GSEC Paper Practical Assignment Version 1.4b

David Hinshelwood

DNS, DNSSEC and the Future

1 Abstract

The domain name system (DNS) is the means by which hosts find out the IP addresses of other machines from their universal resource locator. The key to DNS is its hierarchical nature that makes delegation so easy. It is very important to set-up and document the DNS with best practices firmly in mind or the corporate system will crumble. The aim is to mitigate the risks of mis-configuration and attack so down time is kept to a minimum or compensated for by reducing the single point of failure.

Best practices move us towards secure authentication of the information held in the DNS structure by means of the DNS Security Extension (DNSSEC). Although it is still in need of full implementation, DSSEC is the only viable path to follow for the next generation of the domain name system.

2 Scope

This paper will discuss the Domain Name Service (DNS), the current security issues with DNS, Good Practices needed to secure a DNS server, the use of DNS Security (DNSSEC) additions to DNS to provide added security and possible future directions for the security of DNS. Berkeley Internet Name Domain (BIND) will be used as the example of an implementation of DNS discussed in this paper unless otherwise stated. It is assumed the reader has reasonable knowledge of client/server architecture, networks, operating systems, distributed computing, public-key encryption and TCP/IP.

3 Introduction

The Internet began with the creation of the US Department of Defence's Advanced Research Projects Agency (ARPA and later DARPA) **ARPAnet** in the 1960's. ARPAnet was a large network for the sharing of files, software, email exchange and connection to remote computers. The Transmission Control Protocol/Internet Protocol (TCP/IP), created in the early 1980s, and the virtually free Berkeley's BSD UNIX allowed universities and other (non-government funded) facilities to attach their computers and LAN's (Local Area Network) to ARPAnet. The network expanded to include thousands of computers developing, over the years, into the Internet.

The ancestor of the Domain Name Service (DNS) is the **hosts.txt** file that contained the name to address mapping for ARPAnet. When ARPAnet contained a few hundred hosts, the hosts.txt file was easy to manage but with the addition of universities to the network, editing, compiling and using the file became difficult. The Stanford Research Institute (SRI) in Menlo Park, California administrated the hosts.txt file, which is a simple distributed database. Every computer in ARPAnet needed an updated hosts.txt file to be able to 'see' the other computers. SRI needed to be informed of every host addition, add to and compile hosts.txt then email the updated version to all current participants of ARPAnet. Host naming, network load and consistency quickly became a critical strain on the hosts.txt infrastructure.

The inadequacies of hosts.txt gave rise to advent of a faster and more automatic system for name to address mapping. The core of **DNS** was specified in 1983 in RFCs [1] 882 and 883 but it was not until 1984 in RFC 920 that DNS became fully specified. Paul Mockapetris wrote the first domain name server in 1984. It was called "Jeeves" and was used by SRI and the University of Southern California's Information Sciences Institute. DARPA commissioned a DNS server for Unix machines, which became to be known as the Berkeley Internet Name Domain (**BIND**) package.

BIND, from conception to infancy (version 4.8.3), was maintained by the Computer Systems Research Group at Berkeley with the aid of many other programmers. The maintenance, from infancy to early teens (v 4.9 and 4.9.1) was carried out by Digital Equipment Corporation (Compaq), during adolescence (v 4.9.2) by Vixie Enterprises and from early adulthood onwards (v 4.9.3 and above) by the Internet Software Consortium (ISC).

The current version of BIND is version 9.2.1 [2] released on the 1st May 2002 (as of 6th February 2003).

Summarised from [3] and [4]

4 Domain Name Service

4.1 Overview

DNS is a hierarchical, distributed database that stores information for hosts attached to the Internet to find each other. Without a naming service, such as DNS, mapping information from host name to IP address would be an impossible task. DNS also contains information related to email routing and data for other Internet Applications.

The following components make up the Domain Name Service: **Name Space**; **Name Servers**; **Resolvers**. The Name Space describes the position of the remote host in the hierarchy by means of a **domain name**, the Name Servers contain information on how to navigate the Name Space and the Resolvers query the Name Servers for the location of the remote host. The Resolver is the client running on the local machine that is given the URL (unified resource locator) of the remote machine to map to an IP Address.

Summarised from [3] and [4]

4.1.1 Name Space and Domains

The Name Space is the structure of the DNS database. It is organised in an inverted tree structure with the root node at the top as shown below in Figure 1. The nodes are marked with a dot in the figure. Each node must have a label and that label must be unique within its sibling group or level. Labels can be reused at differing levels on the tree. The root node is represented by the “null” symbol. A “.” Is used to represent the root node for convenience.

A label represents a Domain, which is a sub-tree of the domain name space from that point downwards. As shown below in Figure 2 the entire name space is included in the root domain (the area in light purple). To reference a distinct node the full domain name of that node is used. Figure 1 shows a domain name for *hostA* in University of Example, *hostA.example.ac.uk*. *hostA* is in the *example.ac.uk* domain. *example* is in the *ac.uk* domain and so on up to *root*. A nodes domain name identifies its position in the name space.

