IN THIS CHAPTER

- Understanding how DDoS attacks work
- Learning about cache poisoning attacks
- Discovering how malware uses DNS

Chapter 2
Threats to DNS Security

DNS is becoming a more common target of network attacks. As one of the oldest and most relied-on protocols of the modern Internet, DNS is the cornerstone to almost all other services and protocols. This makes DNS an appealing target to attackers.

Because it is one of the most relied-on protocols, stopping attacks can't be as simple as adding a firewall rule. It's good to know how these attacks work before discussing solutions to stop them. In this chapter, we show you a few common types of DNS-based attacks; in the next chapter, we focus on remediation and solutions.

DDoS Attacks

There are many types of *distributed denial of service* (DDoS) attacks. You probably see them in the news these days, often accompanied by a ransom to induce the attacker to stop the DDoS attack. When it comes to DNS, you can look at specific types of attacks that are used to overwhelm DNS servers, thus rendering the DNS service unavailable. When an attack on the DNS is successful, it can bring an organization to a screeching halt. When a company can't publish the addresses for its web and mail servers, business stops.

The two main attack methodologies you want to look at are *amplification* and *reflection*. While technically two different attack tactics, attackers often combine amplification and reflection attacks.

Amplification

An *amplification attack* is a technique where a small query can trigger a large response, such as querying for a TXT record or a zone transfer when you haven't secured zone transfers to only your trusted sources. By flooding the server with short requests that require long responses, even a relatively weak computer can overload a DNS server. The DNS server is so busy doing the heavy lifting to respond to all these bogus requests that it doesn't have time to respond to legitimate ones.

DNS servers make surprisingly good amplifiers. Here is a simple example:

If a user makes a DNS query for "*isc.org/ANY*," the query is 44 bytes long, and the response is 4077 bytes long. That is around 93 times amplification!

WARNING

This example and some simple math show how devastating amplification can be. Say an attacker is generating queries with a *botnet* (a network of independent hosts, or bots, infected with malware), and each bot has a measly 1 Mbps connection to the Internet:

With a 1 Mbps connection, each bot could send the 44 byte query from the previous example approximately 2,909 times per second.

Because each query results in a response that is 4,077 bytes, you can calculate that the DNS server has to come up with about 93 megabytes each second.

If the botnet contains 11 bots all doing the same thing, that's a total of over 1 gigabytes that the DNS server is supposed to send out every second of the attack!

Amplification alone makes an effective attack, because attackers can use less powerful resources to overload more powerful servers. However, what makes DNS an even greater target for devious DDoS is reflection.

Reflection

A *reflection attack* sends queries that look like they came from the victim of the attack. The response (often a large, amplified answer) is sent to the victim, who never asked, and the amount of the response traffic could potentially overwhelm the victim's network.

In a reflection attack, an attacker sends a query to a recursive name server with a spoofed source IP address. Instead of his real IP address, he places the target (victim) IP address as the source IP address. The recursive name server does the legwork, retrieves the answer to the query from the authoritative name server, and sends the answer to the unsuspecting victim.

Combination attacks

Now the attacker combines the two techniques by spoofing the victim's IP address and sending a carefully crafted query that will result in a large payload. This is a very effective DDoS attack; the authoritative name server provides the amplification, and the recursive name server provides the reflection. This allows the attacker to attack two different victims at the same time. It also causes the victim of the amplification attack to possibly believe he or she was attacked by the second victim, causing potentially even more mayhem.

Figure 2-1 shows an example of combining amplification and reflection to cause DDoS. As you can see, this attack consumes the resources of not only the victim, but also those of the recursive name servers. This is why securing a publicly facing recursive server against recursion from the world is one of the most important best practices in DNS Security. It prevents you from playing the role of amplifier.

Here is how the attack proceeds:

1. An attacker-controlled botnet sends DNS queries with a victim's spoofed IP address to recursive name servers.
2. Recursive name servers follow the standard DNS path and send the query to other authoritative name servers.
3. Recursive name servers receive the (amplified) response from authoritative servers.
4. Recursive name servers send the (amplified) response to the victim, whose IP address was listed as the source address in the original query.

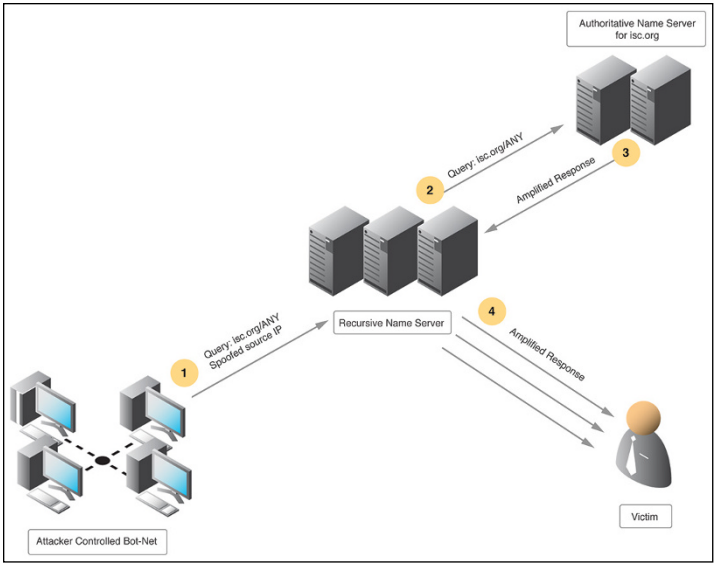

FIGURE 2-1: An amplified reflection attack.

