Operational Implications of the DNS Control Plane

Eric Osterweil
VeriSign Labs
eosterweil@verisign.com

Danny McPherson
VeriSign Labs
dmcpherson@verisign.com

Lixia Zhang
UCLA
lixia@cs.ucla.edu

May 31, 2011

1 Introduction

The Domain Name System (DNS) [7] provides vital mapping services for the Internet. It maps domain names such as `ucla.edu` to values ranging from IP addresses to email servers to geographic locations and more. Virtually every Internet application relies on looking up some form of DNS data. This article first describes a dichotomy that exists between DNS' well structured and ordered *data plane* (the hierarchical tree of domain names) and its, as yet underappreciated, *control plane* (the interconnected graph of name servers). Then the article focuses on the control structure's *dependency graphs*, which are the recursive graphs of the inter-dependencies that exist between the name servers associated with each zone. The goal of this investigation is to understand the implications these graphs have on the security and performance of the overall DNS itself.

DNS' data plane is a name space that is a clearly defined tree hierarchy whose intent is to ensure DNS domain name uniqueness. At the top of the tree, the root zone delegates authority to Top Level Domains (TLDs) like `.com`, `.net`, `.org`, and `.edu`. The zone `.com` then delegates authority to create `google.com`; `.edu` delegates authority to create `ucla.edu`, and so forth. In the resulting DNS tree structure, each node corresponds to a zone. Each zone belongs to a single administrative authority and is served by multiple authoritative name servers, which provide name resolution services for all the names in the zone. One reason that the DNS is so powerful is that its data plane allows administrators a great deal of flexibility: they can manage their name space however they like. However, the control plane's analogous flexibility can lead to operational problems if not managed conscientiously.

For DNS' control plane, operational guidelines require that a zone have multiple authoritative name servers, and that they be distributed through diverse topological and geographical locations to make DNS services robust against unexpected failures [5]. Zone operators face decisions, such as where to place the multiple servers in order to meet these guidelines. While the goal and implications of the diversified redundancy guideline may seem clear at the first glance, a more detailed look suggests the existence of different types of redundancy.

- Server instance redundancy: providing multiple server instances can share the load and provide robustness against host failures. However if all the instances are placed in one location, they are still subject to local failures such as flooding or power outages.

- Topological redundancy: placing DNS server instances in distinct topological locations (different networks/prefixes/ASNs/etc) can help to greatly reduce the impact of local failures. However this topological diversity is likely to introduce administrative diversity as well, i.e. placing one zone's server in different administrative domains. Given the importance of DNS service, hosting one's DNS server in another administrative domain silently implies a high level of trust on the hosting domain, an important issue that deserves attention.

- Domain name redundancy: placing DNS servers under different branches of the DNS tree (e.g. using `ns.bar.com` as a secondary server for the domain `foo.com`) can provide redundancy across different organizations, so that failure of one organization will not impact the availability of its DNS service. This is a common practice today, as DNS operators host each others secondary servers. However this

