The Evolving Threats to the Availability and Security of the Domain Name Service

John Holmblad
SANS GIAC/GSEC Practical
October 5, 2003

Table of Contents

1 Abstract 3
2 Introduction 3
3 DNS Definitions 4
4 DNS History 11
5 DNS Architecture 13
6 DNS Implementation 13
7 DNS Server Side Vulnerabilities 15
7.1 DNS Vulnerabilities: Spoofing of DNS Responses 16
7.2 DNS Vulnerabilities: Cache Poisoning 17
7.3 DNS Vulnerabilities: Email Spoofing 17
7.4 DNS Vulnerabilities: Exploit of Known Security Related Software Faults 17
7.5 DNS Vulnerabilities: Improper DNS Configuration 18
7.6 DNS Vulnerabilities: High Profile and Successful Attacks 18
7.6.1 Microsoft DDOS Attack 18
7.6.2 Nike WWW Site Hijacking 19
7.6.3 Adobe WWW Site Hijacking 19
7.6.4 RSA WWW Site Hijacking 19
8 DNS Client Side Vulnerabilities 20
8.1 Trojan.Qhosts Vulnerability and Exploit 20
8.1.1 Method of Exploit 20
8.1.2 Exploit Severity 20
8.1.3 Exploit Mitigation 21
9 DNS Server Hardening against Attacks 21
9.1 Subnet Diversity 21
9.2 OS Platform Diversity 22
9.3 Separate Public DNS from Internal DNS (Split-Horizon Operation) 22
9.4 Don’t Share DNS server with other Applications 22
9.5 Restrict Zone Transfers 23
9.6 Configuration Hardening 23
9.7 Release Currency 23
9.8 DNS Isolation 24
9.9 DNS Functional Splitting 24
9.10 DNS Version Number Hiding 25
9.11 DNS Application Diversity 25
10 DNS Client Hardening against Attacks 25
11 Alternatives to BIND 26
11.1 Dents 26
11.2 Djbdns 26
11.3 MaraDNS 26
11.4 CustomDNS 27
11.5 Ibnamed 27
11.6 Ibdns 27
11.7 Microsoft DNS 27
12 The Future Evolution of DNS 27
12.1 DNSSEC Support 28
12.2 TSIG Security Improvements 28
13 Conclusions 28

1 Abstract

This objective of this paper provide a concise overview of the role of the Domain Name Server (DNS) system among the essential components that comprise the Internet and the World Wide Web as we know it today, and to examine the security related aspects of its operation and some of the key exploits that have been mounted in the last several years against the system and the services that it provides. Sections 2 to 4 of this paper focus on the reasoning behind the creation of DNS, sections 5 and 6 discuss the network architecture and implementation of DNS, Section 7 reviews a number of vulnerabilities that have been discovered in various implementations of DNS server software, while section 8 does the same with respect to DNS client software, section 9 describes numerous ways in which DNS servers can be made less vulnerable to attack through various “hardening” techniques, while section 10 section provides the same for the client side. Section 11 provides a quick synopsis of other implementations of DNS besides the dominant one known as BIND, and section 12 provides a perspective on the future of DNS.

2 Introduction

Domain Name Service is delivered over the Internet by means of a distributed software system called the Domain Name Server. DNS is of interest to study because of the vital role it performs in the Internet for users of higher layer services (i.e. above the TCP layer) such as the www, ftp, email, and other end-to-end services. In fact, the vast majority of the world’s Internet users don’t know of (nor do they care about) the existence of DNS, or of the vital role it plays every time they type in or click on a Uniform Resource Locator (URL) within their www browser. On the other hand, if DNS is not functioning properly, then for all such users, the www is, for all intents and purposes, unavailable to them. DNS has been likened to a water or electric utility in that the service it provides is not the end service, but, without it, nothing associated with that utility service will function (e.g. sprinklers, lights, A/C, etc.)

From a reliability engineering perspective, DNS represents a serial element in the concatenated chain of systems that must be operating properly in order for the www to be available to a user. For a user connected to the Internet via a dedicated access transmission circuit the availability model (excluding the target www server) would be as shown in **Figure 2.1**
 Below.

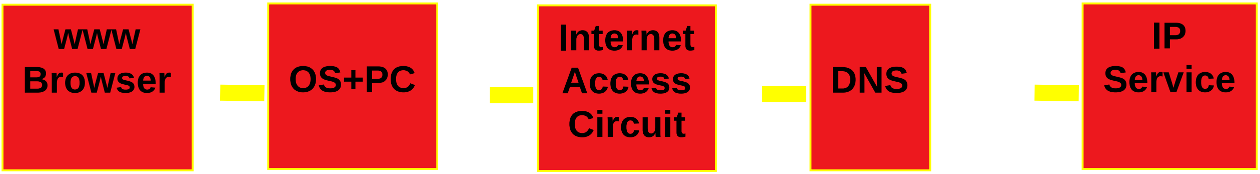


Figure 2.1

As this model implies, DNS is no less important than the other elements in this chain of less than perfectly reliable systems, whose ultimate purpose is to deliver reliable services to the user of the www browser. For high availability networks, a desirable goal for overall availability is the so-called 5 9's of availability target, or 99.999%. Such an availability requirement implies that the system must be unavailable for not more than ~5.26 minutes of outage per year! Relaxing this by an order of magnitude to 99.99% availability would constrain the level of outage to no more than ~52.6 minutes per year. As a consequence of the positioning of DNS in the hierarchy of systems above, the original architects of DNS within the Internet Engineering Task Force (IETF) put forth considerable effort to design a system that is highly reliable, and which takes the possibility of failure of a portion of the system into account in the design. It is fair to say that, in fact, from a reliability engineering perspective, the original architects did succeed in their effort. However much has changed about the Internet since then, not the least of which is the dominance and, therefore, importance of the www. The security architecture of DNS received little attention at the time that DNS was specified and it is the current generation of IETF engineers who have had to grappling with the need for a comprehensive security architecture and how, in the meantime, we can "harden" DNS against attack.

3 DNS Definitions

DNS can best be described by analogy with the U.S. local telephone network, whereby, a user dials a service named, appropriately enough, Directory Service, typically using the dialing sequence 411. The user then provides a query to the operator stating the name for which the user would like the corresponding telephone number. The operator then consults the directory, retrieves the number if it is available, and delivers the response back to the user. In the case of the Internet and DNS, the user can be thought of as the www browser client (e.g. Netscape Communicator or Microsoft Internet Explorer) and the collection of systems and software that deliver DNS to be the operator**. Figure 3.1** below provides an illustration of this query response sequence.

From a more technical perspective, DNS is a distributed hierarchical database application that maps (converts) Internet domain names (e.g. seas.gwu.edu) into IP addresses (e.g. 123.21.51.2) in response to a request for such conversion[1]. It is also able to respond to a request for an inverse mapping, that is, from an IP address into a domain name. The database that contains the mapping information is both distributed and hierarchical in its structure and can be represented as an inverted tree structure. All of the domain names that are registered with a naming authority (more about that later) are held within this database along with their associated IP addresses. The general structure of a domain name is:

.XXX……YYY.ZZZ.WWW

Where

.XXX, .YYY, and .ZZZ represent sub-domains

and

.WWW represents the Top Level Domain or TLD

The TLD's originally specified by the ITEF include the following:

- ➔ .edu (educational)
- ➔ .com (commercial)
- ➔ .org (organizations)
- ➔ .net (network providers)
- ➔ .int (international)
- ➔ .gov (US governmental)
- ➔ .mil (military)
- ➔ .xy where xy = ISO country code (e.g. au for Australia)

Each TLD has associated with it, at least one special server known as a root name server, which, in turn, points to each TLD for which it has authority. **Figure 3.2** provides a visualization of the domain hierarchy. Subdomains can be created to an arbitrary level below each TLD as required by the needs of the organization to which the responsibility for a given subdomain is given.

Domain Registrars have the authority over the TLD's that are used within the Internet. These Registrars, in turn, can and do grant the authority over subdomains to the organizations (e.g. Amazon.Com, Inc. for the .amazon subdomain) that manage those subdomains. Within a subdomain, further delegation of authority is possible. The delegation of authority in this case is managed entirely by the manager of that subdomain. Thus there is a clear principle of division of authority (and corresponding responsibility) within the structure of DNS.

