: while $u \neq r$ do
11:    $P_u \leftarrow 0$
12:    for all $v \in V_d | \exists (u, v) \in A_d$, NS target dep. do
13:       if $Parent(v) \notin D$ or
               ControlledAlias($v, D$) $= False$ then
14:          $P_u \leftarrow P_u + w(u, v)$
15:       end if
16:    end for
17:    $P \leftarrow 1 - (1 - P)(1 - P_u)$
18:    $u \leftarrow Parent(u)$
19: end while
20: return $P$
```

ancestry (lines 10–19).

Algorithm 6 describes the methodology for calculating third-party influence $I_d(U_d - U'_d)$ of d. The TPI of d's alias, if any (line 6), is combined (line 18) with the TPI of its parent zones (line 11) and that of its collective NS target dependencies (lines 14–16).

D. Model validation

To validate the name dependency model presented a sample of over 600 names was selected from the collected database (see Section V-A), and a corresponding active dependency graph, G'_d, was constructed for each name, d. For each name the level of influence of each other domain name in the graph was calculated.

We deployed BIND [8] as a resolver on more than 100 PlanetLab nodes [13], attempting to create an environment diverse enough that queries for each name by the collective resolvers would be uniformly distributed amongst authoritative servers. On each PlanetLab node a query was issued to the name daemon 100 times for each name, d. Before the initial query of each name, the server's cache was flushed, so the source of every name resolved during the process could be identified (rather than relying on existing cached data from unknown sources). All DNS traffic to and from the server was monitored. Any address queries issued by the server were induced because of active influence on d. For every answer received for a name u during the resolution of d, u was mapped to the name of the server from which the answer was received. When the final response was received, containing the address corresponding to d, the names formed a graph of dependency paths from d to r representing the path(s) followed to resolve d, a subgraph of G'_d.

After each iteration, the addresses for any names resolved by induced queries were flushed from the server's cache and explicitly re-queried, before beginning the next iteration. This is equivalent to speeding up the expiration of the cached names. Without this action, the server would always respond with the cached name from the previously acquired source,

Algorithm 6 ThirdPartyInfluence(d)

```
Input: Domain name $d$
Output: TPI of $d$
$D \leftarrow$ FirstOrderDeps($d$)
$P_A \leftarrow 0$
if $d$ is not a zone then
    /* If $d$ is an alias, calculate the TPI of $Cname(d)$ */
    if $d$ is an alias then
        $P_A \leftarrow$ ThirdPartyInfluence1($Cname(d), D$)
    end if
    $d \leftarrow Parent(d)$
end if
/* Calculate the TPI of $Parent(d)$ */
$P_P \leftarrow$ ThirdPartyInfluence1($Parent(d), D$)
/* Calculate the TPI of each NS target of zone $d$ */
$P_{NS} \leftarrow 0$
for all $u \in V_d | \exists (d,u) \in A_d$, NS target dep. do
    $P_{NS} \leftarrow$
            $P_{NS} + w(d,u)$ThirdPartyInfluence1($u, D$)
end for
/* Aggregate the TPI of all name dependencies */
return $1 - (1 - P_P)(1 - P_A)(1 - P_{NS})$
```

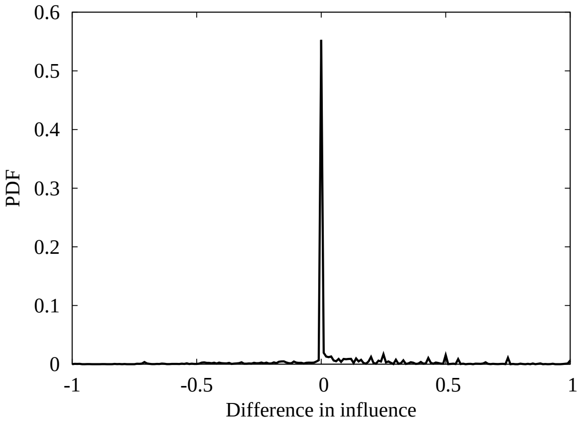

Fig. 3. The distribution of differences between the level of influence for each sample name observed through active querying and that computed using the model. Positive values indicate that the model predicted more influence than was observed.

and the likelihood of exploring other potential paths to the root would be diminished. After the 100 iterations of querying d, the influence of each other name, u, on d is determined by the calculating the fraction of the iterations in which u was included in the experimental graph.

Accuracy was measured by comparing the observed dependency graph with the theoretical dependency graph with $P_{\{s,c\}}(v) = 0$. For each name analyzed we verified that the influential names was a subset of those in V_d. The probability density function (PDF) of the difference in influence of each is shown in Fig. 3. The large peak in the graph demonstrates that 55% of the observed influence was exactly in line with the influence predicted by the model.

