DNS Security

Every machine on the internet has an IP address. These are addresses like 10.40.112.243 or 192.168.8.95. Those examples are IPv4 addresses. IPv6 addresses are much more hideous, like fe80::c44a:fcc0:780b:1c04. Just like the phone system uses numbers to route phone calls to the right place, the internet uses IP addresses to route packets to their destination. And just like your phone has an address book that stores easier-to-remember names with the phone numbers, the internet has a system called the *domain name system* (DNS) for this purpose.

Domain names are generally meant to be human-readable. They are things like `www.msmary.edu` or `mail.google.com`. Domain names get more general as you move from left to right. So in `www.msmary.edu`, `www` is a subdomain of `msmary`, which in turn is a subdomain of `edu`. That `edu` is called a top-level domain or TLD. Other common TLDs include `com`, `org`, `net`, and `gov`, as well as country TLDs, like `uk` or `nl`.

DNS is used like your phone's address book to tie domain names to IP addresses. This is important not just because IP addresses are hard to remember, but because they change frequently and organizations often have several IP addresses tied to the same domain.

When you want to visit a particular site, one of the first steps in the process is usually to use DNS to find out the IP address of the site. Most often, your computer will contact another machine, called a *DNS resolver*, to do the lookup for you. That resolver is usually either a machine on the network you are on or it may belong to your ISP. There are also public resolvers, like Google's 8.8.8.8.

That lookup takes multiple steps. All of the DNS information used to be contained in a single file on the internet, but that quickly become unmanageable, and DNS was broken up into a hierarchical, distributed database. Each part of the database knows about a particular subset of all the information. The terminology people sometimes use is to say that it is *authoritative for its zone*.

At the top of the hierarchy are *root DNS name servers*. They are responsible for knowing the addresses of *TLD name servers*. Those TLD servers are in turn responsible for knowing the addresses of name servers in that domain. For instance, the `edu` name servers are responsible for knowing the addresses of the Mount's name servers as well as those of MIT, Harvard, etc. The `com` name servers are responsible for knowing the addresses of the name servers for `google.com`, `apple.com`, etc. There are many root and TLD name servers spread around the world to handle the bazillions of DNS queries that happen every day.

Below the TLD level, each organization with a domain name is responsible for maintaining its own name servers that know the IP addresses associated with that domain. For instance, the Mount's name servers know the address of `msmary.edu`, `www.msmary.edu`, `email.msmary.edu`, etc. Some larger organizations will even have multiple levels of name servers, with each responsible for a particular set of subdomains. Companies will often use a third party service to manage their name servers.

So when your computer needs the IP address associated with a domain name, it will first contact a resolver. That resolver does a lookup by first contacting a root name server. That root server returns the address of the appropriate TLD server. The resolver then contacts the TLD server, which returns the address of the organization's name servers. The resolver then contacts those name servers to hopefully get the IP address it is looking for, though the process could possibly take a few more steps if the organization has multiple levels of name servers.

Since this process involves contacting several machines and waiting for replies, it can take a decent amount of time. So resolvers will often store (cache) the results. They are usually held anywhere from a few seconds to a few days. Your OS and web browser may also cache the results to avoid having to contact the resolver.

We will look at a variety of things regarding DNS security. One of the overriding themes is people trying to subvert the process to direct you to a phishing site. Often that site will be designed to look like a real site, but the goal is malicious, such as installing malware on your machine, getting you to buy stuff, or stealing your credentials.

Domain name security

Remember that domain names get more general as you move to the right. So in `email.msmary.edu`, `email` is a subdomain of `msmary.edu`. However, in `msmary.email.edu`, `msmary` is a subdomain of `email.edu`. So even though the domain name might look like it belongs to the Mount, it doesn't; it belongs to `email.edu` (if there even is such a place).

Sometimes people buy up domain names that are small variations on a real domain name, like `gogle.com` or `googgle.com`, trying to catch people that misspell a domain name. Companies like Google probably own the most common variations on their name, but smaller companies often don't. Also, there are many TLDs now, so while I may own `brianheinold.net`, I don't own `brianheinold.com` or `brianheinold.io`.

Domain names expire, often every year, and sometimes companies forget to renew their domain names. If someone jumps in right after a domain name expires, they can grab it.

