Global Information Assurance Certification Paper

CURRENT ISSUES IN DNS

GIAC Certified Intrusion Analyst (GCIA) Gold

Author: Craig S. Wright

CraigSWright@acm.org

Adviser: John Bambenek, bambenek@gmail.com

Accepted: … 2008

Index

i. Abstract

There are many ways to attack DNS. Attacks range from denials of service (DOS) to man in the middle (MiTM) to spoofing. The recent inclusion of Unicode entries into DNS may mean a site that looks like “microsoft.com” could exist but actually point to something else. Perhaps the o's in Microsoft would be Cyrillic instead of Latin. This paper will look at the issues facing DNS as well as conduct an analysis of the existing DNS infrastructure to assess its state and weaknesses. This process will also compare results that have been taken from previous DNS testing exercises.

Introduction and objectives

With the suggestion that pharming[1] attacks are on the increase (Leon, 2005), the security of the DNS infrastructure is being questioned again.

Using qualitative research, the following questions are investigated in this paper:

1. Have the Levels of Security (based on patching practices) have improved since 2000 and 2005?
2. How do the TLD[2]'s and Australian servers compare to the general population of DNS Servers worldwide?
3. How is security of the Internet based on the overall level of DNS Security?

A short account of the Domain Naming Service

BIND, the Berkeley Internet Name Domain service, was created by a team of computer scientists at the University of California at Berkeley. The US Defence Advanced Research Projects Administration (DARPA) funded a graduate student project to enable this research. BIND versions up to and including 4.8.3 were maintained by the Computer Systems Research Group (CSRG) at UC Berkeley. The initial BIND project team included Douglas Terry, Mark Painter, David Riggle and Songnian Zhou.

Ralph Campbell and Kevin Dunlap continued the original work at the CSRG for 2 years--from 1985 to 1987. BIND maintenance was subsequently handled by Mike Karels and O. Kure.

---

1 Pharming – Attacks designed to compromise a DNS Server and use it for the attackers purpose

2 TLD - Top Level Domains

BIND Version 4.9.2 was sponsored by Paul Vixie of Vixie Enterprises, who became the principal architect and programmer of BIND. ISC, the Internet Software Consortium, have subsequently taken over support of BIND.

DNS, or domain name system, is the methodology used on the Internet to convert fully qualified domain names (FQDN) into IP (Internet Protocol) addresses. The process of maintaining a central list of FQDN/IP address correspondences by host file was found to be impractical.

DNS was developed as a service to enable this process to be distributed over the Internet as seamlessly as possible. Every name of every computer on the Internet is translated using a DNS server.

What is DNS, Anyway?

DNS is that unknown worker which goes unconsidered until there is a problem. DNS resolves host names to IP addresses (and also conversely, IP addresses to host names). Without DNS the Internet would stop. This is a big claim until you realize that people do not remember complex numbers. We can remember several thousand names but we cannot remember even 50 IP addresses easily.

Even within organizations, DNS is key to security of access, as individuals connect to named servers and (usually) not to IP addresses. To secure a DNS server, it is essential to consider the following points:

Restrict zone transfers. DNS zone transfers are needed from the primary DNS to the secondary. Never allow anything else, not even secondary to secondary zone transfers.

Disable recursive checks and retrievals. There is no reason to allow recursive queries from every host on the Internet. At best it is a waste of resources, at worst an attack path.

Log ALL zone transfer attempts. Any attempt to do an illicit zone transfer should be treated as an incident. This is always going to be someone or some program looking for information about the configurations of systems. This should never be permitted.

Restrict queries. Not all queries are necessary. Information that is not necessary should be restricted on a need-to-know basis.

Restrict dynamic updates. Only authorized hosts should be allowed to change DNS entries.

Deploy split DNS. Split DNS involves logically and physically separating the external and internal address spaces.

External IP addressing should include that information necessary for services on the Internet to function correctly.

Internal IP addressing should be restricted to your organization's own systems.

Recursive

A DNS Server is recursive when it assumes the duty of resolving the answer to a DNS query. DNS servers are generally recursive by default. Exposed recursive servers can be used by attackers (e.g., cache poisoning attacks). At best, they are wasting system resources doing lookups for unrelated entities.

BIND versions 8.x[3] and above provide the capability to configure the server to be non-recursive, with selected exceptions for specific IP addresses. This allows the servers to answer recursive queries for the organization's own hosts

3 If you are running 8.x and not version 9, then you are waiting to be compromised.

while blocking recursive queries from unauthorized hosts on the Internet.

To configure DNS correctly:

- Recursive queries can be allowed for internal DNS
- Recursive queries should be blocked for external hosts

Where there are exceptions (for roaming hosts, for instance) these can be configured separately.

Zone Transfers

Secondary DNS servers use the zone transfer function to update changes to the DNS zone databases. These changes are received from the primary (or SOA--Start of Authority) DNS servers.

In order to maintain some level of secrecy and slow the reconnaissance phase of an attack, DNS servers must be configured to only allow zone transfers between the primary and secondary DNS servers. Secondary DNS servers should never be allowed to respond to a zone transfer request.

It is important not to block TCP 53[4]. This port is used for large transfers and not just zone transfers. Thinking that you are okay since TCP has been restricted to a DNS server is a common mistake. TCP is used for valid DNS queries. The blocking of TCP port 53 will break DNS and not fix the problems related to zone transfers.

Split DNS

4 DNS uses the Transmission Control Protocol (TCP) Port number 53 for both zone transfers and large requests. Zone Transfers use "source=53 destination=53" over TCP. Large Client queries use a combination of TCP 53 and a port greater than 1023 (i.e. src=1023 dest=53). With this knowledge it is possible to filter TCP server communications while allowing normal TCP DNS traffic.

Split DNS involves the logical separation of the external and internal name resolution functions: Information that is necessary for hosts on the Internet is maintained on the external DNS servers, while information about the internal hosts and IP space is maintained and resolved using the internal DNS servers.

When a system is required to support reverse PTR lookups[5], generic information should be provided. PTR records do not matter; they are just required to resolve to something. To have reverse PTRs work requires a name… ANY name. This does NOT need to be the real internal name for all servers. Further, not all servers need to be included in reverse lookups to the Internet. In the experimental section of this paper, inverse DNS queries were used in mapping the domain servers that are used on the Internet.

Split-Split DNS

A split-split DNS is the ideal DNS architecture. A representation of the split-split DNS architecture is displayed in figure 1. The split-split architecture involves a back to back private address DMZ segment with two firewalls (it is possible to do this with a single firewall and 3 interfaces as well, but this is prone to misconfiguration). In the split-split architecture, the DMZ network and internal private network each have:

- Two DNS Advertiser hosts on the DMZ
- Two DNS Resolver hosts on the DMZ
- Two internal DNS servers on the internal network

5 PTR Records map a an IP address instead of a name. These are the in.addr.arpa records.

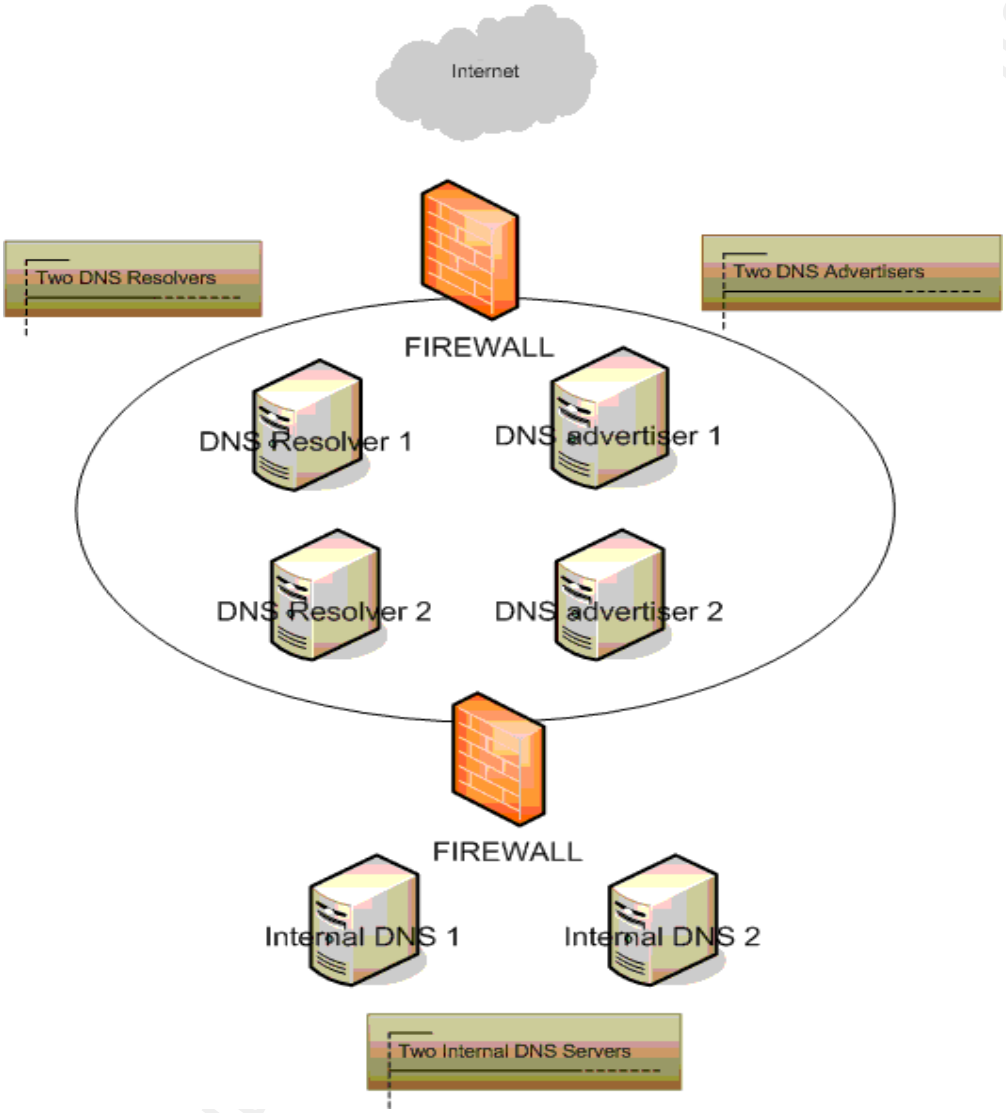


Figure 1 Split-Split DNS

As may be seen in Figure 1, this design consists of at least two of each class[6] of server to provide for fault tolerance and load balancing. At least one of each class of server will be primary and the other a secondary DNS server (Windows Active Directory DNS servers do not use this system). In this system, zone transfers need to be configured such that they are only allowed to occur between the primary and secondary servers. This is:

---

6 The Split-Split architecture consists of DNS servers in the three classes: Internal, Advertiser and Resolver servers.

External DNS. Acts as an advertiser and resolver system.

Internal DNS. Acts to resolve queries for internal client hosts only.

Each zone in the split-split architecture has to contain its own Primary and Secondary DNS. Zone transfers should only be allowed from primary servers to secondary servers (and not the other way).

Split-split DNS has multiple DNS servers located in the DMZ. Separate DNS servers provide name and domain advertising and resolution. A pair of DNS servers is positioned within the internal network, as well. These are all run as duplicates to provide fault tolerance and load balancing.