In reality, the boundary of authority that is granted within DNS is determined by means of the specification and implementation of Zones. This can be best illustrated with a diagram and is shown in **Figure 3.3** below. In this figure there are 4 zones pictured:

- ➔ a root zone
- ➔ a TLD zone (.edu)
- ➔ a subdomain (gwu.edu) which is a zone
- ➔ a subdomain of gwu.edu (seas.gwu.edu) which is a zone.

The remaining subdomain, law.gwu.edu is not, in this case, defined as a separate zone. As a consequence, this subdomain will NOT be granted authority over the namespace management of its subdomain. Rather, the parent subdomain, gwu.edu, which is defined as a zone, will provide this management. The decision as to whether or not to create multiple zones within a zone is determined by the entity (in this case, George Washington University) that has authority over that zone in question. This decision in turn is generally taken in consideration of the size and complexity of the network or networks included in the zone in question as well as the capabilities of subtending subdomains to effect proper management of their allocated zone or zones.

As mentioned above, the root of the DNS is called the root server. In fact, the Internet, today, has a total of 13 root servers that are distributed geographically around the world (U.S, U.K., Japan, and Sweden) in order to effectively manage DNS performance while assuring very high availability. In addition to this geographic resiliency, each of these rootservers is designed with redundancy to distribute load as well as to provide resiliency in the face of hardware or software failure of one or more of the rootservers.

Since 1998 the authority over Internet namespace (IP addresses, domain names, autonomous system numbers, etc.) has been granted to the Internet Corporation for Assigned Names and Numbers, or ICANN. It is a non-profit international corporation that manages:

- ➔ IP address space allocation
- ➔ Protocol parameter assignment
- ➔ DNS management
- ➔ Root Server management functions

ICANN, in turn, delegates address management authority to the regional Internet Registries, which include, at the top level the following organizations:

- ➔ ARIN: Americas (www.arin.net)
- ➔ RIPE: Europe (www.ripe.net)
- ➔ APNIC: Asia/Pacific (www.apnic.net)

Depending upon the region, there may be, for example, country level delegation of authority over domain name allocation services within that country.

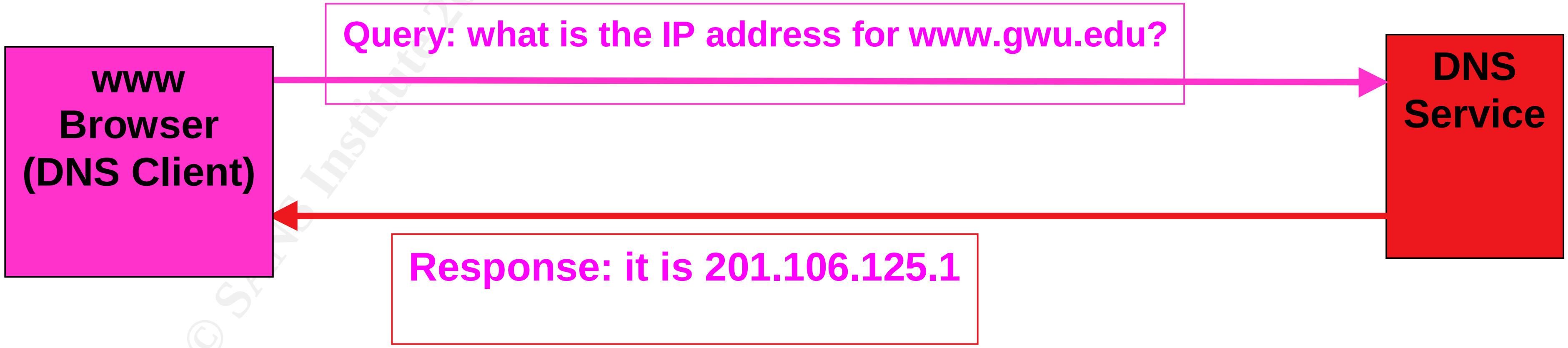

www
Browser
(DNS Client)
Query: what is the IP address for www.gwu.edu?
DNS
Service
Response: it is 201.106.125.1

Figure 3.1

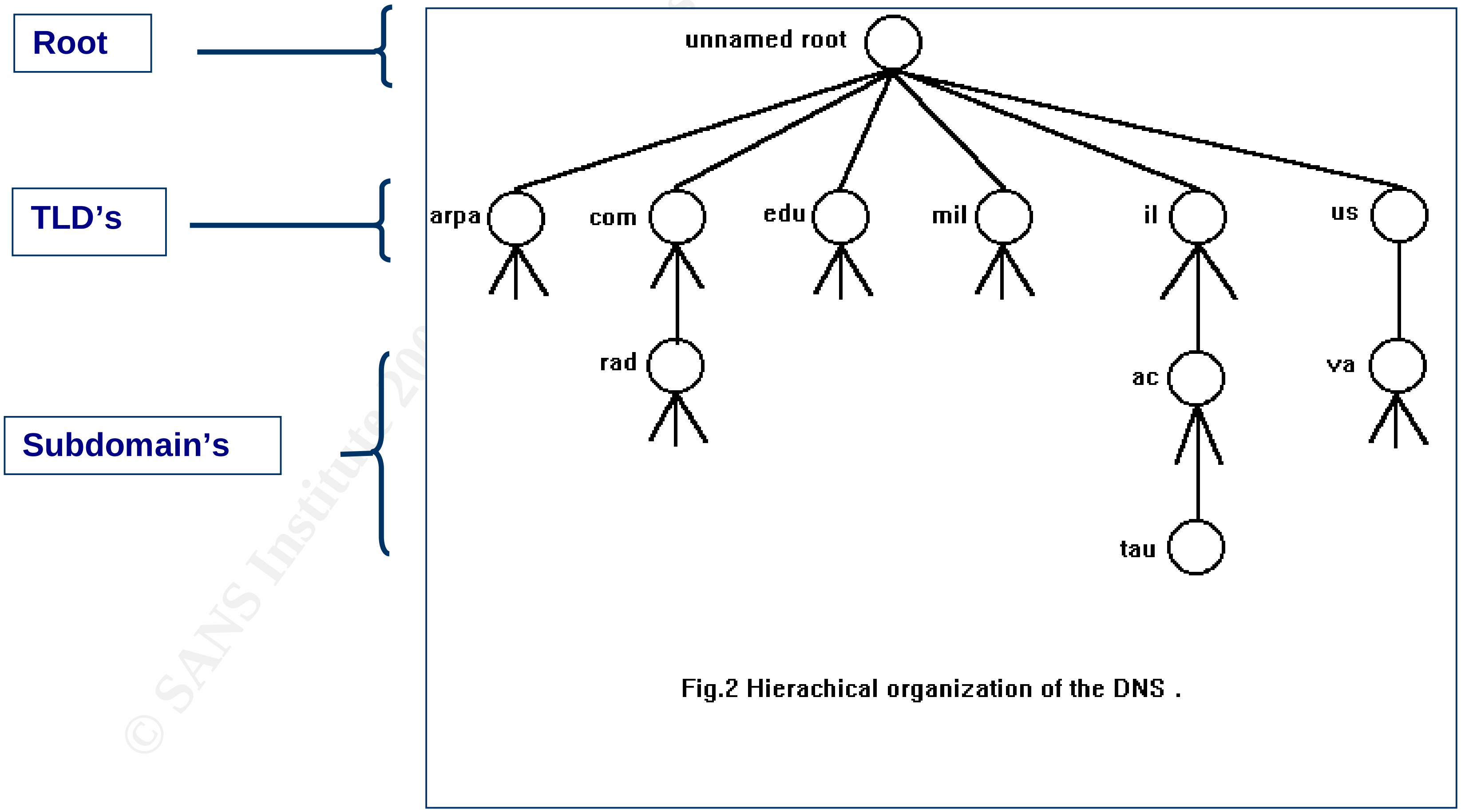


Fig.2 Hierachical organization of the DNS .

