Domain Name System and Security

Bengt Sahlin
April 2005

I.Introduction

The Domain Name System (DNS) is a fundamental component of the Internet. When DNS is unavailable, the user experience is that no connection to Internet is possible. Of course this is not true. The user can still access Internet services if he knows the IP addresses of the servers he would like to connect to. But with DNS being unavailable, the user cannot use easy-to-use domain names instead of hard-to-remember IP addresses for identifying a service.

The objective of this paper is to discuss how to secure DNS servers and DNS communication, in order assure availability of DNS services and reliability of DNS data and communication.

II. Domain Name System

A. Background

Humans are better at remembering names than long sequences of numbers. Machines, on the other hand, are good at handling numbers (bits). The IP address is used to identify the endpoints of the communicating entries in TCP/IP networks. In IP version 4, the address consists of 4 octets (for example 10.0.0.1). Even though the common IPv4 address notation is to represent each octet by a decimal number, it is hard to remember a large number of IP addresses. It is more practical to associate the network entity with a name. With IPv6 addresses, which are commonly represented in hexadecimal notation (for example fe80::a0a1:46ff:fe06:61ee), it becomes even harder to memorize IP addresses.

When Internet was developed, the benefit of easy-to-use and easy-to-remember names to identify services was recognized. As the actual identifier of a network node in the Internet is the IP address, there was a need to store the association between the IP address and the corresponding name. At the beginning, this information was provided for the whole network in a file called HOSTS.TXT. Network Information Center (NIC) of Stanford Research Institute administered the file. Changes were submitted to NIC and the file was updated regularly. Other network nodes were able to fetch new versions of the file from NIC.

As Internet grew, it became impractical to maintain the HOSTS.TXT file. The network traffic and processor load of the equipment at NIC increased and caused problems. Name inconsistencies arouse as nothing prevented from adding a host with a conflicting name. The information was not either propagated timely to all hosts in the network, so inconsistent versions of the file existed in the network.

A more scalable and automated solution was needed. The solution designed was called the Domain Name System (DNS). The main tasks of DNS were to provide the mapping between domain names and IP addresses, and to control e-mail delivery in the network.

DNS is a global, hierarchical and distributed database. As the domain name space is global, the names used in the DNS must be globally unique. The data is divided hierarchically into a tree with a root, similar to the structure of the Unix file system. The root node's name is the null label, written as a single dot. The components of a domain name, called labels, are listed bottom-up, in reverse order when compared to a Unix file name. Each node in the tree represents a partition of the overall database, called a domain. Each domain can be divided into sub-domains, as directories can be divided into subdirectories in the Unix file system. The administration of the names is distributed, allowing local control of the segments of the global database. A DNS server can delegate authority for a sub-domain to another server.

The servers in the DNS are called name servers. The root domain is administered by 13 root name server clusters. The next level domains are the called Top Level Domains (TLD). TLDs can be divided into structural domains and geographical domains. Examples of structural domains are the .com, .net and .org domains. Examples of geographical domains are the .fi, .se and .uk domains. The leaves in the DNS tree contains some specific DNS data, like an actual mapping between a DNS name and an IP address.

A master name server is the name server where DNS data is administered. The master is the ultimate authority for the DNS data (authoritative). Slave name servers are used to provide redundancy. The data is not administered in the slave. Instead, the slave retrieves the DNS data from the master. The data is fetched from the master through a zone transfer.

A name server can also function as a caching name server, storing non-local DNS data. Using caching name servers limits the amount of DNS traffic in the network. As DNS data may change over time, there is a timeout mechanism causing the data to be discarded from the cache eventually.

The DNS client program is called a resolver. It is able to perform DNS queries, that is, messages that are sent to a name server to acquire some DNS data. Typically, the resolvers are simple, expecting name servers to perform most of the name resolution process on their behalf.

A zone consists of those contiguous parts of the domain tree for which a domain server has complete information and over which it has authority. The data contained in a zone file is composed of entries called Resource Records (RRs).

The process of acquiring some DNS data, by possible sending queries to several name servers, is called name resolution.