V. Quality of Name Resolution

In this section we describe the methodology for collection data from the DNS infrastructure, and provide analysis of the data collected. Because no single property of a name dependency graph is useful by itself for evaluating the quality of name resolution for a domain name, an analysis of several different areas is used to assess quality of name resolution.

A. Data collection

We populated a database of name dependencies by crawling the name space of known domain names. A set of over 3,000,000 hostnames was extracted from URLs indexed as part of the Open Directory Project (ODP) at DMOZ [14] dated December, 2008.

Each name was investigated by first surveying each name in its ancestry which had not already been surveyed, beginning with the root. Surveying a domain name consisted of issuing queries to a recursive server to receive an authoritative answer for any matching NS, MX (mail exchange) and CNAME RRs. The dependencies between the name and any corresponding targets returned were recorded and subsequently surveyed.

For each NS RR, we checked the consistency between parent and child zones by using some extra probing. For zone z we found the set of servers only authoritative for $Parent(z)$, $NSA^z_{Parent(z)} = NSA_{Parent(z)} - NSA_z$. For each server in $NSA^z_{Parent(z)}$ we issued an NS query for z, until a response was received that did not have the authoritative answer (AA) flag set. Only if the AA flag was not set could we accurately obtain the set of NS RRs (NS'_z) maintained by $Parent(z)$. If $NS_z \neq NS'_z$ an inconsistency is detected.

The TTL field of additional address records corresponding to targets of NS RRs in the authority section of server responses are used to identify the presence of glue records in the parent zone. When server s returns a non-authoritative response, a second query is issued to s after a two-second delay (both without the recursion-desired flag set). TTL is measured in seconds, and the two-second delay between queries accounts for any skew that may exist and be misinterpreted with a one-second delay. If for an NS target there is no corresponding address record in the additional section, then it is indicative that the parent has not been configured with a glue record. If the TTL of the additional record differs between the two responses, then it is inferred that the record came from an authoritative response in s's cache. Since such a response would take precedence over any glue record configured in $Parent(d)$, we optimistically give the zone the benefit of the doubt that it is configured with a glue record, if the NS target is in-bailiwick.

If the TTL value of an additional record does not vary between the two responses from s, it could indicate one of several things:

- $Parent(z)$ is configured with a glue record for the additional record;
- s is (also) authoritative for the zone to which the additional record belongs; or

Measurement	Values
Seed domain names	3,031,110
Domain names harvested	8,144,487
NS target dependencies	6,633,647
NS targets missing glue	1,146
Additional RRs in-bailiwick from cache (over glue)	12,902
Additional RRs out-of-bailiwick glue	912,791
Additional RRs out-of-bailiwick from cache	37,697
Zones for which $NS_z \neq NS'_z$	575,260
Aliases referencing other aliases	28,721
NS targets aliasing other names	12,450

TABLE IV
A SUMMARY OF RESULTS COLLECTED FROM SURVEYING THE DNS NAMESPACE, SEEDED WITH HOSTNAMES FROM THE ODP.

Metric	$P_{\{s,c\}}(v)$	Avg.	Max.
Influential zones	0	5.13	69
Influential zones	> 0	15.93	161
Non-trivial zones	0	2.62	43
Non-trivial zones	> 0	11.46	127
First-order ratio	0	0.94	1.0
First-order ratio	> 0	0.65	1.0
Third-party influence	0	0.09	1.0
Third-party influence	0.5	0.39	1.0
Third-party influence	1.0	0.56	1.0

TABLE V
TCB AND INFLUENCE STATISTICS FOR ODP HOSTNAME COLLECTION.

- s is authoritative for an ancestor of the NS target and has been configured with a glue record for that NS target.

We assume optimistically in this case that if the NS target is in-bailiwick $Parent(z)$ is configured with a glue record.

If no non-authoritative answers are returned from querying the servers in $NSA^z_{Parent(z)}$, then we cannot determine inconsistencies between NS'_z and NS_z, and their corresponding glue records. However, in practice, if $NSA_{Parent(z)} \subseteq NSA_z$, then consistency is satisfied implicitly since all servers in $NSA_{Parent(z)}$ will send authoritative records from z over corresponding records from $Parent(z)$ [12]. For all zones in our analysis we assumed $P_{NS}(z) = 0$, so NS'_z was used for server selection.

Our analysis did not follow dependencies of general top-level domains (gTLDs), such as *com* and *edu*. There were two reasons for this: all descendants of gTLDs share the same top-level ancestry and was therefore uninteresting from the top level up; and the names of many of the gTLD servers are in the *gtld-servers.net* zone, so as we increased the probability ($P_{\{s,c\}}(v)$) that NS target names—including the names of the gTLD servers—were cached as part of our analysis, the third-party influence of names having non-*net* gTLDs approached 1, which skewed the results. Our analysis did, however, follow country-code top-level domains (e.g., *us*, *fr*). The results from the survey are summarized in Table IV.

