DNS Cache Poisoning: Definition and Prevention

Tom Olzak
March 2006

The Internet would grind to a halt – would not be possible – without a Domain Name System (DNS). As you'll see in this paper, the proper operation of DNS is fundamental to the maintenance and distribution of the addresses for the vast number of nodes around the globe. So it would be too much to hope for crackers (malicious hackers) to ignore DNS as they continuously look for new ways to circumvent your security.

There are several facets to DNS security. In this paper we focus on one of the most dangerous types of attack – DNS cache poisoning. To provide a complete picture of this threat, we'll explore how DNS works, two ways crackers facilitate cache poisoning, what impact this type of attack can have on your organization, and steps you can take to protect your information assets.

What is DNS?

In the world of the Internet and TCP/IP, IP addresses are used to route packets from source to destination. A single IP address, for example 203.192.135.234, is not difficult to remember. But trying to learn or track thousands of these addresses, including which server/node is associated with each address, is a daunting task. So instead, we use domain names to refer to systems with which we want to communicate.

A real-world Internet domain name example is Google.com. When you enter the Google domain name into the address bar of your browser, the Google page appears. This is because your PC executed a process to resolve Google.com to an IP address. Only by having the IP address is a system able to initiate a session with another system across the Internet. Let's look at two ways IP address resolution can occur.

Figure 1 depicts the domain name/IP address resolution process when the target system and DNS server are internal. In this example, a workstation must establish a session with a server with a domain name of *Farpoint.company.com.* In order for a workstation to implement DNS, it must be running a DNS Client or Client *Resolver*. The resolver initiates the following process, resulting in the conversion of the domain name to an IP address (Microsoft TechNet, 2005).

Step 1: The resolver checks the resolver cache in the workstation's memory to see if it contains an entry for Farpoint.company.com. The entry would be present if the workstation had resolved the name to an IP address since the last time it was powered on, and the Time to Live of the entry had not been exceeded. In this example, no entry is found.

Step 2: Having found no entry in the resolver cache, the resolver sends a resolution query to the internal DNS server.

Step 3: When the DNS server receives the query, it first checks to see if it's authoritative for the company.com domain. In other words, is it responsible for managing the zone in which the company.com domain resides? If it is, the server performs a lookup in its internal zone table. In this case, it finds a host Resource Record (RR) that includes the IP address for Farpoint.company.com.

Step 4: The IP address of Farpoint.company.com is returned to the resolver.

Step 5: The resolved domain name and IP address are placed into the resolver cache. Figure 2 is an actual listing of the contents of a workstation's resolver cache. The IP address is used to contact Farpoint.