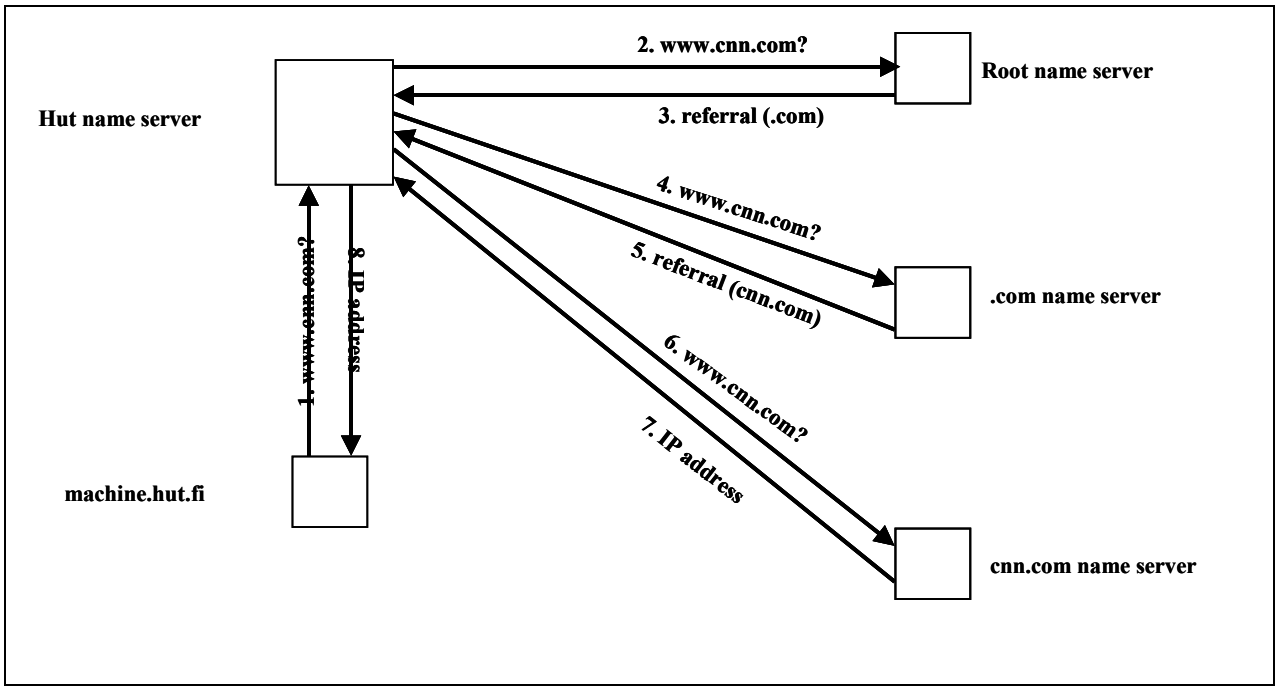

Figure 1: example of name resolution

Figure 1 shows an example of name resolution. The host *machine.hut.fi* wants to connect to the web server of CNN. To be able to do that, it must know the IP address of the web server. The DNS client in the host, the resolver, is configured to contact a name server in the *hut* domain. The resolver sends a DNS query to the name server (message 1). In this example, the *hut* name server has recently been booted up, and it has no non-local information in its cache. As a name server is booted up with information on the IP addresses for the root name servers, it is able to proceed with the name resolution in order to provide a response to the host.

The *hut name server* first contacts a root name server (message 2). The root name server does not have the DNS data queried for, but it has information about name servers authoritative for the TLD .com. Thus, the root name server can return that information to the *hut name server* (message 3).

The *hut name server* next queries a *.com name server* to get the DNS data (message 4). The *.com name server* does not have the DNS data queried for, but it can provide information about name servers authoritative for the *cnn.com* domain (message 5).

Finally, the *hut name server* queries the *cnn.com* name server for the DNS data (message 6), and the *cnn.com* name server responds with the IP address of *www.cnn.com*. The *hut name server* can now return the information to the host.

The *hut name server* can also store the DNS data in a cache. If another host in the *hut* domain would query for the data, the *hut name server* can respond with the data from the cache. Thus, caching the data would reduce the number of messages to resolve the information from 8 to 2 messages.