The node is associated with a **Resource Record** (RR) containing all the data associated with that particular domain, or pointers to the information (see 4.1.2 for more information).

Summarised from [3] and [4]

4.1.2 Delegation and Zones

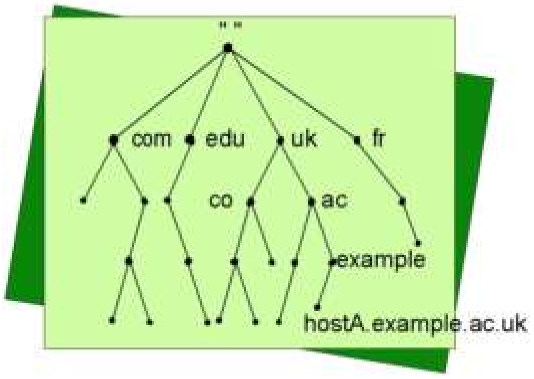


Figure 1: DNS Structure, the Name Space

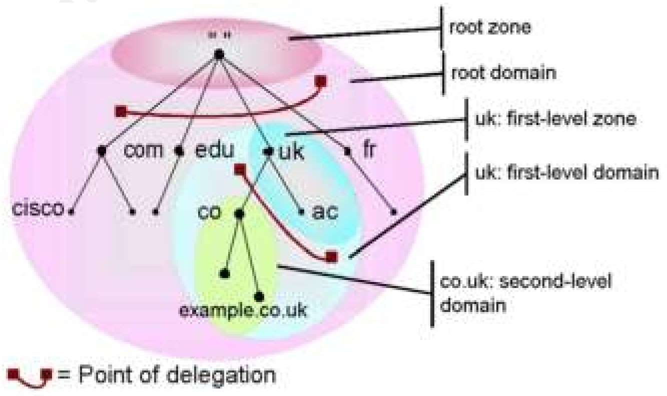


Figure 2: DNS distributed management, delegation by Domains and Zones

DNS is a vast distributed database that would be impossible to administrate (technically and politically) from a central location. To spread the responsibility of administration the name space is split into **zones** or points of delegation. Zones can be equal to a domain or a sub-set of a domain. In Figure 2 the *uk* domain delegates part of its administrative tasks to the *co.uk* domain. This results in the domain being split into 2 zones, one containing *uk* and *ac.uk* data (RR), the other containing *co.uk* data. The *co.uk* node is a sub-domain and a delegated zone of the *uk* domain. Each time an administrator delegates a sub-domain, a new zone or unit of delegation is created. Once a zone is created, the zone and its parent can be administered independently. Both the parent and the child keep a record of the delegation so name servers can navigation up and down the name space (the pointers are stored in the parent and child RR). The created boundary is the point of delegation (see Figure 2). Delegation is the key to the scalability of DNS.

Summarised from [3] and [4]

4.1.3 Name Servers and Resolvers

Name servers store the information about the name space in units called zones (discussed above), represented by a resource record (RR). The RR is stored in the name server at the zones delegation point e.g. for *co.uk* zone the RR would be stored at *nameServer1.co.uk*.

Resolvers query name servers about the information they contain in their RR to locate a remote host.

4.1.3.1 Resource Records

There are different classes of RR depending on which protocol or network the DNS is to run on. Since TCP/IP is the protocol of choice, its associated Internet class RR will be discussed here.

The zone file contains information about its zone, into records. Zone files normally comprise of a Start of Authority (**SOA**) record, a Name Server (**NS**) record, an Address (**A**) record, a pointer (**PTR**) record and a conical (**CNAME**) record plus some other records that will not be discussed in this document.

The SOA record (see Figure 3) indicates where the zone starts, its point of delegation, and states the name or names of the authoritative NS. The record also contains some information for use by slave NS.

The NS record (see Figure 4) indicates the Name Servers for the domain. The A record (Figure 5) points to the IP address for the host. This record is where the work is done and can be readily compared to the hosts.txt file mentioned in section 3. CNAME records provide a means for alias referencing.

Figure 6 shows the PTR records being used to map from IP to host name for the hosts mentioned in see Figure 3, Figure 4 and Figure 5 below. IP to host mapping is discussed in section 4.1.3.4.

```
example.co.uk. IN SOA alfred.example.ac.uk. clara.example.ac.uk. (
                1          ; Serial
                14400      ; Refresh after 4 hours
                3600       ; Retry after 1 hour
                604800     ; Expire after 1 week
                86400 )    ; Minimum TTL of 1 day
```

Figure 3: An SOA record for example.ac.uk

```
example.co.uk.  IN NS  alfred.example.ac.uk.
example.co.uk.  IN NS  clara.example.ac.uk.
```