B. Trusted computing base

The raw size of the TCB for hostnames collected in terms of influential zones and non-trivial zones is shown in Fig. 4 as a cumulative density function (CDF), and the statistics are shown in Table V. Nearly all hostnames have a TCB smaller than 20 zones when $P_{\{s,c\}}(v) = 0$, and the average size of the TCB was 2.62 non-trivial zones and 5.13 total zones—both of which are reasonably small. When $P_{\{s,c\}}(v) > 0$, the average size of the TCB increases several times to 11.46 non-trivial and 15.93 total zones. Only about 80% have fewer than 20 zones; most of the remaining 20% have between 30 and 80 non-trivial and total zones in their TCB. Caching and using NS target names from authoritative sources, rather than glue, can increase the size of the TCB of a domain by several times.

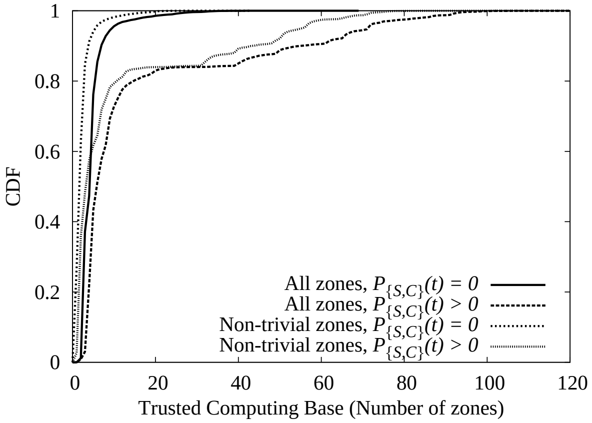

Fig. 4. The CDF for the size of the TCB of hostnames collected from the ODP. Included are the CDF for the number non-trivial and total zones in the TCB, for $P_{\{s,c\}}(v) = 0$ and $P_{\{s,c\}}(v) > 0$.

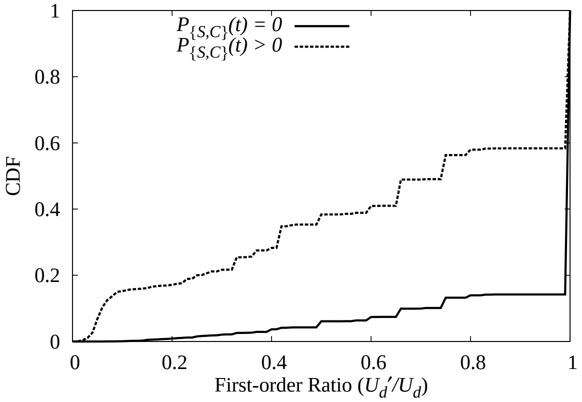

Fig. 5. The CDF for the first-order ratio of the set of hostnames collected from the ODP.

C. Controlled influence

The first-order ratio $\frac{U'_d}{U_d}$, shown in Fig. 5, is used to determine the percentage of non-trivial zones that are expressly configured by the administrators of d. Values closer to 1 indicate that the administrators are largely in control of the zones comprising the TCB. The average first-order ratio was 0.51 for $P_{\{s,c\}}(v) = 0$ and 0.25 for $P_{\{s,c\}}(v) > 0$, indicating that control of the TCB is lost as caching of NS target names is introduced. When $P_{\{s,c\}}(v) > 0$, third-party zones comprise

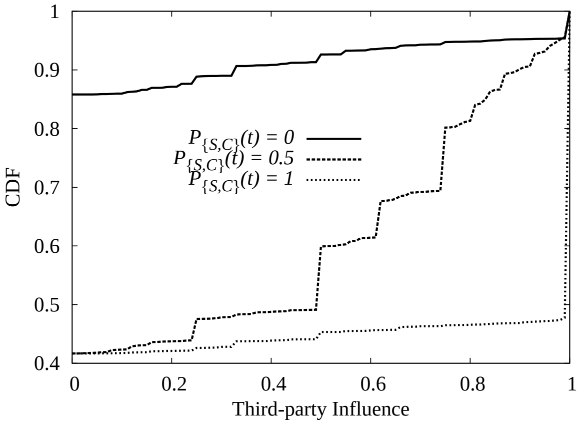

Fig. 6. The CDF for the third-party influence of the set of hostnames collected from the ODP.

more than half of the the non-trivial zones in the TCB of roughly 70% of the hostnames surveyed.