Another thing to be careful of is that domain names, at least when you put them into a browser's URL bar, can have special characters. For instance, the character *a* has character code 97. However, the Cyrillic lowercase *а* looks identical to the ordinary *a* but has character code 1072. In some browsers, if you visit a link that has an ordinary *a* replaced with a Cyrillic one, the domain name will not be the one you expect.

The hosts file

Before contacting a resolver, your computer will look at a particular file on your computer called the *hosts* file. This file has a table of domain names and the IP addresses they map to. Here you can manually say which IP address a domain name on that computer should resolve to, even if it's not actually the real IP address associated with that file. For instance, you could look up the IP address of `hood.edu` and put an entry in your hosts file pointing `msmary.edu` to Hood's IP address. Then if someone tries to go to `msmary.edu` on that computer, they will instead end up at `hood.edu`.

A common use of this file is to "black-hole" certain domains, like tracking sites, by setting the IP to 0.0.0.0. This IP address goes nowhere, which has the effect of preventing that site from loading in your browser. Malware can also modify the hosts file. Sometimes they will black-hole domains associated with antivirus software. They could also put in entries for sites like gmail or your bank to point to the IPs of phishing sites designed to look like the real sites.

The hosts file is at `/etc/hosts` on Mac and Linux and at `windows/system32/drivers/etc/hosts` on Windows. Note that `etc` is a hidden directory in Windows.

Open resolvers

Often your resolver is run by your ISP. On the Mount's network, we have our own DNS resolvers. Recall that these do the DNS lookups for us. If the Mount wants to block people from reaching certain sites, they can do so by having their resolver return bogus IP addresses for queries for those sites. Your cable company ISP could theoretically do this, and ISPs in other countries use this technique to block people from getting to banned sites.

One way to get around this is to use an open DNS resolver. Several companies run these. The most well known is Google's 8.8.8.8. Cloudflare runs one at 1.1.1.1, and there are others. You can go into your OS's settings to change your DNS settings to use one of these resolvers if you don't trust your ISP.

DNS cache poisoning

The basic idea of this attack is to replace the answer to a DNS query with a bogus answer. The goal is often to direct people to a site other than that which they are trying to get to. If an attacker is on a local network with their target, this is pretty easy to do. The attacker sniffs traffic on the network, watching for DNS queries. When they see one, they send a reply that looks like it's legitimate, but it contains the attacker's desired IP address in

the reply. As long as the attacker's answer gets there before the real answer, the attacker's answer will be accepted.

It's possible for an attacker to do this from a remote network, but it's considerably harder. In order for the attacker's answer to be accepted, they must have a transaction ID and port number that matches the original query. That information is plainly visible when sniffing traffic on a local network, but it's not available to a remote attacker. There are over 60,000 possible transaction IDs and over 60,000 possible port numbers, so an attacker could make a lucky guess, but that's not likely. However, the attacker can send a bunch of replies at once, and by a careful application of the birthday problem, it is possible to get this down to a manageable number. However, the attack is difficult on modern systems. But some older DNS systems used predictable transaction IDs and port numbers, which made the attacker considerably easier.

There many places an attacker could insert a bogus answer during the DNS query process. Here are a few:

1. Send a bogus reply to someone making a DNS request (this was discussed above)
2. Send a bogus reply to a resolver. This reply will be cached, and it will affect anyone who uses the resolver as long as the reply is in the cache.
3. Hack a resolver to put a bogus entry on it. Just like above, this affects anyone who uses the resolver.
4. Hack a company's name servers. If someone hacked the name servers for `msmary.edu`, then most people in the world who try to go to `msmary.edu` will get a bogus answer (at least once the entries in their local cache expire).
5. Hack a domain registrar's TLD servers. Each TLD is administered by a particular organization. Chances are the big TLDs like com and edu are well secured, but some of the more obscure ones like pizza or gs might not be. If an attacker is able to put entries into the pizza TLD servers, then people going to `emmitsburg.pizza` could be redirected to a phishing site that would collect their payment details.
6. Hack people's routers and change their DNS settings. Your home WiFi router has a web interface where you can change various settings on it, including what to use as the network's DNS resolver. These web interfaces are protected by a username and password, but many people don't change them from the default manufacturer's values. Those values are easy to look up online, and are often simple things like admin/admin or admin/password. This could be changed by someone with access to your network or even remotely if your router has an easily accessible public IP address.