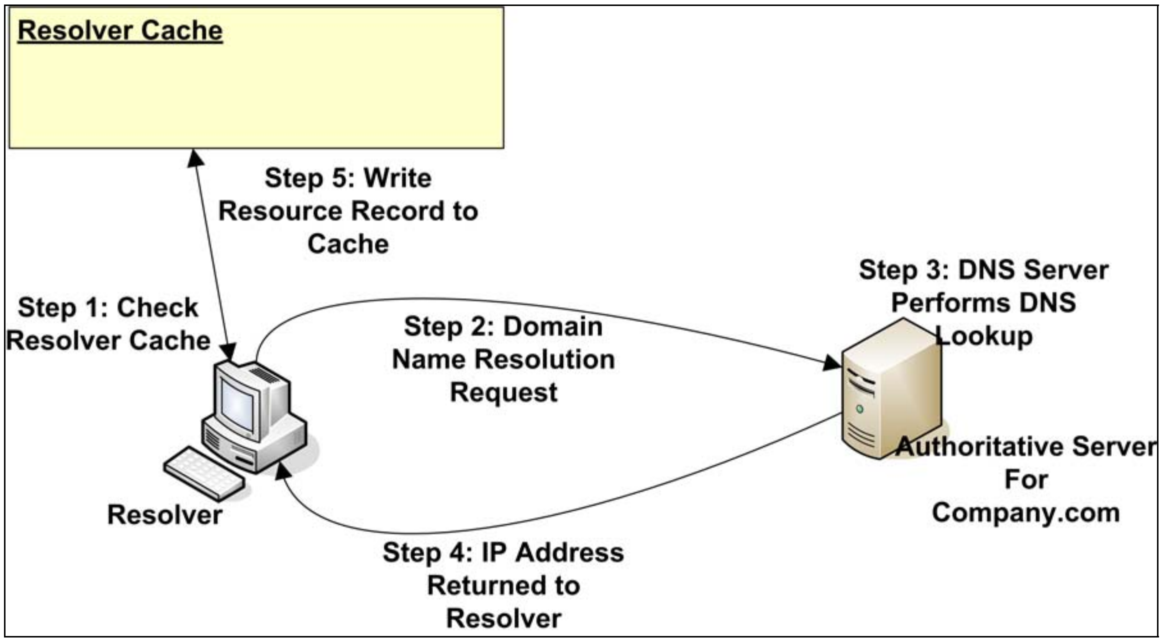
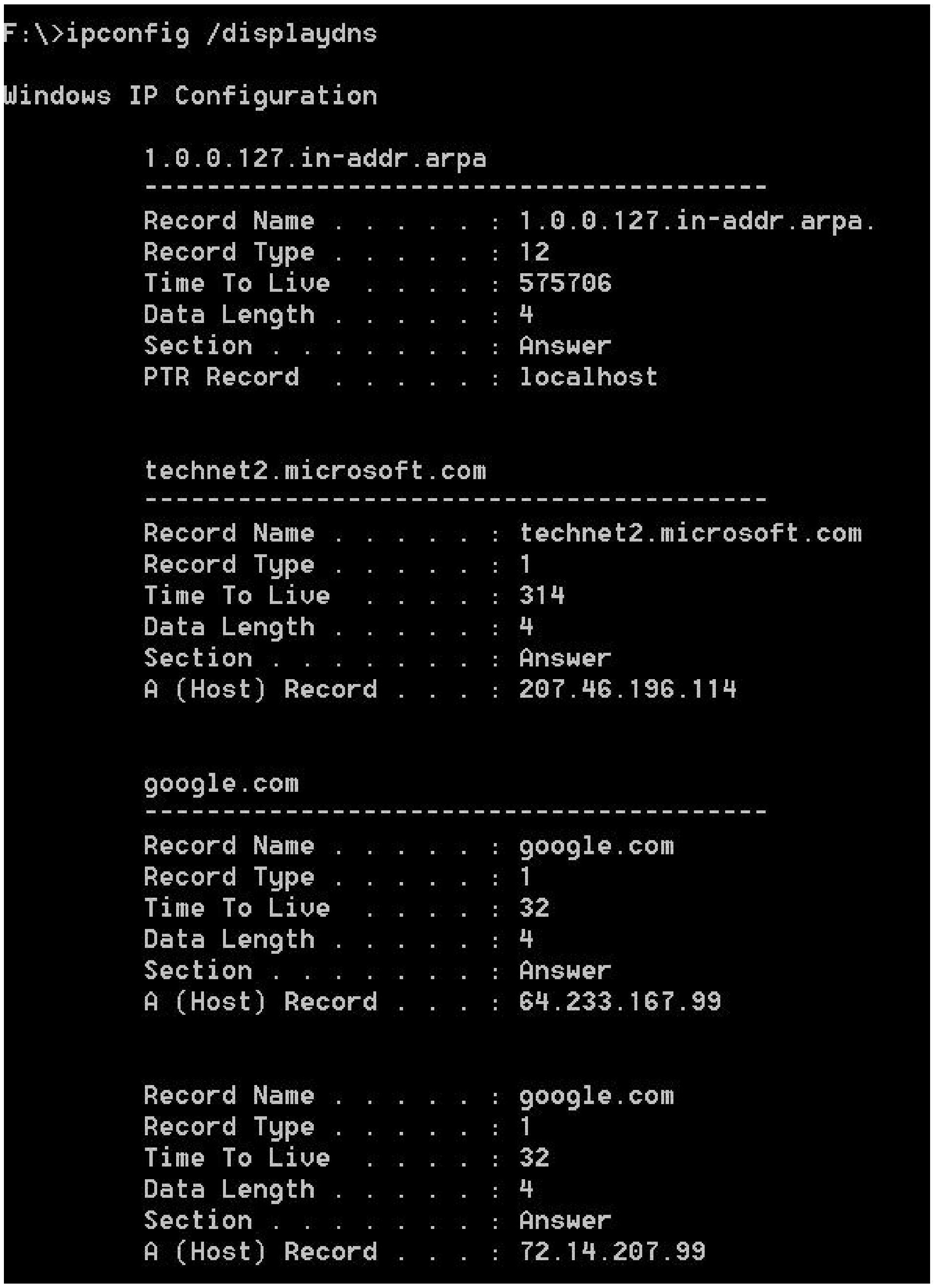

