# Domain Name Service

Timothy Myers
Tennessee Tech
344 East 12th Street Apt. 4
Cookeville, TN 38501
931-206-2255
temyers240@gmail.com

## ABSTRACT

This paper is a brief overview of the Domain Name Service (DNS), a goal which can be difficult considering the pervasiveness of not only DNS as a technology but also of information on DNS because of its widespread nature. As human beings, we people like and enjoy the convenience of names, and this is the service that DNS provides. At its most basic level, it enables us to have and use names for various machines somewhere else on a network. There are also, however, other clever uses of the service that allow for discovering other sorts of information. However, just like any software system, DNS can be vulnerable to attack especially when it is not configured properly, while these attacks can take many forms they all affect the use of names for our networks. I also hope to look at a few of the possible configuration errors and attack vectors.

## Keywords

DNS, human readability, networking

## 1. INTRODUCTION

As a starting point in a discussion on DNS, I believe that I can confidently say that DNS is a technology that has allowed the large networks that we use, especially the internet. This is because people are lazy and do not want to remember numbers, for some reason people like words better than numbers and DNS has been engineered to turn the numbers that computers use to identify themselves with and turn that into names so that we can find the machines, and thus the services that they hold, and use them for ourselves. However because of the database and cache system of DNS there have been some people who have been able to devise ways of retrieving information not normally given by an indirect means. Unfortunately DNS is not a perfect system, it has vulnerabilities and weaknesses that range from simply mis-configuring the system to more subtle things, all of which can allow someone from the outside to damage the system and cause problems for a segment of the users using that network.

## 2. BASICS

As with most services that involve information flow over a network, there is more than one piece involved. The the terminology of DNS these are known as the resolver and the name server [3]. It is the job of the resolver to ask questions to the name server, and the job of the name server to attempt to find an answer to the question, but before we delve deeper into that topic I would like to talk a little bit about the structure of DNS itself. DNS is a distributed database. Most of the time, when we think of a database we consider a large file or program whose purpose is to hold all of the information we need to hold in one single place, a place with multiple redundant backups, but one place nonetheless. DNS is nothing like that at all, there is no one place or file where all of the IP addresses in the world are attached to a name, it would be so large as to be almost impossible to find the address you need efficiently. The actual DNS database is spread out upon many name servers all over the world, with each name server only having authority, or complete knowledge, of one domain or subdomain. In order for this system to work, however, there is one piece that is missing. Each name server has a knowledge of at least one other name server as well, so if one name server is asked by the resolver for information that is not contained locally in his files, he will ask the other name servers that he knows about for the information, a process that repeats recursively.

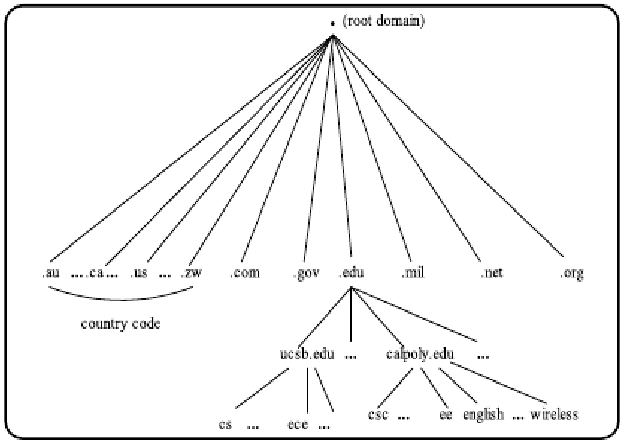

**Figure 1: Domain Name Hierarchy [2]**

This system ultimately works because the name server tree is set up in the hierarchical form shown in Figure 1. With the .edu name server knowing about the name servers for all or almost all domains that end in .edu, for example. This pattern is repeated down the tree with the calploy.edu name server knowing about all subdomains ending in calply.edu[2].

This method alone seems like it would be rather inefficient, as each query could be required to travel up the entire length of the originating branch of the hierarchy and down the entire length of another branch. To this end, both name servers and resolvers can make use of a local cache to hold the results of recently made queries[3], a practice that enforces the locality of temporal reference.

For the sake of robustness, many of the implemented name servers also attempt to retrieve copies of records from neighboring name servers at certain intervals[3]. A practice that can make taking down a certain domain difficult through DNS attacks because there are usually multiple name servers with the correct information regarding any specific domain.

## 3. INFORMATION

The information requested from a name server is normally thought to be, and usually is an address on the network for a particular machine. But within those requests other information about the type of address and other things are to be found as well.

