

Domain Name Server Project

By Tyrone K harmon Jr.

Professor Thomas Re | Operating Systems | 12/26/2017

# DNS BACKGROUND INFORMATION

The domain name server system came about as a result of several ideas about name spaces and their manageability. The proposals varied, however, a common idea was to focus on a hierarchical name space, with the names using the “.” As the character to mark the boundary between hierarchy levels. The DNS scheme was the result of an evolved system of several attempts to implement this scheme.

The primary goal was a consistent name space which is now used to refer to resources. It accomplishes this, with its three major components, domain name space/resource records, name servers, and resolvers. The combined implementation of these three elements assist in avoiding problems with ad hoc encodings, requiring names to be contained within network identifiers, addresses, routes, or similar informational schemes.

The following project focuses on the wrapping and unwrapping of a packet being sent to Hofstra’s DNS. All communication inside of the domain protocol are carried in a single format called a packet. Each package is formatted in 5 sections. The header, question, answer, authority, and additional sections. The header is always present, and includes fields that specify which of the remaining sections are present, and also whether the package is a query or a response. The question section is for the name server, the answer section is for the DNS to write in to answer the question, the authority section is for pointing towards any authority written from the response, and same with the additional section.

From the RFC documentation, one can discover the breakdown and meaning of each header. Each header contains: an ID, QR, Opcode, AA, TC, RD, RA, Z, RCODE, QDCOUNT, ANCOUNT, NSCOUNT, and ARCOUNT all in different bit field sizes to specify certain types of information about the rest of the packet. For example, the RCODE section (which is what helped me to format and understand my packet) is a 4 bit field that is a part of the response and contains values that can be interpreted as different errors received from the response.

## MY DNS PROJECT

The first challenge I had with starting the project, was understanding several different functionalities of C, that were not covered in previous courses. For example, using pragmas or type casting pointers. I found using malloc and sizeof() to help me understand the size of each structure in my packet to be most helpful, as Professor Re displayed in his office hours. Furthermore, by taking that data and casting it as a char array and then casting that array back as an int, allowed me to see if the data was corrupted or not. In addition, I pulled a lot of my resources from the RFC articles, and previous labs we have completed, such as the echo server, and shell programs we completed. For example, in order to send the packet, all we had to do was implement it just as it was implemented in the client-side code from the echo server, except changing the port to UDP port 53, and changing the address we are sending the packet to. Which could be hardcoded to Hofstra’s IP 10.2.2.2 or it can be taken in from the command line. However, if one wishes to implement the command line implementation, then they need to learn how to compress, and change the format of the address, to DNS format and back. Which is as well listed in the RFC.

When it came to creating the packet, that took learning a couple of functionalities as well. For example, I did not know one could assign the bit size of the members of a struct by formatting it as unsigned int ID: 16; (as an example). After I did that for each section of the header, I basically casted it to \*char and added it to a string, and then added that string to a buffer. This size of the buffer, I got from doing the following implementation ~~ header \*thehead = (header\*)malloc(sizeof(header)); except for each section of course. I also discovered the use of ntohs() and htons(). I have learned that these functions do a lot of the bit swapping we need for us. Mainly because, the DNS speaks big N-dian, and our packet is written in little N-dian.

After my first attempt to format the packet I received the following response from the DNS:

A screenshot of a cell phone

Description generated with very high confidence

The error I was running into was, I was receiving the RCODE 2, which means I am receiving a response, however, that response is telling me that I am having slight issues

Connecting to the server. This is due to the fact that I was not completely connected to Hofstra’s Big Lab VPN, however, fixing that was easy and not very relevant to the project. After I fixed that issue, and was able to appropriately connect, I received the following:

A screenshot of a cell phone

Description generated with very high confidence

This time, I was receiving RCODE 1, which means I am definitely successfully receiving a response, however, due to the fact that my packet is not formatted correctly, I have been given this error.