Resolver Cache
Step 5: Write Resource Record to Cache
Step 1: Check Resolver Cache
Step 2: Domain Name Resolution Request
Step 3: DNS Server Performs DNS Lookup
Resolver
Authoritative Server For Company.com
Step 4: IP Address Returned to Resolver

Figure 1: Internal DNS Server Lookup

In the previous example, the target server was located within the requestor's network. But there are many instances in which the target device is located somewhere on the Internet. In these cases, the process is somewhat different. Please refer to Figure 3 as we step through this second DNS resolution process.

Step 1: The resolver checks the resolver cache in the workstation's memory to see if it contains an entry for Farpoint.companyA.com.

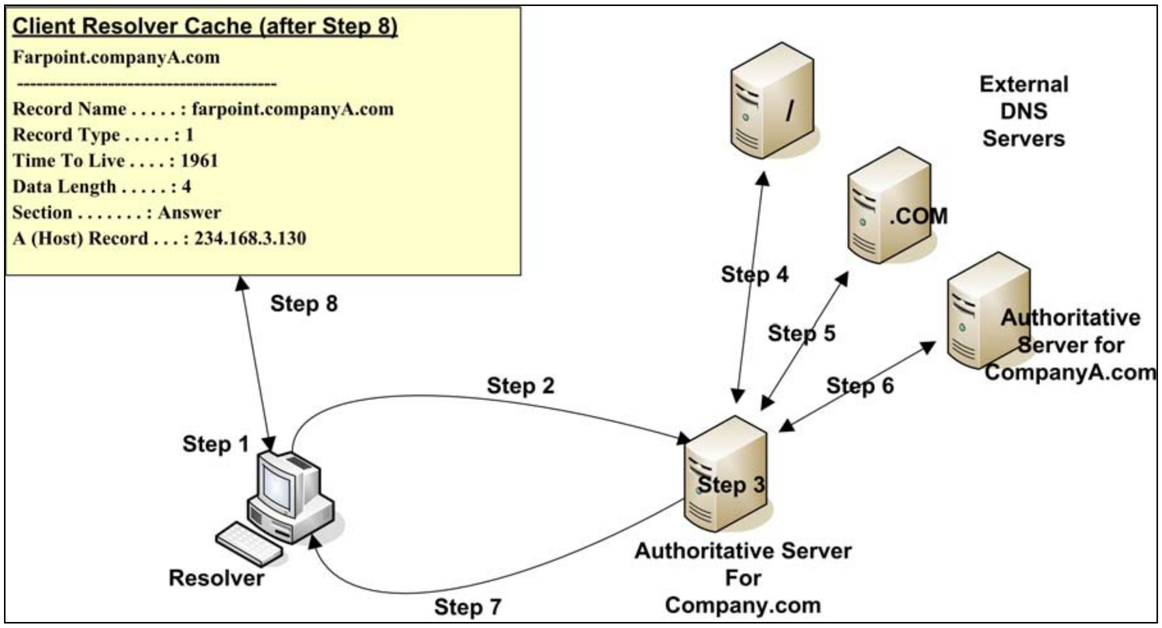
Step 2: Having found no entry in the resolver cache, the resolver sends a resolution request to the internal DNS server.



```
F:\>ipconfig /displaydns

Windows IP Configuration

         1.0.0.127.in-addr.arpa
         ----------------------------------------
         Record Name . . . . . : 1.0.0.127.in-addr.arpa.
         Record Type . . . . . : 12
         Time To Live  . . . . : 575706
         Data Length . . . . . : 4
         Section . . . . . . . : Answer
         PTR Record  . . . . . : localhost


         technet2.microsoft.com
         ----------------------------------------
         Record Name . . . . . : technet2.microsoft.com
         Record Type . . . . . : 1
         Time To Live  . . . . : 314
         Data Length . . . . . : 4
         Section . . . . . . . : Answer
         A (Host) Record . . . : 207.46.196.114


         google.com
         ----------------------------------------
         Record Name . . . . . : google.com
         Record Type . . . . . : 1
         Time To Live  . . . . : 32
         Data Length . . . . . : 4
         Section . . . . . . . : Answer
         A (Host) Record . . . : 64.233.167.99


         Record Name . . . . . : google.com
         Record Type . . . . . : 1
         Time To Live  . . . . : 32
         Data Length . . . . . : 4
         Section . . . . . . . : Answer
         A (Host) Record . . . : 72.14.207.99
```