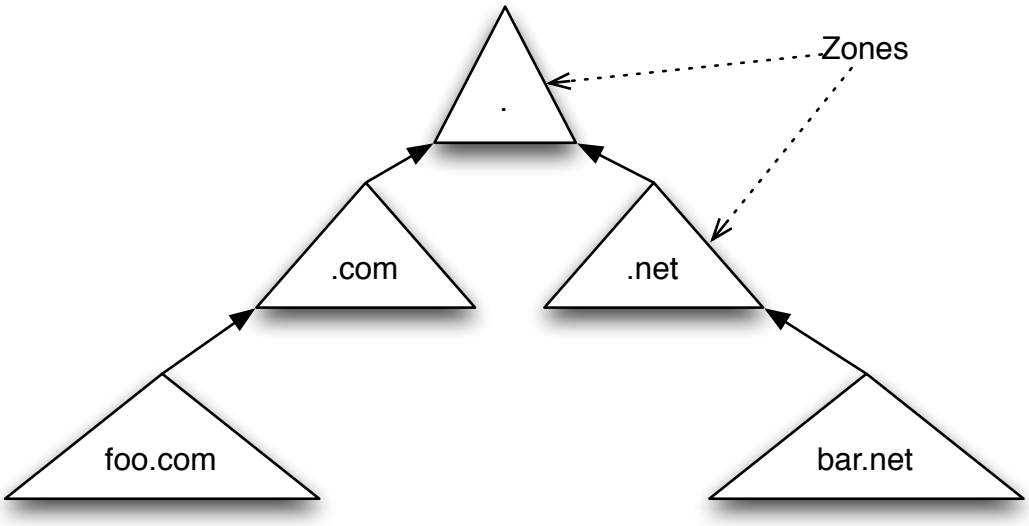

Figure 1: Here we can see that the TLDs `.com` and `.net` fall directly below the root, and `foo.com` and `bar.net` fall directly below their lexical parents.

again can silently lead to unknown transitive DNS resolutions needed to resolve a single name, as we explain later in this article, the resulting impact on security and performance can be significant.

The dependency graph that results from DNS operators hosting each other's secondary servers (both topologically, and by domain name) leads to two operational questions: i) is the transitive trust that a zone operator places in his/her remotely hosted secondaries transitive enough that resolvers should implicitly be able to trust these organizations too, and ii) as the dependency graph grows larger and more work is forced onto resolvers, is the increased redundancy ultimately more hurtful than helpful?

In the remainder of this article we provide observations that show the existence of dependency graphs, a new tool for visualizing them for any zone, we qualify their usefulness to operators, and we outline the magnitude of the performance impact they can have on resolvers. The goal of this article is to raise awareness of the tradeoffs that exists between redundancy and usability of zones with large dependency graphs.

2 Background

The DNS was designed to be (and remains as) the Internet's de facto name mapping system. Thus, among its primary objectives are providing both name uniqueness and a distributed authority model. That is, the *data plane* in the DNS (its domain names) like `iab.org`, `cs.ucla.edu`, and `verisignlabs.com` must all map to unique (and potentially different) administrative organizations, servers, locations, etc. All arbitration about which organizations are allowed to own which domain names are external to the operational system. For example, all accredited colleges in the United States can acquire a name space (or zone) below the `.edu` TLD. However, they must all be designated uniquely. Therefore, `csu.edu` cannot be used by Colorado State University, because it is delegated to Chicago State University. Figure 1 shows a very simple illustration of the data plane.

While the DNS name space maintains order and administrative separation through its strict tree hierarchy, the different zones are actually served by its *control plane*; an uncorrelated graph of name servers. For example, the zone `iab.org` exists under the `.org` TLD, which is under the root, and is totally unambiguous. However, the name servers for `iab.org` exist under both `.org` *and* `.info`. In fact, the TLD `.bg`'s name servers exist under `.bg`, `.se`, `.com`, and `.net`. Thus, while the name space is logically hierarchical, its control structure can be quite complex. Figure 2 shows an example of how the simple zone hierarchy can be served

