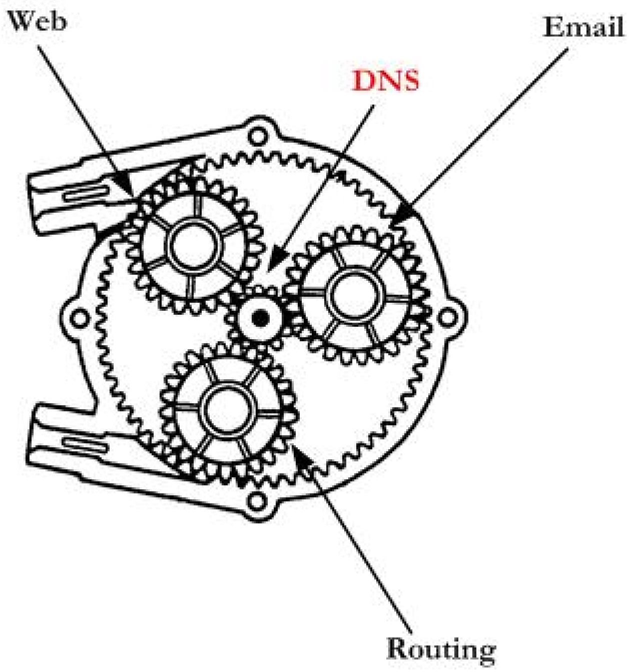
Bound by Tradition

A Sampling of the Security Posture of the Internet's DNS Servers

Mike Schiffman <mike@infonexus.com>
February 2003



DNS servers across the Internet running BIND are not up to date with security patches and software updates. As a result, a significant fraction of the Internet's DNS servers is vulnerable to compromise, subversion, denial of service, and general misuse. Considering that DNS is the lynchpin of the corporate enterprise, the impact of these vulnerabilities is significant and a successful attack could bring down any online business.

Abstract

This Research Report presents an overview of the current security posture of DNS servers found across the Internet. The report also covers the following:

- A summary of some of the finer points of the DNS protocol
- A discussion of why DNS is such a key component in the infrastructure of the Internet
- A summary of the BIND software, the most widely used DNS implementation available
- A presentation of empirical data that underscores the past and present state of security in BIND servers, including correlating the meteoric increase in size of the code-base with the number of publicly-reported vulnerabilities

Introduction

The Domain Name System (DNS) is the protocol that makes up the Internet's distributed name and address database. Originally implemented to make the Internet user-friendly, DNS quickly became the lynchpin in the intricate engine under the hood of the Internet. To understand why, we must first understand a bit about how the Internet is put together at the network level.

Every computer connected to the Internet has a numerical locator that uniquely identifies it, called an IP address [1]. Before an Internet user can connect to a website, he must know this IP address. DNS allows computer users to employ easy to remember mnemonics (called domain names) to reach Internet resources such as “www.cnn.com” rather than requiring them to memorize numerical IP addresses such as “64.236.16.52”. As such, DNS can be thought of as the Internet’s phonebook.

Every Internet user across the world consistently makes use of the DNS protocol whether they know it (by explicitly looking up an IP address using the “dig” program) or not (by visiting websites on the Internet or sending e-mail). The goal of DNS is for any Internet user any place in the world to reach a specific IP address by entering its domain name.

There are two core parts to a DNS implementation: a server and a client. The client, also called a “resolver,” is often built into many of the programs and libraries people use everyday such as web browsers and e-mail programs. The resolver talks to a DNS server on behalf of a program, automatically translating domain names into IP addresses. The DNS server is responsible for maintaining a portion of the name to address database and serves requests to resolver clients.

Critical Infrastructure

DNS is critical infrastructure in the Internet’s complex framework. Without it, everyday Internet use would be arduously difficult, if not effectively impossible. The US Government acknowledges this when it urges near-term federal dollars to be spent on DNS-related security research and development [2]. To appreciate why this is the case, we can reflect on two examples of what happens when someone tampers with DNS, initially on a small scale and then on a larger scale.