Figure 3.2[2]

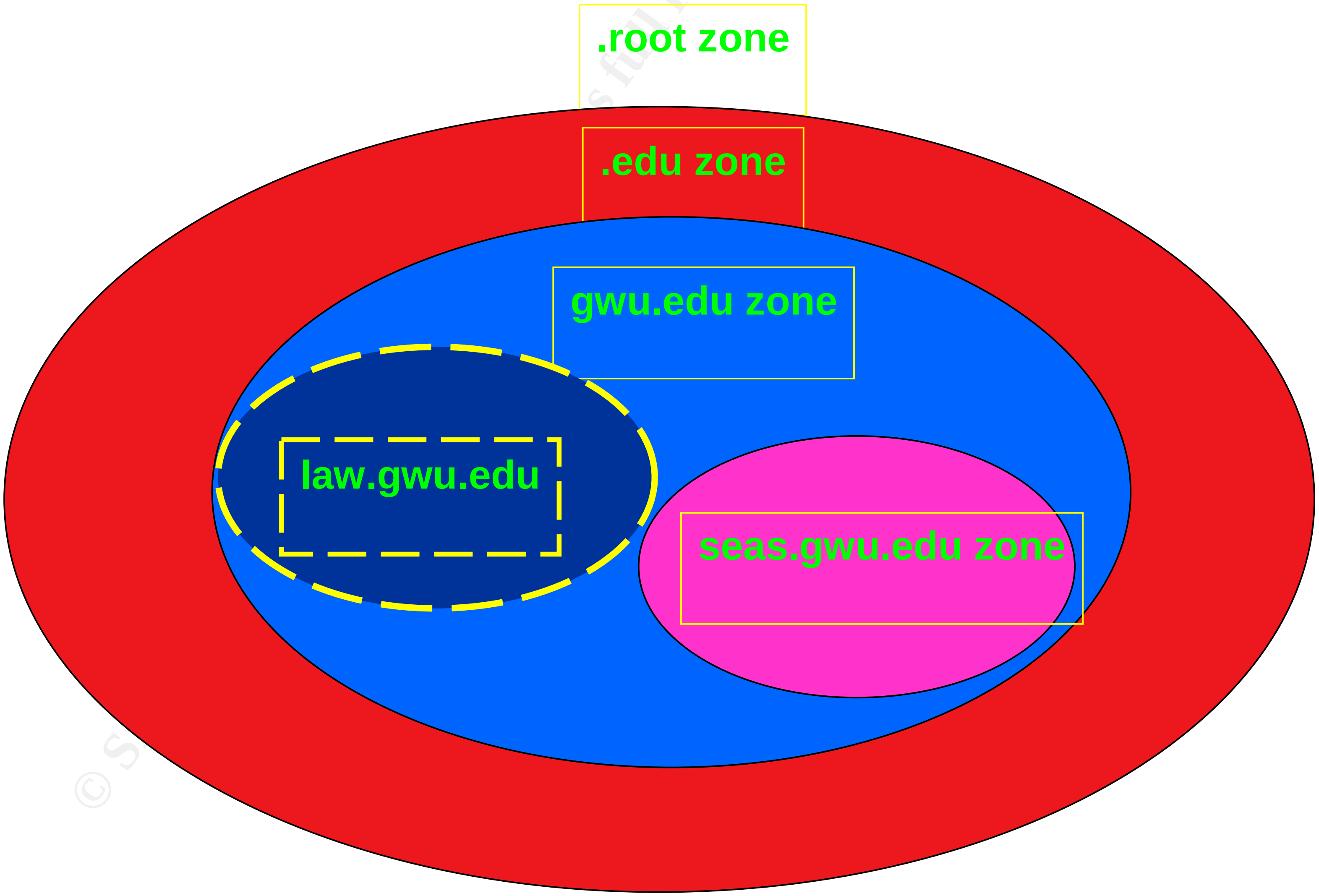
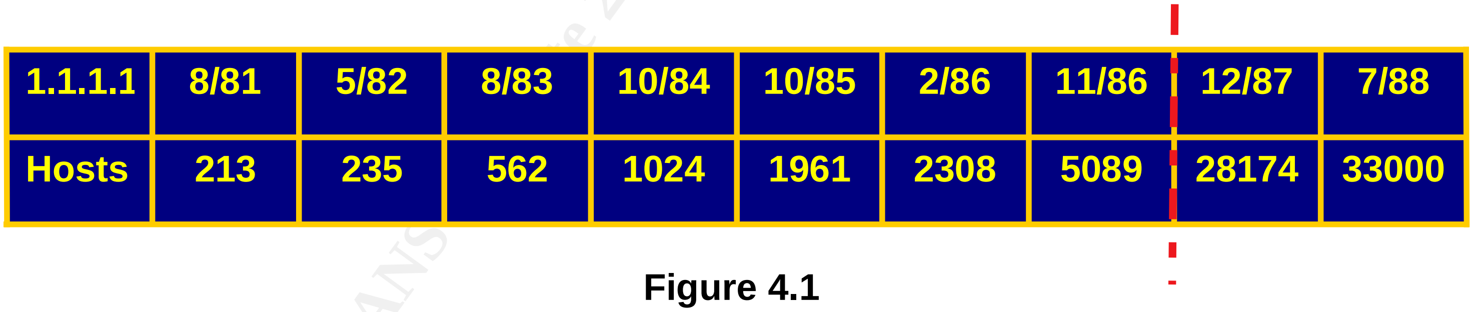


Figure 3.3[3]

4 DNS History

During the development of the Internet, many things have been and continue to be hotly debated. Not in question, however, has been the obvious fact that the Internet and, more specifically, the www have grown phenomenally since the day in the 1960's when the first packet switch (or routers as they are called today) was turned up by engineers working for Bolt Beranak and Newman (BBN) who had won the contract to build the Arpanet, which was itself, the predecessor to today's Internet. Many of the useful innovations spawned by the Internet were, in fact, a result of necessity, as opposed to a foreordained plan, and DNS is one of them.

From the late 1960's through 1970's, the number of host computers (now called servers) connected to the Arpanet/Internet, were a very manageable number. Therefore it was possible to maintain a simple text file, a flat file database if you will, that could be read by humans as well as computers, and which contained the mapping between the textual host names and their corresponding numeric IP addresses. In fact, this task was personally carried out by John Postel[4], one of the pioneers of the Internet and the IETF and a highly accomplished networking engineer. This file was distributed to each of the host computers attached to the Internet using the File Transfer Protocol (FTP). As the network grew, however, and especially once the Internet reached "escape velocity" in terms of host growth, the bandwidth overhead of these FTP's became too burdensome on the network itself. This acceleration of the growth rate of the Internet can be seen in **Figure 4.1** below, first occurring between November of 1986 and December of 1987.



1.1.1.1	8/81	5/82	8/83	10/84	10/85	2/86	11/86	12/87	7/88
Hosts	213	235	562	1024	1961	2308	5089	28174	33000

Figure 4.1

It should be noted that this acceleration in the Internet growth rate took place even before the www was created in the early 1990's, at which point the growth rate picked up even further as shown in **Figure 4.2** below.

Fortunately, the IETF architects of the Internet foresaw this continued growth of the Internet, though few, if any, predicted beforehand, the overwhelming success of the www. This foresight led to the development of the IETF specifications for Domain Names and for the DNS functionality. The current versions of these specifications can be found in IETF documents RFC #1034[5] and RFC #1035[6].

