Practical DNS

WRITTEN BY MICHAEL HUGHES, LEAD SECURITY SPECIALIST, NATIONAL AUSTRALIA BANK

CONTENTS

INTRODUCTION

This Refcard is intended as a practical introduction to DNS, and assumes you are familiar only with networking basics. After a brief explanation of how DNS works, the bulk of this card will explain DNS configuration, security, and common problems, in order to get you up and running quickly, safely, and reliably.

WHAT IS DNS?

Invented in 1982, the Domain Name System is a replacement for a centralized "hostname to numerical address database" (in reality it was a simple text file) that was used in the days of ARPANET, the precursor to what is known today as the Internet.

The purpose of the original database was to provide an easy way to identify a network connected computer, instead of the unfriendly numerical addresses.

It is much easier to remember "google.com" instead of "74.125.237.130".

As the number of interconnected machines grew, maintaining this information in a central database, as well as ensuring that all clients on the network had an identical copy of this database at all times, became increasingly difficult. The modern DNS addressed this issue by replacing the previous central database with a distributed network of systems.

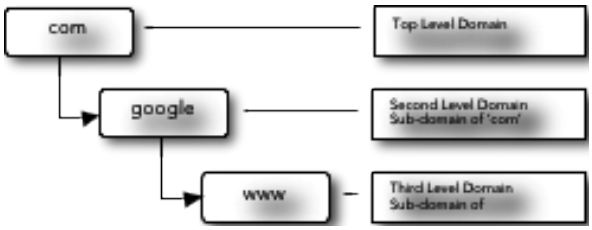
By design, the DNS is more than a simple name-to-IP address mapping database. It allows for many additional properties to be assigned to a domain name, including associated email addresses, anti-spam information, and much more.

DOMAIN NAME HIERARCHY

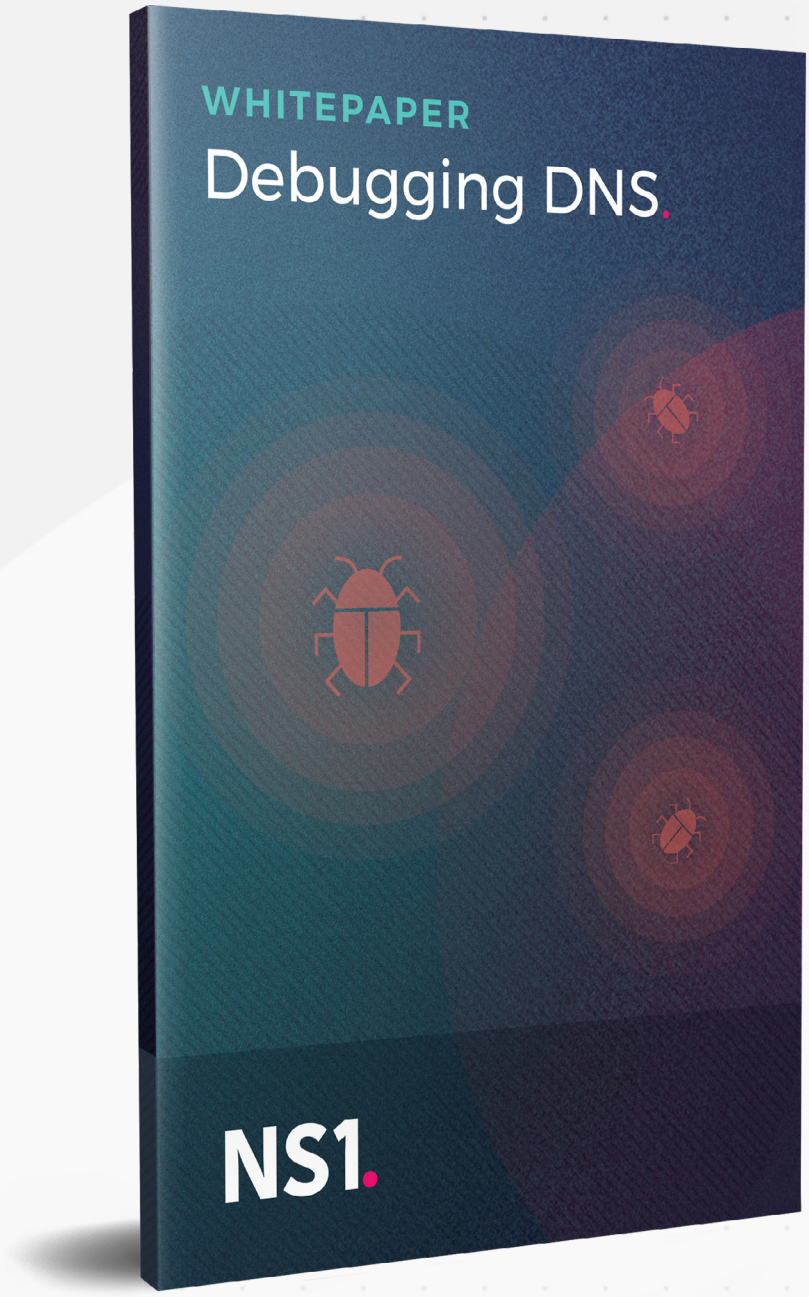
Domain names are the most common method used for accessing websites or any other host on the Internet. Each domain name is made up of a number of elements (called "labels") separated by a dot.