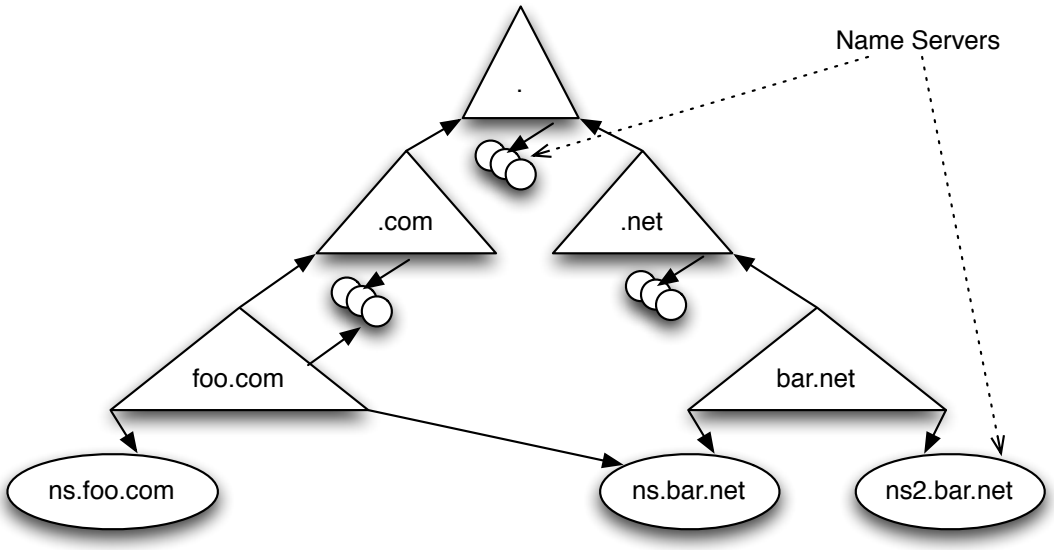

Figure 2: Here we can see that even though the zones follow a clean hierarchy, `foo.com` is served by name servers under the `bar.net` zone, and even some of the name servers that serve its own parent, `.com`. There is no restriction on the control structure of the DNS name servers (like there is over the logical data structure of its name space).

by an interconnection of name servers. Moreover, previous work [8] has even demonstrated that the name server interconnections can lead to cyclic dependencies. Whenever a client resolver wants to traverse any zone in the data plane it must first learn the names and IP addresses of the *name servers* that serve that zone. Thus, as zone operators place the name servers for their zones under more branches of the DNS, they force resolvers to spend a corresponding amount of extra effort querying other zones. Resolvers must learn the mapping of each related zone's name server IP addresses, and the mappings for any name servers that serve *those* zones, and so on (we call this the *Dependency Graph*).

3 Transitive Trust

The term "transitive trust" underscores an implicit tension between one way that zone operators attempt to bolster the redundancy of their deployed zones, and the resultant transitivity that resolvers are saddled with. The crux of the tradeoff is simple: zone operators may want to use trust in their colleagues and diversity of names to bolster their deployments, but resolvers (who have no control over how data is served) may not trust their transactions to the same parties. Some of the benefits to the zone operators are i) they can bolster their name server deployments by trusting friends (who are already deployed in different topological locations), and ii) placing name servers in different branches of the DNS hierarchy increases the diversity of the name space that *zones* rely on (and thus reduces correlated failures). That is, if a zone (say `foo.com`) has name servers under `.net`, `.com`, and `.info`, then any failure of a single TLD will still allow resolvers to locate and query `foo.com`'s zone. However, resolvers may be trusting a zone with sensitive information (such as banking transactions) and may not *trust* any step of their transactions to the servers a zone operator has enlisted.

Another view of adding diverse names to the list of authoritative sources for a zone is that they are new liabilities. Indeed, while adding independent branches and lexically diverse name servers reduces concerns that a zone may have a single point of failure, it also increases the number of points of partial failure, and requires quantifiably more work under normal operations. In order to illustrate this, consider the following example, imagine a zone `foo.com` that is served by the name servers `ns.foo.com` and the secondary `ns.bar.net`. In