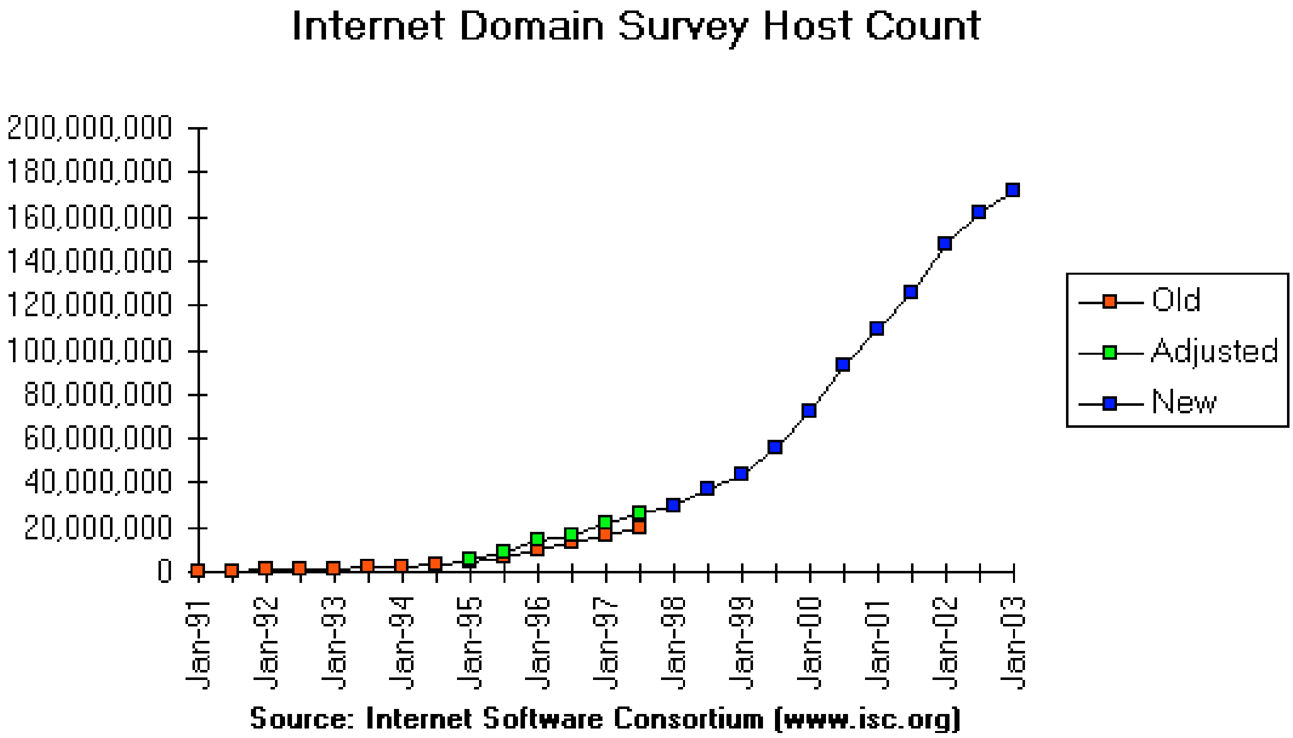


Figure 4.2

5 DNS Architecture

The architecture of DNS, especially considering the timeframe when it was originally specified has proven quite durable. By utilizing the already proven lower layer protocols up to and including TCP, IP, and UDP it was possible to specify a machine and location independent platform for name resolution that could rely on the services of those lower protocol layers. The only material aspect, in which the original DNS architecture can, in hindsight, be said to be lacking, is in the area of security and its explicit trust in the systems with which DNS has to interact to accomplish its tasks.

As mentioned previously in this paper, DNS is properly characterized as a distributed database (DDB) application with a hierarchical structure. DNS implements a client server process architecture, where the client side is represented by a Resolver, which submits queries to and receives responses from the DNS application itself. The Server side responds to queries from Resolvers, downstream DNS servers, and DNS servers performing a DNS Zone transfer. The primary protocol used for communications is the Unsequenced Datagram Protocol (UDP), which is very efficient (one packet in/one packet out) for Resolver⇔DNS Server communications. When multi-packet exchanges are required, which is the case when a DNS Zone Transfer takes place, then the Transmission Control Protocol (TCP) is used to provide a session (oriented) channel over which multi-packet reliable transmissions can take place. To better support the large volume of transactions that a DNS is expected to support, the concept of caching is also implemented within DNS. This not only speeds up the name resolution process, but it reduces the overhead traffic on the Internet caused by DNS queries that cannot be satisfied by the local DNS.

Given the vital role that DNS was designed to fulfill in the Internet, resiliency of DNS functionality was of paramount concern to the original architects. Therefore 1+1 protection of this functionality was considered a critical, if not mandatory requirement. As a consequence, the architecture spells out both a primary and a secondary DNS for each zone. In concert with this architectural specification, most TLD authorities require a minimum of two functional name servers for a zone before they will delegate authority to the zone owner.

6 DNS Implementation

Paul Mockapetris, who is the author of the aforementioned IETF recommendations pertaining to DNS, developed the first actual implementation of DNS, which was called JEEVES. Subsequently, a team of graduate student software developers at UC Berkley developed what is now considered the de-facto standard DNS, called the Berkley Internet Domain package, or BIND. The group at Berkley continued to maintain this software through version 4.8.3, after which, a group at Digital Equipment Corporation, now Compaq, took over the product and issued two versions, 4.9 and 4.9.1. The next version, 4.9.2 was released and sponsored by Vixie Enterprises, and, from version 4.9.3 onwards, the software has been developed and maintained by the Internet Software Consortium and is available for downloading at no cost at their www site, www.isc.org.

The ISC version of BIND consists of three components as follows:

- ➔ **a Domain Name System server (named)**
- ➔ **a Domain Name System resolver library**
- ➔ **tools for verifying the proper operation of the DNS server**

Although the latest version of BIND is version 9.2.2, in fact, the most widely deployed versions at this point in time are versions 8.X.X, of which the latest version is 8.3.1. With version 9.0, ISC undertook to perform a complete rewrite of the software and to also incorporate a number of important new features and capabilities, as well as enhanced security features.

Figure 6.1 below highlights the major new capabilities and features introduced into BIND versions 9.X (see also http://www.isc.org/products/BIND/bind9.html). Space does not permit an extensive discussion in this paper of all of the new features of this version of BIND, however, among the most important, perhaps, are the ones pertaining to security of the DNS itself. These security related features were initiated with the later versions of Bind 8.X.X and have been fully implemented in release 9.X. These security features are briefly discussed in section 12 of this paper.

●DNS Security
●DNSSEC (signed zones)
●TSIG (signed DNS requests)
●IP version 6
●Answers DNS queries on IPv6 sockets
●IPv6 resource records (A6, DNAME, etc.)
●Bitstring Labels
●Experimental IPv6 Resolver Library

●DNS Protocol Enhancements
●IXFR, DDNS, Notify, EDNS0
●Views
●One server process can provide multiple "views" of the DNS namespace, e.g. an "inside" view to certain clients, and an "outside" view to others.
●Multiprocessor Support
●Improved Portability Architecture

Figure 6.1

7 DNS Server Side Vulnerabilities

Because DNS serves such a visible and critical function on the net, it represents an obvious target of opportunity for disruption and, in fact, such disruption has occurred from time to time, much to the dismay of those who are tasked with administering DNS throughout the Internet. Because of its flexibility, resiliency, distributed nature, and, in the case of BIND, having been constructed on a UNIX OS environment, DNS is complicated to administer correctly. Surveys have shown that perhaps as many as 25% of the extant DNS's today are not well managed[7].

It is also important to note that, when the DNS architecture was first specified in the 1980-1983 timeframe, the Internet was in its "early days", and trust in other systems (not to mention individuals) was the rule and not the exception. It is not surprising then, that, since 1997 the Computer Emergency Response Team (CERT) at Carnegie Mellon University has published 12 documents regarding vulnerabilities and exploits with BIND[8].

Given the overall importance of DNS in the "plumbing" of the Internet, ICANN sponsored a 4 day meeting in November 2001 on the subject of DNS security issues[9]. Among the many issues discussed at that meeting, some of which are presented later in this report, was the need to run a live test of the contingency and recovery plans of the operators of DNS. Historically, although DNS operators have such plans, they have been reluctant to run a live test, because of the potential to disrupt DNS service, especially at the root server level. However, the events of September 11, 2001 are causing a reevaluation of the historical position on live testing and, among other things, the use of outside auditors to conduct DNS testing is now being evaluated. One other important outcome of this meeting was to identify a number of security related areas within the scope of ICANN's mission where improvement is needed. These are highlighted below:

- **ICANN should consider procedures to authenticate communications, especially in emergencies.**
- **ICANN should mirror its IANA server to provide a live backup.**
- **ICANN should provide better contact information for protocol delegation.**
- **ICANN should test its crisis procedures.**
- **ICANN should conduct a detailed and public threat analysis.**

➔ **ICANN should base data backup and escrow procedures on an analysis of how long critical services can afford to be down.**

In addition to the work of ICANN with respect to DNS security, the IETF is engaged in an effort to create a best practices standard for root server security, RFC #2870, Root Name Server Operational Requirements[10].