First consider what would happen if a major software vendor’s DNS servers were taken out of commission by a malicious attack. Internet users would be unable to resolve any of the vendor’s website domain names into IP addresses and therefore, the company’s online presence would effectively cease to exist. No one would be able to browse to its website, make an online purchase, or even email technical support to complain that they cannot reach the website! This scenario is exactly what happened to Microsoft in January of 2001 when its DNS servers were taken offline for more than 24 hours. The result was that all of microsoft.com was unavailable to the rest of the Internet for the duration of the outage [3].

Now consider what would happen if DNS were disrupted on a global scale. The majority of Internet connection attempts would fail in what would effectively be a terribly efficient and complete Internet-wide denial of service attack. To understand how this is possible, we need to understand a bit more about how DNS works. The DNS database can be thought of as a giant, inverted, hierarchical tree. At the top of this tree, are 13 roots (the so-called root name servers) [4]. These 13 root name servers contain delegation information for all of the top-level domains (com, net, org, edu, gov, mil and so on). What this means is that these 13 servers contain information on how to find out which DNS server is authoritative for nearly every domain on the Internet. Anytime a DNS server needs to look up an IP address it doesn't have cached (on behalf of a resolver) it will consult one of the 13 root name servers. Effectively, these servers contain the keys to the Internet DNS castle.

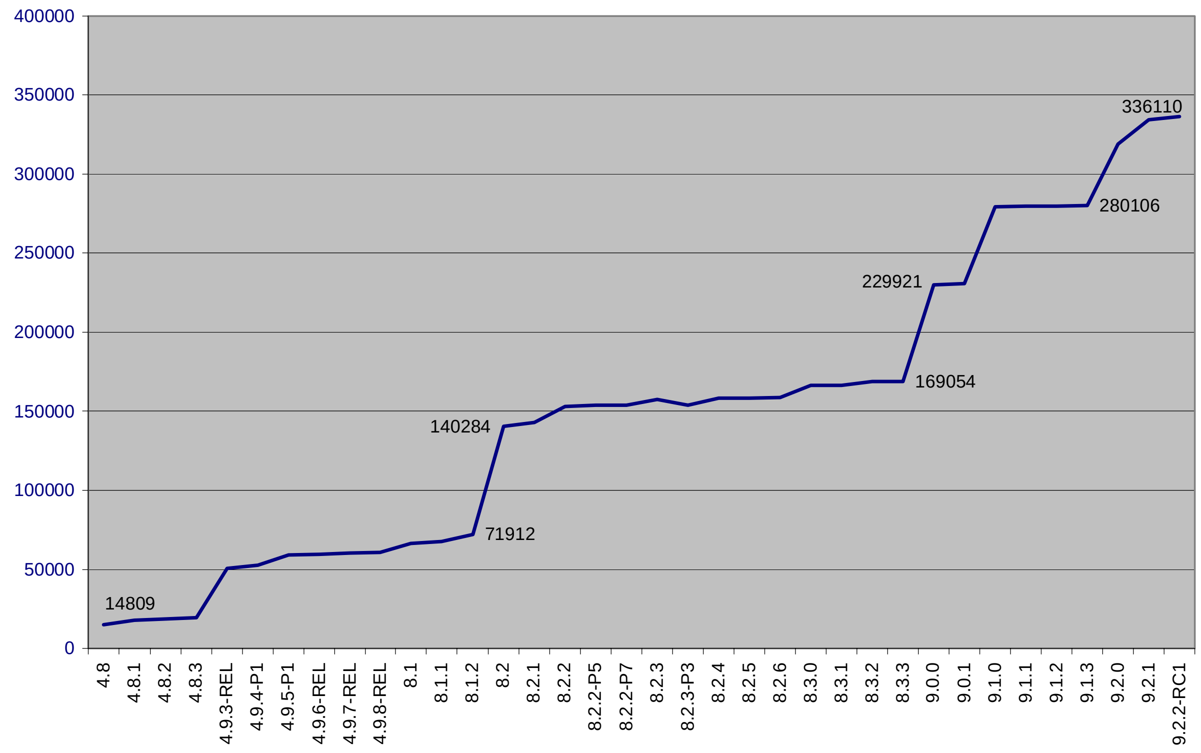
If all of these 13 root name servers were disrupted or otherwise compromised, DNS on a global scale would soon not function. This would in effect grind the great wheels of the Internet to an absolute halt. In October of 2002, a distributed denial of service (DDoS) attack was launched against all of the root name servers, and managed to disable nine of them for a short time [5].