A total of at least six DNS servers (three primary and three secondary servers) are required for a split-split DNS configuration. The three classes of DNS servers are:

DNS Resolvers. In the split-split architecture, DNS resolvers only provide DNS caching. These systems need to be configured such that they act as DNS forwarders and allow access only from the internal network hosts. They should be configured so that they do not maintain a DNS zone database and are not authoritative for any domains. This setup allows split-split DNS to aid in stopping DNS hijacking attacks.

DNS Advertisers. The DNS advertisers in the split-split system are responsible for maintaining domains that are "advertised" over the Internet by the organization (the organization's authoritative zones). They should be configured such that they don't allow recursive queries to be preformed.

Internal DNS Servers. In the split-split configuration, the Internal DNS servers resolve queries that originate from the internal network hosts. They function identically to the internal DNS servers in a "**split DNS**" setup.

The Basics of making a DNS server secure

DNS queries primarily operate over UDP port 53. DNS Zone transfers (and certain other longer queries) operate over TCP port 53. TCP is still required for normal DNS operation. In addition to this, zone transfers may be restricted more effectively than using filtering over TCP port 53. The use of a security enforcement device, such as a packet filter or firewall, is essential. These devices filter traffic by restricting the allowed queries. Restricting access to the DNS server to just the required ports is the first step in securing your DNS.

Next, the operating system of the server running the DNS software has to be securely configured and locked down. To do this, restrict access to authorised users only and prevent access from unauthorised users. This may seem like an obvious statement but it is one that seems to be frequently missed and is a common flaw (Dept. of Commerce, 2004).

Also, the DNS server operating system should be installed such that it has the bare minimum of functionality to do the required role. That is, no other services should run on the DNS server--it should be a bastion host. Further, file and directory permissions should be the leanest possible for normal operation. These points are just a start. This is why many organisations have taken to outsourcing DNS (Booth, 2004).

The version of software you should run on your DNS is the latest supported version available at the time. Make sure you have all of the latest applicable security patches applied.

Securely configuring your DNS software is necessary. Without this important step, the integrity of your DNS will be compromised. A general rule of thumb when configuring DNS (as with most other Internet Systems) is to *“enable only that*

which is required, from only the locations it is required, and disable the rest" (Ashbury, 2000).

Primary DNS

The DNS software needs to be configured such that it allows:

- Anyone, anywhere, to resolve the names of your externally visible hosts to IP Addresses and vice versa;
- A primary DNS to forward queries for hosts it does not know to a root server (or ideally not handle forwarded requests); and
- The primary DNS for your domain to update the configuration of the secondary DNS servers for your domain.

Secondary DNS

The software on the secondary DNS needs to be configured in such a manner as to allow:

- Anyone, anywhere, to resolve the names of your externally visible hosts to IP Addresses and vice versa;
- The DNS to forward queries for hosts it does not know to a root server; and
- The primary DNS for your domain to update the configuration of the secondary DNS servers for your domain.

Active Directory and DNS

Active Directory domains expand the range of items that are resolved through DNS. DNS and Active Directory allow for the resolution of both DNS names and NetBIOS names. In general, both names are visible to end users.

For the most part, DNS and Active Directory function the same way on the Internet as any other DNS server. The primary

difference comes from the addition of SRV records and a Microsoft-specific subdomain (the _msdcs DNS subdomain) that permit the domain to find the location of selected domain controllers with particular roles in the Active Directory domain or forest.

Multiple Domain controller SRV resource records are registered in a Windows domain through Active Directory. The DNS SRV resource records support services such as “NetLogon” and provide domain-specific locations and service details. Although it was possible to integrate these records into other DNS servers such as BIND, the manner in which they are handled varies enough that it is possible to determine whether the type of server running the SRV records is based on Windows DNS or an alternate server type (e.g., BIND).

For the most part, this is of little concern other than the process of fingerprinting the system. Most systems on the Internet running Windows-based DNS are not running on Active Directory. The biggest vulnerability here is where organisations expose internal DNS structure to the Internet directly. The registration of SRV and _msdcs records, which contain internal information that could be of value to an attacker on an external DNS server, makes information gathering simpler.

An attacker who can get access to an Active Directory DNS server that records internal server details can bypass noisy scanning techniques and start attacking services directly.

The Microsoft paper, “*How DNS Support for Active Directory Works*” (March 28, 2003) introduces the functioning of Active Directory DNS and details the complete list of Windows-specific DNS records.

The threats of an insecure DNS

The threats are as limitless as one's imagination, and we do not plan to cover all of them in this document. We will briefly cover a few in the following sections.

The threats mentioned below have been broken down into the categories of those against Confidentiality, Availability and Integrity.

Threats to Confidentiality

By controlling where hosts connect, the DNS infrastructure becomes critical to the confidentiality of systems on the Internet. If DNS can be compromised, an attacker can do a man-in-the-middle attack, monitor traffic, and generally break the security of many systems.

A man-in-the-middle attack is also known as the bucket-brigade attack and Janus attack. It is an attack based on actively eavesdropping and controlling the communication. By compromising DNS, the attacker can easily take over most systems and protocols.

Eavesdropping Attacks

If you run any system and you use a Fully Qualified Domain Name(FQDN) to connect, then DNS is critical to the security of your systems. Overall eavesdropping is a straightforward attack.

Mail

If you run any other software on the DNS server, and the DNS is compromised in a way that allows the attacker operating system-level access to the server hosting the DNS, any data traversing the server that runs the DNS will be able to be intercepted and captured by the attacker.

For example, if your DNS server was also a mail relay for your organisation, the attacker could read all mail messages entering or exiting your domain. If there was a lack of

adherence within your organisation to obeying your security policy, and sensitive information was being regularly transmitted via email, the attacker could collect a lot of valuable information from this attack.

General Traffic Sniffing

If the DNS was poorly located in such a way that all traffic entering or exiting your organisation had to pass it, the compromised server could be used to eavesdrop on all inbound and outbound traffic, such as:

- E-Commerce transactions,
- Remote access sessions, and
- File transfers.

Trust Relationship Exploit

If the security of your organisation had been poorly configured to allow the DNS to access other servers within your organisation, or even in your bastion zones, it could be used as a springboard by a successful attacker from which to launch attacks against other, more valuable information assets.

Threats to Integrity

The following are examples of what could happen in the event of a compromise of integrity of your DNS:

Mail Redirection

If an attacker can alter the address of your primary mail exchanger (MX) record, they can effectively:

- Deny your ability to receive mail,
- Receive all of your mail and reply to it, making it look like it came from your organisation, and bring your organisation into disrepute by sending obscene or inaccurate replies,

- Publicise sensitive mail messages on newsgroups or other media, thereby causing loss of trust from your customers/shareholders,
- Receive and forward emails such that only some emails are altered.

Web Redirection

In this scenario, if an attacker can alter your DNS records, they could redirect your customers to:

- Your competitor's site,
- A bogus site containing anti-social content,
- A site that looks like your site but contains inaccurate content,
- A site that states your site has gone out of business,
- a spoof/phishing site and capture their credentials (e.g., Internet banking)

E-Commerce Redirection

In this scenario, if an attacker can alter your DNS records, they could redirect your customers to another site:

- which takes their orders and accepts the payment but doesn't provide the goods, or
- proxies all traffic back to your real e-commerce server to capture customer details and credit card information.

Masquerading

In this scenario, an attacker places their address into your DNS so it looks as if the attacker's host is one of your systems. They can then use this host to commit acts against other hosts on the Internet while pretending to be from your domain.

This is as simple as removing one of your IP Addresses and inserting theirs in the DNS configuration files. When their address resolves in someone's log files, it will look like a server from your domain. They commit the attacks on others and then change it back to normal, and when the person suffering the attack goes hunting for the attacker, it looks like you did it.

This type of attack could cause very bad publicity for your organisation and subsequent loss of customers/shareholders.

Threats to Availability

Your DNS is probably **the most critical part** of your organisation--without it:

- People cannot determine where to send mail to you, and
- People cannot determine how to get to any of the services you provide.

Redirection

In this attack, the attacker simply redirects the address of any of your servers to a non-functioning address, thereby making your site inaccessible.

Another twist on this attack is to direct all of your web and mail traffic to a server within your domain, on another DMZ, which was not running a mail relay or web server. This would have the added effect of causing an additional load on your security enforcing devices such as packet filters and firewalls, as the traffic bounced in towards the server not running the services and out again as it got rejected, resulting in twice the traffic levels normally experienced by your organisation .

Deletion

A deletion attack comes about when an attacker removes entries from your DNS servers, thereby making those hosts inaccessible. At its simplest, this is a form of DoS (Denial of Service). At its worst, this can be used as a targeted attack designed to stop communications to and from a secure server, monitor, alerting engine or other security device. For instance, if the attacker has obtained intelligence showing that all router logs in an organisation go to a server at **syslog.company.com,** deleting this DNS entry will make the logging on these devices fail. This leaves the attacker able to conduct their attack on the routers without having to worry about detection.

DNS is Critical

DNS is arguably the most important service on any Internet-based network (VeriSign, 2003). The domain naming service is more crucial than even the web server or mail. Without DNS, the Internet stops (McCahill et al., 1995)--no mail, no web and no e-commerce (RFC1862).

DNS at the packet level

A DNS message will contain a header and up to four data sections. The following figures demonstrate the format and contents of a DNS packet:

ID	FLAGS
Number of Questions	Number of Answers
Number of RR Authority	Number of Supplementary RR
Question	
Answer	
Authority / Additional information	

Table 1 The DNS Packet internals

The DNS packet is contained within the IP packet (Figure 2). This fits into the IP packet in the following manner (note that the relative field length provides the field lengths in this diagram):