In order to fix this error, I looked through my packet, and noticed that I was indeed not completely formatting the header in the correct way. Which, is very unfortunate because the header and wrapping it correctly are the main components of this project. Especially, considering the fact that the header contains the information needed for interpreting the rest of the packet. In my packet, I discovered that I was not giving the opcode, rcode, qdcount, ancount, nscount, and arcount the appropriate amount of bit space. I gave them all 1 bits, when the opcode and rcode should have 4, and the rest should have 16. After finally formatting my packet correctly, I finally received an error free response from Hofstra’s DNS…. As listed below.

A screenshot of a cell phone

Description generated with very high confidence

It was very fun and exciting to see that I was actually receiving information from the DNS, and that my packet had successfully been wrapped, sent, received, sent back, and unwrapped! After this, I started researching more on DNS zone transfers (especially, because they will be taught in my future cyber security courses, and because it could be a fun side project), and I have come across a lot of useful information. I know, instead of UDP, I Have to use TCP, and implement many other things. However, I wasn’t sure if it would be safe or legal to experiment with that using Hofstra’s DNS, and attempting to get my packet to have Hofstra’s DNS dump its whole DNS table back, which would be interesting. From what I understand, the zone transfer is the method a secondary DNS server uses to update its information from the primary DNS server. Furthermore, DNS servers within a domain are organized using a master-slave method where the salves get updated DNS information from the master DNS. Now the problem is, many do not configure master DNS servers in a way that only allow zone transfers from secondary DNS servers. Which is why we can use this method to get a list of all the ip addresses as well as aliases assigned within the domain. By using the nslookup client, I think it is possible to issue this zone transfer request. However, that extends a little bit beyond this project, and I haven’t had a lot of time to experiment with it. However, I will continue to work on it after the duration of this course has ended. However, I am going to include as much research as I can on the topic, and attempt to do one myself.

From what I discovered, in order to attempt a zone transfer, a possible implementation would be to (on windows) just use the nslookup utility that is contained in windows. From here we just have to enter the target DNS server and the domain we are attempting to implement this on.

So it would implemented as:

Server [www.somedomain.com](http://www.somedomain.com) (this would be our target domain)

Set type=any (which would allow us to get all types of DNS records)

Ls -d somedomain.com (here is where the actual transfer is done)

HOWEVER, in linux this is far easier to do, by just using the dig utility to perform the DNS zone transfer

It would be as easy as dig somedomain.com axfr

After implementing either of these methods, I believe we would be able to tell if the DNS zone transfer is allowed, because you would receive an entire list of all DNS entries that have been made in the DNS server for that domain. However, if it wasn’t allowed, then the I would just receive the error indicating that the Zone transfer did not work.

I have actually found a website that helps teaching DNS zone transfers, by allowing its students to perform zone transfers on its DNS. I have attached below, the successful zone transfer that I implemented on the website on the next page, listed below.

All the information starts with the SOA recorded that is listed in one of the top lines. What is listed is information about the primary name server, containing contact details for the administrator and other key information.

From everything that was dumped we can find useful information, that we can use to perform other securities attacks. For example, there is a field that gives email addresses which can be used as a part of other attacks, and from the serial number that was dumber, if it is date based and checked regularly, we could use that to understand different kinds of activity in the company. This is probably why it is always worth checking for zone transfers on all available name servers, because there could always possibly be one or multiple servers that allow zone transfers. Which could all possibly leak useful information.

There is other useful information listed below such as MX records, which from my research, indicates where mail should be sent, I think. Which can indicate what email service a company uses. There is also a location value that was dumped, which gives the exact latitiude/longitude value of the client. There are also TXT records that were dumped, which could contain phone numbers, email addresses, different site verifications. To summarize, there is a lot of information you can use if you have a DNS server dump its table to your computer. That’s why it is important to secure all servers from this. Even test servers, which many companies use, and are many times not preventing zone transfers from being blocked. To verify the things I stated above, Here is the data from the zone transfer I did:

A screenshot of a computer

Description generated with very high confidence