Bound by tradition

BIND (Berkeley Internet Name Domain) is the largest and most widely-implemented DNS package available today. Available for the past fourteen years, BIND is by far the most mature open-source implementation of the DNS protocol and is used almost universally by every enterprise deploying DNS servers.

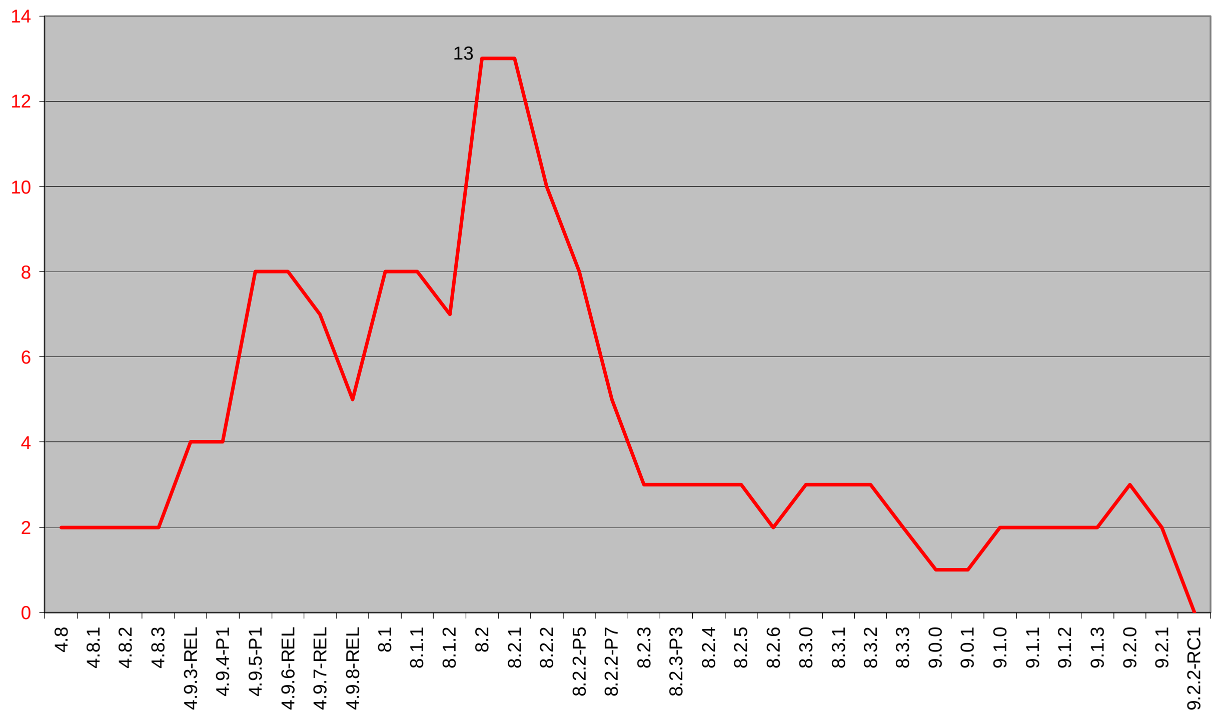
Like most software applications, BIND has grown in size as it has matured. Since the first public release of BIND in 1988, its code-base has grown from 14,809 lines of C code to 336,100 lines, an increase of 2200%. Chart 1 shows the size trend of lines of source to BIND software version [6].

Chart 1: BIND lines of source per version
2003.01.05



Large software applications are inherently complex. Complexity, from a computer security perspective, is often detrimental. As software becomes more sophisticated, the number of potential security breaches grows. Research has shown the number of errors in computer code is proportional to the square of the lines of code of the program. Many of these errors are potential security leaks [7]. Additionally, the more complicated a system is, the harder it becomes to comprehensively audit and ultimately guarantee its security. As a large and complex application, BIND has a long history of security issues. Below, Chart 2 summarizes the number of publicly known and acknowledged security vulnerabilities BIND has suffered