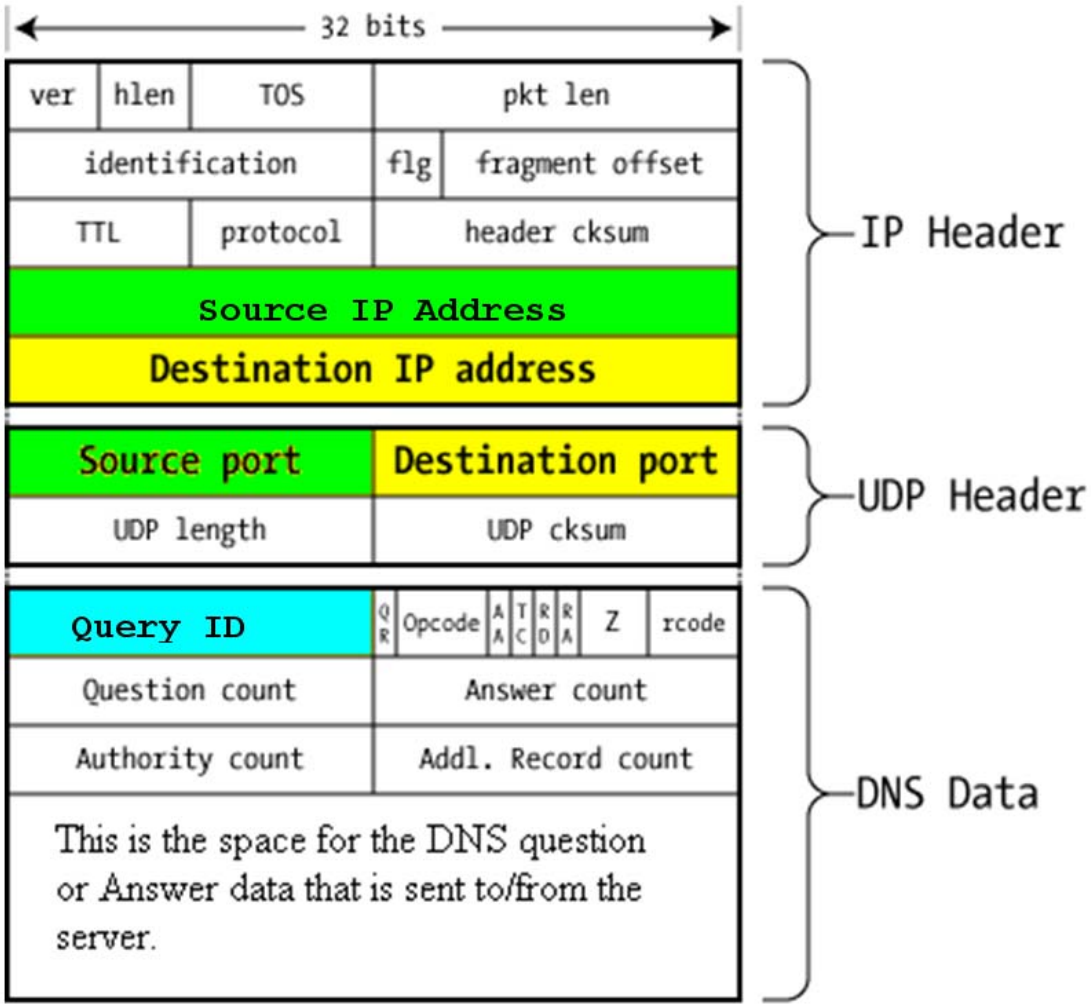


Figure 2. The configuration of an IP packet for DNS

Note that not all DNS traffic uses UDP. DNS Queries can use TCP when there is a large amount of data that needs to be added to the reply. Many sites simply restrict TCP to the DNS server; this is a mistake and causes many issues with domain resolution. Like obscuring the header, this approach does not secure the DNS server but rather creates a false sense of security.

The information contained within in the DNS section of the packet is further as follows:

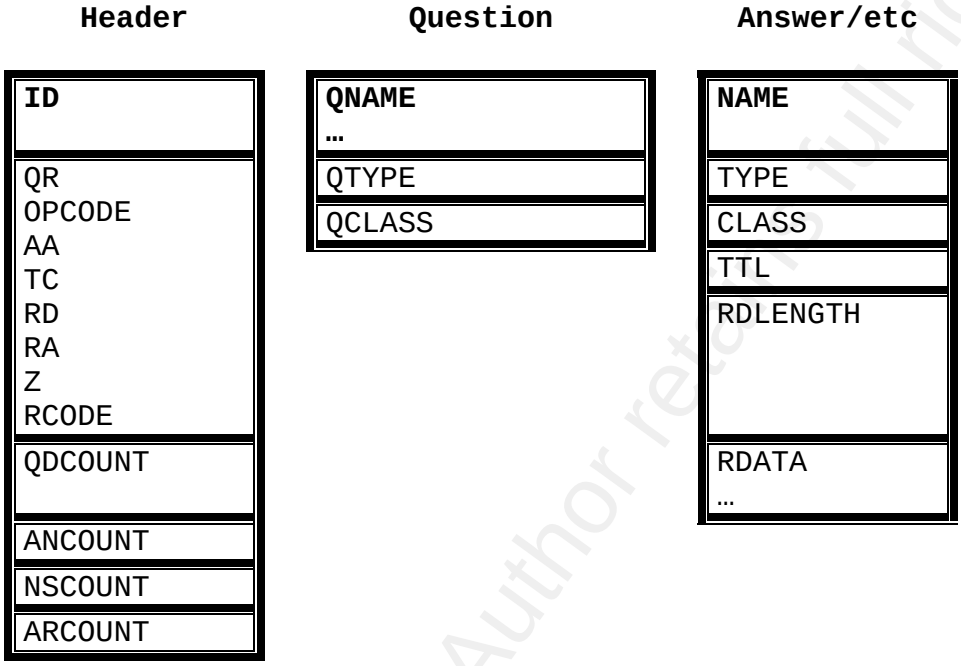


Figure 3. The DNS Fields

The Query ID identifies each DNS packet. A 16 bit indenter is assigned by the program that generates any kind of query. In modern computing terms, 16 bits is insufficient as a security control, hence the recent DNS poisoning exploits.

The fix would be to increase the number of possible Query ID values (that is, to make this a 32 bit field). This will require that all systems on the Internet are updated--a process that is unlikely to succeed in the short term. The fix is to randomize the source port and the Query ID. DJBDNS[7] has used this process from its initial version and has not suffered many of the common DNS vulnerabilities.

- The **QR** bit is used to denote whether the message is either a query (QR=0) or a Response (QR=1).

7 DJBDNS is a simple and security-aware DNS implementation.

- The **OPCODE** word is four bit word that instructs the server as to what the message contains:
  - 0 = standard query (QUERY),
  - 1 = inverse query (IQUERY),
  - 2 = server status request (STATUS),
  - Values “3” to “15” are reserved for future use.
- The **AA** bit is only legal in a response packet. This field indicates that the responding DNS server is authoritative for the FQDN that is listed in the question section of the packet.
- The **TC** bit identifies whether a DNS message has been truncated.
- The **RD** bit specifies if recursion is desired by a query.
- The **RA** bit species if the name server queried supports recursive requests.
- The **Z** field is 3 bits in length and is currently reserved for future use.
- The 4 bit word containing the **RCODE** holds the response values/codes:
  - 0 = No Error Condition (this is the normal response)
  - 1 = a Format Error
  - 2 = a Server Failure
  - 3 = a Name Error
  - 4 = an error indicating that the requested attribute is **Not Implemented** on this name server

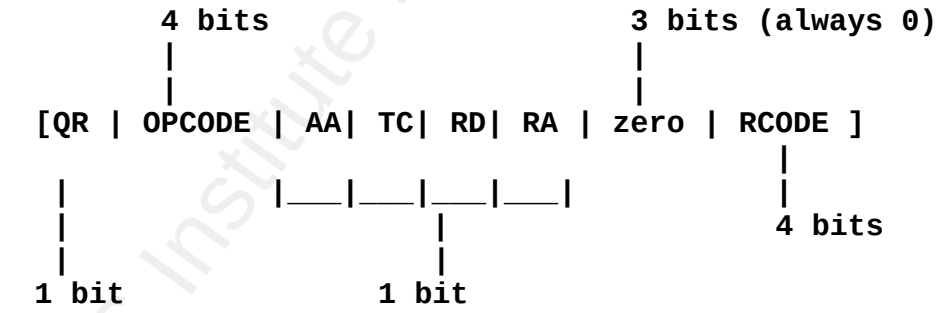
  - 5 = the name server has declined the operation (such as a query that has been administratively prohibited)
  - Values “6” to “16” are reserved fields for possible future use

- The following four unsigned 16 bit integer values specify the number of entries

The DNS packet format is always the same. This results in selected sections of the packet being left empty. This fact is useful in server version analysis (as is covered in a subsequent section of this paper). As these responses are not defined, altering the unset flags provides a part of the puzzle in determining version information for DNS Servers.

The content of the question, answer, authority, and additional sections of the DNS packet serve separate goals. They are, however, always formatted in the same order and are always structured the same.

The flags are divided as follows:



The DNS Question Section holds the query name, query type and query class values.

The Name of the Question	
Type of Question	Type of Query

The class mnemonics and values include the following:

- 1 for IN Internet
- 2 for CS CSNET
- 3 for CH CHAOS
- 4 for HS Hesiod
- 255 for wildcards

The structure of the question is demonstrated in the following examples:

www.microsoft.com =

[3|w|w|w|5|m|i|c|r|o|s|o|f|t|3|c|o|m|0]

And:

201.21.57.203.in-addr.arpa will be encoded as:

[2|2|0|1|2|2|1|2|5|7|3|2|0|3|7|i|n|-|a|d|d|r|4|a|r|p|a|0]

There is also a compression format that is supported in DNS but that is beyond the scope of this paper.

The type of question holds the values are used most frequently in DNS queries. RFC 1035[8] initially defined the types and classes of Resource Records that can be requested. These include:

- A The Host address (IP address)
- AAAA The IPv6 Address
- NS A Name Server (authoritative) record
- SOA Start of Authority
- PTR A Pointer to another location
- HINFO Host Info (common no longer used)

---

8 See www.ietf.org/rfc/rfc1035.txt

- TXT Text records for the domain/system
- CNAME The canonical or primary name (Alias)
- MX Mail Exchange (Email record)

RFC 1035 defines the following table for the RR types and values (other values, such as the SRV record added with Active Directory from Microsoft, have been incorporated into later RFCs):

TYPE	Value	Meaning
A	1	a host address
NS	2	an authoritative name server
MD	3	a mail destination (Obsolete - use MX)
MF	4	a mail forwarder (Obsolete - use MX)
CNAME	5	the canonical name for an alias
SOA	6	marks the start of a zone of authority
MB	7	a mailbox FQDN
MG	8	a mail group member
MR	9	a mail rename FQDN
NULL	10	a null RR
WKS	11	a well-known service description
PTR	12	a FQDN pointer
HINFO	13	host information
MINFO	14	mailbox or mail list information
MX	15	mail exchange
TXT	16	text strings

These are but a small sample of the supported records. The addition of new record types (such as Microsoft's SRV record) aids in fingerprinting the DNS version and type.

The type of query section holds the same values as are contained in the type of question field.

DNS as defined in the RFCs from 1033 to 1035 and 1037 are the place to start if you want to learn more on these records and fields.

The DNS Answer Section holds the resource records that answer the question supplied to the DNS Server. The authority section holds any resource records that detail additional authoritative servers.

The format of an answer (RR) is:

Name of the Domain	
Type	Class
TTL (Time to Live)	
Resource Data Length	
Resource Data	

The class flag = 1 for Internet data.

The time to live flag contains (in seconds) the time-life of the information--how long it is valid for.

The additional section contains any resource records that are not explicitly requested. These can aid the DNS system in the use of the resource records in the further sections.

Attacks Against DNS

With the recent announcements by Dan Kaminsky, DNS is again in the spotlight. As noted above, DNS is arguably the most critical service on the Internet today.

DNS rebinding attacks

There are many ways to attack DNS. Attacks range from denials of service (DOS) to man in the middle (MiTM) to spoofing. The recent inclusion of Unicode entries into DNS may mean a site

that looks like 'microsoft.com' could exist but actually point to something else. Perhaps the o's in Microsoft would be Cyrillic instead of Latin. Such attacks are a concern but are beyond the scope of this paper.

The focus of this section of the paper is an attack originally known as the Princeton attack and its derivatives. The Princeton attack is a DNS-based attack on JavaScript's domain-based security scheme. It's normally accepted that the Princeton attack can not be barred by a useragent and needs to be solved by firewalling. This does not mean that useragents should not attempt to protect against the attack. There is no bulletproof solution to protect against this, and the more people understand what it is about, the higher the chance that a solution will be found.