Typically, the DNS messages are fairly small. Thus, DNS messages are mainly transported over UDP. The exception is zone transfers, where a large amount of information is transferred. Zone transfers are thus transported over TCP.

```
verkot.example.         IN      SOA     ns.verkot.example. dnsadmin.verkot.example. (
                        6 28800 7200 604800 86400 )
                        IN      NS      ns.verkot.example.
                        IN      MX      10 mail.verkot.example.
$ORIGIN verkot.example.
localhost               IN      A       127.0.0.1
ns                      IN      A       10.10.10.1
mail                    IN      A       10.10.10.2
www                     IN      A       10.10.10.3
                        IN      TXT     "Our web server"
ftp                     IN      CNAME           mail.verkot.example
```

Figure 2: zone file example

DNS data is typically stored in the so-called zone file format. Figure 2 shows an example of a simple zone in zone file format. The data in the DNS is organized into resource records. All the resource records have the same base format, described in RFC 1035.

Every zone has a Start of Authority (SOA) RR. This RR informs about the authoritative name server of the domain and provides the e-mail address of the administrator (dnsadmin@verkot.example in Figure 2). Furthermore, the SOA record contains the serial number, which is incremented when changes are made to the zone. The SOA record also includes information on how often a slave name server needs to query the master for updates, how often the slave should retry if the master cannot be reached, and provides a limit for how long the slave can store the zone information without being able to contact the master.

The A record type contains a mapping between a name and an IP address. In figure 2, the name *ns.verkot.example.* is associated with the IP address *10.10.10.1.* Another typical record is the CNAME RR. In figure 2, the domain name *ftp.verkot.example.* is an alias for the domain name *mail.verkot.example.*

The most common name server implementation in use is Berkeley Internet Name Domain (BIND). It is a freely distributable software package. BIND is currently managed by Internet Software Consortium (www.isc.org). There are several estimates on how widely BIND is used as a name server, varying from 88-99% of the zones being administered by the BIND name server software.

One of the main problems with DNS is that despite that it is a fundamental component of Internet and used widely, there is only sparse documentation about managing and maintaining DNS data. The master zone file format is error-prone. Care has to be taken to ensure that modifications of the DNS data are correct. According to domain health surveys by the company Men and Mice (http://www.menandmice.com), the majority of the .com zones contain some incorrect data, which might lead to failure in DNS lookups.

III.Extensions to DNS

The immense growth of Internet raises a need to improve or amend the underlying network technology. DNS has served its purpose well, but as Internet evolves, new requirements are also put on DNS. The new IP protocol, IPv6, requires a new resource record for storing the mapping between domain name and IPv6 address. DNS security extensions are being standardized to meet the growing security demands. To support auto-configuration of a host being attached to a network, dynamic updates of the DNS data needs to be made possible. Domain names are constructed in ASCII, and the Internationalized Domain Name (IDN) initiative aimed at making it possible to support a wider range of characters for use in domain names.

IV.DNS Threats, Vulnerabilities and Attacks

The basic DNS protocol does not provide any security services. Thus, it is clear that the actual DNS protocol is vulnerable against a number of threats. It is possible to intercept packets and eavesdrop on DNS traffic. It is possible to modify DNS messages, to inject DNS messages or even to destroy DNS messages. The DNS protocol includes a certain amount of predictability, and an attacker might in certain circumstances be able predict resolver behaviour and provide false responses to the resolver. In some circumstances, it might even be possible to perform blind attacks on the resolver, without being able to intercept the DNS traffic from the resolver.

These threats make DNS spoofing possible. If the attacker can intercept DNS messages, he might be able to reply with a false reply before the authentic DNS reply is sent. The goal of the attack might be to provide a resolver with false DNS data that directs the victim to a server that is under control of the attacker. Or, the goal of the attacker might be to poison a cache of a caching name server, by providing false DNS data with a long time to live value.

DNS can be used to mount DOS attacks. An attacker can send a large number of DNS queries to a name server with a spoofed IP address. The victim of the attack - the node whose IP address is used - might get a lot of DNS answers and experience DOS.

Typically, DNS reply message are larger in size than the DNS queries, so the victim might be flooded by large responses based on rather simple queries generated by the attacker. Potentially, the name server that answers to the bogus queries might also experience DOS.

DNS software vulnerabilities introduce threats against DNS or nodes that run DNS. Traditionally, name server software has been run using super-user privileges. Compromising name server software might thus in the worst case make it possible to run arbitrary code on the name server node. The attacker might in the end gain control of the name server node.