Figure 4: A NS record for example.ac.uk

```
; Host addresses
localhost.example.co.uk.  IN A     127.0.0.1
alfred.example.ac.uk.       IN A     192.0.0.2
…
; Multi-homed hosts
clara.example.ac.uk.   IN A     192.0.0.4
clara.example.ac.uk.   IN A     192.0.0.5
…
; Aliases
ns1.example.ac.uk.        IN CNAME alfred.example.ac.uk.
ns2.example.ac.uk.        IN CNAME clara.example.ac.uk.
…
```

Figure 5: A and CNAME records for example.ac.uk

```
2.0.0.192.in-addr.arpa.  IN PTR alfred.example.ac.uk.
4.0.0.192.in-addr.arpa.  IN PTR clara.example.ac.uk.
5.0.0.192.in-addr.arpa.  IN PTR clara.example.ac.uk.
```

Figure 6: PTR records for example.ac.uk

4.1.3.2 Authoritative Name Servers

Each zone must have at least one authoritative name server, that is, a name server containing the complete information about its zone. A zone can have more than one authoritative name server; a name server can be authoritative for more than one zone.

If an authoritative name server is queried about the information it holds about its own zone, the response is an authoritative response called the "authoritative answer" and would have the AA flag set in the response.

The **Primary Master** server contains the master copy of the zone data. When it starts, the zone file is loaded from a local file called the "zone" or "master" file. Humans can edit this file and, when loaded, the changes will be propagated to the slave or **secondary** servers. The propagation of the loaded zone file to a secondary server is called a zone transfer. Secondary servers can receive the zone file from other secondary servers, creating a chain of replication.

Both the primary and secondary are authoritative for the zone they inhabit; the only difference being the secondary receives the zone file from the primary. Name servers can be authoritative for one zone and secondary for another.

Summarised from [3] and [4]

4.1.3.3 Recursive Name Servers

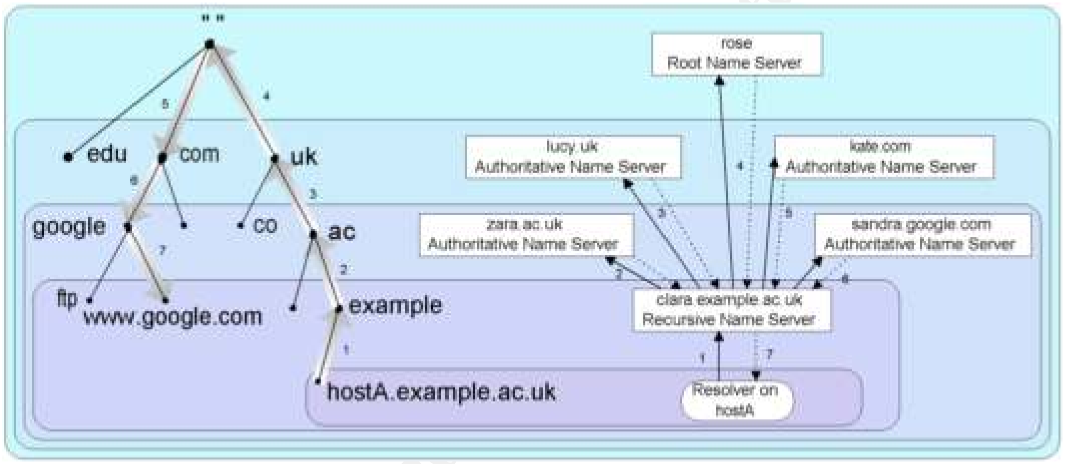
Most resolvers (see 4.1.3.4) are "stub resolvers", meaning they cannot perform a complete DNS host name to IP address resolution. The resolvers rely on a type of name server to do the resolution for them, called recursive name servers (RNS). These servers perform a recursive lookup by querying other name servers (NS) for the name resolution information until the information is found.

Name resolution would be a lengthy process if the RNS had to start from fresh every time a new query was received. To speed up the process the RNS cache the information they receive from other NS when they do lookups. After a lookup the RNS will know the IP address and authoritative domain for all NS it talks with. This information is useful for subsequent queries as shown below in Figure 8.

The cached information is kept for a certain period of time before it expires, enabling updates to be propagated from the authoritative NS. This duration is called the **time to live (TTL)** of the cached record. The administrator of the zone from which the record originated sets the TTL. Too short a TTL would mean exhaustive updates and network congestion, too long a TTL and the record could never be updated – a happy medium should be reached. Negative caching or "not found" responses are also cached but for a set time, 10 minutes.