Note: I have "picked on" keygen.us, as this is a known spyware site active in the distribution of software cracks and other illegal material. In fact, the site attempts to load a number of Java and other applets for this and other attacks.

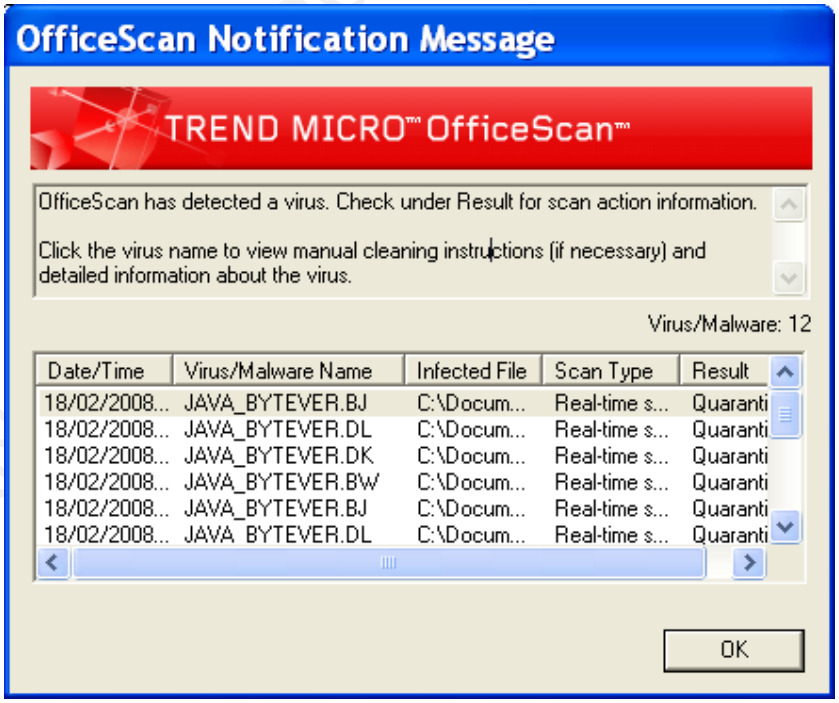


Figure 4. Malware code

They also attempt to load a number of signed activeX and other code segments.

No malicious code was run outside an isolated sandbox for the purpose of writing this paper. The site did attempt to run though. So please do not go to this site if you are on a production host (or one that you in any way care about).

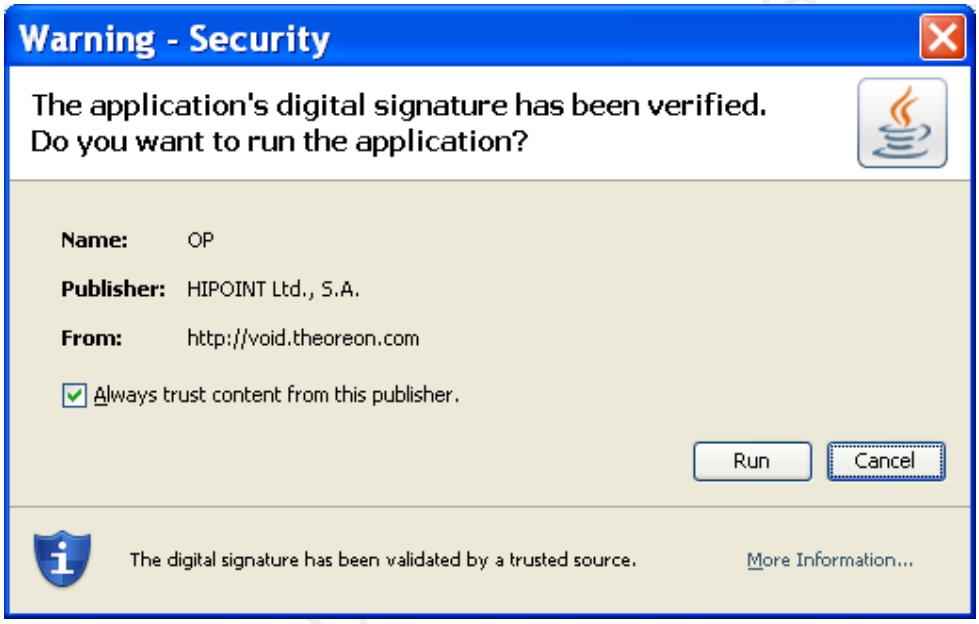


Figure 5. Digital Certificates

What is the same-origin policy?

The same-origin policy prevents documents or script loaded from one origin from getting or setting properties of a document from a different origin. The policy dates from Netscape Navigator 2.0.

Mozilla considers two pages to have the same origin if the protocol, port (if supplied in the call), and FQDN are the same for both pages. To illustrate, this table gives an example of origin comparisons to the URL http://store.microsoft.com/dir/page.html.

URL	Outcome	Reason
http://store.microsoft.com/dir2/other.html	Success	

http://store.microsoft.com/dir/inner/another.html	Success	
https://store.microsoft.com/secure.html	Failure	A different protocol was used in the URL
http://store.microsoft.com:81/dir/etc.html	Failure	A failure is due to the altered port in the URL
http://news.microsoft.com/dir/other.html	Failure	A failure is due to the changed host in the URL

There is one exception to the same-origin rule. A script can set the value of **document.domain** to a suffix of the current domain. If it does so, the shorter domain is used for subsequent origin checks. For instance, assume a script in the document at http://store.microsoft.com/dir/other.html executes this statement:

```
document.domain = "microsoft.com";
```

After execution of that statement, the page would pass the origin check with http://microsoft.com/dir/page.html.

However, using the same reasoning, company.com could NOT set document.domain to othersite.com.

What is DNS Pinning?

DNS Pinning involves storing the DNS host lookup result for the lifetime of the browser session. The basis of this attack is old. It was described by Princeton University in 1996.

The same-origin policy is an access restriction implemented in most modern browsers that prevents a script loaded from one origin to access documents from a different origin in any kind. Hence, it is neither possible to set nor get information from that foreign origin. Security researchers have spent a significant amount of time to find ways to bypass this restriction. One result was Anti DNS Pinning and later on Anti-Anti-Anti DNS Pinning, both exploiting another security mechanism of modern browsers called DNS Pinning.

First we need to explain what DNS Pinning is. This requires a bit of background information on the Domain Name System (DNS). When someone requests a website such as www.microsoft.com, the browser needs to perform a DNS lookup on that domain to get the associated numerical address (IP) of the server that hosts the website in question. In the next step, the browser sends a query to that IP that contains the domain, a specific Web page and other variables to be able to ultimately retrieve the requested data.

So let's assume the DNS lookup on www.microsoft.com provided the IP 207.46.193.254. A normal HTTP request sent by the browser to www.microsoft.com may look like this:

GET / HTTP/1.1

Host: www.microsoft.com

User-Agent: Windows-RSS-Platform/1.0 (MSIE 7.0; Windows NT 5.1)

MSIE /7.0

Accept: */*

Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Connection: keep-alive

Cookie: secret authentication token 12345

Now for DNS Pinning. As a protection attempt against Anti DNS Pinning, the browser caches the FQDN-to-IP address pair until the browser window gets closed, regardless of what the actual DNS time to live (TTL) is set to. See the example below where an attacker runs keygen.us pointing to IP address 85.17.52.48.

The attacker has full access to the DNS server entry, which is set to a TTL (DNS timeout) of 1 second. When viewing his Web site in a browser, malicious JavaScript will be

executed that tells the browser to connect back to its current location in 2 seconds and then pull the returned data to a different server that the attacker controls.

1) The user's browser connects to keygen.us and performs a DNS lookup for that URL, receiving 85.17.52.48 with a TTL of 1 second.

2) JavaScript tells the browser to connect back to keygen.us after two seconds, shortly after the TTL expired.

3) Since the DNS is not longer valid, the user's browser connects to the DNS server to ask where keygen.us is now located.

4) The DNS server responds with 207.46.193.254, which points to www.microsoft.com

5) The user's browser connects to 207.46.193.254, sending a header like:

GET / HTTP/1.1

Host: keygen.us

User-Agent: Windows-RSS-Platform/1.0 (MSIE 7.0; Windows NT 5.1)

MSIE /7.0

Accept: */*

Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Connection: keep-alive

Notice that the host has been changed to keygen.us instead of www.microsoft.com and furthermore, the cookie is missing. Due to the cached FQDN-to-IP pair, DNS Pinning prevents the second lookup of keygen.us.

Normally requests from code embedded in Web pages (JavaScript, Java, Flash) are limited to the website they are

originating from (same-origin policy). DNS rebinding attack can be used to improve the ability of JavaScript based malware to penetrate private networks, subverting the same-origin policy.

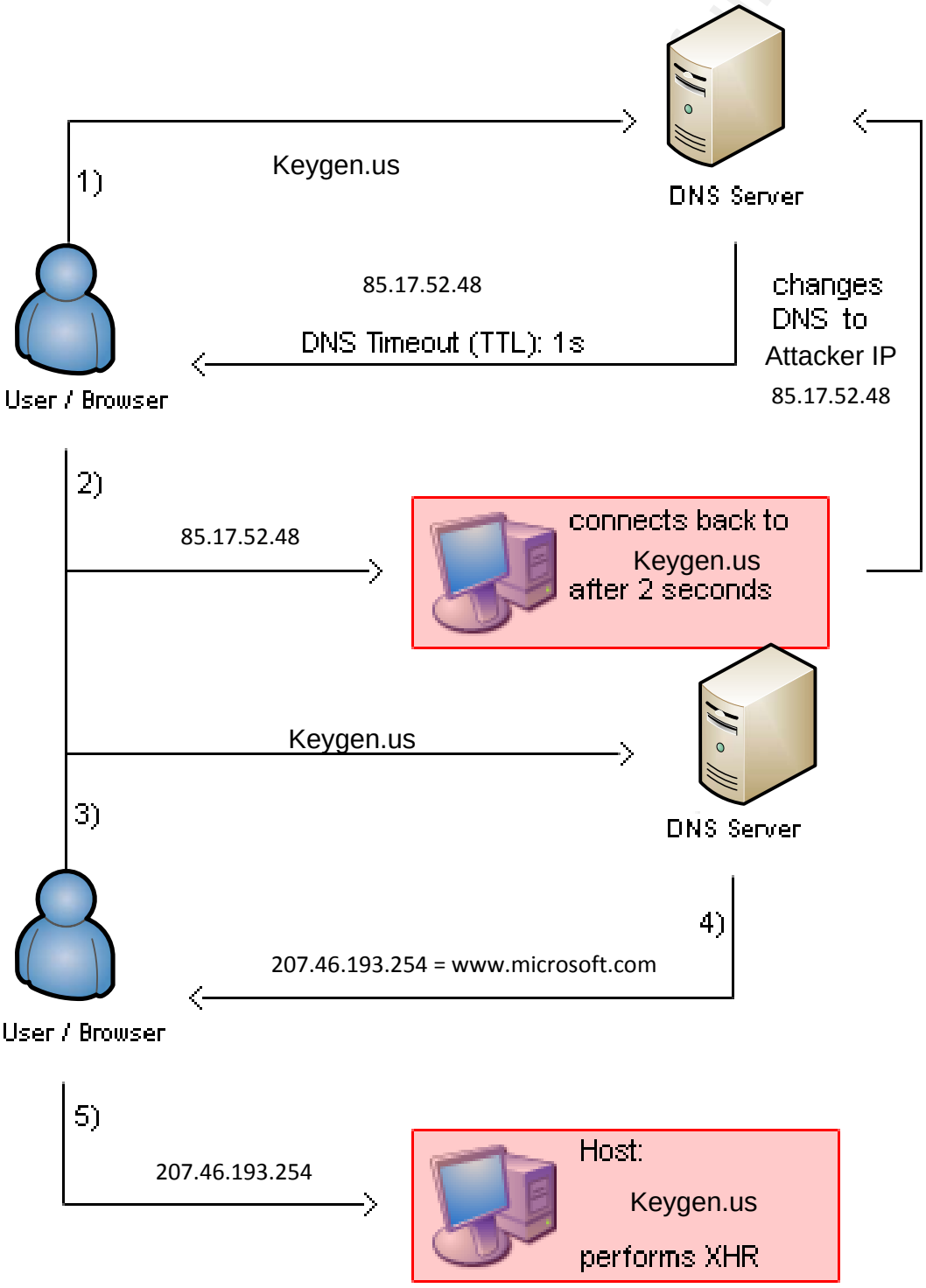


Figure 6. DNS Pinning

Anti DNS Pinning (Re-binding)

Anti-DNS Pinning is what DNS Pinning was meant to defend against. This involves forcing the browser to request a manipulated DNS entry again, e.g., by making it seem that the cache expired.