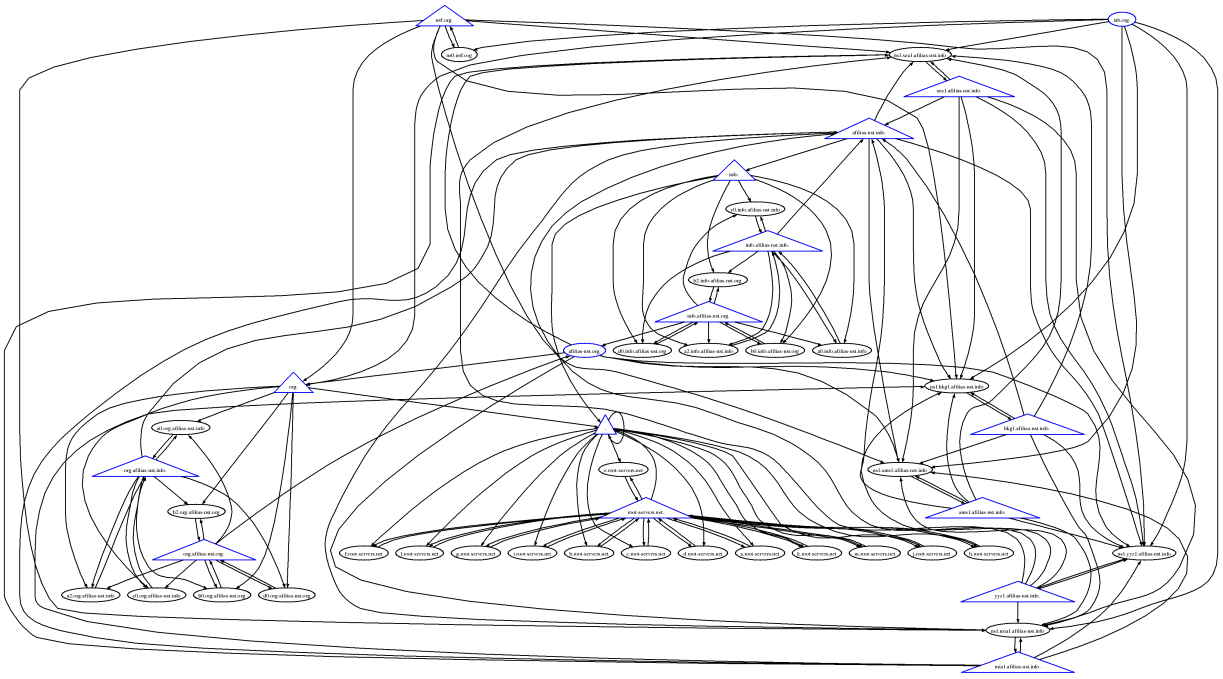

Figure 3: The dependency graph of the IAB has a large number of zones (15) and, thus, requires more work to resolve, and presents a larger attack surface.

order for a resolver to query `foo.com`, it must first lookup the zone for `ns.bar.net` (`bar.net`). Further, if `bar.net` has a secondary in the zone `baz.com`, and that zone has a secondary in `qux.com`, then the operator of `foo.com` has directed resolvers to transit *a form of* their trust in the zone's authorities all the way to `qux.com`. We call the full set of interconnections the *Dependency Graph*.

3.1 The Dependency Graph

Earlier work [9] described the Transitive Trust process in some detail. However, one important operational clarification to this earlier work is that while resolvers do trust `qux.com` for part of the resolution process of `foo.com` (in the above example), they do not directly query `qux.com` for any authoritative information *about* `foo.com`. Thus, an adversary cannot inspect queries to `qux.com` and thereby know if they are even related to `foo.com`. Therefore, what needs to be clarified is that one cannot assume that compromising any name server in a dependency graph will necessarily enable an attack against any other portion of that graph. Nevertheless, this view of transitive trust does present a liability to `foo.com`. For example, if an adversary were determined enough, and perhaps had some out of band information, she could interpose in early communications and spoof the entire resolution chain between `qux.com` and `foo.com`, or perhaps she may even control `qux.com`. Here we simply observe that adding dependencies across wider areas of the DNS name space increases the *attack surface* over which an adversary may try to launch an attack.

While the above is just a simple fictional example, we can use VeriSign Labs' Transitive Trust Portal [6] to consider the *actual* dependency graphs of several real zones. First, in Figure 3, we consider the zone of the Internet Architecture Board (IAB) [1], which is an organization that provides leadership to the IETF [2] for Internet standards and operational best practices. Here we can see that this relatively important zone has a rather large dependency graph, and the resulting attack surface includes name servers outside the IAB's direct administrative control. Perhaps more alarming is the graph for the TLD `.bg`, seen in Figure 4. By contrast, consider the relatively smaller surface of Starbucks' zone, in Figure 5.