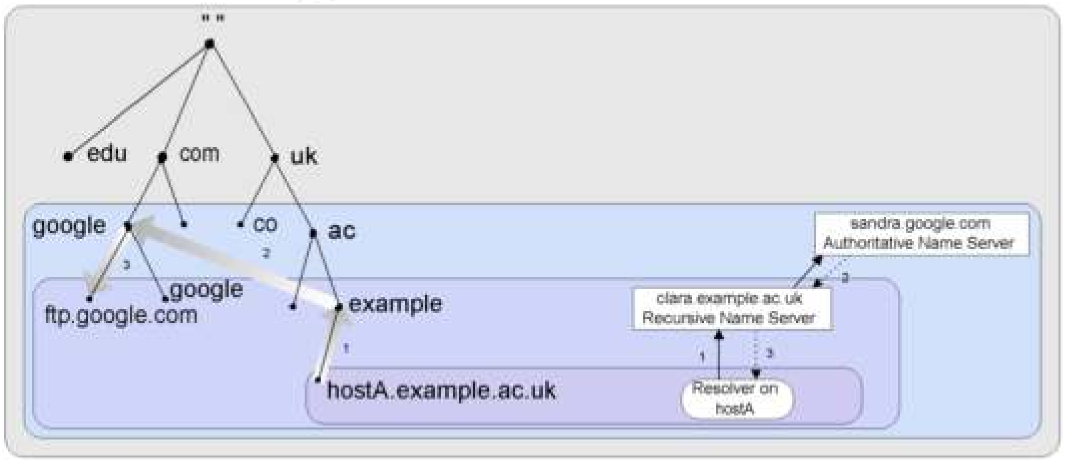
Summarised from [3] and [4]

4.1.3.4 Name Resolution


" "
rose Root Name Server
edu
com
uk
lucy.uk Authoritative Name Server
kate.com Authoritative Name Server
google
co
ac
zara.ac.uk Authoritative Name Server
sandra.google.com Authoritative Name Server
ftp
www.google.com
example
clara.example.ac.uk Recursive Name Server
hostA.example.ac.uk
Resolver on hostA

1. Q: Resolver query: what is www.google.com?
2. Q: What is www.google.com? A: referral to *uk* server.
3. Q: What is www.google.com? A: referral to *root*.
4. Q: What is www.google.com? A: referral to *com* server
5. Q: What is www.google.com? A: referral to *google* server.
6. Q: What is www.google.com? A: here's the IP for www.google.com.
7. A: Here's the IP for www.google.com.

Figure 7: The Resolution Process (no caching)


" "
edu
com
uk
google
co
ac
sandra.google.com Authoritative Name Server
google
ftp.google.com
example
clara.example.ac.uk Recursive Name Server
hostA.example.ac.uk
Resolver on hostA

1. Q: Resolver query: what is ftp.google.com?
2. I know *google.com* server. Q: ftp.google.com? A: Here is the IP for ftp.google.com.
3. A: Here's the IP for ftp.google.com.

Figure 8: The Resolution Process (with caching)

Since most resolvers cannot perform a lookup for themselves the RNS has to do almost all the work. A RNS will try to answer a query authoritatively (i.e. an IP of a host in its own zone) but if it were not authoritative for the information required, the RNS would go “hunting” (unless recursion is disabled). The “hunting” is called **recursive resolution** (described in Figure 7). Another form of resolution exists, normally carried out by authoritative and root NS, **iterative resolution**. Figure 7 shows iterative answers to a query for www.google.com. The NS return the best answer they know, instead of “hunting” for the answer.

Mapping Addresses to Names

We have discussed how to get from a name to an address, but not the converse. At first this seems to be a needle in the haystack problem as the only way to find a host based on its IP would be to query every host for its IP and try to find a match. Fortunately DNS provides a method to calm this madness, the **in-addr.arpa** domain. The nodes under the *in-addr.arpa* domain are labelled after the numbers in the dotted-octet IP address in the range 0..255. The *in-addr.arpa* domain has 256 sub-domains corresponding to first octet of the IP. The sub-sub-domain corresponds to the second octet and so on. The IP in Table 1: **IP Address split into octets** would be written: *1.0.168.192.in-addr.arpa*.

192.	168.	0.	1
1st Octet	2nd Octet	3rd Octet	4th Octet

Table 1: IP Address split into octets

The exhaustive search mentioned above is called an **inverse query**. To perform an inverse query the resolver would send a query to a RNS. The RNS would search its local data for the subject of the query. If no result is found the RNS would stop looking, returning a “not found” error.

Summarised from [3] and [4]

4.2 Security Issues

Summarised from [5] and [6]

The DNS server is prone to the same threats from malicious attack and poor configuration as any server or host attached to the Internet. Older versions of Operating Systems, not staying current with patches, poor server set-up and poor infrastructure configuration all play a part in leaving your DNS directory open to improper updates, slow responses, bad data, denial of service and other problems. This section will concentrate on the specific security issues relating to DNS, which may be more general threats taken from the DNS viewpoint.

4.2.1 Packet Interception and Query Prediction

If an attacker can sniff the traffic on a network used by DNS, the attacker can intercept the DNS packets and use that information to attack by taking on the "piggy-in-the-middle" role. This attack depends on the ability to **spoof*** the IP address, or assume the identity, of the NS.

