CYBER ATTACKS EXPLAINED: DNS INVASIONS

We often read about defaced websites whose pages get changed to some malicious content. How do hackers do it and how do we protect our infrastructure from them? This article is about how hackers invade DNS (domain name systems). DNS invasion is technically advanced and a harmful attack on a network or Web infrastructure. Network administrators are urged to learn more about it and strive to secure the infrastructure they manage.

As we all know, the DNS exists because it's impossible for humans to remember IP addresses for sites, but easy to remember alphanumeric names. The DNS system was created when the Internet was a friendly place — leading to quite a few issues.

Figure 1 shows how name resolution fundamentally works. When an application (like a browser) wants to connect to a destination service, it queries the DNS server, asking for the IP address. This query is sent over UDP port 53 as a single request and receives a single-packet reply from the server. (Note that since UDP data space is limited to 512 bytes, the protocol stack automatically uses the TCP protocol for queries and replies.) When the client receives a reply, it updates its local cache with the received entry, speeding up subsequent queries to the same domain. Entries in local cache are automatically purged after their TTL (Time to Live) expires.

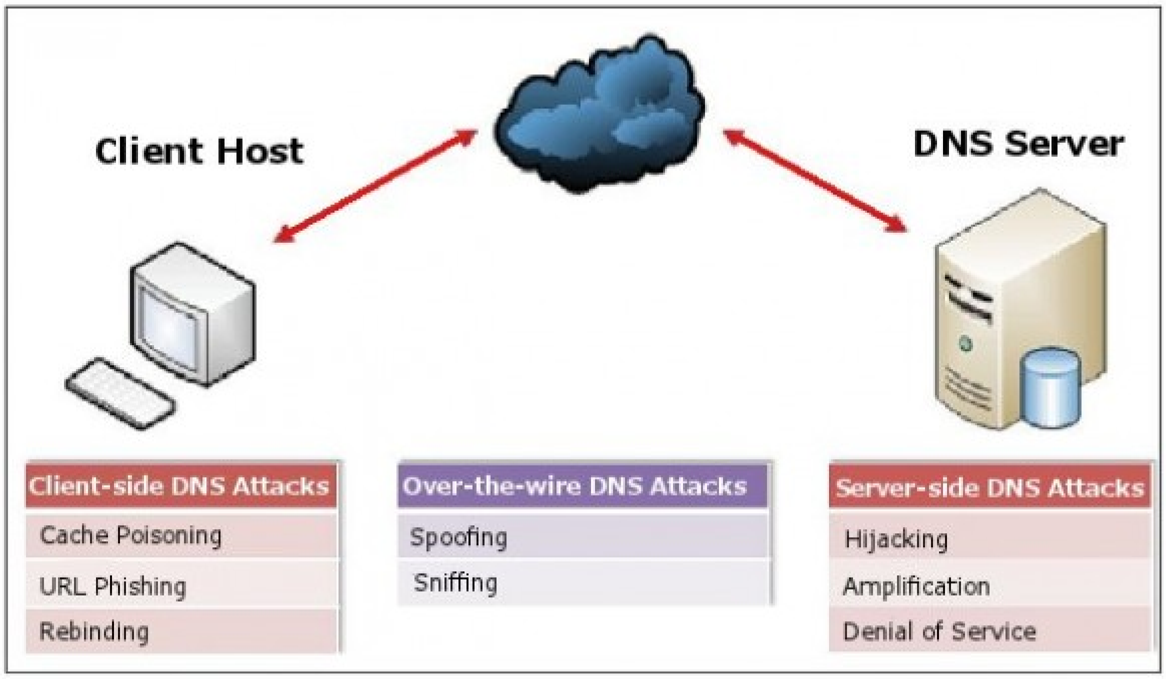

Figure 1: Name resolution

The DNS uses record types like A, CNAME, SOA, MX, etc. While explaining these is beyond the scope of this article, it is important for administrators to know the usage of each, and before implementing them, they should be evaluated from the security standpoint. Before we learn about DNS-based attacks, we need to know about two types of queries — iterative and recursive.