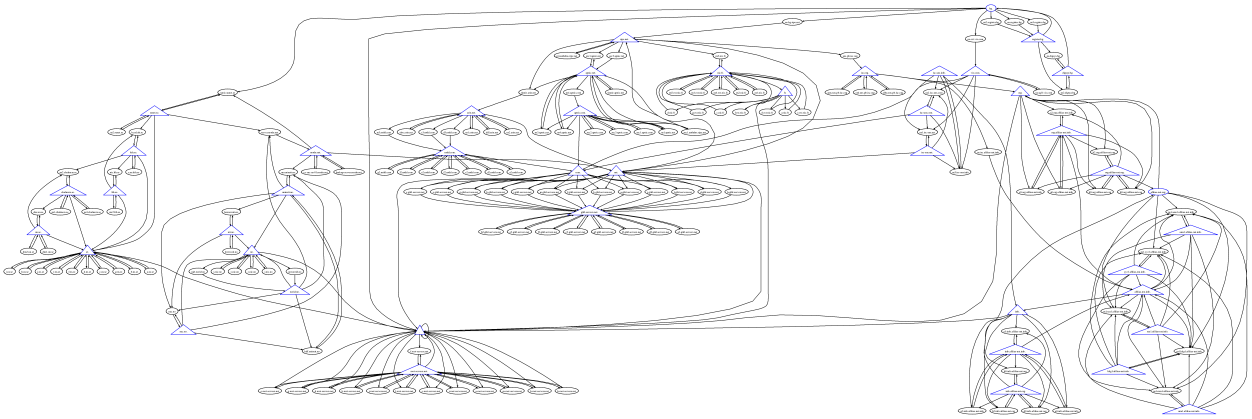

Figure 4: The dependency graph of `.bg` has a very large number of zones (42) and, thus, requires a great deal of work to resolve, and presents a significant attack surface.

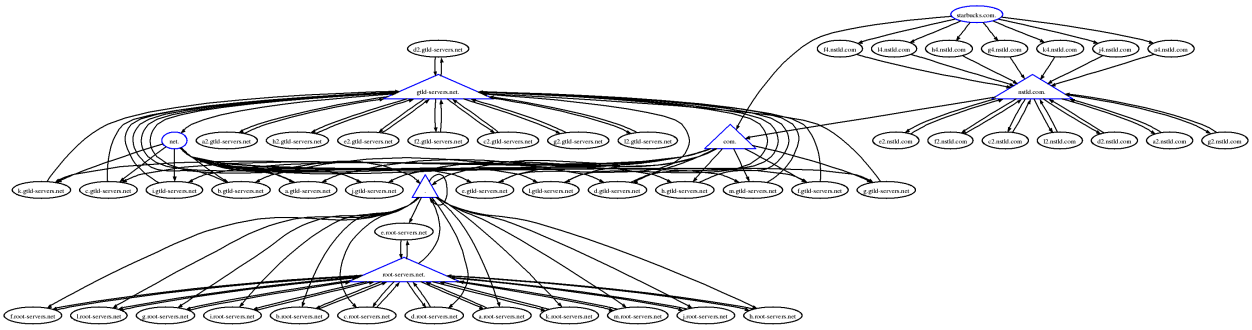

Figure 5: The dependency graph of Starbucks is only 5 zones and is about as large as that of its TLD (`.com`) and the root (which could be considered to be requirements for any zone under a TLD), and thus presents relatively little resolution work, and a relatively small attack surface.