In the remainder of this section we review first, three specific types of attack that have been successfully perpetrated on DNS and then some selected, specific instances of attack on well known corporate networks,

7.1 DNS Vulnerabilities: Spoofing of DNS Responses

DNS has a mechanism, that is enabled in most DNS servers on the Internet, by which a DNS server can perform a query to an upstream DNS server (upstream, that is, in the sense of the inverted tree structure shown in Figure 3.2 above) in order to resolve a URL. This process by which this is accomplished is called a recursive query. In such situations, it is possible that a DNS conducting such a query could end up making a series of sequential queries to upstream DNS's before it obtains a fully resolved URL. In a DNS where this recursive query feature is disabled, and under the scenario where such DNS cannot resolve a given URL, the DNS then simply forwards the query (a single UDP packet) to the next DNS server higher up in the inverted tree of servers and has no further involvement in the URL resolution process for that particular request. This mechanism is called DNS forwarding.

DNS's that perform recursive queries are vulnerable to a DNS response spoofing type of attack[11]. In this attack, the DNS server is fooled into thinking that it is receiving a response from a trusted DNS server when, in fact, it is being "spoofed". The spoofing server issues a command to change the IP address associated with a particular URL to an IP address of its own choosing. The spoofing process can be achieved in networks that use older versions of BIND by means of guessing what the next DNS response sequence number will be and sending a reply with the guessed sequence number to a DNS that has just sent out a recursive DNS query to a legitimate DNS server. Once the spoofed DNS server has the incorrect Domain Name⇔IP address mapping, then the attacker can, if it so desires, fake the operation of the www whose Domain Name has just been hijacked. This vulnerability has been corrected in later versions of BIND in which BIND utilizes a random number generator to produce the aforementioned sequence numbers, thus making it extremely difficult for the attacker to predict the next sequence number.

7.2 DNS Vulnerabilities: Cache Poisoning

As mentioned earlier in this paper, a fundamental component of the DNS architecture is the ability of DNS to cache responses to queries in order to improve the performance (throughput and delay) associated with the DNS service. The downside of this is that if the cache gets corrupted with malicious, but otherwise well-formed data (e.g. a spoofed URL⇔IP address mapping) then the compromised cache will continue to be used by the unknowing DNS server until the offending cache entries are flushed. Thus if a DNS server is spoofed, for example, by the means described in section 7.1 above, and the Time To Live (TTL) parameter associated with that information is set to an artificially high number by the attacker, then that compromised information will remain in the cache, waiting to be served, whenever the now compromised DNS responds to a query for the URL associated with that compromised data.

7.3 DNS Vulnerabilities: Email Spoofing

This exploit involves spoofing a trusted source email address in formal email correspondence with, for example, ICANN itself[12]. The attack is accomplished by sending the registrar a request via an e-mail message containing the spoofed return address (i.e. the trusted source) to update the URL⇔IP address mapping for a particular URL under the authority of the spoofed source. The attack is strengthened if the attacker is able to hijack the email response to the spoofed source from the registrar, so that it never reaches the actual source (i.e. the victim organization). This hijacking of the email response can be achieved indirectly by flooding the legitimate recipient's email (inbox) so that the confirming email from the registrar never gets through. That way the source will not even be aware that this bogus transaction took place, and they will only discover the attack when, for example, the number of visits to their, say e-commerce www site suddenly go to zero! This attack often works because such administrative email requests are often verified only by inspecting the return email address. New procedures are already in place with many domain registrars (e.g. Verisign/Network Solutions) that provide much stronger authentication methods than an easily spoofed email address[13]

7.4 DNS Vulnerabilities: Exploit of Known Security Related Software Faults

As with many complex software applications, and especially ones that deal with variable length inputs (e.g. character strings), BIND has had its share of latent software defects, some of whose effects were buffer overflows as a result improper processing of such variable length inputs. Buffer overflow attacks are well documented in many on-line applications and in the case of earlier versions of BIND, these failures could then result in an attacker gaining root user access to the underlying server running the attacked DNS. With such root access, the attacker can then do pretty much as they please with respect to the functioning of DNS, because the can then take on the identity of a highly trusted network application. Once a DNS has been hijacked by this means, it is then possible to induce other DNS's (say the secondary DNS) which are authoritative for the same domain as the hijacked DNS server, to provide a complete copy of their domain URL⇔IP address mapping table. By this means, the attacker can then easily find out the IP addresses of the client and server computers that are on the attacked network. This type of attack can be mitigated by using a split-DNS system as described below in section 9.3 below.

7.5 DNS Vulnerabilities: Improper DNS Configuration

Two surveys from early 2001, the first, of 978 www sites in the Fortune 1000, and a second, of 5000 random sites in the .com domain suggest that 25% and 38% of the respective sites had DNS configurations that were either incorrect or weak from a security perspective[14]. Of course, improper configuration, resulting in insecure applications is a problem that is not limited to DNS, but, the effects of the attacks that can exploit these insecurities can be significant as I have already described in this paper. One of the recommendations for securing DNS is covered in section 9.6 below.

7.6 DNS Vulnerabilities: High Profile and Successful Attacks

In his paper, "Has Your Domain Been Hijacked Lately?"[15] , Michael Patrick, discusses a number of high profile exploits of commercial www sites where security weaknesses in DNS were utilized as the "gateway" to effect the compromise of those sites. One of those cases, the one pertaining to the Microsoft www sites, and reviewed in section 7.6.1 below is explained in greater detail in "DNS Vulnerabilities – Nine Days in the Spotlight", by Cheryl Culpepper Olusada[16] as well as in the paper, "Recent Developments and Emerging Defenses to D/DOS: The Microsoft Attacks and Distributed Network Security, by Jay L. Koh[17]. These exploits got press coverage, no doubt because they represent high value, and therefore, highly visible www sites.

7.6.1 Microsoft DDOS Attack

This attack came about one day after Microsoft's www sites were isolated from the Internet for most of the day due to an improper configuration change in the two routers on the edge of Microsoft's network. This change caused the Microsoft DNS's (both primary and secondary!) to become isolated and therefore unable to serve the IP addresses for Microsoft's www sites in response to DNS queries for those sites. The cause of this outage was that both DNS's were connected by the same routing path, that is, the aforementioned two routers. So once these routers were mis-configured, the Microsoft DNS's and, consequently, their www sites became unreachable.

This problem was very serious and it resulted in a 24 hour outage of Microsoft's www services as well as Hotmail which itself served millions of customers. To make matters worse, because of the visibility of the outage, it set the stage for the subsequent Distributed Denial of Service (DDOS) attack the next day on the very same routers that had mis-configured the day before. The DNS outage clearly signaled to the attackers that Microsoft had a single point of failure in its routed network and, using information gained from Microsoft's domain registration records, the attackers were able to flood those two routers at the edge of Microsoft's network, thus effecting a successful DOS attack[18] and effectively shutting down Microsoft's www sites, this time for several hours.

7.6.2 Nike WWW Site Hijacking

The Nike www site was hijacked on June 21, 2000 when an attacker caused www traffic destined for Nike's www site to be diverted to another IP address of a www hosting company in Scotland, whose servers then went offline due to traffic overload. This attack lasted for between six and twenty-four hours and was implemented by spoofing an email to Network Solutions, directing the change of IP address for Nike's www site. Despite claims that were made at the time by Network Solutions that changes to Nike's www site could only be made via an encrypted and password-protected channel, the attackers were able to bypass this means of protection and thereby trick Network Solutions into making the request changes.

7.6.3 Adobe WWW Site Hijacking

The Adobe www site was hijacked in a similar way to the aforementioned hijacking of Nike's www site. Through bogus correspondence with Network Solutions, the attacker caused the domain record for adobe.com to be transferred to Paycenter, and ICANN-accredited domain registrar in China. In the same correspondence, the name-servers for the www site were modified.