For example: www.google.com

The domain name system works in a hierarchical model, with the right-most elements classed as the "Top Level Domain" or TLD, followed by the second element, which is classed as the "Second Level Domain." This structure continues from right to left with each element being classed as a subdomain of the element to its right.



NS1.



Because DNS is a critical part of your network infrastructure, it pays to become familiar with the process of debugging DNS.

Download this e-book to learn a number of approaches and techniques to debug DNS issues, and new concepts, tools, and a basic process to follow.

Visit our Debugging DNS Resource Center for more
https://ns1.com/debugging-dns

USA TODAY NETWORK · yelp · jetBlue · theguardian · LinkedIn · salesforce · SQUARESPACE

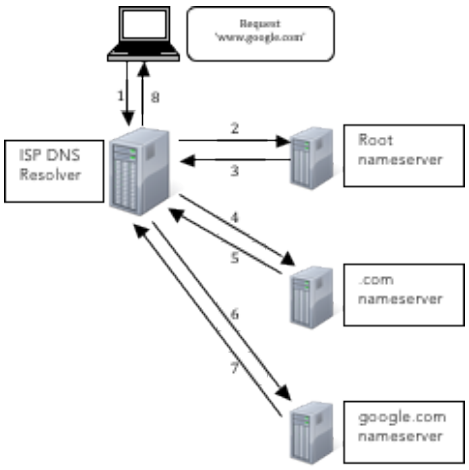
In addition to the structure above, any element may be classed as a hostname should it be associated with one or more IP addresses, and provided it meets the following basic rules as stated in the DoD Host Table Specification as well as RFC1123:

- A hostname must be a text string consisting of only the letters A through Z (upper or lower case), digits 0 through 9, the minus sign (-), and the period (.)
- A hostname cannot contain any spaces
- The first character must be an alphabetic character or a digit
- The last character cannot be a minus sign or a period
- The recommended length for a hostname is up to 24 characters

HOW DNS WORKS

DNS REQUEST PROCESS

When a user attempts to load a website by entering the site's URL into their browser (or any URI with a hostname), a series of requests take place in the background. The illustration below outlines the structure and sequence of these events, and demonstrates how entering a domain name into a browser eventually results in the browser knowing exactly what IP address to connect to at the network layer.