- **An iterative DNS query:** When a client queries a DNS server, asking if it has the answer for a given domain name, the DNS server may or may not have the answer ready. If the DNS server doesn't have an answer, instead of shutting the request down, it sends the name of an upstream DNS server that might have the answer. This is usually called a DNS referral. The client sends the query to the next (referred) server; if that one too doesn't have an answer, it sends a referral to yet another upstream server. This process continues till either the client gets an IP address or gets a "query failed" error message.
- **The recursive DNS query:** In this case, the query begins by a client host requesting a name resolution to its immediate DNS server. If the DNS server does not have the answer, it is supposed to do the job of talking to upstream servers, instead of merely providing their referral names. Again, if the upstream server does not have an answer, it needs to take on the responsibility further. This continues till either the root domain server is reached, which must have the answer, or if the queried name itself does not exist, in which case an error message percolates down the chain to the client. Unlike the iterative method, a recursive query proves to be more aggressive in getting query results.

Iterative queries are usually made by DNS servers while recursive queries are made by clients, which helps to reduce the burden of referral searches. From the security perspective, it is important to know the basics of DNS, such as, there can be multiple DNS servers in an organisation replicating their zone records to each other in order to maintain name resolution consistency.

DNS data can be updated dynamically without needing any service to be restarted, and when a change is made on the master server, it triggers replication to partner servers automatically. The actual time required for replication is defined by the TTL of each record. In case of geographically dispersed DNS servers, this time period can be as long as a day, since all servers in the chain maintain their own cache to speed up replication.

DNS security attacks

It has been observed that systems administrators spend a lot of time designing security around applications, servers and other infrastructure components, but unfortunately tend to forget hardening DNS servers. Please refer to Figure 2, which shows possible breach points where the DNS can be vulnerable to attacks. By design, the DNS heavily relies on UDP, does not contain security by itself, and does not have foolproof built-in authentication — which makes it more susceptible to hacking than other network-based services. Let's look at a few very common DNS attacks.