7.6.4 RSA WWW Site Hijacking

It is considered a badge of honor within the hacker community when a www site associated with a security systems or service provider is successfully attacked. In this case, it was the site of RSA Security, whose customer's were diverted to a spoofed www site. In this case the hijacking involved the spoofing of a DNS outside the domain of authority of RSA, so that RSA had no direct control over the occurrence of the attack. One remedy for this kind of attack is to watch for obvious (your www URL takes you to a defaced www site) as well as more subtle (www page requests for your servers suddenly decrease in quantity) changes in the behavior of your system.

8 DNS Client Side Vulnerabilities

So far this paper has focused on the vulnerabilities and exploits associated with the server component of the DNS service. Recall however from the simple system availability model shown in **Figure 3.1** above that for the complete service to be available then the client side of the DNS service must be operating as well. The client side components of DNS reside in the protocol stack and the www browser of the PC which is requesting the services of DNS. Thus, any malware exploit on a PC that allows an attacker to run "code of their choice" will make these client side components vulnerable to exploit as well. A common exploit of this type is one where the attacker redirects the settings for the IP address of the DNS server or servers (primary and secondary) that are maintained for each Network Interface Adapter (NIC), generally by the TCP/IP protocol stack. The net effect of this kind of exploit will be that domain names will be resolved to the IP address of the attacker's choosing. Below we take a look at one such exploit which is "in the wild" as this paper "goes to press".

8.1 Trojan.Qhosts Vulnerability and Exploit

This particular virus is a non-self-replicating Trojan exploit of the Microsoft Windows operating system (Windows 2000, Windows 95, Windows 98, Windows Me, Windows NT, Windows Server 2003, and Windows XP)[19]. The exploit modifies the Windows Registry settings associated with the DNS server so that such settings will no longer point to the proper DNS server for that client but to the IP address specified by the attacker. It also modifies a file called the "hosts" file to insert the domain names of about 20 well known search www sites (e.g. www.google.com) and associate those URL's with the aforementioned attacker IP address.

8.1.1 Method of Exploit

A popup ad at http://www.fortunecity.com/ /fc728x90smartad. is known to load a remote site containing this trojan. This trojan relies on an Microsoft Internet Explorer vulnerability to get installed on the local system. Once installed, the trojan redirects Domain Name requests to a specified address.

8.1.2 Exploit Severity

Symantec's www site classifies the damage caused by the viral payload as "Degrades Performance". In general, the severity of such an attack on the client side DNS functionality can vary from a frustrating annoyance to something much more serious if the attacker's DNS were designed to subsequently spoof legitimate www pages such as, say, www.citibank.com, www.firstusa.com, or some other financial services www site in order to harvest account information. As a consequence any such malware must be treated as a potentially significant threat.

8.1.3 Exploit Mitigation

As of October 2, 2003 both of the major providers of PC A/V software, Mcafee, and Symantec have released A/V signature file updates to contain this exploit. Furthermore, Microsoft has released on October 3, 2002 a cumulative patch to the Microsoft Internet Explorer software that purports to eliminate the vulnerability that is exploited by this particular Trojan[20]

9 DNS Server Hardening against Attacks

As is almost always the case, there are lessons to be learned from each instance of successful penetration of a network or server by an attacker. And sometimes the lesson learned, is to quickly implement the lessons learned from a prior attack (see section 9.7 below for a graphical explanation of this simple concept). The eleven recommendations explained below encapsulate many years of expert opinion on the most effective ways to harden a DNS, whether for a public network service provider such as WorldCom or Sprint, or at the edge of a corporate network such as Microsoft or Nike. Recommendations 9.1 - 9.9, and 9.11 are taken from the paper, "DNS Security Considerations and the Alternatives to BIND", by Lim Seng Chor[21], while recommendation 9.10 is taken from the text, "The Concise Guide to DNS and BIND"[22].

9.1 Subnet Diversity

This recommendation which was learned painfully by Microsoft with the outage of their www sites that was described earlier in this paper, says that your DNS's (primary and secondary) should always be configured to operate on separate sub-networks within your network and each sub-network must have a separate route to the Internet. The irony for Microsoft is that, as a company at least, they already had learned this lesson because it was contained in their own user documentation for their products. In an article[23] published at Zdnet's www site, Robert Lewis refers to a survey made subsequent to the Microsoft attack which found that 38 percent of the companies in the.com domain have the same DNS design flaw, that is, both DNS's on the same subnet and/or connection to the Internet.

9.2 OS Platform Diversity

This recommendation says that the primary and secondary DNS should be implemented on an operating system, each one, different from the other. So, if, for example the primary DNS is implemented on Sun Solaris, then the domain authority should implement the backup DNS on Open BSD. Doing this will generally make it more difficult for an attacker to take down both DNS's by means of the same attack, say, on a specific known buffer overflow condition in one of the two operating systems.

9.3 Separate Public DNS from Internal DNS (Split-Horizon Operation)

As is often the case, a network will provide DNS to the Internet for its domain as well as providing DNS to the internal network of the company. This recommendation says that the Internet facing (public) DNS functions should be placed on a server that is on the perimeter network (sometimes referred to as the DMZ), while the internal facing (private) DNS functions should be placed on a separate server that is inside of your firewall (or inner firewall if you are operating a DMZ with a second, outer firewall). That way, if the public DNS is compromised, it will not be possible for the perpetrators to discover the URL's and IP addresses associated with the internal network. This mode of operation is referred to as "Split-Horizon DNS".

9.4 Don't Share DNS server with other Applications

This paper has already outlined the damage that can be caused by an attacker that exploits a software flaw in the DNS software. If there are other applications running at the same time on the server that is providing DNS, then, there is the possibility, however, remote, that that other software can be compromised, thereby allowing the attacker to gain control of the hardware and OS of the server that is running DNS. This recommendation mandates that no other software application should be hosted on the same server as one that is hosting DNS.

It should be noted that this practice is at odds with certain bundled operating system products such as Microsoft Windows Small Business Server, which bundles a complete server operating system, a WWW server, and Exchange email server, and, optionally, an Structured Query Language (SQL) data base server all on one machine along with Microsoft Active Directory services. Of course it would be possible to turn those extra services off, and run the DNS on a separate server, but the whole purpose of this operating system bundle is to reduce overall cost for the user. It should also be noted that such a bundled product will violate the hardening technique referenced below in section 9.9 "DNS Functional Splitting", for the very reason that, in such bundled systems, there is only one DNS server unless the enterprises chooses to implement stand alone servers at a higher up front cost for, at a minimum, the extra hardware for a separate computer.

9.5 Restrict Zone Transfers

This paper has already described an attack scenario where a compromised DNS requests a zone transfer from its domain peer (primary or secondary). By restricting (i.e. accepting) zone transfers only from authorized name servers, the likelihood of success of this kind of attack can be minimized.

9.6 Configuration Hardening

As mentioned earlier in this paper, BIND is a relatively complex software application that runs on UNIX, which is, itself, a complex, though well understood operating system. By carefully following prudent guidelines for the initial configuration of a DNS, it is possible to minimize the chance of compromise of the system once it is put into operation. Rob Thomas, maintains a www site which provides a template for securing BIND, (www.cymru.com/~robt/Docs/Articles/secure-bind-template.html)[24]. A corollary recommendation to this one is to employ a surveillance application such as Tripwire and schedule it to run every day in order to verify the integrity of the DNS binaries, configuration files(s), zone data and other important files stored on the DNS server. For users of BIND version 8.1.x, Psionic Technologies (www/psionic.com) provides a guide for securing that release of BIND when operating in the OpenBSD/FreeBSD environment[25].

9.7 Release Currency

As will all network accessible software systems, it is critical to maintain currency with the latest release or patch update for the software that is providing DNS. This is oftentimes easier said than done, considering the variety of such network accessible systems that must be regularly updated and the common practice of many systems security managers and technical experts to leave the administrative work as a lower priority in an oftentimes overloaded schedule.

