DNS Servers

Domain Name System (DNS) (TCP and UDP Port 53)

The Domain Name System (DNS) is a distributed database that is used so that computers may determine IP addresses from hostnames, determine where to deliver mail within an organization, and determine a hostname from an IP address. The process of using this distributed system is called *resolving.*

When DNS looks up a hostname or other information, the computer performing the lookup contacts one or more nameservers seeking records that match the hostname that is currently being resolved. One or more nameserver records can be returned in response to a name lookup. The table below lists some of the kinds of records that are supported

Record Type	Purpose
A	Authoritative address. For the IN domain, this is an IP address
AAAA	IP Version 6 authoritative address
CNAME	Canonical name of an alias for a host
NS	Nameserver record: specifies the name of the nameserver responsible for resolving a domain
SOA	Source of authority: specifies the name of the primary nameserver for the domain, the email address of the person responsible for it, and several configuration values for the domain
PTR	Pointer record: maps IP addresses to a hostname for IP host
MX	Mail exchange: specifies a different computer that should actually receive mail destined for this host

DNS-Supported Record Types

For example, using DNS, a computer on the Internet might look up the name *www.cs.purdue.edu* and receive an A record indicating that the computer's IP address is 128.10.19.20. An MX query about the address *cs.purdue.edu* might return a record indicating that mail for that address should actually be delivered to the machine *newman.cs.purdue.edu*. You can have multiple MX records, sorted by priority, for robustness. If the first host is unavailable, the program attempting to deliver your electronic mail will try the second, and then the third. Of course, a program trying to deliver email would then have to resolve each of the MX hostnames to determine that computer's IP address.

DNS also makes provision for mapping IP addresses back to hostnames. This reverse translation is accomplished with a special domain called IN-ADDR.ARPA, which is populated primarily by PTR records. In this example, attempting to resolve the address 20.19.10.18-IN-ADDR.ARPA would return PTR record pointing to the hostname, which is lucan.cs.purdue.edu.

Besides individual hostname resolutions, DNS also provides a system for downloading a copy of the entire database from a nameserver. This process is called a *zone transfer*, and this is the process that secondary servers use to obtain a copy of the primary server's database.

DNS communicates over both UDP and TCP. Because UDP is a quick, packet-based protocol that allows for limited data transfer, it is typically used for the actual process of hostname resolution. TCP, meanwhile, is most commonly used for transactions that require large, reliable, and sustained data transfer, i.e. *zone transfers*. However, individual queries can be made over TCP as well.

DNS Nameserver Attacks

Because many UNIX applications use hostnames as the basis for access control lists, an attacker who can gain control of your DNS nameserver or corrupt its contents can use it to break into your systems.

There are three fundamental ways that an attacker can cause a nameserver to serve incorrect information.

1. **Loading Erroneous Information**

   Incorrect information can be fraudulently loaded into your nameserver's cache over the network, as a false reply to a query. This is often referred to as **cache poisoning**.

   If your nameserver has contact with the outside network, there is a possibility that attackers can exploit a programming bug or a configuration error to load your nameserver with erroneous information. The best way to protect you nameserver from these kinds of attacks is to isolate it from the outside network so that management operations, including DNS updates, are handled over a different interface from the one used to serve DNS query responses.

   If you have an internal network and a firewall, you can limit the damage that an outsider can do to your internal network by running two nameservers: one in front of the firewall and one behind the firewall. The nameserver in front of the firewall contains only the names and IP addresses of your gateway computer. The nameserver behind the firewall contains the names and IP addresses of all of your internal hosts. If you couple these nameservers with static routing tables, damaging information will not likely find its way into your nameservers. Of course, depending on how you have built your firewall and what you allow your users to do on the network, this may not be a workable solution.

2. **Changing the Configuration Files**

   An attacker can change the nameserver's configuration files on the computer where your nameserver resides. To change your configuration files, an attacker must have access to the filesystem of the computer on which the nameserver is running and be able to modify the files. After the files are modified, the nameserver must be restarted by sending it a **kill -HUP** signal. As the nameserver must run as superuser, an attacker would need to have superuser access on the server machine to carry out this attack. By having control of your nameserveer, a skillful attacker could use that control as a stepping stone to controlling your entire network. Furthermore, if the attacker does not have *root* access but can modify the nameserver files, then he can simply wait until the nameserver is restarted by somebody else, or until the system crashed and every program is restarted.

3. Using Dynamic DNS

An attacker can use the DNS dynamic update facility to provide your DNS server with a fraudulent update.

Modern DNS servers have facilities for dynamically updating DNS tables. This feature is very useful when IP addresses are dynamically assigned or shared among large numbers of people. Dynamic DNS allows a running DNS server to update its DNS tables without manually uploading a domain text file and asking the server to restart.

To be secure, dynamic DNS updates must be properly authenticated, or otherwise an attacker could attack your system by simply changing the mapping between your domain names and IP addresses. Most dynamic DNS servers make provisions for authentication by IP address, i.e. only certain IP addresses are allowed to provide updates, through the use of shared key or updates that are signed with a public key algorithm. In general, combining an IP source address with one of the two cryptographic techniques provides for the highest level of security

If you enable dynamic DNS and it is not correctly implemented, an attacker may use it to update your server without your permission. Many domain nameservers suffer a constant stream of fraudulent dynamic DNS update attacks.

DNS Best Practices

You can minimize the possibility of an attacker's modifying or subverting your nameserver by following these recommendations:

- Run your nameserver on a special computer that does not have user accounts.

- If you must run the nameserver on a computer that is used by ordinary users, make sure that the nameserver's files are all owned by *root* and have their protection mode set to 444 or 400 depending on your site's policy. Any directories that are used to store nameserver files should be owned by *root* and have their protection mode set to 755 or 700, again depending on your site's policy. And all parent directories of those directories should be own by *root*, mode 755 or 700. If your nameserver can be configured to run as a nonprivileged user as modern versions of BIND can, you should take advantage of this option and keep the nameserver's file accessible only to that user.

- If your nameserver can be configured to run in a *chroot* or jail area of the filesystem, as modern version of BIND can, use this option to limit its access to other files on your host.

- Configure your nameserver to ignore requests from bogus IP ranges such as 10.0.0.0/8 if your subnet does not use these addresses. In BIND the *blackhole* directive in *named.conf* can be used to do this.

- Configure your nameserver not to perform recursive DNS queries for outsiders. In a recursive query, if your DNS server cannot find the information for the client, it issues its own queries to try to resolve it. When recursive queries are not allowed, it is up to the client to do the follow-up

work. Recursive queries consume nameserver resources and should not be performed for outsiders. In BIND the *allow-recursion* directive controls which client hosts may request a recursive query.

- Remember that there are many files that are used by the nameserver. For example, the Berkeley **named** nameserver by far the most common on UNIX systems first looks at the file */etc/named.conf* when it starts up. This file specifies other files and other directories that may be located anywhere on your computer. Be sure that all of these files are properly protected.

- If you use dynamic DNS updating facilities, require that updates be appropriately encrypted or cryptographically signed. Do not rely on IP addresses for appropriate authentication

- If you know of a specific site that is attempting to attack your nameserver, you can use BIND's *bogusns* directive to prevent the program from sending nameserver queries to that host.

You can further protect yourself from nameserver attacks by using IP addresses in your access control lists, rather than by using hostnames. Unfortunately, raw IP addresses can be somewhat harder to manage, as IP addresses can change more frequently than hostnames. Furthermore some programs do not allow the use of IP addresses instead of hostnames.