Figure 2: Actual Resolver Cache Listing

Step 3: When the DNS server receives the request, it first checks to see if it's authoritative. In this case, it isn't authoritative for companyA.com. The next action it takes is to check its local cache to see if an entry for Farpoint.companyA.com exists. It doesn't. So in Step 4 the internal DNS server begins the process of iteratively querying external DNS servers until it either resolves the domain name or it reaches a point at which it's clear that the domain name entry doesn't exist.


Client Resolver Cache (after Step 8)
External DNS Servers
/
.COM
Authoritative Server for CompanyA.com
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Step 7
Step 8
Resolver
Authoritative Server For Company.com

```
Farpoint.companyA.com
 ----------------------------------------
Record Name . . . . . : farpoint.companyA.com
Record Type . . . . . : 1
Time To Live . . . . : 1961
Data Length . . . . . : 4
Section . . . . . . . : Answer
A (Host) Record . . . : 234.168.3.130
```

Figure 3: Recursive DNS Query

Step 4: A request is sent to one of the Internet "root" servers. The root server returns the address of a server authoritative for the .COM Internet space.

Step 5: A request is sent to the authoritative server for .COM. The address of a DNS server authoritative for the companyA.com domain is returned.

Step 6: A request is sent to the authoritative server for companyA.com. The IP address of Farpoint.companyA.com is returned.

Step 7: The IP address for Farpoint is returned to the client resolver.

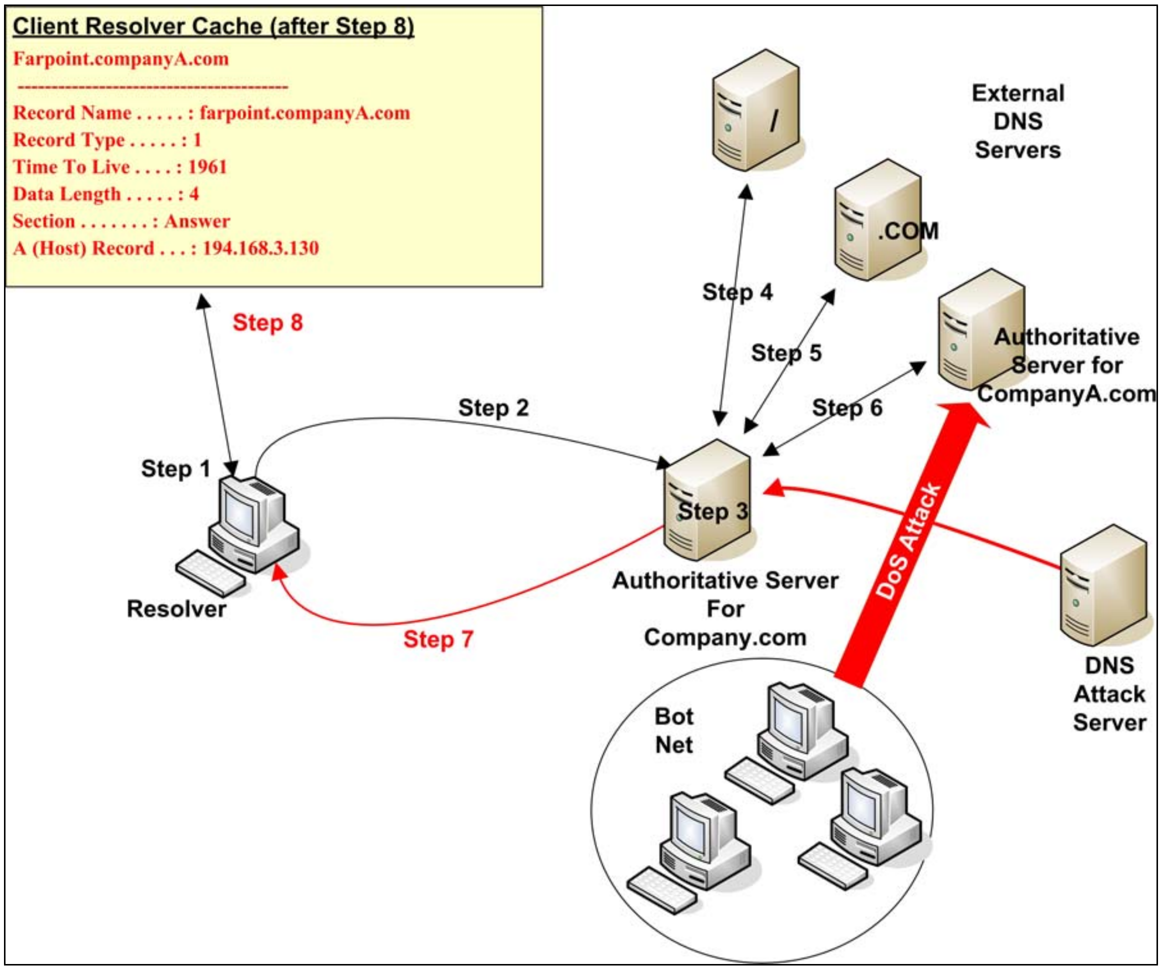
Step 8: An entry is made in the resolver cache, and a session is initiated with Farpoint.companyA.com.