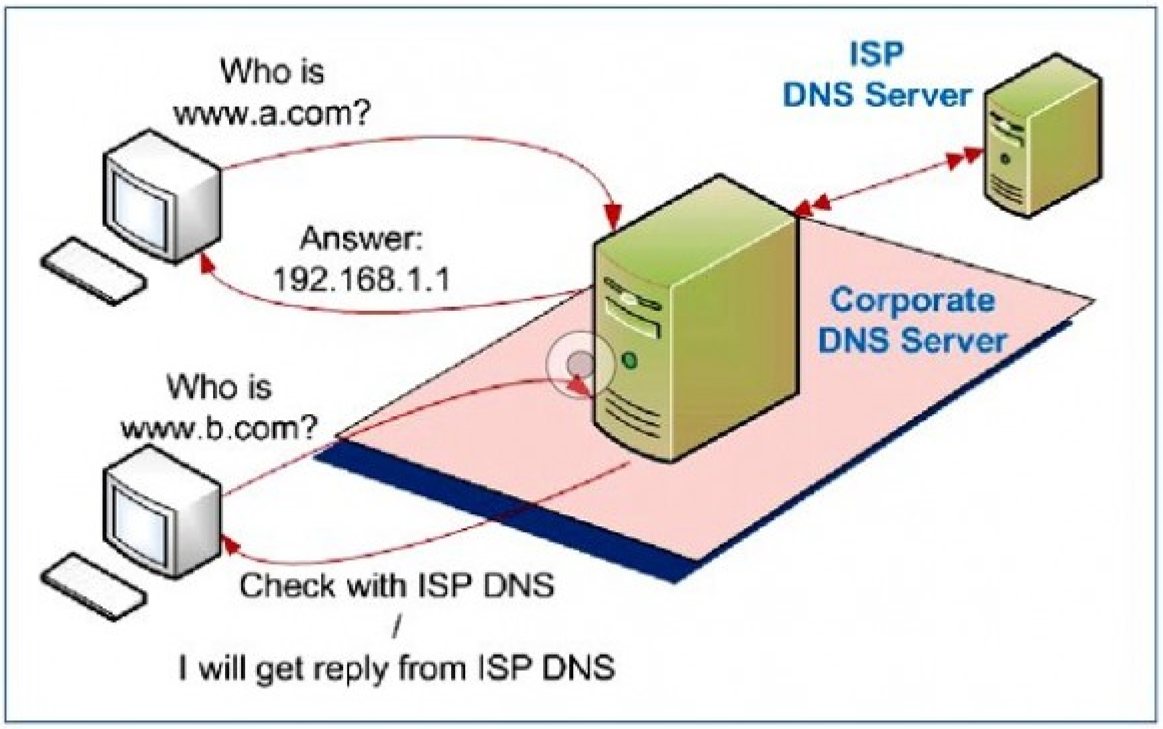

Figure 2: Possible attack points

DNS cache poisoning

This attack lets name resolution to be tweaked in two ways. In one method, the hacker installs a rootkit or a virus, which is intended to take control of the local DNS cache of the client. Once done, entries in the local DNS are altered to point to a different IP address.

For example, if a browser tries to access `http://www.cnn.com/`, instead of getting the CNN IP address, it gets an IP set by the hackers' script, which is usually in the hackers' own Web farm, and hosts viruses or shows some derogatory information.

In a different and more dangerous approach, the hacker attacks a DNS server and alters its local cache — so all servers using that DNS server for resolution end up at a wrong IP address, causing a system-wide failure, apart from information loss or theft.

In rare cases, hackers can access a root DNS server, which holds the base entries that form the root domain, such as `.com`, `.net` or any country-specific name system. Hackers then modify entries on that server, which triggers automatic replication, and can cause serious global outages for multiple businesses and websites. Though such situations are rare, they have occurred — and that too, very recently, involving a famous social community chatting website.

DNS hijacking

This attack is also commonly used to bend the DNS system. Here, the client DNS cache on a client is not altered, but instead the client's DNS settings are changed to point to the hackers' own DNS server. Usually the purpose is not to steal data, but to gather statistical data from the client computer. All name resolution requests going to the hacker are resolved to the correct addresses, but the hacker learns of the typical sites visited by the client.

This information can further be used by online advertisers to target that client with Web-visit-specific advertisements. Some ill-behaved e-thieves also redirect users to their own websites, or search engines, either to gain money from advertisements, or simply to steal data and use it for social engineering. While it is inappropriate to use this feature for any personal gain, it is being used by many well-known websites and ISPs to collect user browsing statistics.

DNS spoofing

This refers to merely a man-in-the-middle type of attack in which the hacker gains access to the network the DNS server is on, and performs ARP cache poisoning and spoofing on that network. Once MAC-level control is achieved, the hacker then fetches the IP address of the DNS server, and starts sniffing and spoofing requests meant for the real DNS server.

The hacker's machine resolves all DNS queries, completely bypassing the real DNS server. This has serious consequences, because all machines on that network can be completely unaware of this, and end up sending DNS traffic to the hacker's machine.

There is an alternate method called DNS ID spoofing. Each DNS request and response carries a unique identifier, to differentiate between various simultaneously generated requests to a DNS server. This unique ID is usually a combination of the MAC address and the date/time stamp, and is created by the protocol stack automatically.

A hacker uses a sniffer to look at one or more DNS requests and responds with their respective unique number, but with a false IP address. This results in the client's local cache being updated to this fabricated address. Further damage can be caused by hosting a virus on the machine at that IP address.

DNS rebinding

Also called DNS pinning, this is an advanced type of attack. In this method, the hacker first registers his own domain name and sets the TTL value of that domain at a lower value, which prevents that domain name from being cached.