When a resolver creates a query packet there are several things that it includes, things such as flags requesting recursion, the number of questions it is asking, the questions themselves, and one of the most important parts the type of answer is want back [3].These types in the query usually match up to the types of the resource records in the local master file of the name server. A few examples of the more common types found in master files, from [3]:

A – (Host Address) The address for a particular machine

NS – (Name Server) The address of another name server on the network

CNAME – (Alias) An aliased name for the given query

SOA – (Start of Authority) Information about the name server's authoritative domain

MX – (Mail Exchange) Address for a mail server

In addition, there are a few types only for queries, such as the type "*" which asks for all resource records matching the request name.

When responding to a resolver, a name server will look for all records matching both the query name and the query type, or all types matching the query name if the query type is "*" and send that information back to the resolver. The response includes the question(s) that the resolver asked, presumably for verification purposes as well as any responses. Because the actual return data varies with each type, each section of the return data does include the type of the return record so that correct parsing can occur, whether it is simply a 4 byte address or a name in the case of a CNAME record.

This however is not the only way to discover data from a DNS name server. As web applications have become so prevalent in recent times, many seek to discover ways of capturing the popularity of these web services [1]. One way to get a rather rough estimate of relative popularity is through a technique of "poking" a name server's cache. This technique sends a request to the name server with the recursive bit in the flags turned off. With this flag turned off, any information returned that is not in the authoritative zone for the name server must be in it's cache. The name server is "poked" once per Time-To-Live(TTL) interval of the requested record, and it is determined whether the record is in the cache and if it is when it entered the cache and when it will leave based upon the TTL[1]. With this information the times when that record was not in the cache can be determined, and thus a measure of relative popularity for that web service can be obtained for the users of that name server[1].

## 4. VUNERABLE

As robust as DNS is with its distributed database, human errors can still crop up and hurt the overall system. Especially where system configuration, both physically and in software, is concerned[4]. This results in both severe delay in system responsiveness and in some cases the negation of the distributed nature of the system for a specific domain. An example that is used in [4] is a case of diminished server redundancy where the Microsoft DNS services became unavailable when the switch in front of all of their DNS servers went bad, a case where this error caused the advantage of having multiple DNS servers to be negated as none of them were available for use. While diminished server redundancy seems to be primarily a physical configuration issue, the next problem of lame delegation is a software configuration issue. This problem occurs when a server is listed as an authoritative server for a particular zone, but it cannot actually provide an authoritative answer[4]. Other than referring the request to a higher server in the hierarchy, another form of this is an "authoritative" server which simply does not respond at all, or gives an error saying that it is not set up correctly and cannot respond to any requests [4]. At the best one of the incorrect servers can return a cached value for the address, but at the worst, a domain or subdomain thought to be connected through several name servers could actually be hanging one only one, which not only introduces brittleness into the system, but also can dramatically increase response time from the servers. One more issue noticed by the authors of [4] is cyclic zone dependency, which occurs when two zones decide to help each other out and provide DNS services for each other. Unfortunately this involves looking for name server A which needs information from name server B, which need information from name server A. In complex environments this error can even occur when both systems are configured properly and the dependencies are made to be cyclic. The effects of this are really bad when all of one organization's name servers go down at once, possibly through diminished server redundancy mentioned earlier. Not only do that organizations services become disrupted, but also those of the other organization in the cycle, as their name servers depended on the first organization's name servers[4].

## 5. CONCLUSION

Even with the human caused errors that appear in DNS configurations, DNS is still a relatively stable workhorse. Not only does it allow us to use names to communicate across networks, but it also allows us to discover more about the networks and the traffic on them. From the beginnings of using names on a network with the host file, to the DNS system we have today has been a giant, well placed leap in the usability and convenience of the internet and other vast networks. One that has and will stand the test of time.

## 6. REFERENCES

[1] Craig E. Wills , Mikhail Mikhailov , Hao Shang, *Inferring relative popularity of internet applications by actively querying DNS caches*, Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, October 27-29, 2003, Miami Beach, FL, USA

[2] Liu, M. I. *Distributed Computing Principles and Applications*. Pearson Addison-Wesley, Boston, 2004

[3] P. Mockapetris "Domain Names - Implementation and Specification," RFC 1035, November 1987. [Online]. Available: http://www.faqs.org/rfcs/rfc1035.html

[4] Vasileios Pappas , Zhiguo Xu , Songwu Lu , Daniel Massey , Andreas Terzis , Lixia Zhang, *Impact of configuration errors on DNS robustness*, ACM SIGCOMM Computer Communication Review, v.34 n.4, October 2004