IETF is working security extensions for DNS. This work is not trivial, and some characteristics of the DNS protocols (for example wildcards) make the security extensions work even more challenging. Potential weaknesses in the security extensions might introduce threats against DNS.

Lately, so called phishing attacks have become more common in the Internet. Recently, it has been found out that IDN is vulnerable against phishing. The idea of the phishing attack in this context is to use a character set where characters look almost the same as the known ASCII characters. The translation from the character set to ASCII will then result in an entirely different domain name. The attacker aims in this way to re-direct the victim to a fake site, which looks similar to the original site. For example, if you use the Cyrillic letter “a” instead of the Latin letter “a”, “www.apple.com” would be translated to an entirely different domain name.

V.DNS Security

IETF is progressing security work on DNS. The current set of security mechanism for DNS offer the following security services:

- Data origin authentication and integrity, including the ability to prove non-existence of DNS data. The DNS security extensions are designed to provide these security services.
- Means for public key distribution. RFC 2538, currently being updated currently in draft-ietf-dnsext-rfc2538bis, describes how certificates are stored in the DNS.
- Transaction and request authentication and integrity. RFCs 2845 (Secret Key Transaction Authentication for DNS (TSIG)) and 2931 (DNS Request and Transaction Signatures (SIG(0)s)) describe the mechanisms that provide these services.

DNS security offers no confidentiality. This has been a conscious choice, since the DNS is intended to be public information available to anybody, and thus the value of confidentiality for DNS traffic is limited. For the same reasons, DNS security mechanisms do not offer any access control. Individual DNS implementations might offer some level of access control, though.

The DNS security mechanisms offer no protection against attacks on name server node itself, and it does not offer protection against DOS attacks. Other means need to be provided to protect against these kinds of attacks.

It should also be noted that DNS security does not assure the correctness of the DNS data in any way. DNSSEC provides for integrity of the data, but the DNS data can still be misconfigured, and potentially lead to failure in the name resolution process.

A. DNS Security Extensions

The DNS Security Extensions (DNSSEC) describe the mechanisms needed for providing data origin authentication and integrity. These services are provided using four new resource records.

Signature record (RRSIG)

The signature record contains a signature of a DNS record. The RRSIG record contains information about the algorithm used, about the validity interval, the signer name and the actual signature. The signature is created using public key cryptography. Both DSA and RSA can be used in DNSSEC. In addition, private algorithms can be used.

The owner of the DNS data performs the signing process. The RRSIG records are created automatically during the zone signing process. The signatures are created using the owner's (zone) private key. The signature can be verified using the owner's public key. It should be noted that the signature covers both the actual DNS record and the values of the RRSIG record.

DNSKEY record

In order to be able to verify a signature, the verifier needs to have access to a public key, which corresponds to the private key that was used to sign the data. The DNSKEY record is intended for storing public keys for use in DNSSEC. Like the other records in a DNS zone, the DNSKEY record is protected by a corresponding RRSIG record.

Delegation Signer record (DS)

The DS record is needed for indicating which public key can be used for verifying the signatures for the DNS data in a child zone. The DS record contains a key tag for identifying the correct public key, and a digest over the public key. The DS record resides in the parent zone and is covered by an RRSIG record generated in the parent zone. This provides a cryptographic binding between the keys in the parent zone and the keys in the child zone. Thus, the DS records are used when verifying signatures in the name resolution process.

NSEC record

The NSEC record is used to prove non-existence of DNS data. NSEC records are created automatically as part of the zone signing process. The NSEC resource record contains the name of the next name in the zone, thus stating that there can be no resource records between the owner name and the next name. To be able to use the concept of a next name in the DNS implies that the records in a zone have to be canonically ordered. As with DNSKEY and DS records, the NSEC records are covered by an RRSIG record.

B. Secure Name Resolution

When introducing DNS security extensions, the name resolution process changes. As part of the name resolution, the corresponding signatures need to be verified, in order to verify the integrity and data origin authentication of the data. The verification starts from the highest level RR and continues through a chain of verifications, until the zone signing key for the DNS data is verified. When the zone signing key has been verified, the signature of the actual DNS data can be verified.