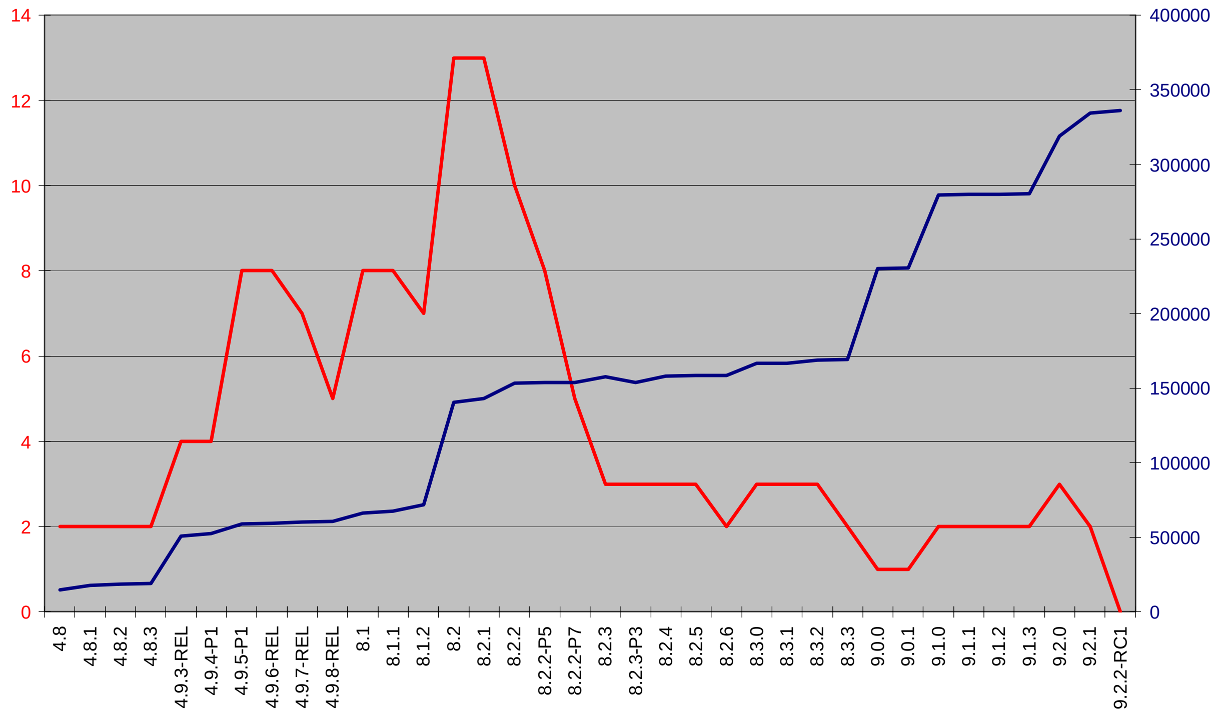
Chart 2: BIND vulnerabilities per version
2003.01.05



from over the past fourteen years [8]. Note that this chart only takes into account vulnerabilities that stem from programming errors. It does not consider configuration or setup errors.

Interesting to note is the fact that the BIND codebase almost doubled in size from version 8.1.2, which

Chart 3: BIND lines of source per version with vulnerabilities
2003.01.05



comprised 71,912 lines of code to version 8.2 with 140,284 lines. BIND version 8.2 peaked with the most security vulnerabilities at 13. The trend of complexity as a function of code size to vulnerabilities can be seen in Chart 3.

To further highlight the rift between complexity and security, consider that as this paper was being written, three more serious BIND vulnerabilities were publicly announced [9].

What's the worst that can happen?

Vulnerabilities due to programming errors come in all shapes and sizes. Over the years, the eighteen publicly disclosed bugs seen in BIND fall in three categories:

- Remote code execution (nine instances)
- Denial of service (eight instances)
- Information disclosure (one instance)

By far the most dangerous of these are those that enable remote code execution to take place; half of the eighteen bugs are of this type. A successful exploit of a bug in this category will result in arbitrary code being executed within the context of the DNS server, which is frequently run as a privileged user. This can give the attacker complete control over the machine and its data and often enables additional access to adjacent machines. Furthermore consider that a successful compromise of an enterprise's DNS sever not only affects that system, but allows an attacker to modify zone file information and redirect users away from the enterprise's servers to systems of his or her choosing.

With all of this in mind, as of January 2003, the SANS (SysAdmin, Audit Network, Security) Institute had BIND vulnerabilities listed among the top ten most critical Unix-related Internet security risks[10].