DNS Pinning only works on the condition that the Web server in being accessed is online and available. This is a result of the belief that if the server appears to be down, a new DNS lookup is necessary to find out whether it has changed or moved. However, an attacker can shut down any server that he controls any time he desires, thereby circumventing the user's DNS Pinning in the browser.

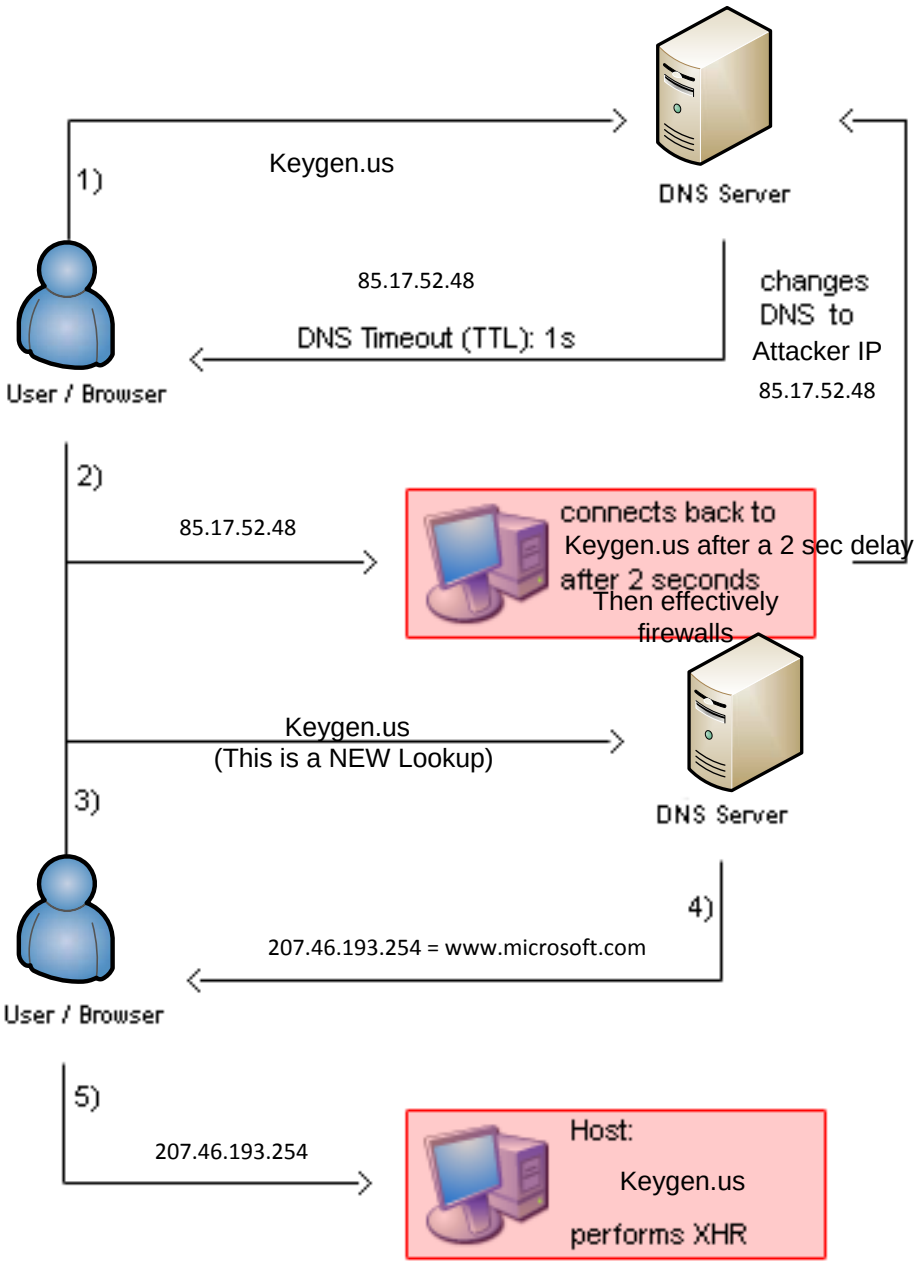


Figure 7. Anti-DNS Pinning

1) The user's browser connects to keygen.us and performs a DNS lookup for that URL, receiving 85.17.52.48 with a TTL of 1 second.

2) JavaScript tells the browser to connect back to keygen.us after two seconds, shortly after the TTL expired. After this, the server is instructed to firewall itself.

3) Now DNS Pinning is dropped due to Anti DNS Pinning. As the

DNS is no longer valid, the user's browser connects to the DNS server to ask where keygen.us is now located.

4) The DNS server responds with 207.46.193.254, which points to www.microsoft.com

5) The user's browser connects to 207.46.193.254, sending a header such as:

GET / HTTP/1.1

Host: keygen.us

User-Agent: Windows-RSS-Platform/1.0 (MSIE 7.0; Windows NT 5.1)

MSIE /7.0

Accept: */*

Accept-Language: de-de,de;q=0.8,en-us;q=0.5,en;q=0.3

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Connection: keep-alive

As the IP address has changed, the attacker's XMLHttpRequest is reading a different Website (www.microsoft.com), even though the browser believes it is still the same. We are able to break same-origin provisions for Javascript, etc. using Anti DNS Pinning.

Note, however, that the host entry has changed to keygen.us instead of www.microsoft.com, plus there is no cookie data sent in the header. Taking this into account, one may wonder why anyone would do Anti DNS Pinning instead of requesting www.microsoft.com. As a consequence, Anti DNS Pinning isn't doing the attacker any good unless the attack is against an intranet or otherwise IP restricted websites, which the attacker could usually not connect to himself because the site is just inaccessible to the public.

This is where Anti DNS Pinning becomes dangerous. Instead of targeting www.microsoft.com, we could possibly launch an

attack against intranet.microsoft.com, which was actually considered to be secure, since it is hosted behind a corporate firewall.

Not only we can read data from those protected pages but also use the so gained information to launch CSRF[9] attacks against intranet applications.

Anti Anti DNS Pinning

The name already suggests what this technique is about. Attackers and researchers have started to investigate how Anti DNS Pinning could be prevented and have come up with the checking of the correct Host header. Remember that this has been changed to keygen.us and so indicates an attack, not only because it is keygen.us, but simply because the Host header differs from the one(s) that has been allowed by the server administrator.

Anti Anti Anti DNS Pinning

Regrettably, the header can easily be spoofed using a variety of methods. Thus the previously described technique is not very effective.

Amit Klein published a posting to Bugtraq demonstrating how to spoof the Host in Microsoft Internet Explorer using XMLHttpRequest or Flash.

```
<*script>
var x = new ActiveXObject("Microsoft.XMLHTTP");

x.open(
"GET\thttp://keygen.us/\tHTTP/1.0\r\nHost:\twww.microsoft.com
\r\n\r\n,"http://keygen.us/", false);

x.send();
alert(x.responseText);
<*/script>
```

---

9 Cross Site Request Forgery (CSRF) Attacks

Browser attacks have become the focus of many attacks as it becomes more difficult to attack the server.

The first question is why?

You visit a site (say you decided that you need a key for that software you downloaded without considering the ethical considerations of not paying). While you are getting the key off the Web page, JavaScript code is downloaded and executed by your Web browser. The script scans your entire internal network, detects and determines your Linksys router model number, and then sends commands to the router to turn on wireless networking and turn off all encryption.

This is why rebinding has again become a current issue. It lurked in obscurity for about a decade following the original Princeton attack, but with ad nets and client attacks all the rage, it has again reared its ugly head. So what, and why?

- Javascript has built-in restrictions to limit abuse-- “Same Origin Policy.” This will only allow a script to interact with the site from which it originated by default.

The issue occurs when there is one site on the Internet where a user can download content from multiple sites. This can occur as a result of translation sites, proxies, etc.

DNS Re-binding is an attempt to subvert the “same origin” policy in a browser. It is based on changing DNS resolution on-the-fly, to alter what the browser considers to be “same origin.” This allows an attacker to “drop” an attack behind a firewall, effectively bypassing it.

No re-binding from a non-RFC 1918 address to an RFC 1918 address is allowed, but beyond this, the fixes tend to break

little, unimportant things like “Akamaized” websites (see http://research.microsoft.com/~ratul//akamai.html).

The browser doesn’t know microsoft.com from the external IP is any different from microsoft.com from the internal IP by design.

Major web sites have IP addresses spread across the world, and resources acquired from them need to be able to script against one another.

Detecting that there’s a cross-IP scripting action happening is only the beginning--what to do after that is what people are trying to figure out

Varieties of DNS Rebinding attacks

What this attack can do?

- Circumvent firewalls to access internal documents and services.
- Sending spam and defrauding pay-per-click advertisers.
- Obtain the (internal) IP address of the hosting web browser
- Port scan the LAN to locate intranet http servers
- Fingerprint these http servers using well known URLs
- And (sometimes) exploit them via CSRF (Cross-site request forgery).

Traditional Re-binding

DNS records have a TTL field that lets you declare how long a record should live in the infrastructure before a second query causes a new request to the original server. By declaring a “0” TTL, DNS records will hypothetically not cache. At this point, each time the browser has a slightly

different DNS request, you get an opportunity to provide a different location.

A problem will occur for the attacker, as many networks won't respect the low TTL. The attacker could wait until the network-enforced minimum TTL expires, but that takes time and makes the attack more difficult.

Spatial Rebinding

DNS responses can contain multiple addresses.

When system.microsoft.com is asked for its IP address, it returns both its address and the address of the printer, which can have an infinite TTL. There is now a question as to which record the browser will choose. The choice is totally random.