Secure name resolution indicate that the resolver needs to trust at least some key; it needs to have some trust anchor from where it can start the verification process. As the structure of DNS is hierarchical, a natural trust anchor would be the zone signing key for the root domain. However, it is today recognized that deployment of DNSSEC will take time, and thus it cannot be assured that all the zones are secure. Rather, there will be islands of security: parts of the DNS tree will be secured and part will not. In practice, resolvers thus need to be configured to have several trust anchors.

VI.Transaction and Request Authentication and Integrity

The DNS security extensions provide data origin authentication and integrity for the DNS data. However, these extensions are not able to protect a complete DNS message, including the message headers. Protection of a complete DNS message is needed especially in the case of using DNS dynamic updates (RFC 2136), to ensure that only authorized clients can dynamically update the data in a zone. Protection of complete DNS messages might also be useful for example in the case of zone transfers.

Two mechanisms have been specified for providing transaction and request authentication and integrity. RFC 2845 defines a method based on shared secrets (TSIG). The shared secret is used for calculating a Message Authentication Code (MAC) over the complete DNS message.

The main benefit of a method based on shared secrets is that signature calculation (generating the MAC) and signature verification (verifying the MAC) are relatively simple and inexpensive. A method based on shared secrets is however not very scalable. If scalability is desired, the SIG(0) mechanism described in RFC 2931 can be used. In

this method, public key cryptography is used to create a signature (SIG(0)) that covers a complete DNS message.

VII.Implications of the DNS Security Mechanisms

The use of DNSSEC implies a growth of the DNS databases. RRSIG records need to be provided for every DNS record, and NSEC records need to be added between each of the resource record sets. Consequently, the number of records increases roughly by a factor of three. As the DNSSEC records are larger in size than the other DNS records, the size of the actual data grows even more than by a factor of three.

As part of the name resolution, a number of signatures need to be verified. This increases the processing time needed for name resolution and implies increased DNS query response times.

DNS uses a hierarchical structure, but in practice the tree is not very balanced. For example, today the .com domain contains millions of entries. Signing such large domains will take time. The hierarchical structure also introduces issues with performing a key rollover at the root or the TLD name servers.

The RRSIG signatures have a certain lifetime, bound by the signature creation time and the signature expiration time. This implies that strict time synchronization is needed between the communicating DNS entities.

The NSEC records make it effectively possible to list all the contents of a zone. Alternatives that would not make it possible to list the contents of a zone are being discussed in IETF.

RFC 1035 limits the size of DNS UDP packets to 512 octets. When including RRSIG records, this limit might be exceeded. To make it possible to support larger messages, the Extension Mechanism for DNS has bee specified, and it must be supported in DNSSEC.

Allowing dynamic updates of a zone implies that the zone signing key needs to be online in the name server, so that the zone can be re-signed after the dynamic update has been performed successfully. Care need to be taken to protect a zone signing key that is online properly.

VIII. Recommendations for improving DNS Security

For DNSSEC to be effective, it should be widely deployed. That deployment can be expected to take some time. However, even without wide deployment of DNSSEC, there are many measures that can be taken to improve DNS security. In fact, it is very important to provide basic security even before deploying DNSSEC. If a name server, for example, is easy to compromise, deploying DNSSEC might not improve security at all, since an attacker might compromise the name server and might even be able to alter the

DNS data and re-sign the data. To improve DNS security, the following recommendations can be given.

1. Run latest version of the name server software

   It is good security practice to ensure that a software system is up-to-date and all the patches against know faults or vulnerabilities have been applied. Thus, it can be recommended to run the latest version of the name server software and ensure that the latest patches are applied.

   There are useful resources for keeping up-to-date about recent vulnerabilities. CERT organizations (for example www.cert.org, www.cert.fi) are good sources for finding information about vulnerabilities. Monitoring relevant e-mail lists is also useful. ISC keeps up-to-date information about BIND vulnerabilities (http://www.isc.org/sw/bind/bind-security.php).

2. Firewall protection

   It is considered good security practice to protect services in a network. Although a firewall might not protect against all DNS specific threats, it can help in protecting against attacks on the name server node itself.

3. Run the name server with least privileges.

   A common security practice is to run services with least privileges. Usually, the name server needs super-user permissions to reserve the DNS port (port 53) and read the configuration file. Other than that, no need for super-user permissions can be seen. Therefore, it is recommended to run the name server without super-user permissions. The most used DNS software, BIND, provides this kind of functionality. BIND can be started up with super-user permissions, but it is able to relinquish these permissions after the DNS port has been reserved and the configuration file has been read.