From chaos comes order

The initial aim of this report was to query the DNS server version of every host across the Internet running a DNS server. Correlating this information with the vulnerability information in Chart 2 could then provide us with an evenhanded assessment of the security posture of the Internet's DNS servers. However, given the IP address space of almost 4.3 billion (less the 17.6 or so million RFC 1918 internal hosts) in conjunction with the unreliable nature of UDP over IP, sending a 58 byte UDP datagram and waiting for a variable sized response was considered to be an intractable solution to complete in a reasonable time period. An alternative plan was devised using current zone files obtained from the InterNIC.

In order to obtain the DNS server version, a special type of DNS query message is used; the "Chaos" class. Programs that want to request the software version from a given DNS server construct and send these Chaos class queries [11].

Custom code was written on top of libnet and libpcap [12] to rapidly execute Chaos class queries and extract and collate the responses. In order to combat the aforementioned network unreliability issues, up to three probes were sent for each unresponsive server. The scanning took place between January 1 2003 and January 5 2003. Initially, all unresponsive and non-resolving DNS severs were rescanned in up to four subsequent passes (again of up to three probes per server) in order to account for possible transient network issues.

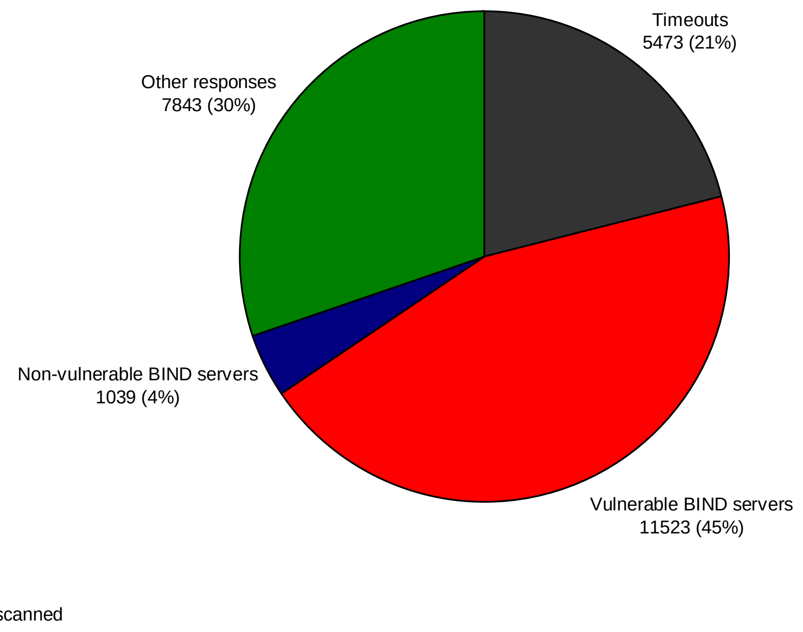
It is important to note that it is actually trivial for a DNS hostmaster to specify an arbitrary response string for incoming Chaos class queries. In practice, this adds noise to our empirical data. Proof that this happens is found with the numerous "humorous" and extraneous responses [13]. While it is likely many of these servers are in fact running BIND, we cannot be sure which version, and therefore we cannot factor them into our analysis. The data used in this paper is mainly dependent on what appear to be legitimate BIND server responses, and it is assumed that the rate of incidence for a hostmaster to actually spoof a legitimate version of BIND is low; all BIND version responses that seem genuine are taken to be such.

All of the raw scan data (sans IP addresses) that was mined for the paper, including a complete listing of all of the responses from every server queried may be downloaded from http://www.packetfactory.net/papers/DNS. Please note that server IP address information can made available, but only to organizations that can demonstrate a specific need for the data.

Substantiation

As shown in Chart 4 below, a total of 25,878 servers were scanned and 20,405 (79 percent) returned some sort of a response. Of the responders 12,562 (49 percent) responded with what appeared to be valid BIND version information, and of these, 11,523 (45 percent) were running a version of BIND with one or more vulnerabilities. We note that the only versions of BIND that do not have any publicly disclosed vulnerabilities are versions 9.2.2, 9.0.1, 9.0.0 and 8.3.4. Unfortunately, these servers made up only four percent of those found across the Internet.