DNS denial of service

As we learnt in the very first article of this series, bombarding the UDF port 53 or TCP port 53 with DNS queries can cause a DoS attack. Another method is to perform a ping of death or a TCP SYN flood attack. The idea behind this is to overwhelm server resources (CPU and memory) to stop it responding to queries. Though DNS servers are protected by firewalls, if care is not taken to block DNS UDP ports from non-trusted networks, it exposes the name resolution system to this attack.

DNS amplification

Amplification means to provide the DNS server with a task heavier than it is capable of handling. There are multiple ways to stress the server and eventually make it non-functional. In one method of amplification, a Trojan is written to poison and populate the local cache of multiple client hosts. This forces all infected clients to send their name requests to a particular name server, which is being targeted by the hackers.

Each server can only respond to a certain number of queries (based on CPU speed and configuration) and eventually starts queuing up requests. As more and more clients get infected, the increasing number of queries ultimately makes the server give up.

In another type of attack, a hacker poisons the DNS server’s cache; instead of changing the associated IP address of an A or CNAME record, a change is made to the domain name. To make it worse, the domain name is made to contain a few hundreds or thousands of characters. This starts the replication process, and hence the download of multiple kilobytes of data from the main name server to its replicating partners, and eventually to clients.

Upon expiration of the TTL, the replication process initiates again, and results in the breakdown of one or more DNS servers in the chain. This trick actually simulates a distributed denial of service attack, and hence is very dangerous and hard to control.

Protecting FOSS systems

In the FOSS world, the DNS service is a well-known implementation across the globe, simply because it proves to be the fastest available name resolution mechanism. A widely used and famous example is the Bind utility/service. However, since most DNS attacks exploit the basic design lacunae, it becomes a tougher task to protect FOSS-based name resolution systems.

The very first step to protect a FOSS DNS server is to lock it down at the network level. Besides the server management ports, only the DNS query ports should be allowed and the rest must be blocked on the firewall as well as in OS-based port filtering.

The second important step is to not install any other software on a DNS server, other than the name server service itself. This precaution must be followed especially in the case of an externally facing corporate root name server that holds all internal name spaces, and resolves external name queries for the local area network.

It is often found that vulnerability in another program on the name server leaves a back door open, resulting in intrusion into the name service. While most critical infrastructures implement a firewall, a UTM device and powerful anti-virus or anti-Trojan software, it becomes imperative to have an intrusion detection system (IDS) in place. An IDS is capable of filtering out sneaky Layer 2 and Layer 3 attacks such as ARP spoofing, IP spoofing, packet sniffing, etc.

Besides the above crucial precautions, there are a few advanced methods that can be followed too. As we learnt earlier, each query carries its own unique identifier and is marked in the UDP packet. Unfortunately, due to the design of DNS stacks based on RFC standards, these identifiers are easily predictable, and hence randomising those can be a good idea to prevent spoofing attacks. Similarly, the UDP port on which the name server responds is predictable too, and can be randomised.

There are open source tools available on the Internet for just this purpose; however, please note that it adds a bit of a delay in query resolution. A fairly recent and popular protection technique is DNSSEC (DNS Security Extensions). It protects clients and systems from cache poisoning attacks by digitally signing records using public key encryption. While working in a similar fashion to SSL, the querying and answering hosts need to establish a digital trust between each other; once it is achieved, name resolution takes place.

Once the process is completed, the session is torn down, thus protecting the security at either ends. DNSSEC is being implemented by most ISPs in the world.

DNS invasion is a common phenomenon in the IT security world. It involves exploiting DNS design loopholes to gain access to the IT infrastructure or to lure the client computers to a phishing farm. FOSS is also susceptible to such attacks and hence network administrators must understand the techniques to protect their infrastructure from information loss or theft.

Source : http://www.opensourceforu.com/2012/02/cyber-attacks-explained-dns-invasions/