Figure 9.7

, below, however, clearly points out the risk of delaying action once an advisory pertaining to DNS concerning a security vulnerability and its countermeasures has been released[26]. As the bar graph shows, within 30 days of the CERT announcement, there was a dramatic rise in the number of reported incidents pertaining to the DNS vulnerability that was communicated in the associated CERT Advisory. The number of incidents tapered off approximately 8 months after the date of the original announcement. It is impossible to judge, from this data, what higher priorities prevented the affected system operators from acting more quickly to implement the countermeasures indicated in the Cert Advisory but the risks of inaction seem clear.

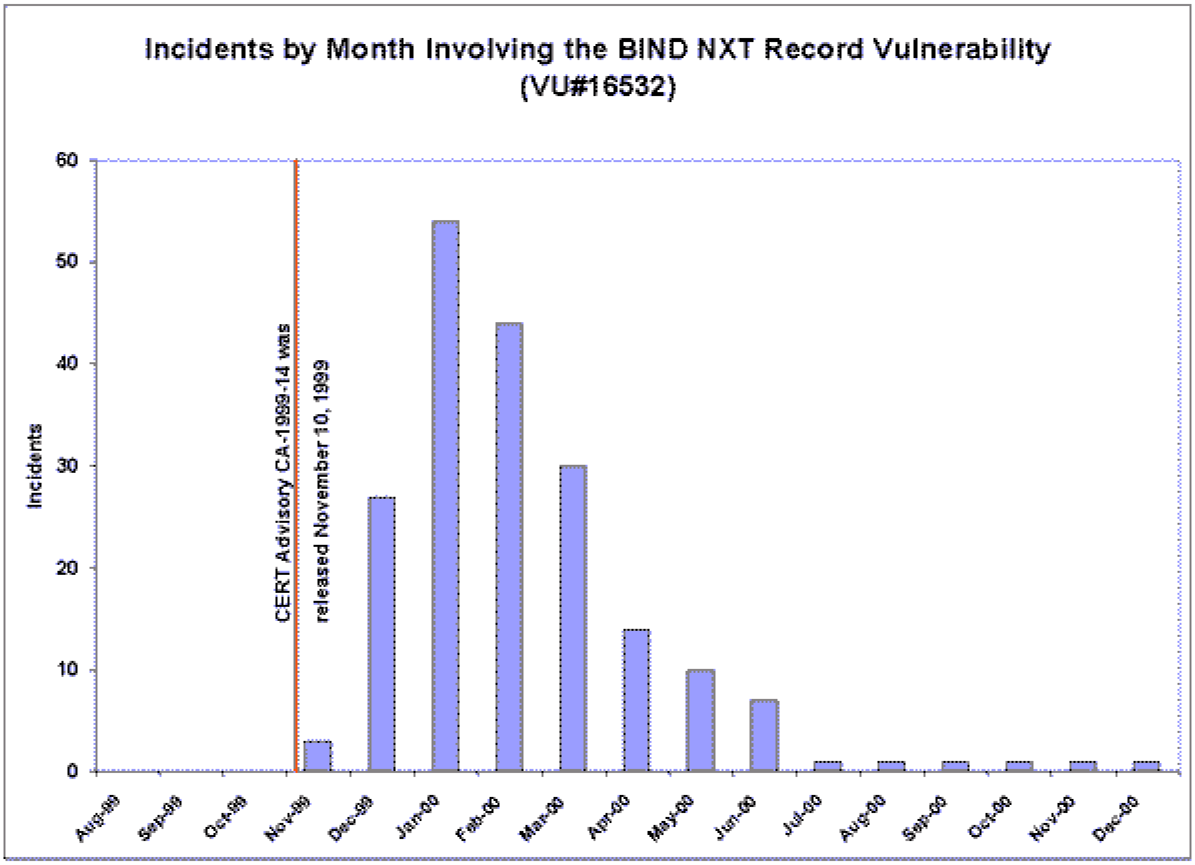


Figure 9.7

9.8 DNS Isolation

This recommendation mandates that the DNS application be isolated (the more technical phrase is to run DNS in a “chroot jail”) and to always run it as a non-root user. Doing this will further protect a server, whose DNS application has been compromised, against the attacker subsequently gaining root privileges to the underlying OS and server hardware.

9.9 DNS Functional Splitting

In addition to Split-Horizon DNS is possible to separate DNS functions into Advertising Name Server and Resolving Name Server functions. By running these functions on two separate servers, it is thereby possible to customize the behavior of each server and to implement more stringent purpose specific restrictions on each one.

9.10 DNS Version Number Hiding

This recommendation suggests that, in order to make the job of a potential intruder to the DNS more difficult, responses to queries which request the version number of the software should always be hidden from external access.[27]

9.11 DNS Application Diversity

The last recommendation is to consider using a version of DNS software for one of your two DNS's that has been developed by a different supplier from the supplier that developed the other version of DNS software that you are using. A number of such alternatives are presented in section 10 below.

10 DNS Client Hardening against Attacks

Most of the hardening that can be performed on client computers falls into the non-specific category of "safe computing" practices which are well documented at the www sites all of the anti-virus software providers and will not be repeated here. Some might argue that given the large number of vulnerabilities in Microsoft's Internet Explorer Browser, which are well documented at the Pivx Security www site[28], and, not all of which pertain to directly to DNS vulnerabilities, ,that the user is better off using a different browser, such as Netscape Navigator, when operating with a Microsoft OS. Certainly, given the overall functional overlap between these two software products, that is Netscape Navigator, and Microsoft Internet Explorer it would be a smart practice for the user to be familiar with both products, and to maintain an awareness of the latest exploits against one versus the other so that the one can be "fired up" when the other is in "high seas" as a result of a particular exploit for which the A/V signature file update is not yet available. Unfortunately, the Trojan.Qhosts

One specific protection that can be implemented at the firewall level, however, to stop the exploit from succeeding by blocking DNS queries to IP addresses other than those that are known "valid" DNS addresses for the network in question. That way the system administrator can be a) alerted to such invalid attempts, and b) begin remedial action on the system that has been compromised.

11 Alternatives to BIND

Although BIND is the dominant software product that is used on the Internet for DNS, there are a number of alternative systems, which have been fully or partially constructed and which provide DNS capabilities as well. These are discussed in some detail in a paper written by Lim Seng Chor[29] and will only be briefly mentioned here. URL's to the pertinent www sites are also provided below. Sam Trenholme, the author of one of the DNS products, MaraDNS, also maintains a www page with a brief summary of DNS implementations of which he is aware (http://www.maradns.org/dns_software.html)

11.1 Dents

Dents is an implementation of the server side of DNS and it was developed for higher performance and better server management. According to Chor, the design of Dents is very clean, which should contribute to its overall security. (http://www.dents.org)

11.2 Djbdns

Djbdns is a secure replacement for Bind in which security was a forethought as opposed to an afterthought in its design. It is structured as a collection of small, independent, and mutually distrusting programs, each of which runs in its own chrooted jail. Its author, Daniel J. Bernstein, has actually offered a monetary award to the first person to publicly verify a security weakness in the latest version of the system. (**http://cr.yp.to/djbdns.html**)

11.3 MaraDNS

MaraDNS by design goes after the problem of buffer overflow by using specially written software to perform string handling, which, presumably, performs the necessary limit checks that are not performed by the widely available standard library routines that software engineers often use to get the job done. It is designed to operate on both Linux and Unix systems as well (**http://www.maradns.org**).

Buffer overflow attacks are well documented and the text, "Building Secure Software", by John Viega and Gary McGraw devotes a full chapter to the root causes of buffer overflow vulnerabilities and the various programming techniques that can be utilized to eliminate these kinds of vulnerabilities. In these authors' opinion the true root cause of such buffer overflow attacks is the non-existent bounds checking on arrays and pointer references in both the C and C++ programming languages, which have been in use for many years[30].