Once the attacker is positioned "in-between" the resolver and the NS and has assumed the identity of the NS, it can intercept the queries sent to the NS. This attack relies on the NS being silenced by either a Denial of Service (DoS) attack or simply by the attacker being "closer" than the NS. This tends to be an information gathering stage for more serious attacks.

If the attacker is not positioned "in-between" the resolver and the NS, a similar attack can be attempted. The attacker must then predict the ID number [+] of the resolver request to be able to pretend to be the NS. Since communication between resolver and NS is done through single unsigned, unencrypted UDP packets, it is not difficult to sniff a few packets and predict the ID number.

Even without a history of ID numbers the ID number sequence can be hacked. In a DNS header the ID field is 16 bits, giving only 2^16 possible combinations, and with DNS working from a well-known UDP port, brute force guessing can easily crack the code.

Once the ID number is know the attacker is most of the way to sending the resolver bogus information. To complete the attack the attacker needs to know what type of query and the state of the query before the resolver can be fooled into believing the bogus information. This can be done by forcing the target into a known state through exploiting bugs in the operating system or crashing the target with a DoS attack, forcing a restart.

[*] Spoofing is the means by which the attacker assumes the IP address of a machine, the spoofed machine. The target machine (for instance the resolver) will be fooled into believing any packet sent to it from the attacker will have come from the spoofed machine.

[+] A simple means by which the resolver and NS keep track of the DNS conversation. Traditionally the number was simply incremented. To make things more difficult for an attacker a random number is used to start and increments of more than one is applied to the ID number of each dialogue.

4.2.2 Name-based Attack

Name-based attacks or **cache poisoning** are a very serious threat to the data stored in the Resource Record (RR) in a NS. Attackers use the entries in the RR with host name on the right hand side (see Figure 3, Figure 4 and Figure 5 above) to add false information to the NS. The most attacked records are the CNAME, NS and DNAME [§] that can be used to direct the resolver to false hosts or IP addresses.

This attack comes after the information gathering attacks mentioned in section 4.2.1. The attacker would induce the victim to make a query, silence the authentic NS, provide an answer to the query and at the same time add information to the cache of the resolver/RNS. The victim can then be induced to use this bogus information. This attack will only work if the victim has less reliable information about the same domain name i.e. the information was gained from a non-authoritative NS but this is scant protection from the attack.

Now the victim has bogus information that could be used to lead the victim to a mock up of well-known and well-used sites. All the end user has to do is enter his/her credit card details and the attacker has all the details necessary to defraud. The attacker may even go to the real site to order the same product to fool the credit card owner.

[§] The DNAME record is use to indicate a point of delegation in the *in-addr.arpa* domain.

4.2.3 Voluntary Name-based Attack

This is a strange but entirely probable form of “attack” perpetrated by the servers you dial-in to or the NS for the network you are plugging into. This kind of “attack” may be badly configured servers, bug or virus-ridden servers or deliberate misinformation to favour a partner company.

4.2.4 Denial of Service Attack

DNS is particularly useful to the attacker for DoS attacks since the reply is much greater than the query. The attacker can use the NS as a multiplier to carry out DoS attacks on other hosts or networks.

4.2.5 Zone Stealing

The DNS architecture is set up so one or many Secondary NS (see section 4.1.3.2) can retrieve a copy of the zone file direct from a Primary NS through zone transfer. If a bogus NS can pretend to be a Secondary to the Primary NS, the zone file will be transferred and all the information gathered can be used for future attacks. The next step could be to use this information to discreetly poison the caches of caching NS and resolvers as in section 4.2.1.

Summarised from [7]

4.2.6 BIND weakness

The NXT records, QUERY INVERSE records and NAMED (the names server software in BIND) allow immediate root compromise on many UNIX and Linux systems. The NXT and QUERY INVERSE problems are validation and buffer overrun bugs in the BIND code while the NAMED problem is a weakness in the name server software leaving the NS open to various “back door” exploits such as Trojans. Not all versions of BIND are vulnerable.

This weakness is part of the SANS top 10 vulnerabilities [8].

4.2.7 Microsoft Windows resolver weakness

Systems running Windows 98, NT, 2000 or XP are vulnerable to intended or accidental attacks due to the resolver on these systems accepting responses from any IP address [9]. This means any server can reply to queries sent to the resolvers RNS and will be accepted, as long as the reply is in the correct format and the ID number sequence is correct.

4.2.8 BIND Surveys

Date of Survey	% with at least one server running bad BIND (v8.2.x older than v8.2.3)
30th January 2001	33.3%

Table 2: Fortune 1000 companies with bad BIND

Date of Survey	% of bad BIND servers
31st January 2001	40.27%

Table 3: Bad BIND servers for .com's

Top level domains	Date of Survey	% of bad BIND (v8.2.x older than v8.2.3)
.co.uk (UK)	7th February 2001	18.6%
.de (Germany)	7th February 2001	28.6%
.ch (Switzerland)	7th February 2001	22.5

Table 4: Bad BIND servers for national Top Level Domains

On 29th January serious vulnerabilities were announced for BIND versions 8.2.x. The surveys, from Mice&Men [10], above in Table 2, Table **3** and Table **4** show the percentage of servers vulnerable to that particular exploit just after the announcement. These surveys demonstrate the necessity for best practices and continued revision of these practices in response to the changing environment. Even after a considerable time, in the tech world anyway, the % of servers running vulnerable versions of BIND stabilised at ~10% [10].