This process, from the client resolver perspective, is known as a recursive query. With the proper software, the client resolver could perform the iterative query illustrated by Steps 4, 5, and 6.

Now that we have a good idea how DNS is supposed to work, it's time to look at how this process can be used to co-opt one or more DNS caches.

What is DNS Cache Poisoning?

DNS cache poisoning consists of changing or adding records in the resolver caches, either on the client or the server, so that a DNS query for a domain returns an IP address for an attacker's domain instead of the intended domain. To demonstrate how this might work, let's step through Figure 4.


Client Resolver Cache (after Step 8)
Farpoint.companyA.com
Record Name . . . . . : farpoint.companyA.com
Record Type . . . . . : 1
Time To Live . . . . : 1961
Data Length . . . . . : 4
Section . . . . : Answer
A (Host) Record . . . : 194.168.3.130
/
External DNS Servers
.COM
Authoritative Server for CompanyA.com
Step 8
Step 4
Step 5
Step 6
Step 2
Step 1
Step 3
Resolver
Authoritative Server For Company.com
Step 7
DoS Attack
DNS Attack Server
Bot Net
Figure 4: DNS Cache Poisoning

Step 1: The resolver checks the resolver cache in the workstation's memory to see if it contains an entry for Farpoint.companyA.com.

Step 2: Having found no entry in the resolver cache, the resolver sends a resolution request to the internal DNS server.

Step 3: When the DNS server receives the request, it first checks to see if it's authoritative. In this case, it isn't authoritative for companyA.com. The next action it takes is to check its local cache to see if an entry for Farpoint.companyA.com exists. It doesn't. So in Step 4 the internal DNS server begins the process of iteratively querying external DNS servers until it either resolves the domain name or it reaches a point at which it's clear that the domain name entry doesn't exist.

Step 4: A request is sent to one of the Internet root servers. The root server returns the address of a server authoritative for the .COM Internet space.

Step 5: A request is sent to the authoritative server for .COM. The address of a DNS server authoritative for the companyA.com domain is returned.

Step 6: A request is sent to the authoritative server for companyA.com. This is identical to the standard process for an iterative query – with one exception. A cracker has decided to poison the internal DNS server's cache. In order to intercept a query and return malicious information, the cracker must know the transaction ID. Once the transaction ID is known, the attacker's DNS server can respond as the authoritative server for companyA.com.

Although this would be a simple matter with older DNS software (e.g. BIND 4 and earlier), newer DNS systems have build-in safeguards. In our example, the transaction ID used to identify each query instance is randomized. But figuring out the transaction ID is not impossible. All that's required is time. To slow the response of the real authoritative server, our cracker uses a botnet to initiate a Denial of Service (DoS) attack. While the authoritative server struggles to deal with the attack, the attacker's DNS server has time to determine the transaction ID.