1. The user queries their Internet Service Provider's (ISP) DNS resolver asking for the IP address for "www.google.com."
2. The DNS resolver asks the root nameserver where it can find details for "www.google.com," unless it already has the information cached.
3. If it is asked, the root nameserver responds that this information is handled by the .com nameserver.
4. The DNS resolver asks the .com nameserver where it can find details for "www.google.com," unless it already has the information cached.
5. If it is asked, the .com nameserver responds that this information can be found at the nameservers of google.com.
6. The DNS resolver asks the google.com nameservers where it can find details for "www.google.com," unless it already has the information cached.
7. If it is asked, the google.com nameservers have this information and respond with a DNS record containing the IP address for "www.google.com."
8. The ISP's DNS resolver then sends this information back to the user. The user then knows what IP address to connect to in order to access "www.google.com."

Once the user knows which IP address to connect to in order to retrieve the content of "www.google.com" the DNS job is complete.

DNS ROOT SERVERS

As demonstrated in the previous diagram, DNS infrastructure is designed in a distributed manner, with different servers responsible for different sections of the DNS namespace.

At the top of the hierarchical structure are the DNS "root" servers. These servers are responsible for the initial delegation of requests received from DNS resolvers to the correct top level domain nameservers.

At the time of writing of this guide, there are 13 root servers defined in the DNS root zone. The hostnames of these servers are in the following format: `.ROOT-SERVERS.NET`. The letter values are A-M and each of these root server hostnames is managed by a different organization (with the exception of A and J, which are both currently managed by Verisign). The following table represents the current details of the DNS root servers:

LETTER	IPV4 ADDRESS	IPV6 ADDRESS	OPERATOR
A	198.41.0.4	2001:503:ba3e::2:30	Verisign
B	192.228.79.201	N/A	USC-ISI
C	192.33.4.12	N/A	Cogent Communications
D	199.7.91.13	2001:500:2d::d	University of Maryland
E	192.203.230.10	N/A	NASA
F	192.5.5.241	2001:500:2f::f	Internet Systems Consortium
G	192.112.36.4	N/A	Defense Information Systems Agency

LETTER	IPV4 ADDRESS	IPV6 ADDRESS	OPERATOR
H	128.63.2.53	2001:500:1::803f:235	U.S. Army Research Lab
I	192.36.148.17	2001:7fe::53	Netnod
J	192.58.128.30	2001:503:c27::2:30	Verisign
K	193.0.14.129	2001:7fd::1	RIPE NCC
L	199.7.83.42	2001:500:3::42	ICANN
M	202.12.27.33	2001:dc3::35	WIDE Project

The reason the number 13 was chosen as the number of root server hostnames is primarily the desire for the details of all root servers to fit in one DNS packet. DNS messages were originally limited to 512 bytes of data, and each IPv4 address is exactly 32 bytes long, so the number 13 was chosen. This requires 416 bytes of space, leaving 96 bytes for future use or other supporting information.

While there are only 13 root server names, in reality there are many more DNS root servers than this. A majority of the IP addresses assigned to the root server hostnames (as seen in the above table) are actually "anycast" addresses.

Anycast is a technique that uses the same address at multiple (physical) servers at the same time. For example, the root server "L.ROOT-SERVERS.NET" which is managed by ICANN is actually a cluster of over 130 physical servers distributed around the globe. This type of redundancy serves two main purposes:

1. To ensure speedy responses to DNS queries (by providing network-topologically short distances to the root servers); and
2. To minimize the likelihood of an outage of the entire DNS system.

Management of the DNS root zone itself is actually controlled by the United States Government"s Department of Commerce. The zone is managed by the Internet Assigned Numbers Authority (IANA); the IANA contract is currently held by the Internet Corporation for Assigned Names and Numbers (ICANN). Any changes proposed for the DNS root zone must be approved by the US Department of Commerce before they can be implemented.

CONFIGURING DNS

NAMESERVERS

In the example given above, we can see that nameservers play a key role in the DNS request process. When you register a new domain name, the nameservers for that domain are one of the first items you may configure. This is normally done through the same company with which you registered your domain name.

Nameserver configuration page at Godaddy.com

Configuring the nameservers provides the parent domains (e.g. .com) with an address at which the rest of the records for that domain name can be found. As was seen in step 5 above, these details are given by the top level domain nameservers whenever they are queried by a DNS resolver.