11.4 CustomDNS

CustomDNS is a modular DNS server that is written in the Sun Java programming language and in Perl programming language. (**http://customdns.sourceforge.net/**)

11.5 Ibnamed

Ibnamed is a load balancing DNS that is written in the Perl programming language By Roland Schemers whose source code is available for download. .(**http://www.stanford.edu/~schemers/docs/lbnamed/lbnamed.html/**)

11.6 Ibdns

Ibdns is a load balancing DNS which is similar to Ibnamed. (**http://cr.yp.to/djbdns.html**)

11.7 Microsoft DNS

Of course the Microsoft Server operating systems including Windows NT Server, 2000 Server, and Server 2003 contain an implementation of DNS and, in addition, since the release of Windows 2000 Server, and the introduction of Microsoft's Active Directory Service, it is possible to integrate the Zone files of DNS into the Active Directory. This form of operation is referred to as Active Directory Integrated DNS operation and is explained further along with its benefits in the following Microsoft document, (www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windowsserver2003/proddocs/server/sag_dns_und_activedirintegration.asp)

12 The Future Evolution of DNS

Given the overwhelming "market share" of BIND in the extant DNS's of the Internet, it does not appear that there will be a near term migration to any other software platform, despite the efforts of others as documented in section 10 above to develop viable alternative implementations for DNS. This speaks to both the durability of the DNS architecture as specified in the relevant IETF RFC's and the quality of the implementation of BIND, despite the exploits carried out to date against that system.

The most recent release of BIND from ISI is version 9.2.2 and Section 6 of this paper highlights the features contained in that release. In addition to supporting the next generation IP protocol, IPV6, and support for multiprocessor configurations, which will facilitate the implementation of higher performance implementations of BIND using commercial multiprocessor systems, BIND 9.2.2 provides important improvements in the security of the application. These improvements are discussed briefly in the remainder of this section. It is anticipated that the security related improvements, once fully propagated throughout the DNS hierarchy of the Internet, will reduce the exposure of DNS to DNS spoofing attacks. Of course, this transition will take some time (perhaps a few years) unless some high profile successful attack occurs on DNS that could have been prevented with the features available in BIND 9.2.2 and above.

12.1 DNSSEC Support

The IETF has published an RFC, # 2065, "Domain Name System Security Extensions"[31], which describes the extensions to DNS that provide integrity and authentication to security aware resolvers and applications through the use of cryptographic digital signatures. The extensions also provide for the storage of authenticated public keys that is necessary in order to allow security aware resolvers to learn the authenticating key of zones that are present in addition to those zones for which such resolvers are initially configured. The extensions described in the RFC also provide for the optional authentication of DNS protocol transactions and requests. DNSSEC provides for the digital signature of responses to DNS queries, which will eliminate the problem of DNS spoofing and the consequent potential for cache poisoning (unless the DNS is compromised by some other software failure). DNSSEC does not, however, address the question of bogus queries, which would typically come from an attacker who is mounting a DOS/DDOS attack on a particular DNS[32].

12.2 TSIG Security Improvements

TSIG (Transaction Signature) is the means by which BIND signs transactions that it issues. In BIND 8.X.X TSIG is supported for update requests from one DNS to another. TSIG uses a mechanism called HMAC-MD5 to authenticate the sender and message content of each update request. HMAC-MD5 is a symmetric key encryption algorithm, and, therefore, requires that both the sender and the recipient have the same key which must be kept secret.

In BIND 9.x, TSIG support is added for queries, the NOTIFY protocol and for zone transfers.

13 Conclusions

This paper has examined a key service associated with the Internet, the Domain Name Service, which must be continuously available in order to allow efficient navigation across the Internet for www browsing and email services. DNS service has been available on the Internet since the early 1980's but by the late 1990's as www growth exploded, DNS became a target of opportunity for malicious hackers. The security challenges that arise with respect to DNS are relatively well understood and reasonable countermeasures have been made available by various organizations to deal with most of these challenges. New versions of DNS software are under different stages of development by various groups and organizations, but it is anticipated that the dominant version, called BIND, will continue to be used for the foreseeable future for the vast majority of the DNS servers extant in the Internet. It is therefore important that BIND be continually improved, in particular, with respect to security and resiliency features. This imperative has led the current custodian of BIND (which is after all an open source product), ISI, to undertake a complete rewrite of BIND and to incorporate a number of security related improvements previously established by the IETF and documented in RFC #2535. As with most network accessible applications, prudent system administration is essential in order to provide the most effective protection against cyber attacks against DNS.

References

[1] Nicolai Langfeldt, The Concise Guide to DNS and BIND, (Indianapolis: QUE, 2001)

[2] Cheryl Culpepper Olusada, DNS Vulnerabilities – Nine Days in the Spotlight, February 15, 2001, (**http://www.giac.org/practical/gsec/Cheryl_Olusada_GSEC.pdf**)

[3] Nicolai Langfeldt, Ibid, p10.

[4] Katie Hafner, Matthew Lyon, Where Wizards Stay Up Late: The Origins of the Internet, New York, Simon & Schuster, 1998

[5] P. Mockapetris, Domain Names – Concepts and Facilities, IETF RFC 1034, November 1987, (**www.ietf.org/rfc/rfc1034.txt?number=1034)**

[6] P. Mockapetris, Domain Names – Implementation and Specification, IETF RFC 1035, November 1987, (**www.ietf.org/rfc/rfc1035.txt?number=1035)**

[7] Cheryl Culpepper Olusada, Ibid

[8] Ron Baklarz, In Yet Another BIND, February 20, 2001, (**http://www.giac.org/practical/gsec/Ron_Baklarz_GSEC.pdf**)

[9] James Sweetman, Current Issues in DNS Security: ICANN's November 2001 Annual Meeting, November 28, 2001, (http://www.sans.org/rr/paper.php?id=568)

[10] R. Bush et al, Root Name Server Operational Requirements, IETF RFC 2870, June 2000, (www.ietf.org/rfc/rfc2870.txt?number=2870)

[11] Sinéad Hanley, DNS Overview with a discussion of DNS Spoofing, November 6, 2000, (http://www.ogobin.org/internet/%5BPaper%5D%20dns_spoofing.pdf)

[12] Michael Patrick, Has Your Domain Been Hijacked Lately?, February 15, 2001, (http://www.giac.org/practical/gsec/Michael_Patrick_GSEC.pdf)

[13] Kurt Seifried, DNS Security, email correspondence, (**http://archives.neohapsis.com/archives/bugtraq/1999-q4/0545.html**)

[14] Cheryl Culpepper Olusada, Ibid

[15] Michael Patrick, Ibid

[16] Cheryl Culpepper Olusada, Ibid

[17] Jay L. Koh, Recent Developments and Emerging Defenses to D/DoS: The Microsoft Attacks and Distributed Network Security, February 9, 2001, (http://www.giac.org/practical/gsec/Jay_Koh_GSEC.pdf)

[18] Cheryl Culpepper Olusada, Ibid

[19] Symantec www site explanation of Trojan.qhosts Virus (http://securityresponse.symantec.com/avcenter/venc/data/trojan.qhosts.html)

[20] Microsoft Security Bulletin MS03-040 (http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-040.asp)

[21] Lim Seng Chor, DNS Security Considerations and the Alternatives to Bind, October 2, 2001, (http://www.sans.org/rr/paper.php?id=567)

[22] Nicolai Langfeldt, Ibid

[23] Robert Lewis, Too many holes in the Net, January 26, 2001, (www.zdnet.com/filters/printerfriendly/0-,6061,2679081-2,00.html)

[24] Rob Thomas, Secure Bind Template Verison 3.2, 16 Nov 2001, (http://www.cymru.com/Documents/secure-bind-template.html)

[25] Psionic Technologies, Securing DNS (OpenBSD/FreeBSD Version), (http://www.psionic.com/papders/bindbsd.html)

[26] I.E. (Jon) Naumann, DNS Attacks: An Example of Due Diligence (http://www.giac.org/practical/gsec/Jon_Naumann_GSEC.pdf)

[27] Nicolai Langfeldt, Ibid

[28] http://www.pivx.com/larholm/unpatched/

[29] Lim Seng Chor,, Ibid

[30] John Viega and Gary Mcgraw, “Building Secure Software, Addison-Wesley, 2002

[31] D. Eastlake, Domain Name System Security Extensions, IETF RFC 2535, March 2002, (www.ietf.org/rfc/rfc2535.txt?number=2535)

[32] Christopher Irving, The Achilles Heal of DNS**,** August 2, 2001, (http://www.sans.org/rr/paper.php?id=565)