Once the ID is determined, a query response is sent to the internal DNS server. But the IP address for Farpoint.companyA.com in the response is actually the IP address of the attacker's site. The response is placed into the server's cache.

Step 7: The rogue IP address for Farpoint is returned to the client resolver.

Step 8: An entry is made in the resolver cache, and a session is initiated with the attacker's site. At this point, both the workstation's cache and the internal DNS server's cache are poisoned. Any workstation on the internal network requesting resolution of Farpoint.companyA.com will receive the rogue address listed in the internal DNS server's cache. This continues until the entry is deleted.

Another method used to poison a DNS cache is the use of a recursive query sent by the attacker. The query can force the target server to connect to the authoritative source of the domain in the query. Once connected, rogue information about one or more domains might be sent to the querying server and posted to the server's cache.

There are other methods attackers use to poison DNS caches, but the objective is the same. Now we'll explore the consequences of using a poisoned DNS cache.

Potential Consequences of Cache Poisoning

Pharming is the primary risk associated with cache poisoning. Crackers employ pharming for four primary reasons (Hyatt, 2006): identity theft, distribution of malware, dissemination of false information, and man-in-the-middle attacks.

Identity Theft

Once an attacker gets you to his site, he'll try to trick you into leaving behind information he can use to impersonate you. One way to do this in our first example is to create a site identical to the real Farpoint.companyA.com. When the user connects using the poisoned cache information, she might be fooled into entering information about herself through apparently legitimate requests for her name, social security number, address, etc.

Distribution of Malware

Another of objective of attackers using cache poisoning is the automatic distribution of malware. Instead of releasing malicious code into the Internet and realizing random results, the use of rogue IP addresses to redirect unsuspecting users to the attacker's site can be a more focused attack vector. Once a workstation initiates a session with the malicious site, malware is uploaded to the workstation without intervention by or the knowledge of the user.

Dissemination of False Information

This aspect of pharming is useful to attackers who want to spread self-serving information about an organization. It's also been used to manipulate stock prices in an attempt to realize a large profit.

Man-in-the-middle Attack

In this attack type, the workstation initiates a session with the attacker's server. The attacker's server initiates a session with the actual target site. All information flowing between the workstation and the genuine site passes through and is intercepted by the cracker's server.

There can be serious consequences when security is an afterthought during the configuration and deployment of DNS servers. The next section provides guidelines that can help prevent cache poisoning.

Protecting Your Assets

The first layer of defense against cache poisoning is the use of the latest version of DNS. DNS based on BIND 9.3.x or Microsoft Windows Server 2003 is far more secure than DNS implemented with earlier versions. Successful completion of our first poisoning example would have been more difficult, because these systems also randomize the port used for the DNS query in addition to the transaction ID.

Recursive queries should be limited to internal DNS servers. If Internet facing recursive queries are required, only queries from internal addresses should be accepted. This will help prevent outside systems from sending queries with malicious intent. The following list provides additional guidance (Hyatt, 2005):

- ▲ Physically separate external and internal DNS servers
- ▲ Restrict zone transfers to authorized (secondary servers) devices

- ▲ Use TSIG to digitally sign zone transfers and zone updates – one of the best ways to prevent poisoning is to force identification of the sending authoritative source
- ▲ Restrict dynamic DNS updates when possible
- ▲ Hide the version of BIND being used on the DNS servers
- ▲ Remove unnecessary services running on the DNS servers
- ▲ Where possible, use dedicated appliances instead of multi-purpose servers

Conclusion

DNS cache poisoning is a large risk for organizations running early versions of DNS solutions. Elevated risk extends to companies that carelessly deploy DNS, regardless of the DNS software version used. Careful attention to version management and the secure configuration of DNS devices should reduce risk from cache poisoning to an acceptable level.

Works Cited

Hyatt, M. (2005, June). *Five steps to minimize DNS cache poisoning.* Retrieved March 14, 2006 from http://searchwindowssecurity.techtarget.com/tip/1,289483,sid45_gci1101998,00.html

Hyatt, R. (2006, January). Keeping DNS trustworthy. *The ISSA Journal*. January 2006, p. 37-38.

Microsoft TechNet (2005, January). *How DNS query works*. Retrieved March 14, 2006 from http://technet2.microsoft.com/WindowsServer/en/Library/1fd5b3be-ca29-43d0-b5e2-8a65192d5b781033.mspx