DNS ZONES

All DNS information regarding a domain name (IP addresses, mail server details, etc.) are stored on the configured nameservers in what is known as a DNS Zone. These zones are generally bundled into a zone file, which contains details of all domains for which the given nameserver is authoritative.

As there are a large number of different DNS server software packages, the format of these zone files can vary slightly between different implementations. The most widely-used DNS software on the Internet today is the Berkeley Internet Name Domain (BIND) DNS server.

Originally written in 1984 by four Berkeley College students as a graduate project, the BIND DNS server was eventually rewritten in 2000 with contributions coming from a number of large organizations including IBM, Hewlett Packard, Compaq, and Sun Microsystems.

This rewrite was labeled version 9 and is currently still the supported version of the BIND DNS server in use on systems around the world (and also used by the majority of DNS root servers themselves). Below is an example of a zone file for the BIND 9 DNS server:

```
$TTL 14400
$ORIGIN example.com.
@ 14400 IN SOA ns1.example.com. admin.example.com. (
                                2012121902 ;
                                3600; refresh seconds
                                600; retry
                                86400; expire
                                3600; minimum
                                );
IN A 12.34.56.78
IN NS ns1.example.com.
IN NS ns2.example.com.
ns1 IN A 11.11.11.11
ns2 IN A 22.22.22.22
www IN CNAME example.com.
ftp IN CNAME example.com.
mail IN MX 10 example.com.
```

DNS zone file example (BIND DNS Server)

RECORD TYPES

Each piece of data stored within a zone file is called a resource record. There are a large number of different record types that can exist for a domain. The most common record types are as follows:

"SOA" RECORDS

The DNS "SOA" record is a mandatory entry for any DNS zone. It is the "Start of Authority" record; its purpose is to inform DNS resolvers that this server is the authoritative server for the requested domain name.

The structure of an SOA record is as follows:

```
@ 14400 IN SOA ns1.example.com. admin.example.com. (
            2012121902 ; serial number
            3600; refresh seconds
            600; retry seconds
            86400; expire seconds
            3600; minimum TTL seconds       );
```

The first entry in the "SOA" record contains the name of the host where the zone file is located (ns1.example.com above) as well as the administrative email contact for the DNS zone (admin.example.com above — the "@" symbol is replaced with a "." within DNS zone files).

The "serial number" value is used as a type of version control on the DNS records. Where multiple nameservers are configured, ensuring that each server"s SOA serial number value is the same is a good indication that DNS changes have been replicated on all nameservers.

The "refresh" value is how long a secondary nameserver will wait before checking whether updates have been made on the primary server. When checking, the secondary server will check the serial number of the zone on the master server, and if different will request a zone transfer to update its local copy.

The "retry" value is how long a secondary nameserver will wait to retry a zone transfer in the event the initial attempt fails.

The "expire" value is how long a secondary nameserver will continue to attempt a zone transfer before giving up. If this value is reached before a successful zone transfer is made, the secondary nameserver will expire its local zone file and stop responding to user queries.

The "minimum TTL" value is an indication of how long a DNS cache may hold a negative value (e.g. an error message) before requesting fresh copies.

"A" RECORDS

"A" records are one of the key record types within the DNS. Their purpose is to define a relationship between a human-friendly name and an IPv4 address. These records are useful for pointing different portions of your domain to different IP addresses (e.g. pointing "www.example.com" to your web server).

Each DNS "A" record consists of both a host name and IP address. The format of these records is as follows:

```
<NODE NAME> IN A <IP ADDRESS>
```

The "node name" value above will be the portion of the address that precedes ".example.com" (for example "www"). The IP address is the location that the resulting hostname ("www.example.com") will point to.

"AAAA" RECORDS

"AAAA" records are similar to DNS "A" records.The only difference is that "AAAA" records point to IPv6 addresses instead of IPv4 addresses. Configuration of the record and layout is the same: simply replace "A" with "AAAA" and use an IPv6 address.