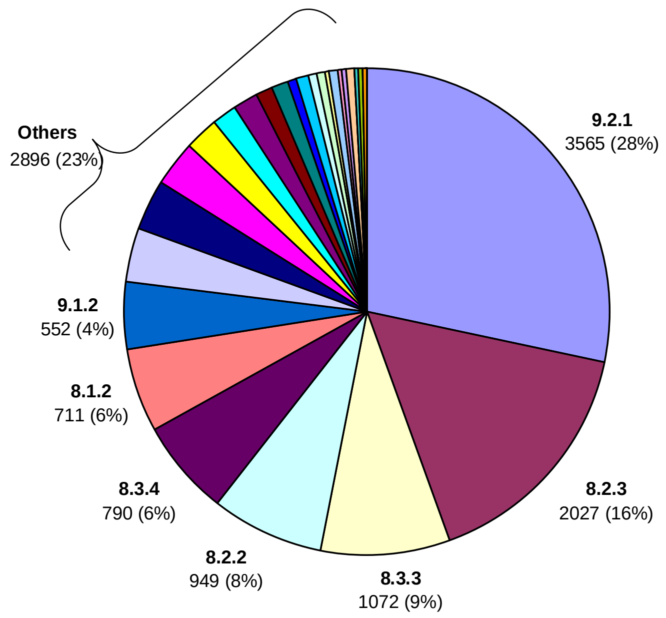
Chart 4: Overall Chaos class scanning results
2003.01.05



Three-thousand seven hundred four (14 percent) of the 7,843 servers that returned something other than a BIND version string responded with a reply code of 4 (not implemented) which is usually indicative of a Windows-based DNS server. The evil that may lie beneath is far beyond the scope of this paper. 204 (less than one percent) servers returned a reply code of 1 (format error) which is usually returned by Dan Bernstein's stalwart tinydns DNS server (more on this below). The rest of the 3,935 (15 percent) unknown responses were made up of random strings, threats, warnings, or otherwise witty replies that could have only been dreamt up in the mind of the DNS hostmaster.

To gain insight into the distribution of BIND servers across the Internet we can breakdown the most prevalent server versions, as shown in Chart 5 (in order to make this data a bit more manageable patch-level information was aggregated to the major and minor version numbers only). At the time of this writing, the most recent (and vulnerability free) BIND version was 9.2.2 released four and a half months prior to the scan. It didn't even make into the top ten list of servers found, which as it happens, had a combined total of 26 vulnerabilities. Version 8.2.2, at number four on the list with 949 occurrences, has anywhere from five to ten vulnerabilities depending on the patch-level.

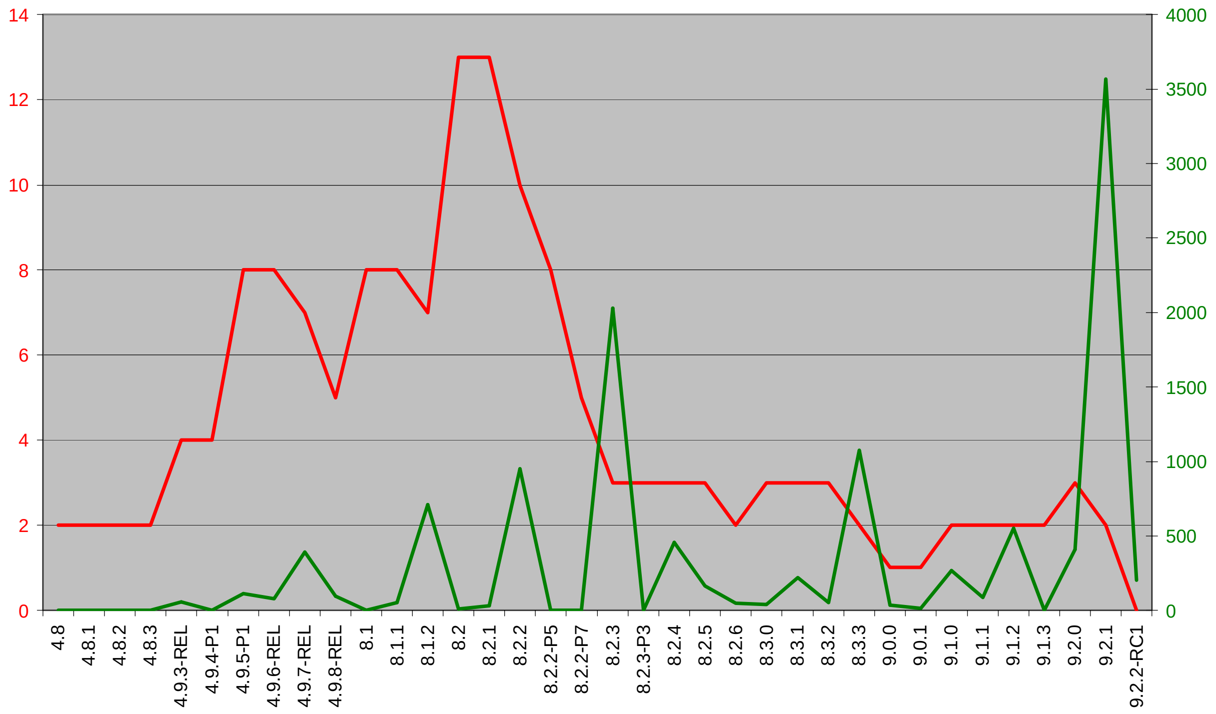
Chart 5: BIND server distribution across the Internet
2003.01.05