November 2002	
Errors	Percentage (to 2 significant figures)
Overall Errors	69%
Single Point of Failure	
All Name Servers in same subnet	28%
Only one Authoritative name server	6.6%
Zone Transfer	
No server allowed to zone transfer	39%
Some servers blocked zone transfer	6.0%
All servers allowed zone transfer	55%
Delegation/Zone Access Problems	
Incorrect delegation configuration	19%

Delegation data and zone data do not match	17%
None of the authoritative name servers answered	17%
2 authoritative name servers have same IP address	5.1%
There is only one NS record in the zone data	5.8%

Table 5: Survey of .COM name server errors and bad configurations, November 2002 carried out by Men & Mice

Table 5 shows a survey carried out by Men & Mice [11] late last year. 69% of .COM name servers with bad configurations is very high. More will be mentioned about these problems in section 4.3.

4.3 Best Practices

When building a server for use as a domain name server (or analysing the current setup) the implementer/administrator needs to think about the complete picture. If the domain name service is absolutely as secure as it can be but the server itself is running on an insecure, unpatched operating system and the secure entry door to the room is wedged open, there is no security. Physical security, logical security, change management, risk assessments, vulnerability assessments, standard policies and disaster recovery must all be in place on the server and on the network at large before any element of security is achieved. Remember there are enough problems facing experienced administrators without the good-intentional unsanctioned configuration updates made by the not-so-experienced "handy man".

4.3.1 System Best Practices

I will quickly run through some very important concerns when setting up the server that will run as a name server (authoritative or otherwise).

Knowledge really is the key to secure system. There is no such thing as too much knowledge in this case, but be careful to choose certified sources for the information and keep the updates regular.

Physically secure your system by **restricting access** to the box and the room to authorised personnel only. Bare in mind how easy it is for someone to accidentally turn off your box. If this in done before you have implemented your backup and disaster recovery procedures the system could be down for some time.

If an operating system component/service is not absolutely necessary it should be uninstalled or turned off, **the leaner the system the better**. Remember to turn off all unnecessary TCP/IP ports; an attack can't succeed if the ports are closed. If at all possible the name servers should be run on dedicated boxes so the leanest system can be achieved.

The root should have a **very strong password**, as a compromise here would leave the whole system open to the attacker. No service (unless absolutely necessary should be run with root privilege).

The necessary services should be run with their own user names giving the **baseline privileges** for the service to run. If the service is compromised, the attacker will be limited by those privileges. Remember to turn off default user accounts.

Keep your operating system software up-to-date with all current revisions, patches and service packs. **Security updates** should be downloaded as soon as they are available but functionality upgrades may not be so immediately important.

The system should be **tested and monitored** periodically for vulnerabilities, break-ins, miss-configurations, bugs and errors. Maintaining logs and proper documentation concerning the server can be the difference between a high page fault warning suggesting someone has hacked the server or the current version of explorer is a bit flaky.

Without **documentation** there is little point in attempting to secure the server, as there is little hope of the server being maintained in a secure state for very long. Without a history for the server all investigations into an incident are reduced to guesstimates. Document the server build, document all changes, document the procedures for making changes, document the scans, document the tests… document everything!

Risk management is an ever-increasing hole companies are getting used to throwing them selves down without truly thinking things through. Having 20 servers vulnerable to the same attack is as good as having a pop-up on the corporate website marked "confidential files this way". Having **no single point of failure** is the Holy Grail of securing servers. Not only must functionality be replicated over many servers but also these servers should be managed by separate teams, run different operating systems and software e.g. authoritative DNS server 1 running Linux/BIND v8.3.1, authoritative DNS server 2 running Solaris/BIND v9.2.1 and authoritative DNS server 2 running windows 2000 server and an alternative DNS software. Should these servers share the same subnet, router, leased line, building or power grid, the Holy Grail is still at the end of the rainbow.

Summarised from [12]

4.3.2 Domain Name Service Best Practices

Domain Name Servers should be extended greater consideration when planning the security of the corporate site due to the special nature of the facilities provided; the NS should be available to persons inside and outside the corporation.

Section 4.2 mentions all the bad things waiting to happen to a NS secured or otherwise, what we are trying to do is mitigate the risk of the servers being attacked and the impact should a server (or servers) fall. What follows is a candid introduction to the pitfalls of NS security. The reader should follow the references at the end of this document to gain a broader and deeper understanding of the subject.