Case 1: Browser wants an external IP but it gets internal address.

Attacker Fix 1: External resource is hosted on an unusual port, so the internal connection will fail and thus retry to external. This has problems with outbound firewalls, though.

Attacker Fix 2: Immediately after connecting, look for evidence in the connected session that attack has actually reached the correct server. If not, destroy the object that did the incorrect retrieve and keep trying until success.

The trick: Retrieve the content with XMLHttpRequest so that you can actually destroy the object that guessed incorrectly.

Case 2: Flash/Java wants an internal address but receives an external one.

Attacker Fix: Look for magic token on incoming session. If magic token is returned, destroy the object and try again. If no token has been issued, retry the applet a number of times to ensure that the issue is a consequence of having an extrusion firewall that is blocking the attack.

Ridiculous or Far-fetched?

Many sites deploy DNS TTLs as a security technology. However, DNS TTLs are not a security technology! Overriding a TTL is simple for an attacker when they control the record.

The issue is that this was never far-fetched and attacking TTLs has been done in many ways.

CNiping (pronounced “Sniping”)

CNAME Records: DNS Aliases

Instead of returning an address, many requests will return the “canonical,” or official name and then the address of that canonical name. An attacker acting as the resolver for that canonical name has the capability to add an additional record that can override any value in the cache, even if the TTL hasn’t expired. This works against most, but not actually all, name servers.

What are open network proxies?

Normally, a proxy server allows clients within a defined network group to store and forward internet services such as DNS or web pages, so that the bandwidth used by the group is reduced and controlled. An "*open*" proxy, however, allows any system on the Internet access to its forwarding service.

Through the use of selected open proxies (the so-called "*anonymous*" open proxies), an attacker can conceal their true IP address from the accessed service and host. This is used in access attacks, DoS, and other abuse. Open proxies are therefore often a problem without a solution. The legislative solution fails due to jurisdictional issues in many cases and in others, the site administrators may not know that they are running an open proxy. This can be the result of misconfiguration of proxy software running on the computer, or

of infection with malware (viruses, trojans or worms) designed for this purpose. One such proof of concept proxy was slirpie.

Slirpie (Proxy)

This proxy and attack by Dan Kaminsky requires 3 components:

- The Browser, which has access to internal resources
- The Attacker, which wants access to those internal resources
- The Proxy, which sends code to the Browser to copy messages from the Attacker

The Proxy, which is software designed by Dan, is called Slirpie. It is a Multiprotocol Server, which accepts TCP streams for Browser delivery, containing routing data. It also

- Accepts HTTP requests for those routable streams
- Accepts DNS requests to direct routing
- Accepts XMLSocket requests to determine routing policy

This may be used to subvert controls in Flash. It is designed to allow an attacker that connects to the Proxy to effectively subvert the appropriate resources in Browser to service the Attacker’s connections.

JSON

JSON (JavaScript Object Notation) is a lightweight computer data interchange format. It is a text-based, human-readable format for representing simple data structures and associative arrays (called objects). The JSON format is specified in RFC 4627 by Douglas Crockford. The official Internet media type for JSON is application/json.

The JSON format is often used for transmitting structured data over a network connection in a process called

serialization. Its main application is in Ajax web application programming, where it serves as an alternative to the traditional use of the XML format.

A dns rebinding JSON script is formulated as:

```
{
  "10.0.0.1" : {
    "3" : {
      "from_browser_seq" : -1,
      "server_state" : "CONNECTED",
      "from_browser_ack" : -1,
      "to_browser" : {
        "1" : "YQo=",
        "0" : "Zm9vCg==",
        "3" : "Ywo=",
        "2" : "Ygo="
      },
"dport" : 80,
      "dproto" : 6,
      "browser_state" : "CONNECTING",
      "to_browser_seq" : 3,
      "to_browser_ack" : -1,
      "from_browser" : {

      }
    }
  }
}
```

Javascript alone cannot open the necessary Sockets and thus Flash is necessary. HaXe, a metalanguage, is used to compile both a Flash object and a Javascript interface to it. The Flash object is loaded and directed to create a connection to 10.0.0.1:80

- QUERY ONE: Load the flash image from 10.0.0.1.proxyhost.com (actually Proxy's IP)
- QUERY TWO: Load the security policy controlling <1024 port access from 10.0.0.1.proxyhost.com (this remains as the Proxy's IP)
- DNS REBIND: Instruct the Proxy to return a different address with the next query, using a special HTTP query.
- QUERY THREE: Connect to 10.0.0.1.proxyhost.com:80 (now finally returning 10.0.0.1).
- Connection is in the applet loaded by the proxy, using the security policy provided by the proxy.

Distributed malware

It has been predicted that within the next two years, a cross-site, javascript-based worm will be released. This could be used to exploit XSS injection vulnerabilities using AJAX and to subsequently actively hunt for other, similar, systems through the use of a search engine (similar to googlescanning but in the worm).

Defending Against DNS Rebinding

There are a number of possible solutions:

- You can defend against these attacks for a site by
  - o disabling the Flash plug-in,
  - o disabling JavaScript, and
  - o disabling any other plug-ins.
- Implement host-based (or personal) firewalls in conjunction with a gateway system to restrict browser access to ports 80 and 443. On internal systems, allow access to the Internet through a corporate proxy and not from each host.

- Ensure that all websites you manage do not utilise a default virtual host, but instead require a valid Host header.
- RANDOMISE – that is, make it as difficult as possible!

The latest attack

DNS queries consist of variable-length packets, which are made up of a header, metadata in flags and resource records (RRs). A DNS packet can consist of up to three sets of RRs alongside the originating query:

- Answer RRs: These are the answers to the initial query (e.g., an A record stating that WWW.BIGBANK.COM is A.B.C.D),
- Authority RRs: These records inform the resolver systems where they need to go to (that is, the name servers) in order to obtain an authoritative answer to the initial query,
- Additional RRs: These are what is commonly known as "glue." The glue holds any additional information that is required to make the response succeed.

The Kaminsky DNS Cache Poisoning vulnerability was made into an exploit before it was due to be officially released. As the exploit post states,

> *"This exploit caches a single malicious host entry into the target nameserver. By causing the target nameserver to query for random FQDNs at the target domain, the attacker can spoof a response to the target server including an answer for the query, an authority server record, and an additional record for that server,*

causing target nameserver to insert the additional record into the cache."[10]

It has long been known that weak TTLs pose a flaw in the DNS system. In fact, SANS[11] and many other sources have been commenting on similar weaknesses for many years. Ian Green also posted a similar flaw in his 2005 SANS GSEC Gold paper.[12] These types of weaknesses have been noted and then overlooked as theory for more than 15 years.[13]

The recent exploits and press give the impression that DNS will now be secured. This is the hope. However, in the next section of this paper I shall provide evidence from past research (including my own) that demonstrates otherwise. Schiffman (2003) reflected the poor posture of the DNS through quantitative testing. This is still reflected in the patching practices after the exploit (such as the 50%-60% patched results reported by many CERTS and Kaminsky at his site).

The situation is, however, worse than first suspected (as will be demonstrated below). Many sites have been heavily reliant on the use of obscurity. Rather than patching, changing information such as host headers has created a situation based on hiding rather than securing systems.

DNS Man in the Middle

None of the patches and recent updates fixes the issue of MiTM (Man in the Middle Attacks) against DNS. An attacker with the capability to intercept traffic on the wire (using a

---

10 http://www.caughq.org/exploits/CAU-EX-2008-0002.txt

11 http://www.seoconsultants.com/tools/dns/cache/. Has a list of the various SANS incident handler diaries on DNS cache poisoning.

12 Green, Ian; (2005) "DNS Spoofing by The Man In The Middle," SANS Reading Room.

13 Schiffman, 2003.

compromised router, for instance) and inject traffic can still read and update queries, changing these to suit their desires.

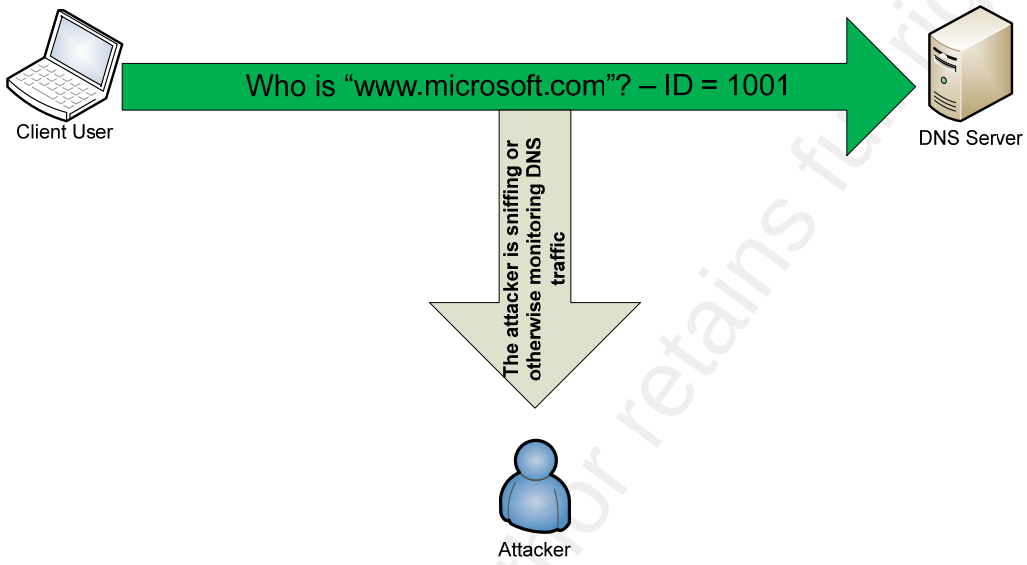


Figure 8. MiTM DNS Attacks

Other than intercepting a router, an attacker could hijack a connection from a host on a local area network in order to:

1. Poison the ARP cache of the victim's host (there are many tools that will aid in this - http://www.arp-sk.org)
2. Step 1 causes the outgoing packets from the target hosts to be redirected to the attacker. Caning routes will allow this attack on a WAN.
3. Tools (such as WinDNSSpoof by Valgasu) can be used to alter and resend the DNS packet. Hping is also useful for this.

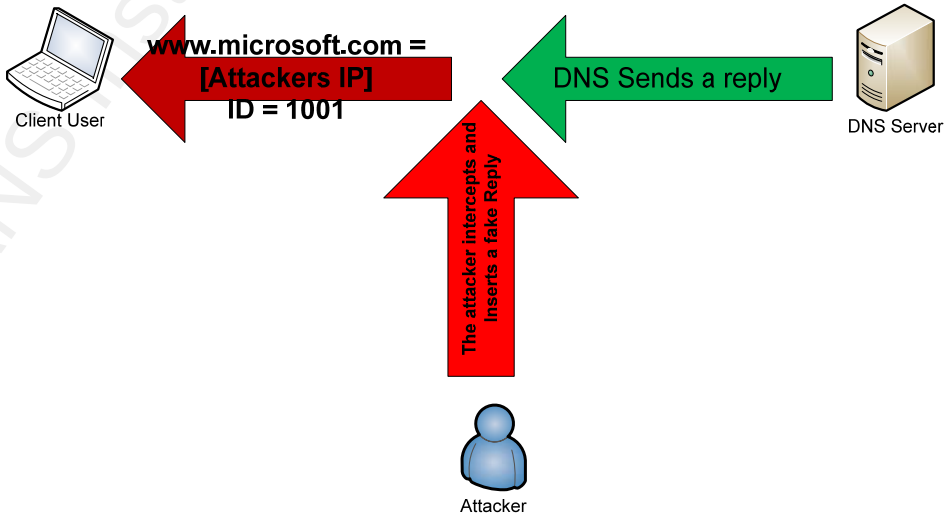


Figure 9. MiTM Attack continued

The recent updates to the DNS infrastructure will not help defend from this attack, as the attacker has both the Query ID and the source port.