"NS" RECORDS

"NS" records are used to specify what the nameservers for the domain name are. These should ideally match the values set at the domain name registrar.

Each DNS "NS" record consists of both the domain name and nameserver hostname and is formatted as follows:

```
example.com. IN NS <HOSTNAME OF NAMESERVER>
```

The hostname will generally be a value such as "ns1.yourdomain.com" or "ns1.yourwebhost.com". If these values do not match those configured with the domain name registrar, it can lead to problems accessing your website in certain circumstances.

"CNAME" RECORDS

"CNAME" is short for canonical name. These records are used to define an alias for another domain or hostname. These records are particularly useful when you have multiple hostnames that are located on the same IP address.

Each "CNAME" record needs to point to another valid hostname, either for the same domain or even a completely different domain. Generally "CNAME" records point to a hostname that is configured in a DNS "A" record. The format of these records is as follows:

```
<HOST NAME>  IN CNAME <EXISTING HOSTNAME>
```

The hostname is the value preceding the ".example.com" as in the "A" record example and is typically something like "www." The existing hostname could be another hostname for the current domain, for example "blog.example.com." In this case, when a user attempts to visit "www.example.com" they will see the content associated with "blog.example.com."

If the existing hostname value is set to a hostname on a different domain — for example "www.domain2.com" — when a user visits "www.example.com" they will see the same content as a user visiting "www.example.com."

"MX" RECORDS

"MX" is short for Mail Exchanger. These records are used to identify the server that handles email address for your domain name.

Each "MX" record contains three pieces of information: the hostname, the priority and the mail server"s hostname. The format of an "MX" record is as follows:

```
<HOSTNAME> IN MX <PRIORITY><MAIL SERVER HOSTNAME>
```

The hostname value is the domain for which the MX record exists. This is generally set to the value of your domain name (e.g. "example.com.").

The priority value is a numerical value that signifies the priority of this particular MX record — and hence the mail server. The values used for this are only important if you have more than one mail server. The lower the value of the priority field, the higher the priority of the mail server.

The mail server hostname is then the hostname that handles email for this domain. It could be a google.com address if the domain uses Google Apps for email hosting, but the simplest setup is to have the hostname set to "mail.example.com". If this is the case, it is important to ensure that a valid DNS "A" record exists for the "mail" hostname. If this isn"t configured properly, you may experience issues receiving email at your domain.

"TXT" RECORDS

"TXT" (or text) records serve multiple purposes. They allow for virtually any free text to be stored and can be assigned to specific hostnames. The typical format for a TXT record is as follows:

```
<HOSTNAME> IN TXT <TEXT DATA>
```

"TXT" records often store sender policy framework (SPF) data . SPF data is used to inform email servers which actual systems are authorized to send mail on behalf of the given hostname. This is useful in the prevention of spam emails being sent with a forged sender address originating from your domain. (RFC 4408 discourages this practice as "not optimal," however, because SPF now has its own DNS resource record type (code 99).)

Another common use of the "TXT" records is for DomainKeys Identified Mail (DKIM) data. These records allow a receiving mail server to authenticate entities that have signed a specific email message. DKIM is similar to SPF in that it can help reduce spam email from containing forged email addresses originating from your domain, but it also contains a large amount of additional functionality.

"PTR" RECORDS

The "PTR" record type is used to perform the exact opposite functionality of the DNS "A" record. Where the "A" record allows for the translation of a domain name or hostname into an IP address, the "PTR" record allows for an IP address to be translated into a domain name.

The format of a "PTR" record is as follows:

```
<IP REVERSED>.in-addr.arpa PTR <DOMAIN>
```

The first field is the IP address in reverse format, with the "in-addr.arpa" value appended to the end, for example "1.0.168.192.in-addr.arpa". The domain value would then be the domain name you want the IP address (192.168.0.1 in this example) to resolve to.