4.3.3 Securing a Domain Name Server (BIND)

As ever the newest version of the generation of BIND should be used, with all the up-to-date patches correctly applied. This is a good launching pad to security but by no means have we taken off.

Consider creating two types of physically separate name server: The Advertiser; The Recursive Resolver. Separating your servers physically separates domain name resolution into two distinct roles. The Advertiser is available to the Internet, authoritative for the corporate zone and only provides answers for information it holds authority over. It is not recursive and will not go looking for answers. The Recursive Resolver is reserved for recursively answering queries from corporate users. It caches answers and acts as the workhorse of the DNS. Since it caches information it is prone to cache poisoning but not being the authority for the corporate zone, the damage is limited – the risk is mitigated.

Since the servers are on different machines they can be better protected from vulnerabilities in other software (not that there will be much there as all unnecessary services and software has been deleted. Section 4.3.1) running on the machine. DNS uses UDP port 53 and TCP port 53 so everything else should be filtered by the machines interface and by the attached router. Closing all unnecessary ports is the best way to shut out attacks. We can be a little cleverer here and make sure external clients can't gain access to or receive answers from The Recursive Resolver. Within the NS, access control lists can be set-up to only allow authorised IP addresses to access the service, adding another layer of protection.

An attacker can learn a lot about the corporate infrastructure from a stolen zone file (Section 4.2.5). Restricting who can initiate a zone transfer will tie one hand of the attacker. In BIND restricting access to the file is based on IP address. This is ok but spoofing is very easy. Authenticating the transfer is a much better approach. This is achieved through TSIG [*], coupling something you are (IP address) with something you know (shared secret). Bare in mind any server that is authoritative for the zone must guard its zone file from attackers; this means both the master and slave.

Dynamic update is a mechanism to break the overhead of static zone transfers, each time a piece of information is changed the change is sent to the other authoritative servers. It may reduce network and server load but it opens the floodgates. TSIG authentication and IP restrictions must also be used with dynamic updates or you will realise all too late that although you double locked the front door, the cat flap has let in a lion.

Even with all these precautions the NS can and will be compromised. When this happens the attacker will be given free rein of the server, possibly using this server as a privileged tool to attack other servers. To prevent such a catastrophe the domain name service should be run in a "jail". The service should be run under a restricted user and limited to a subdirectory of the file system. If the attacker takes over the service resources will be limited to that user and subdirectory.

[*] Transactional Signature (TSIG) is an addition to DNS that facilitates the authentication of a host through a shared secret. It is used to secure zone transfers and possibly to secure the relationship between the resolver and RNS.

Summarised from [3], [4], [13] and [14]

5 Domain Name Service Security Extension

5.1 Overview

The DNS Security Extension (DNSSEC RFC2535 [1]) was conceived to address the limitations and intrinsic exploits in DNS described in Section 4.2.

DNSSEC has 2 main aims: to confirm the integrity of the information; to authenticate the source of the information. DNSSEC dose not guard against poor configuration or bad information in the "real" authoritative name server. Authentication is obtained by means of digital signatures created using the MD5/RSA and DSA algorithms.

In DNSSEC each resource record entry is signed allowing verification by the zone's public key. The zone file is also signed so servers attempting a zone transfer can verify all information is present and correct.

The SIG and KEY records are added to the RR for authentication support. There must be a SIG record for each entry in the RR plus a SIG for the whole file (regenerated each time the zone file is updated). The KEY record stores the public key of the name server, used to verify the SIG records. Figure 9 shows a simple zone file and Figure 10 shows the equivalent file in DNSSEC.

The NXT record is meant to authenticate the non-existence of a RR. The DNSSEC RR is conically ordered with NXT specifying the next record. This is a defence against replay attacks.

```
example.co.uk.          IN   SOA     alfred.example.ac.uk. (
                                     1
                                     14400
                                     3600
                                     604800
                                     86400 )
example.co.uk           IN   NS      alfred.example.ac.uk.
alfred.example.ac.uk.   IN   A       127.0.0.1
```

Figure 9: A simple zone file in DNS

```
example.co.uk.          IN   SOA     alfred.example.ac.uk. (…)
                        IN   SIG     SOA 1 86400 (           ; RR, alg. type, TTL
                                     20030305120312          ; SIG expiration time
                                     20030205120312          ; SIG inception time
                                     87964 example.ac.uk     ; key tag, signer's name
                                     jk86HGD67*gs-%…)        ; the signature
                        IN   SIG     AXFR 1 86400 (          ; zone transfer signature
                                     20030305120312
                                     20030205120312
                                     7tRRe#3si09*'d?/… )
example.co.uk.          IN   NS      alfred.example.ac.uk.
                        IN   SIG     NS 1 86400 (            ; NS signature
                                     20030305120312
                                     20030205120312
                                     ?tgRe#jk86HG?!… )
                        IN   NXT     alfred.example.ac.uk.   ; NXT record
                                     NS SOA SIG NXT
```

```
alfred.example.ac.uk.    IN   A      127.0.0.1
                         IN   SIG    A 1 86400 (
                                     20030305120312
                                     20030205120312
                                     ^09PoqE4#tty… )
                         IN   KEY    Jjhu$^Uld49875*…    ; the alfred public key
                         IN   SIG    KEY 1 86400 (
                                     20030305120312
                                     20030205120312
                                     jG6^09PoqE4#… )
                         IN   NXT    example.ac.uk.      ; the next record is the
                         IN   SIG    KEY 1 86400 (       ;start of the RR.
                                     20030305120312
                                     20030205120312
                                     wwg%qw45… )
```
Figure 10: The same zone file in DNSSEC