Steve Friedl's Unixwiz.net Tech Tips – “An Illustrated Guide to the Kaminsky DNS Vulnerability” (http://www.unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html) - provides a simple (visual) walk-through of the DNS poisoning attack released by Dan Kaminsky.

A Quantitative Study of the State of the DNS Infrastructure

TTLs have been used as a pseudo-security control by many DNS operators. This has been a recent concern in the press,[14] gaining much public attention.

The current issue presented in this paper is patching. Patching is not done to anywhere near an acceptable level. DNS is critical and yet the quality and level of patching was found to be extremely poor.

Methodology used in the study

The design of the following experiment was used to provide the necessary rigour for a detailed experimental study. It was created to determine the state of the DNS Servers on the Internet from a statistically valid sample.

The survey initially used **nmap**[15] to collect a random statistical sample of DNS hosts (in the IP[16] range between 11.0.0.1 and 213.255.255.255). 5,000,000 systems were randomly scanned to determine the overall sample of DNS servers worldwide. “Nmap” has a randomisation feature which enabled the “slicing” of the Internet space to be able to complete this test. Ping was disabled for the tests.

The test limited scanning to UDP port 53 on the preliminary scans to find active DNS servers only. Any results showing access as closed or filtered were ignored, as these (though they may have been DNS Servers) were not publicly available.

---

14 “Exploit code for Kaminsky DNS bug goes wild” - 24 Jul 2008 (http://www.theregister.co.uk/2008/07/24/dns_exploit_goes_wild/).

15 NMAP is an advanced Port scanner available from http://www.insecure.org/nmap.

16 IP is the Internet Protocol.

When a system responded (that is, it was discovered), tests were also conducted using TCP to check responses. All of the protocol flags and packets were fed into an analysis system that created a neural network designed to determine the version of the DNS server being run. These tests did not rely on the version query to discover the version of DNS server being tested.

The scan servers were configured outside the firewall[17] to mitigate any interference. The servers ran the latest version of nmap (nmap-4.68) on Redhat Linux (Red Hat Enterprise Linux 5 Desktop). The Linux servers were "locked down" in order to minimise exposures, as they were externally located in relation to the firewall. All services were patched to the latest levels and only SSH was available remotely on the systems. No access from the Internet was allowed to the servers and an IDS (intrusion detection system) was implemented to determine if the system had been compromised.

No attempted compromise was successful against the test hosts. The servers ran without incident, without rebooting or needing the scan to be restarted.

The scan was also conducted against a control set of 25 known DNS servers from Australian ISPs. Each of the server and patch levels for these systems was known in advance of the scan.

A subsequent test was re-run, designed to collect information on BIND DNS servers running with known shell exploits. This test used a collection of "harvested" IP addresses for known DNS servers. The harvesting process involved collecting "NS" records from "in-addr.arpa" domains.

---

17 It should be noted that the scan servers were "scanned" themselves and that over 10,000 attempts to break in were recorded.

This process allowed for the collection of DNS servers configured to provide services to the “net.” Many systems do not have valid reverse name lookups configured, so there are still many more systems that have not been tested; these, however, are likely to be less well configured than those tested. Good practice calls for the configuration of reverse names for DNS, and many sites do not allow systems configured without a reverse name to connect to them.

As a consequence, it is believed (though not tested) that those DNS servers not set up and using reverse addresses will be even less well configured and have a greater number of vulnerabilities than those tested.

Procedures to determine higher level domain servers

The higher-level domain servers where determined using “ns” records from *nslookup*. The following domains were tested to determine the name servers for these higher-level domains:

1. All base root servers (.)
2. All Australian TLD servers (.au)
3. All COM and EDU servers (.com and .edu)
4. All Australian COM and EDU Servers (.com.au and .edu.au)
5. Miscellaneous higher level systems (.BIZ, .GOV, .Gov.au and .Org)

A list of these servers was entered into both nmap and Nessus[18] to determine the levels of security and the versions of the systems software.

This was then collated against a set of results created by the versioning method detailed below. Where the versioning

---

18 Nessus is a open source vulnerability scanner available from http://www.nessus.org.

information was available (that is, the DNS server version was not obscured), the nmap and Nessus scans provided an accurate response as to which version of software was found to be running. In the event that the version was obscured, the results were less accurate (with the Nessus results being more accurate where a vulnerability was easily determined).

Description of experimental procedure

This was a controlled trial with randomised subjects, selected using the randomisation function of nmap to “discover” the Domain systems with as little bias as possible.

The ISC (Internet Systems Consortium) BIND Vulnerabilities page (http://www.isc.org/index.pl?/sw/bind/bind-security.php) was used to determine the vulnerabilities which could affect the BIND systems discovered. Microsoft’s Security Site (Http://www.microsoft.com/security) and the vulnerability site for Security focus (http://www.securityfocus.com/vulnerabilities) were used to determine vulnerabilities on Microsoft DNS and other DNS servers in the sample.

NMAP was used to verify the results and conduct an examination of the overall levels of security associated with the systems.

No hosts were scanned more than once, to minimise impact that the scanning would cause. Of the servers discovered in the nmap scan, 12,240 systems were determined to be “active” domain servers running a DNS Service for Internet domains. A total of 264,125 DNS servers were discovered using reverse records.

These servers were analysed for susceptibility[19] to:

1. **Denial of Service Attacks**
2. **Testing if the server is susceptible to Cache Poisoning**
3. ***Remote exploit* of the Operating System (at a root or administrative level)**
4. **Remote compromise of the system (at a user level)**

Also, the security of the system was rated based on the overall levels of effort to lock the systems down. These levels have been rated (for the purpose of this paper) as:

1. **High Level Security**
2. **Medium Level Security**
3. **Low Level Security**

High-level security was determined to be a system which had been obviously locked down. Medium was defined as a system with an ample amount of controls in place to secure, and Low level was a determined minimum level of security which would be expected to deter an attacker (such as patching and basic filters).

Any hosts which could not be determined by these tests (either from good security practice, mis-configuration or obscure systems) have been designated as "*Undetermined*."

Finally, systems were categorised into one of two security categories:

1. *Secure*: All systems in the High, Medium and Low Security categories. This field has also been designated as "Adequately Secure" within the document.

---

19 A complete verification of the vulnerabilities was not undertaken due to legal and ethical constraints.

2. *Insecure*: All systems which have been determined to be vulnerable to “Denial of Services Attacks,” “Cache Poisoning“ attacks or which are remotely vulnerable by either exploit or compromise.

The tests took 12 days to run and discovered 281,743 hosts in total, from the 5,000,000 IP addresses used in the test. Testing was significantly faster (at 12 days) than in 1999/2000 (at 48 days).

Results and Data Analysis

The results from the scans were collected and collated using Microsoft Excel. In the test results analysis from 2000, the fields, “**Vulnerable Denial of Services**” and “**Exploitable Remotely**” where not exclusive, with the field “**Exploitable Remotely**” being a subset of the former.

The current analysis has separated these two fields due to a perceived benefit. This change does not invalidate the data but must be taken into account when running comparisons between the “**Vulnerable Denial of Services**” designations in each year.

Results from 2000

Server Types	Vulnerable Denial of Services	Exploitable Remotely	"Adequately Secure"	Percentage of Total Servers Sampled
ISC BIND	86.31%	72.59%	12.08%	65.72%
Microsoft	64.97%	10.90%	35.03%	15.09%
Mac	46.41%	0.00%	53.59%	3.77%
Other	38.01%	18.71%	61.99%	15.42%
Total / Overall	74.14%	52.24%	24.80%	

The results from the 2000 test as documented above clearly show that ISC BIND was the most widely implemented DNS Server in 2000 with 65.72% of the total install base (or 63.54 & 67.90% at a 99% C.I.).

Further, it is clear that with only 24.80% being tested to be “adequately secure” in 2000, there was room for improvement in security practices with these servers.

Results from 2005

Server Types	Vulnerable Denial of Services	Can be Exploited Remotely	"Adequately Secure"	Percentage of Total Servers Sampled
ISC BIND	12.91%	23.85%	63.24%	33.49%
Microsoft	87.53%	0.00%	12.47%	51.11%
Tiny DNS	100.00%	0.00%	0.00%	0.38%
Other (or Unknown)	NA	NA	NA	15.02%
Total / Overall	58.18%	9.40%	32.42%	

Due to constraints, testing of the vulnerabilities on the unknown or “other” servers was not conducted. It was found in 2000 that this process was intrusive and could be seen to breach the ethical constraints placed on this experiment.

What is surprising is the change in composition. In 2000, as was noted, most DNS Servers were running on ISC BIND. Over 50% of DNS Servers are now running on Microsoft DNS. At a 99% Confidence Level (49.2%, 53.0%) Microsoft has between 49-53% of the total deployment of DNS Servers on the Internet. At the same 99% C.I. BNID was found to be (99% C.I. 31.7, 35.3) installed on 33.50% of hosts. This demonstrates a marked drop

in the deployment of ISC BIND in comparison to the total number of systems in the last 5 years.

There is very strong statistical evidence to demonstrate that the security of systems has improved, though with only 1 in 3 hosts being "adequately secured" (up from 1 in 4) this is not something to be too conceited about.

Results from 2008

Server Types	Vulnerable Denial of Services	Exploitable Remotely (shell)	Percentage of Total Servers Sampled
ISC BIND	36.77%	23.85%	65.55%
Microsoft	78.31%	0.00%	16.56%
Tiny DNS/PowerDNS	38.91%	0.00%	2.22%
Other (or Unknown)	---	---	15.67%
Total / Overall	37.93%	15.63%	

What these results have demonstrated is that 15.63% of DNS servers tested are running software versions that have a known shell exploit associated with these. This came to a total of 41,283 servers.

This process was confirmed through an analysis of audit client servers. A total of 156 DNS servers that are audit clients and are thus tested in-depth were used to validate this approach. A total of 102 DNS servers in this class were determined to have a vulnerability. Of these, 57 were tested to have a shell exploit.

These results demonstrated an R^2 value of 98.635 for the methodology. This demonstrates that the test set of audit clients validates the approach and that it is feasible to

determine the version and patch level of DNS software remotely using statistical methods.

Results from 2008

Server Types	Vulnerable (MiTM, Phishing, etc.)	"Adequately Secure"	Percentage of Total Servers Sampled
ISC BIND	79.55%	21.87%	65.55%
Microsoft	84.15%	15.50%	16.56%
Tiny DNS/PowerDNS	NA	NA	2.22%
Other (or Unknown)	NA	NA	15.67%
Total / Overall	-1-	-2-	

A total of 264,125 DNS servers were tested. It is likely that many of these are associated with small sites and home users. Nevertheless, these systems are vulnerable to attack and found the basis for many of the attacks mentioned in this paper.