12562 BIND servers total

To get a better feel for just how vulnerable these BIND servers are, we can merge the vulnerability per version chart with server distribution. This trend, as shown in Chart 6, yields an accurate picture of the vulnerability profile of the Internet's DNS infrastructure.

Chart 6: BIND server distribution across the Internet with vulnerabilities
2003.01.05



From this chart it is easy to see the legions of servers out there that are still running old software that contains serious vulnerabilities. For example, while only 9 servers were found to be running the most insecure version of BIND, 8.2 (as mentioned earlier, containing 13 vulnerabilities) there are still 768 servers running BIND4. Version 4! The version 4 code tree has not seen any continued development since 1998. The README files from the final two distributions gives the following warnings:

"This is ISC BIND 4.9.8, hoped to be the last of 4.*, which we are releasing since it has an important security bug fixed. Other less important security bugs in BIND4 remain *unfixed*. You should not be running it. You have been warned."

"This is ISC BIND 4.9.9, hoped to be the last of 4.*, which we are releasing since it has an important security bug fixed. Other less important security bugs in BIND4 remain *unfixed*. You should not be running it. You have been warned."

Indeed, users have been warned. BIND 4.9.8 still has five vulnerabilities, and 4.9.9 still has four.

The root of the problem?

Finally, since they are so important, let's have a quick peek at the 13 root name servers. Table 1 gives a summary of each root name server and its reported version.

Table 1: Reported root name server DNS versions

2003.01.05

Root Name Server	Location	IP address	DNS Version
A.ROOT-SERVERS.NET	VA, US	198.41.0.4	VGRS1
B.ROOT-SERVERS.NET	CA, US	128.9.0.107	8.2.5-REL
C.ROOT-SERVERS.NET	VA, US	192.33.4.12	8.3.3-REL
D.ROOT-SERVERS.NET	MD, USA	128.8.10.90	8.3.1-REL
E.ROOT-SERVERS.NET	CA, USA	192.203.230.10	8.3.3-REL
F.ROOT-SERVERS.NET	CA, US	192.5.5.241	9.2.2rc1
G.ROOT-SERVERS.NET	VA, US	192.112.36.4	*server failed*
H.ROOT-SERVERS.NET	MD, US	128.63.2.53	9.2.2rc1
I.ROOT-SERVERS.NET	Stockholm, SE	192.36.148.17	8.3.2-REL
J.ROOT-SERVERS.NET	VA, US	198.41.0.10	VGRS1
K.ROOT-SERVERS.NET	London, UK	193.0.14.129	8.3.3-REL-NOESW
L.ROOT-SERVERS.NET	CA, USA	198.32.64.12	BIND-8.3.1-MA-PATCH-JMB-01
M.ROOT-SERVERS.NET	Tokyo, JP	202.12.27.33	8.3.4-REL

Interesting to note that, even though Verisign has stated its intent to phase BIND out in favor of proprietary software dubbed "ATLAS" [14] most of the root name servers are still reporting versions of DNS that have serious vulnerabilities.