Evil twin access points

An access point is a device people use to connect to a WiFi network. For about $50, you can buy a USB device to turn your computer into an access point. You could then observe the broadcasts of a legitimate access point, copy all the details, and make your access point look just like the real one. This is an evil twin access point.

When people connect to an access point, that access point sends them various network details, like an IP address on the network and the address of the WiFi network's DNS resolver. The person running the evil twin can also use free DNS software called BIND to turn their computer into a DNS resolver. So when people connect to their evil twin access point, the attacker will send them the address of the resolver running on their computer. Since the attacker controls this, they can put whatever entries they want into it.

The last step of the process is for the attacker to start up their access point and start spoofing deauthentication frames to boot people off the real network. These are easy to create. The attacker makes sure that their network has a stronger signal than the real one, and since their network looks just like the real one, the deauthenticated people's computers will try to connect to the attacker's network.

This is a fairly easy attack to pull off. It's one of the reasons you have to be careful using public WiFi in places like hotels, airports, and coffee shops.

Solutions

There are a few solutions to some aspects of DNS attacks. One is DNSSEC, which adds some cryptographic authentication to the process. However, even though it has been around for a while, it has some issues and has not been widely adopted.

A recent solution is DNS over HTTPs. Instead of going through a local resolver, your web browser will send the DNS request over HTTPs to an open resolver, like 8.8.8.8 or 1.1.1.1, who will return the answer back over HTTPs. The whole query is encrypted, and it would be quite difficult for an attacker to insert a fake answer into the process. However, DNS over HTTPs makes it harder for some organizations to use DNS to block certain sites, which they may do for security purposes. Other people are concerned with giving more power to big companies like Google over basic internet services like DNS.

DNS and denial of service attacks

A *denial of service attack* (DoS attack) is any attack that prevents people from using a resource. The most common type involves sending a lot of network traffic to a device to either crash it or chew up so many resources that legitimate users can't access it. A particular class of DoS attacks are *distributed denial of service attacks* (DDoS) that involve many machines all sending traffic to the target. This could be done by a group of people coordinating their efforts, or it could be done by a single attacker using a botnet of hacked machines under their control.

One particular DoS attack was directed a company called Dyn, which hosts name servers for a large number of organizations. When you have a domain name, you are supposed to have name servers so that DNS can work to resolve your domain name to an IP address. Many companies outsource this to other companies, with Dyn being one of the biggest players. A DoS attack on Dyn in 2016 made large parts of the internet, especially on the east coast of the US, inaccessible for a few hours. The name servers were taken offline by the DoS attack, which meant DNS queries could not be answered. Without the IP addresses they would get from DNS, people were not able to visit the sites they wanted to.

DNS reflection/amplification DoS attacks This is a somewhat common type of DoS attack over the last decade or so. It's called either DNS reflection or DNS amplification. The reflection in the name comes from the fact that instead of the attacker directly sending traffic to the target, they bounce it off another machine. The idea of an amplification attack is for the attacker to use a small amount of traffic to turn into a large amount of traffic. To flood a target with a ton of traffic, you first have to generate a ton of traffic, and there's a limited amount of traffic most devices can generate. Certain features of DNS allow an attacker to make a query of maybe 50 bytes and get a large response, like around 4000 bytes. The attacker directs that response to the target.

There are four machines involved: the attacker, the target, a misconfigured DNS resolver, and a DNS name server with a large DNS resource record. The resolver is misconfigured in that resolvers really only should perform DNS queries for people on the resolver's network, but some are set to perform them for anyone who asks. As for the large record on the name server, there are various types of DNS records. Some are simple like A records, which just contain an IP address, but there are other records called TXT records, which can be fairly large, as well as DNSSEC records, which can also be very large.

The attacker starts by asking the misconfigured resolver to query the DNS name server with the large record. The attacker "spoofs" their address to be that of the target. When sending a packet it's possible to put a fake "return address" (source IP address) on it. Spoofing it to be the target's IP means the target will get the DNS response, which will be that large record.

The attacker sends a query that is around 50 bytes, and it results in up to around 4000 bytes at the target. This is an 80 times amplification factor. So if the attacker can generate 10 Mbps of traffic, that results in 800 Mbps of traffic at the target. However, this requires a lot of effort on the part of the resolver and the name server, so the attacker would use several of them. The attacker would also not likely use their own device to send the queries, but instead would use their botnet. There is a related attack called NTP reflection that uses the Network Time Protocol (NTP) in a similar way.