3.2 Security Concerns

The term "*attack surface*" describes the relative number of places that an adversary can attack something (in this case, a DNS zone). As a general rule of thumb, operators should try to reduce their attack surface by reducing any points of vulnerability (i.e. limiting any would-be adversary's targets of opportunity). Not all points on an attack surface need to have known glaring holes in order to be of concern to an operator. Rather, the existence of an external dependency can, itself, be enough for an adversary to find a weakness that an operator is unaware of. For example, consider the previous example of `foo.com`'s dependency on `qux.com`. While an operator may not think that an adversary can know which of the queries that are seen by `qux.com` relate to `foo.com`, that doesn't mean that the adversary *cannot* know. Thus, a security conscious operator faces an implicit challenge to guard their zone against threats that result from attacks against his/her infrastructure *and* the external infrastructure the zone relies on. In addition to the increased surface, the operational overhead of traversing these graphs can be an important factor in *every day operations* when there is no failure or attack underway.

3.3 Overhead Concerns

While including multiple branches of the DNS hierarchy bolsters a zone's independence from failures, operators must also consider that the amount of work needed to *resolve* names in multiple branches is on the same order as the relative independence. That is, if one has put secondary servers in different branches of

the DNS, then the entire predecessor graph for each secondary must *also* be looked up. This is because when many modern resolvers (BIND, unbound, Secure64, etc.) encounter a zone, they lookup all name servers, with varying degrees of parallelism. This allows them to use various proprietary algorithms to select the most efficient responder to queries (lowest latency, fewest dropped packets, etc.).

While increasing the overhead that resolvers face may not be considered a "problem" by many, it *does* have a measurable impact on performance. The aforementioned resolvers must all issue separate queries for each name server in a zone, and as the dependency graph grows larger, the number of concurrent queries (and therefore, the amount of work done by a resolver) has been observed to result in larger lags in resolution time. In essence, resolvers pay a measurable price every day for these deployment choices even if a failure never capitalizes on their existence.

Perhaps one of the most daunting problems of large dependency graphs is simply a lack of awareness. With zones like `.com` that have hundreds of servers at multiple geographic locations and an official up-time of 100%, most zones gain no measurable resilience by placing name servers under additional TLDs. In the event of a TLD failure, this may help their up-time, but every resolver is constantly faced with the increased lag and load time *every time* it encounters such a zone. Many Best Common Practices (BCPs) exist for manging DNS deployments [5, 3, 4], but no document currently describes the liabilities and tradeoffs of dependency graphs in the DNS.

4 Conclusion

Given the importance of DNS services in Internet operations, it is our responsibility as Internet researchers to gain a thorough understanding of the system's security and performance qualities. In this article we use specific instances of DNS dependency graphs to highlight an important aspect of the existing DNS system that has yet to be fully understood, documented, and dealt with.

As the evidence in this article shows, excessively large dependency graphs exist, and they can lead to both underappreciated trust considerations and notable performance penalties. We would like to call the community's attention to these issues in order to put them on the table, so that necessary operational guidelines can be developed. Further, there is a balancing act between the complexity, overhead, and resilience of DNS' control plane, and finding an operational sweet spot is an important open research challenge.

References

[1] Internet Architecture Board (IAB). `http://www.iab.org/`.

[2] Internet Engineering Task Force (IETF). `http://www.ietf.org/`.

[3] D. Eastlake 3rd. Domain Name System (DNS) IANA Considerations. RFC 5625, March 2011.

[4] R. Bellis. DNS Proxy Implementation Guidelines. RFC 5625, August 2009.

[5] R. Elz, R. Bush, S. Bradner, and M. Patton. Selection and Operation of Secondary DNS Servers. RFC 2182, July 1997.

[6] VeriSign Labs. Transitive Trust Portal. `http://trans-trust.verisignlabs.com/`.

[7] P. Mockapetris and K. J. Dunlap. Development of the domain name system. In *SIGCOMM '88*, 1988.

[8] Vasileios Pappas, Zhiguo Xu, Songwu Lu, Daniel Massey, Andreas Terzis, and Lixia Zhang. Impact of Configuration Errors on DNS Robustness. In *ACM SIGCOMM*, 2004.

[9] Venugopalan Ramasubramanian and Emin Gün Sirer. Perils of transitive trust in the domain name system. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, IMC '05, pages 35–35, Berkeley, CA, USA, 2005. USENIX Association.