Ensuring that both forward lookups (using "A" records) and reverse lookups (using "PTR" records) match is important: in some instances, applications will not work correctly if this is not configured correctly. For example, some mail servers will not accept mail from a mail server if the reverse lookup does not match the domain name the email is originating from.

SECURITY ISSUES

As DNS is one of the core technologies used in the delivery of the modern Internet, coupled with the high percentage of DNS servers using the same DNS server software (BIND), the DNS protocol and servers implementing it are under constant scrutiny from hackers. The following are two common DNS related attacks:

DNS CACHE POISONING

One of the more common attacks against the DNS protocol is the DNS cache poisoning attack. In order to reduce unnecessary Internet traffic, most Internet Service Providers (ISP) configure their DNS servers to cache DNS responses for the period defined in the TTL value of the requested record set.

This allows for all concurrent requests to be served from the local cache at the ISP and not require the series of lookups normally required. This results in faster DNS responses for the end user, as well as a decrease in data costs for the ISP.

This mechanism, however, is the target for the DNS cache poisoning attack. In this attack, the hacker aims to have their IP address cached by the ISP"s DNS resolvers for a record of their choosing.

For example, the attacker would seek to have the IP address of the hostname "login.example.com" be cached with their own IP address instead of the legitimate IP address. The result of this attack is that anyone using that DNS resolver (typically most of that ISP's customers) would be loading the site "login.example.com" from the hacker's server rather than the legitimate server. Once this is achieved, the hacker could potentially display a fake login page and harvest users" logins and passwords.

DNS REFLECTION ATTACKS

DNS reflection attacks differ from DNS cache poisoning attacks in that their primary goal is not to compromise the integrity of a DNS resolver"s data, but rather to completely flood a system with DNS responses, rendering it unable to respond to legitimate queries. This type of attack is known as a Denial of Service (DoS) attack.

This attack is conducted by a hacker spoofing the sender IP address of a DNS query to mimic that of the victim"s server. When the DNS resolver then replies to the query, the response will go to the victim"s server rather than the hackers. This attack works due to the fact that a small DNS query can actually return a response that is many times larger than the query itself.

Hackers leverage this in order to direct a large amount of network (DNS) traffic at the victim"s machine. This results in the victim machine being unable to accept or respond to legitimate traffic.

DNSSEC

The original DNS protocol was never designed with security in mind; user access to the nascent Internet was tightly controlled and not open to public access.

The Domain Name System Security Extensions (DNSSEC) are an expansion of the DNS protocol aimed at mitigating a number of security issues that have been identified within the DNS since this time. The DNSSEC provide the ability for DNS clients to determine that the DNS response they are receiving actually comes from the server authoritative for the given domain name. This feature is a direct response to the DNS cache poising attacks described above.

HOW DNSSEC WORKS

DNSSEC is implemented by adding a number of additional records to the DNS zone of a domain. These records are as follows:

RECORD NAME	FUNCTION
RRSIG	The signature of the DNS resource record set. This record is returned in the response to any DNS query and can be verified by checking against the public key for the domain.
DNSKEY	Contains the public key for the domain.
DS	The "Delegation Signer" record, used in authenticating the DNSKEY for a domain.
NSEC	Used to prove a name does not exist.
NSEC3	An alternative version of the traditional NSEC record also provides proof a name doesn"t exist but prevents zonewalking.
NSEC3PARAM	Parameter record for use with NSEC3.

In order for DNSSEC to be functional in any given DNS query, every authoritative server from the trust anchor (generally the TLD nameservers) all the way down to the domain"s nameservers must support DNSSEC and offer signing of record sets. If any of these do not support the extensions, then it is not possible to utilize the benefits of DNSSEC for that query.

In order to sign a DNS zone with DNSSEC, a private and public key pair is generated by the owner. The public key is listed in the DNS zone in the "DNSKEY" record whilst the private key is stored securely on the authoritative nameserver. Once this is done, the parent domain server is informed that this zone has been signed.