   Effects of a compromise of the name server software can be further mitigated by running the software in a sandbox. The principle of least privileges is followed here since running the software in a sandbox means that the software is able to access only parts of the file system in the node, and thus any damage in case the software is compromised can be mimimized. In UNIX like systems, this can be implemented by running the name server software in a chroot environment.

4. Eliminate single points of failure

   To ensure availability of DNS data, it is recommended that redundant name servers are used. For a zone, at least two name servers should be run. The name servers should also be placed in separate sub-networks and behind separate routers. This mitigates effects on DNS of attacks against a certain sub-network or

a certain router.

5. Avoid running any other services on the name server node.

   It is advisable to avoid running other services on the name server node. This mitigates effects on other services in case the name server software is compromised. Also, this mitigates the possibility of gaining control of the name server software by compromising other services on the node.

6. Consider non-recursive behavior and restricting queries

   Name server software often implements some means for access control. This can be useful in mitigating against cache poisoning attacks. Restricting queries and running the name server in non-recursive mode are measures that can be taken to decrease the probability of cache poisoning.

7. Use random message Ids

   Using random message Ids makes it possible to mitigate against ID guessing and query prediction attacks.

8. Hide details about the name server software

   It is advisable to try to configure the name server software so that if provides as little information as possible about itself. This might make it harder for the attacker to mount specific attacks against that specific name server software. For example, BIND makes it possible to hide the version number of the name server software.

9. Prevent unauthorized zone transfers

   Attackers usually try to gain as much DNS information as possible in order to try to find out as much as possible about the network topology and the services of the victim. Preventing unauthorized zone transfers might make it harder for the attacker to gain this kind of information.

   It should be noted that the effectiveness of preventing unauthorized zone transfer can be challenged from a security perspective. Firstly, the DNS data is supposed to be public and available to anybody. Second, the attacker can get access to all the DNS data of a zone anyway, since the NSEC records make it possible to traverse all the DNS data in a zone. This might be a bit harder than doing a zone transfer, but the process can be automated, and from that point of view it is debatable if preventing zone transfer really help. Furthermore, the attacker might be able to eavesdrop on DNS data and gain the information in that way. On the other hand, a zone transfer implies transferring a lot of DNS information, and thus preventing unauthorized zone transfers might be a means for mitigating against

DOS attacks.

TSIG (RFC 2845) can be used to provide for authentication and integrity of the zone transfer exchange.

10. Use Split DNS

    Split DNS refers to a DNS configuration where the view of a DNS zone is different in the public Internet when compared to the view of the zone within an organization. Split DNS makes it possible to only list the services that are publicly available in the external zone. The internal zone can then include all the DNS information about all the nodes in the internal network. This has the benefit that the internal DNS data would only be available in the internal network. Furthermore, private IP addresses can be used in the internal network. If the internal nodes need access to the public Internet, NATs are needed for translating private addresses to public addresses. Organizations also often run HTTP proxies for restricting the possibility of accessing HTTP services directly from the internal network.

11. Restrict DNS dynamic updates

    If the name server is configured to accept dynamic updates (this might be useful for example if DHCP is used to allocate IP addresses), the name server should be configured to allow access to authorized dynamic update clients only. TSIG (RFC 2845) is recommended for protecting the dynamic updates. A shared secret is thus created between the name server and the client, and the shared secret is used to provide authentication and integrity for the exchange of DNS dynamic update messages.

12. Deploying DNSSEC

    To protect against DNS threats, the DNS security extension can be deployed. This helps provide assurance about the integrity of the DNS data of the organization. Many attacks, like DNS spoofing attacks, can be mitigated by using DNSSEC. Of course, the client needs to trust the zone signing key for the zone in order for the security extensions to provide security for the DNS data. A wider deployment of DNSSEC would make the DNS data more trustworthy.

IX.Bibliography

Further information about DNS and DNS security can be found from the following resources.

- Cricket Liu, Paul Albitz, DNS & BIND, 2001
- Cricket Liu, DNS & BIND Cookbook, 2002

- http://www.ietf.org/html.charters/dnsext-charter.html
- http://www.dns.net/dnsrd
- http://www.menandmice.com
- http://www.idns.org