Unbind

While definitely the most popular, BIND is certainly not the only open source DNS package available. Dan Bernstein, well known for his ultra-secure open-source software, offers to the public the tinydns program via his djbdns suite of DNS tools. Unlike BIND, tinydns was built with security in mind and, according to the documentation: "every step of the design and implementation has been carefully evaluated from a security perspective". Bernstein is also extremely vocal in pointing out some of the deficiencies in the BIND software [15]. Since its introduction in 1999, there has yet to be a publicly disclosed security vulnerabilities in djbdns. Bernstein backs up his software by offering an as yet unclaimed $500 bounty to the first person who can report a verifiable security vulnerability with djbdns.

Remediation

As it turns out, when simplified, there are actually two core problems that need solving. Poor programming is obviously the main issue enabling the vulnerabilities, but poor patch and upgrade management prevents servers from becoming secure once fixes are made available. The bottom line is that the developers need to get better at writing code that isn't insecure, and, administrators need to get better at upgrading and patching their systems when they are found to be insecure.

When considering BIND as a case study in the software development lifecycle, it is a perfect example of what happens when security is retrofit as opposed to designed into the product at the design and requirements phase [16].

Denouement

While this report is not the first to comment on security vulnerabilities relating the Internet's DNS infrastructure, it is the first to present substantive proof quantifying just how vulnerable they are. Given that DNS is essential to the inner-workings of the Internet and its stability is of paramount importance, the conclusions to be drawn from this paper are portentous. A decisive attack against an organization's DNS servers can be damaging to that company. A decisive attack against the Internet's root name servers can be devastating to the entire Internet. In today's world of political and global uncertainty administrators and programmers alike can no longer afford to leave security as an afterthought, bolt-on, or luxury. It is a requirement.

Endnotes

[1] Advanced readers will note this isn't always the case. Consider that RFC 1918 networks enable properly separated internal networks to contain identically numbered systems. Additionally, many devices can be IP-aware without actually having a valid IP address such as network protocol analyzers and Network Intrusion Detection Systems

[2] The National Strategy to Secure Cyberspace (draft): http://www.whitehouse.gov/pcipb/cyberstrategy-draft.pdf

[3] A detailed discussion of the Microsoft DNS outage: http://www.menandmice.com/dnsplace/healthsurvey.html

[4] There is a special reason why there are 13 root name servers. It is the maximum number of servers that can exist given that the maximum size of a UDP datagram is 512 bytes. Only 13 NS records and their corresponding A records can fit inside of a 512 byte DNS packet

[5] Washington Post article on the DDoS attack against the 13 root name servers: http://www.washingtonpost.com/wp-dyn/articles/A828-2002Oct22.html

[6] The author downloaded all of the source code from ftp.isc.org and wrote a small script to recursively descend into the directories and get a line count of all files ending in ".c" and ".h". Not exactly scientific method, but it worked.

[7] David Kahn: http://www.fas.org/irp/eprint/kahn.html

[8] ISC BIND Vulnerability table: http://www.isc.org/products/BIND/bind-security.html

[9] Unholy trinity puts users in a BIND: http://online.securityfocus.com/news/1640

[10] SANS/FBI 20 Most Critical Security Vulnerabilities: http://www.sans.org/top20/#U9

[11] Chaos class queries can be executed in a variety of ways; one of the most common is at the Unix command line via the 'dig' program: '`dig @server-ip-address version.bind chaos txt`'

[12] Mike Schiffman, Building Open Source Network Security Tools, http://www.amazon.com/exec/obidos/tg/detail/-/0471205443/qid=1035574926/sr=8-1/ref=sr_8_1/102-8983346-7283338?v=glance&n=507846

[13] For example, some of the feature film responses seen were: "`Klaatu, Barada, Nikto`" (Army of Darkness, 1993), "`the phantom menace`" (Star Wars, 1999), "`Negative, I am a meat popsicle.`" (The 5th Element, 1997)

[14] VeriSign to give BIND the boot: http://www.nwfusion.com/news/2002/133242_06-10-2002.html

[15] BIND, the Buggy Internet Name Daemon: http://cr.yp.to/djbdns/blurb/unbind.html

[16] @stake Return on Security Investment (ROSI): http://www.atstake.com/research/strategic_security/rosi.html