Each record in the zone is now digitally signed with the signature of the record set stored in the "RRSIG" record. This record is the encrypted hash of the original records value. Sent with the response to every DNS query, this "RRSIG" records allows the client to determine if the value received is the same as the one sent by the server.

When the client receives the DNS response, it takes the "RRSIG" value, decrypts it with the public key (retrieved from the "DNSKEY" record) and then compares the result with the hash of the record value received. If these values match, then the record received is identical to the one sent.

The next feature of DNSSEC is the ability to ensure that the server that sent the record and public key is actually the legitimate authoritative domain server for that domain. This is achieved using the "DS" record. This "DS" record stores a digest of the domain"s DNS key in the domain"s parent zone that is protected by the parent zones "DNSKEY." This configuration then continues in a hierarchical structure up to the DNS root zone. The data for the DNS root zone is treated as a "Trust Anchor."

Using this record, and hierarchical authentication method, it is possible to ensure that the "DNSKEY" received for a domain has not been spoofed by an attacker.

DNS REQUEST PROCESS WITH DNSSEC

The DNS request process remains quite similar to the standard process of requesting DNS data, albeit with a few more pieces of data added in order to verify the responses received.

The following is an example workflow and description of a typical DNS request with DNSSEC enabled. Changes from a normal (non DNSSEC enabled) DNS request are in italics.

1. The user queries their Internet Service Provider"s (ISP) DNS resolver asking for the IP address for "www.google.com."
2. The DNS resolver asks the root nameserver where it can find details for "www.google.com," unless it already has the information cached. It also sets the "DNSSEC OK" (DO) bit to signify that it is requesting the DNSSEC records be returned also.
3. If it is asked, the root nameserver responds that this information is handled by the .com nameserver. It also sends the DS record for the .com zone and RRSIG records for any records returned with the result. The resolver would then validate that the DS record value received from the root nameserver matches the digest of the DNSKEY record value for the .com zone.

4. The DNS resolver asks the .com nameserver where it can find details for "www.google.com," unless it already has the information cached.

5. If it is asked, the .com nameserver responds that this information can be found at the nameservers of google.com. It also sends the DS record for the google.com zone and RRSIG records for any records returned with the result. The resolver would then validate that the DS record value received from the .com nameserver matches the digest of the DNSKEY record value for the google.com zone.

6. The DNS resolver asks the google.com nameservers where it can find details for "www.google.com," unless it already has the information cached.

7. If it is asked, the google.com nameservers have this information and respond with a DNS record set (RRSET) of all "A" records configured for "www.google.com." It also sends the RRSIG value for the RRSET that was returned. The ISP"s DNS resolver can then validate that the RRSIG value is indeed the signature of the RRSET that is returned by checking it against the DNSKEY record in the google.com zone. The resolver has now validated that the records themselves have not been tampered with, and also previously validated that the servers that have been involved in receiving the data are trusted by validating the DS records against the associated parent"s DNSKEY record.

8. The ISP"s DNS resolver then sends the "A" record for "www.google.com" back to the user. The user then knows what IP address to connect to in order to access "www.google.com."

DNSSEC AVAILABILITY

Unfortunately, not all top level domains currently support DNSSEC. As was previously mentioned, each server in the chain needs to both support DNSSEC as well as have the appropriate records configured in order for DNSSEC to have any effect.

As of April 2013, less than a third of all top level domain servers (including ccTLD's) are configured with the appropriate settings to support DNSSEC.

Of the 19 globally assigned top level domains, the following do not currently support DNSSEC:

- .aero,
- .coop
- .int
- .jobs
- .mobi
- .name
- .pro
- .tel
- .travel
- .xxx

COMMON DNS PROBLEMS

DNS PROPAGATION

The most common problem encountered with the DNS protocol is the situation known as "DNS Propagation." When a change is made to a DNS zone, this change is not immediately seen by the rest of the world.