Cache Poisoning

Also known as *DNS spoofing*, cache poisoning focuses on corrupting the cached answers on the recursive name servers, either through software exploits or protocol weaknesses.

Figure 2-2 describes an older-style DNS cache poisoning by attaching malicious answers to the ADDITIONAL section of the DNS response. An older, less secure recursive name server would accept the RRset in the ANSWER section, and also accept the RRset in the ADDITIONAL section, even if those records were unrelated to the question being answered.

Here's how that variant of the cache poisoning attack works:

1. Query a recursive server for foo.example.com.
2. Send replies spoofing the example.com name server's IP address and with various query IDs, including malicious records in the Additional Section, in this case redirecting all queries to a com name server to the bad guy's DNS server at 10.17.34.25. (Of course, this would need to be a reachable IP address in a real attack.)

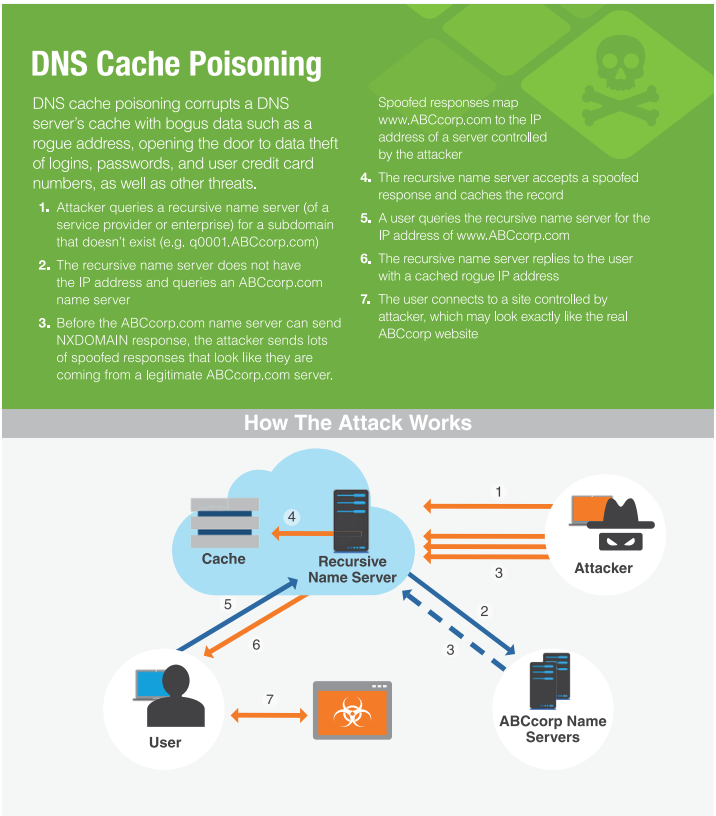

FIGURE 2-2: A cache poisoning attack can land unsuspecting website visitor on nefarious pages.

```
;; ANSWER SECTION:
foo.example.com 3600 IN A 10.17.34.25
;; ADDITIONAL SECTION:
a.gtld-servers.net.   1540000 IN  A
  10.17.34.27 ; (bad guy's IP address)
```

REMEMBER

Name servers have been patched against this older cache poisoning attack for some time. This was accomplished by requiring DNS servers not to accept additional data that is unrelated to the original query.

One of the most (in)famous cache poisoning attacks was discovered by an American computer security expert, Dan Kaminsky,

in 2008. The attack Kaminsky discovered takes advantage of a weakness in the DNS protocol itself. When a recursive name server makes a query, it sets the value of five fields. If it receives a response with matching values for these fields, with an answer to the question that it asked, then it accepts the answer as legitimate, returns it to the client, and stores it in its cache.

As it turns out, four of the fields are easy to figure out and spoof, so only one field is actually meaningful: the Query ID or TXID. The Query ID has relatively few possible values, by computer standards. It is a 16-bit number, or a value between 0 and about 64,000. This makes it possible for an attacker to initiate an information request, then when the recursive server asks the next server up the tree, the attacker tries his luck firing a lot of responses at the recursive name server that look like they came from that authoritative server. If the attacker gets lucky and guesses the right query ID, the original server stores the bad data from the attacker's bogus response in cache and passes it on if any other servers ask for it.

A well-crafted Kaminsky attack can insert a bogus entry in the cache within minutes. To make matters worse, the attacker could choose to poison an NS record, thus taking command of an entire domain rather than just a single entry. The most common fix for the vulnerability sends queries from random source ports, since the response must arrive at the same random port. Having two values to guess correctly makes crafting an acceptable bogus response much more difficult. The attack becomes more time-consuming, but there is no surefire way to guard against this type of attack without changing the protocol itself. Fortunately, the enhanced DNSSEC helps with this problem. We talk about this in Chapter 3.