Now we have a signed RR with associated public key but we must trust the source of the key to trust the information. DNSSEC provides a chaining mechanism to enable this trust. Each zone is signed by its parent's signature.

So if the resolver queried its resolving name server, the RNS would search for the authoritative information. Once found the RNS would verify the authenticity and integrity of the information by retrieving the signature of the zone's parent and the parent's parent until a trusted signature was found. The worst case would be the chaining all the way up to the root (the roots signatures would need to be available in a well known and trusted place). The chain of NS to the trusted ancestor would then become trusted NS. With trust established, the individual record required would be authenticated by its SIG. Now the RNS can give the resolver true information.

Summarised from [3], [4], and [15]

5.2 Issues

DNSSEC is the answer to many issues threatening DNS today. Its ability to authenticate the source and validate the information integrity prevents the attacks mentioned in Sections 4.2.1, 4.2.2, 4.2.3, 4.2.4 and 4.2.5 but it does have some problems of its own.

It is complex to implement. Configuring delegation is fraught with problems, compounded by DNSSECs poor error reporting.

DNSSEC zone files are very big, loading both the network and RNS or resolver (if its DNSSEC aware). When vulnerabilities are found in DESSEC, the large record sizes would aid any denial of service attacks.

As with any algorithm, the public/private keys used with DNSSEC can be hacked with time. In response the keys should be changed at various intervals to reduce the risk of a compromise. This is very easy to implement at the lower levels of the DNSSEC hierarchy as the public keys are not cached for very long but the root key are a problem. Authentication is based on the root keys being known. It will be difficult to regularly change the root keys without impacting the DNSSEC structure.

DNSSEC RNS use absolute time to verify a signature is still valid. If an attacker can change the RNS opinion of the current time, he/she can trick the resolver into believing an expired signature.

Summarised from [5]

6 The Future

DNSSEC solves the current top issues with the domain name system. It is the way forward. Without it companies, online shops, banks, governments and home users will continue be susceptible to threats the original DNS was never designed to cope with. Without a method of authenticating the public information held in name serves attackers will have a very easy and effective weapon in their arsenal. The future is DNSSEC.

---

[1] RFCs located at http://rfc.sunsite.dk/main.html

[2] BIND v 9.2,1 release notes held at http://www.isc.org/products/BIND/bind9.html

[3] Liu, Cricket & Albitz, Paul. DNS and BIND Third Edition. O'Reilly, September 1998, Chapters 1,2, 4, 10 and 11.

[4] BIND 9 Administrator Reference Manual. Internet Software Consortium, 2001, Chapter 2, 4, 7 and Appendix A

[5] Atkins, D & Austein, R. Threat Analysis of The Domain Name System. Network Working Group, November 2002. (2 Feb. 2003)

[6] Sweetman, James. Current Issues in DNS Security: ICANN's November 2001 Annual Meeting. SANS, 28 Nov. 2001. (5 Feb. 2003)

[7] Bellovin, Steven M. "Using the Domain Name System for System Break-Ins", Proceedings of the Fifth Usenix UNIX Security Symposium, Salt Lake City, UT, June 1995. (20 Jan. 2003)

[8] How To Eliminate The Ten Most Critical Internet Security Threats: The Experts' Consensus, Version 1.33. The SANS Institute, 25 June 2001. (1 Jan. 2003)

[9] Vulnerability Note VU#458659, CERT Advisory, 14 July 2000. (3 Mar. 2003)

[10] Men & Mice Research on BIND Security: BIND Vulnerability. Mice & Men. (28 Feb. 2003)

[11] Domain Health Survey for .COM. Mice & Men, November 2002. (28 Feb. 2003)

[12] Setty, Harish. System Administrator – Security Best Practices. SANS, 16 August 2001. (6 Mar. 2003)

[13] Householder, Allen & King, Brian. Securing the Internet Name Server. CERT Coordination Centre, August 2002. (1 Mar. 2003)

[14] Liu, Cricket. Transactional Security in BIND 9: Securing DNS. Linux Magazine, November 2001. (20 Feb. 2003)

[15] Lioy, A & Maino, F & Marian, M & Mazzocchi, D. DNS Security. Dipartimento di Automatica e Informatica, Politecnico di Torino, May 2000. (7 Feb. 2003)