The name itself (specifically the "propagation" component) suggests that changes propagate outwards across the Internet away from the server on which the change was made. This is a common misconception: DNS changes do not propagate outwards like ripples. Instead, when a change is made on a DNS zone, the only real automated propagation of that change is to secondary or child nameservers. Due to the nature of DNS, all DNS queries for a particular domain will follow the hierarchical approach to determining which server has the data for that domain, eventually arriving at the nameserver for the domain.

As the change to the DNS zone has already occurred on the primary server, the updated values are immediately available and are included in all DNS responses made by that server after the change has been made. Once the change has been made on the primary server, each additional authoritative nameserver will update its stored values the next time it attempts a zone transfer.

The "propagation" effect is actually caused by ISP"s DNS resolvers caching the original, now outdated, DNS records and returning that to the end user, rather than the updated value that could be retrieved directly from the domain"s nameservers.

The TTL value assigned to each record indicates to DNS resolvers how long they should cache the values of the retrieved DNS records.

Ideally, if a change were made to a DNS record immediately after it was cached by a DNS resolver, the delay before that resolver retrieved the updated record would be at most the value of that record sets TTL.

While not as common in recent times, not all DNS resolvers adhere to the values contained in the TTL values, with many choosing to implement their own TTL values for cached records (in order to reduce bandwidth use). This is where the "propagation" effect comes from.

In order to reduce the likelihood of a long DNS propagation time, try this:

1. Ensure that the TTL values of all records are set to a low value (e.g. 300).

2. Ensure that when changing nameservers both the old and the new nameservers are configured with the same DNS records. This will ensure that regardless of what nameserver the user is directed to, the results they receive will be consistent.

WEBSITE NOT AVAILABLE ON "WWW." HOSTNAME

Another common issue with DNS configuration is users' experiencing difficulties accessing their website using "www.example.com" rather than just "example.com."

The cause of this issue is actually quite simple to diagnose as well as resolve. The ideal configuration of the DNS zone to allow the "www." hostname to work is as follows:

1. Ensure that there is an "A" record for the domain itself. example.com. IN A 192.168.0.1
2. Next, create a CNAME record for the "www." hostname. www.example.com. IN CNAME example.com

The above steps will (as far as DNS is concerned) ensure that the website is visible on both "www.example.com" as well as "example.com." It is then the responsibility of the web server to determine whether to force redirect the user to one or the other.

There are other methods to solve this issue, including adding an A record for both the domain itself as well as the "www." hostname (instead of a CNAME record); however for ease of maintenance, the use of a CNAME record is recommended as if the IP address needs to be changed in future, it only needs to be changed once.

NAMESERVER REDUNDANCY

For most domain name owners, configuring a domain's nameservers is as simple as entering the values given to them by their web hosting company.

For example: ns1.webhost.com and ns2.webhost.com

It is important to check, however, that your provider"s nameservers are:

a. Not located on the same server as the website itself; and
b. Located in geographically diverse locations.

A large number of providers use the web hosting control panel "cPanel" which by default allows the provider to configure the nameservers, web server and mail server to be located on the same physical server. This means that should the server itself go offline for whatever reason, your website and email are completely offline.

Ideally, you should ensure that your web host has at least its DNS servers located on separate physical servers as well as networks to your web server and mail server. This provides a layer of defense in that if the DNS service fails to work on one of your nameservers due to a network or hardware issue, the other nameserver is still able to respond, and the website and mail server will remain unaffected.

Written by **Michael Hughes,** Lead Security Specialist, National Australia Bank

Michael Hughes has worked for energy companies, web hosting companies and large financial institutions as a technical specialist and, more recently, a security specialist. Michael has a Bachelors Degree in Telecommunications Engineering, and has developed a deep knowledge of many technologies over his career. While working for one of the larger web hosting companies in Australia, Michael developed ViewDNS.info with the aim of providing a free and simple interface to a large number of DNS tools.

DZone, Inc.
150 Preston Executive Dr. Cary, NC 27513
888.678.0399 919.678.0300