Malware and Exfiltration

The term *malware* comes from combining the words "malicious" and "software." Malware is a category of software designed with malicious intent. Often, malware is installed without the user's knowledge and operates in the background, safely hidden from attention.

REMEMBER

Although malware is typically distributed via web servers, disguised as part of the web content, its foundation is in DNS. Modern attackers don't rely on a single, registered domain name such as malware.not-evil.example.com. Instead, they create a complex system that automatically registers hundreds or thousands of domain names on the fly and configures them to point to web servers that distribute malicious content. Changing their domain names constantly makes it difficult to block them at a DNS level.

Once it has infected a computer, malware uses DNS to resolve a list of domain names of possible command-and-control servers until it finds one that is available and responds. Command-and-control servers are special servers maintained by the malware owners that tell the infected client what to do. The list of command-and-control servers can be compiled into the malware, or synthesized using what's known as a domain-generation algorithm.

Malware can also sneak sensitive data out of your network, a technique known as *data exfiltration*. Because it is done over DNS, data exfiltration is also known as *DNS tunneling*.

Although there are many DNS tunneling implementations out there, all of them rely on the capability of clients to perform DNS queries. DNS tunneling software allows users to do:

- Relatively innocuous things, such as getting free airport Wi-Fi
- Potentially dangerous acts, such as using SSH to evade corporate firewalls
- Outright malicious activities, such as stealing sensitive information

In the case of the last, clients evade detection by breaking data down into query-sized chunks, disguising sensitive data as DNS queries, and sending them to malicious DNS servers on the far end who can unpack these queries and reconstruct the data.

It's like stealing someone's car without opening the garage door: You have to break the car down into small chunks that fit through the doors and windows, and then rebuild the car outside. Except in the case of data exfiltration, the malware breaks down files, sometimes even encrypting each chunk, before sneaking them off your premises to reassemble.

Figure 2-3 illustrates a simplified flow of how data exfiltration works over DNS. The steps in Figure 2-3 go like this:

1. The attacker registers the domain name ZG5ZC2VJDXJPDHKK.COM, and sets up name server NS1.ZG5ZC2VJDXJPDHKK.COM.
2. The infected client encodes stolen information, in this case the text "Pa$$w0rd" into "UGEKJHCWCMQK."
3. The client makes the DNS query for the domain with the encoded password as a subdomain: UGEKJHCWCMQK.ZG5ZC2VJDXJPDHKK.COM.
4. A recursive name server finds the authoritative name server NS1.ZG5ZC2VJDXJPDHKK.COM and sends the query there.
5. The attacker recognizes the subdomain value as the encoded password. The attacker decodes the information UGEKJHCWCMQK back to recover "Pa$$w0rd."

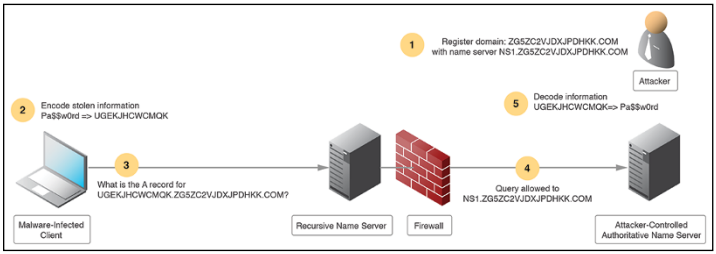

FIGURE 2-3: DNS tunneling can sneak your passwords into DNS queries that look like gibberish.

In this example, it is not necessary for the client to receive a response from the malicious server, because the goal was to send information out; however, the process can easily include the malicious server sending back an exploit to be executed on the infected client.

An example of a well-publicized malware attack that leveraged DNS is WannaCry. WannaCry was (or is) *ransomware* — a type of malware that encrypts valuable data on a computer and charges the user a price to get it unencrypted — that used a vulnerability in the Windows platform at the time. Once inside the network, it could infect any unpatched Windows computer using SMB, Microsoft's remote file sharing protocol. Users of the infected computers saw a screen like in Figure 2-4.

FIGURE 2-4: Unlike attacks that just cause mayhem, ransomware attacks your pocketbook directly.

Where DNS was noticeably involved was in WannaCry's decision to encrypt data. The malware incorporated a kill-switch using DNS, first checking for the existence of an obscure domain name via a DNS query and, when it was not found, encrypting files and holding the computer hostage. If the creators of the malware needed to stop the ransoming they could just register the domain.

Once the "good guys" discovered this fact, they could address WannaCry in two different ways.

1. DNS providers could look for queries for that domain name to identify clients infected with WannaCry.
2. Good guys could register the kill-switch domain name that told the malware not to ransom data, in effect neutering the ransomware until the code could be changed to point to other not-yet-registered domain names.

In this example, the malware developer's use of DNS led to the possibility of easy remediation. In the next chapter, we look at more way to prevent and resolve DNS attacks.