Fig. 6 shows the third-party influence for the hostnames collected from the ODP. When $P_{\{s,c\}}(v) = 0$, over 85% of the hostnames are not influenced at all by third parties. At $P_{\{S,C\}}(v) = 0.5$ only 60% of the hostnames are influenced less than 50% by third parties. When $P_{\{S,C\}}(v) = 1$ more than half of the hostnames are influenced almost certainly by third parties. Again the behavior of caching preference of NS target names from authoritative sources at the resolver and authoritative servers greatly affects third-party influence.

VI. Future work

Given the impact of caching of NS target names from authoritative sources at resolvers and authoritative servers, a field study of how often this type of caching occurs would provide a useful supplement to this research. Also, the model proposed in this research can be expanded to include details such as the dynamics of time-to-live (TTL) values of related RRs and statistical modeling of the cache contents, such as in [15]. For example a BIND resolver will not necessarily query every server in NSA_z for names in z before repeating queries to servers, as RFC 1035 suggests [11]. Rather, it only considers the addresses from servers names in NS_z that are currently in its cache (while it simultaneously begins queries to acquire others), so some servers might be preferred (perhaps exclusively) over others, given certain configurations. This is not the case with djbdns, which obtains all the addresses for names in NS_z before selecting a server for query. This analysis will provide guidance as to whether the BIND or djbdns method of server selection results in higher levels of vulnerability.

A resolver's knowledge of the complete set of dependency paths for a zone provides the basis for answer checking, such as that presented in [7], but cross checking by following unique dependency paths. This model also lays the groundwork for formal methodology to detect misconfigurations, such as cyclic zone dependencies [2]. This could be done by measuring the level of self-influence in the dependency graph.

VII. Conclusion

In this paper we have presented a graph model for analysis of name dependencies in DNS, which was based on specification and behavior of deployed DNS servers. We defined the trusted computing base (TCB) of a domain name in terms of namespace, and particularly zones. Methodology for calculating the level at which domain names influence the resolution of others was described and used to determine third-party influence—the probability that resolution of a domain name will utilize namespace outside the explicit configuration by the domain administrators.

We observed that the TCB of domain names, when measured by influential zones, is much smaller than previously thought. On average 94% of the non-trivial zones in the TCB of a domain name were explicitly configured by the domain administrators. However, caching of NS targets at the resolver and authoritative server can increase the size of the TCB and the influence of third-party namespace significantly, and should be considered when configuring DNS servers.

References

[1] V. Ramasubramanian and E. G. Sirer, "Perils of transitive trust in the domain name system," in *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, USENIX Association. Berkeley, CA, USA: USENIX Association, 2005, p. 35.

[2] V. Pappas, Z. Xu, S. Lu, D. Massey, A. Terzis, and L. Zhang, "Impact of configuration errors on dns robustness," in *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, ACM. New York, NY, USA: ACM, 2004, pp. 319–330.

[3] R. Liston, S. Srinivasan, and E. Zegura, "Diversity in dns performance measures," in *In Proceedings of the SIGCOMM '02 Symposium on Communications Architectures and Protocols*. ACM Press, 2002.

[4] J. Pang, J. Hendricks, A. Akella, R. D. Prisco, B. Maggs, and S. Seshan, "Availability, usage, and deployment characteristics of the domain name system," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2004, pp. 1–14.

[5] V. Ramasubramanian and E. G. Sirer, "The design and implementation of a next generation name service for the internet," in *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2004, pp. 331–342.

[6] K. Park, V. S. Pai, L. Peterson, and Z. Wang, "CoDNS: Improving DNS performance and reliability via cooperative lookups," in *OSDI '04: Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation*, USENIX Association. Berkeley, CA, USA: USENIX Association, 2004, pp. 14–14.

[7] L. Yuan, K. Kant, P. Mohapatra, and C.-N. Chuah, "DoX: A peer-to-peer antidote for dns cache poisoning attacks," in *Communications, 2006. ICC '06. IEEE International Conference on*.

[8] ISC BIND. [Online]. Available: http://www.isc.org/products/BIND/

[9] djbdns. [Online]. Available: http://cr.yp.to/djbdns.html

[10] P. Mockapetris, "RFC 1034: Domain names - concepts and facilities," 1987. [Online]. Available: http://tools.ietf.org/html/rfc1034

[11] ——, "RFC 1035: domain names - implementation and specification," 1987. [Online]. Available: http://tools.ietf.org/html/rfc1035

[12] R. Elz and R. Bush, "RFC 2181 - clarifications to the DNS specification," 1997. [Online]. Available: http://tools.ietf.org/html/rfc2181

[13] PlanetLab. [Online]. Available: http://www.planet-lab.org/

[14] Open Directory Project. [Online]. Available: http://www.dmoz.org/

[15] L. Yuan, K. Kant, P. Mohapatra, and C.-N. Chuah, "A proxy view of quality of domain name service," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. Proceedings*.