As not all types of DNS servers have been tested for vulnerabilities, it is not possible to determine the exact range of vulnerable systems. Of those classified and tested to be running BIND or the Microsoft Windows DNS server (85.11% of DNS servers found on the Internet), the total of vulnerable and secure systems is as follows:

-1- Vulnerable (MiTM, Phishing etc.) 30.44 %

-2- "Adequately Secure" 17.56 %

The results demonstrate that only 17.56% of DNS Servers are patched and secured to an acceptable level.

TLD's

Of the 50 High-level servers (2nd level and top level), only one was found to be vulnerable to a software-based "Denial of Services" attack. This is a marked improvement over 2000, as is shown below.

2000 39.15% vulnerable

2005 5.26% vulnerable

2008 2.00% vulnerable

Additionally, no root-level systems were discovered to be vulnerable to a system-level remote compromise at the time of testing. This is a manifest improvement, and the domain operators have shown commendable improvements in their practices.

Security of Systems

It was determined in 2005 that ISC BIND is generally deployed in a more secure configuration. Testing at a 99% C.I. showed Microsoft as being secured correctly in 12.47% of cases, compared to ISC BIND at 63.24% of cases. This does not mean that Microsoft is more secure, just that the deployments are not as secure. There is no evidence to support causation and it cannot be assumed from these results alone that UNIX is more secure than Microsoft.

The 2008 results demonstrated a marked decrease in overall levels of patching and security. It remains the case that ISC BIND is generally deployed in a more secure configuration. Testing at a 99% C.I. showed Microsoft as being secured correctly in 15.50% of cases compared to ISC Bind at 21.87% of cases. Mirroring the results obtained in 2000, this does not mean that Microsoft is more secure, just that the deployments are not as secure. There is no evidence to support

causation and it cannot be assumed from these results alone that UNIX is more secure than Microsoft.

What is alarming is the large increase in unpatched ISC BIND systems. Many of these are likely associated with personal and small site implementations of Linux, however the issues remain of great concern.

The results do demonstrate that DNS systems are not being adequately secured. This may be a direct result of inadequately trained personnel deploying systems without the experience to effectively manage and secure them. This is an area that needs further research.

In testing the Microsoft servers, scans of additional ports (such as RPC) were not able to be effectively conducted due to legal constraints. It is theoretically possible (and likely in the tester's judgment) that a number of systems would have been reported as "**Exploitable Remotely**" had this line of testing been achievable.

Bound to the past…

The discovery of 56 versions of ISC BIND running concurrently was unanticipated. It was generally expected that there would be numerous versions of BIND running at any given time, but the unexpectedly large quantity of reported versions and patch levels was beyond all expectations.

The primary cause of ISC BIND vulnerability was determined to be inadequate patching. Again, this demonstrated a lack of effort expended on the management and maintenance of these systems.

Discussion

Security has improved from year 2000 to now (2008). The change from 24.80% of DNS servers being secure in 2000 to 32.42% in 2005 is a noticeable improvement. The drop back to 17.56% in 2008 poses a definite problem. The issue is that this is still

not good enough. Two out of every three DNS servers were not secure in 2005 and this has increased to three of every four in 2008. The patching effort following Dan Kaminsky's vulnerability may have improved this situation somewhat, but testing was conducted prior to the release of this patch.

Farmer and Spafford have defined security as:

> *to protect the information from disclosure; protecting it from alteration; preventing others from denying access to the machine, its services and its data; preventing degradation of services that are present; protecting against unauthorized changes; and protecting against unauthorized access.*[13]

DNS is the backbone of the Internet and very little works without it. Though security is stronger than it was in 2000, it is still very flimsy. Security is an ongoing process and cannot be taken for granted.

The high level of servers readily susceptible to Cache Poisoning attacks and the servers remotely able to have code run (15.63%) without permission is a serious concern. This is particularly disturbing, as nearly all of the vulnerabilities found on both Microsoft and UNIX hosts were directly linked to inadequate patching.

It is noted that recursion was discovered to be open on over one in four DNS servers tested.

The discovery of 56 versions of the ISC BIND service has demonstrated that the level of knowledge about IT systems needs to be improved. Patching is often overlooked as an unnecessary expense, but it is critical to the security of all systems. Internet accessible systems (such as DNS) are especially prone to attack and need extra care.

A correctly secured and fully patched host is immune to over 90% of vulnerabilities immediately. As with all security

controls, though, there needs to be a trade-off. The efforts of monitoring every host individually are beyond all but the smallest of sites.

It is crucial that the administrator knows the difference between security and general patches and hotfixes. It is also important that all patching be done in an organised manner. A risk management approach needs to be taken to patching systems. Some guidelines are as follows, remembering that it is too late to patch a system after it is compromised. The only way to clean a compromised system is to rebuild it (from a system format).

Some points to remember when patching include:

1. Security Patches need to take precedence over other patches. First determine if the following conditions apply:
    a. Is the patch required for an active service (i.e., an IIS patch for a Microsoft Web Server)? If the service being patched is not installed on the system, then it may not be necessary to patch the system;
    b. Is the Service externally vulnerable? It is important to apply security patches for services that are not available externally as well, but the level of risk is lower;
    c. Does the patch affect other services on the host? Has the patch been tested on a development or QA[20] system and been found to function correctly in your organisation’s environment?

20 QA, Quality Assurance

2. If the patch affects the system in a non-desirable manner (i.e., causes a crash) then it would be better to look at other alternatives, based on the risk to the system and its value. It may be a better option to filter the service, for example.
3. If the patch is determined to be required to ensure the security of the system, then formal patch procedures should be followed for its implementation.

Contrary to the views of many in the field (Mullins, 2005; Ateniese & Mangard, 2000), DNSSec[21] is not the answer. It may be a good addition, but like the majority of security issues, people are the root cause. Technology is touted as the panacea for all our ills. The real cure for many security problems is an efficient and effective patch and maintenance regime. Just as “unpatched Windows operating system vulnerabilities are the top threat to security” (Aste, 2004, p59) on a Windows-based network, unpatched DNS software is the main threat to DNS Security on the Internet.

Conclusion and Future Research

The rise of Microsoft as a foundation for Internet DNS systems was surprising in 2005, but this has dropped in 2008. More research needs to be conducted on this subject. Other services, such as the change in Web server composition and the number of Apache, Netscape and IIS servers, should be investigated.

As can be seen by the results in the previous section, most servers tested still had vulnerabilities. Some vulnerabilities were caused through misconfigurations due to lack of DNS knowledge by support staff (Crespo, et al 2001; Chapman, 1995, Liebke, 1999), some were caused from old

21 See http://www.dnssec.net

versions of BIND being used (once again, through lack of knowledge) and some servers had both. Of prime concern is the lack of adequate patching.

All of the detected vulnerabilities were ones that could have been prevented by appropriately trained staff and an effective maintenance regime.

Well formulated audit and compliance processes need to be developed. “Ethical Attacks” and basic “scans” are not adequate. Organisations need to be able to rely on the security of their infrastructure. For the expansion of commerce into electronic mediums to continue unabated, the general population needs to be confident about the security of the Internet Infrastructure. This means DNS must be managed more effectively.

The odds of finding an unpatched DNS server are too high. Two thirds of servers remaining unpatched is unacceptable. It does not matter what DNS runs on as long as it is secure.

Next steps:

A detailed methodology was presented and published at SANS NS 2008. This details the processes and stages necessary to determine application version in DNS.

Bibliography

1. Ashbury, Adrian; Wright, Craig, 2000, “DNS Security in Australia,” Net-security.

2. Ashe, James P., 2004, “A Vulnerability Assessment of the East Tennessee State University Administrative Computer Network,” East Tennessee State University.

3. Ateniese, Giuseppe & Mangard, Stefan, “A New Approach to DNS Security (DNSSEC),” ACM 2001.

4. Bellare, M, Canetti R., & Krawczyk, H. 1996, “Keying hash functions for message authentication.” In Advances in Cryptology - Crypto 1996 Proceedings, LNCS Vol. 1109, N. Koblitz ed, Springer-Verlag, 1996.

5. Bellovin, Steven M., 1995, ”Using the Domain Name System for System Break-Ins,” Proceedings of the Fifth Usenix Unix Security Symposium, pp. 199{208, June 1995.

6. Booth, Nick, 2004, “The Domain Game,” Computing, CRN (01 Mar 2004), viewable from http://www.computing.co.uk/crn/features/2010587/domain-game

7. Crespo, William, Lauria, Vincent & Theriault, Michael, 2001,"FIREWALLS AND NETWORK SECURITY,” SC 546, May 4, 2001

8. Chapman, D. B., and Zwicky, E. D., *Building Internet Firewalls*, O'Reilly & Associates, Sebastopol, CA, 1995.

9. Cheswick, W. R., and Bellovin, S. M., *Firewalls and Internet Security: Repelling the Wily Hacker*, Addison-Wesley, New York, 1994.

10. Davis, D & Swick, R., 1990, “Network Security via Private-Key Certificates,” USENIX 3rd Security Symposium Proceedings, (Baltimore; Sept. '92). Also in ACM Operating Systems Review, v. 24, n. 4 (Oct. 1990).

11. Dept. of Commerce (NSW, Australia), “Information Security Guideline for NSW Government - Part 3 Information Security Baseline Controls,” section 6.6, 2004

12. Eastlake, D., “Bigger Domain Name System UDP Replies,” Internet Draft, www.ietf.org/proceedings/98aug/I-D/draft-ietf-dnsind-udp-size-02.txt

13. Farmer, Dan & Eugene H. Spafford, July 10, 1992, “The COPS Security Checker System.”

14. Green, Ian; (2005) “DNS Spoofing by The Man In The Middle,” SANS Reading Room.

15. Information and statistics about F.root-servers.net, www.isc.org/services/public/F-root-server.html.

16. Keizer, Gregg, “Phishers get devious on DNS,” TechWeb (viewed 25th June 2005, Dated 05th May 2005); http://www.itnews.com.au/newsstory.aspx?CIaNID=18725&eid=1&edate=20050505.

17. Leon, Mark, 2005, “The looming threat of pharming,” Computerworld, viewable from (01st July 2005) http://www.computerworld.com.au/index.php/id;1181452367;fp;4;fpid;16.

18. Liebke, David Edgar, March 23, 1999, “TSRI Information Security”, Scrippts Research Institute.

19. Lioy, Antonio; Maino, Fabio; Marian, Marius & Mazzocchi, Daniele “DNS Security,” Terena Networking Conference, May 22-25, 2000.

20. Lottor, M.,“Domain Administrators Operations Guide,” RFC 1033, November 1987.

21. Lui, C & Albitz, P, 2001, “DNS And BIND 4/e,” May 2001 by OREILLY.

22. Marsan, Carolyn Duffy, “DoD faces hurdles with DNS security” (viewed 25th June 2005, Dated 08th Oct 2002); http://www.computerworld.com.au/index.php/id;2106933301;fp;512;fpid;272399337.

23. McCahill, M. Schwartz, M. Sollins, K. Verschuren, T. Weider, C. Romkey, J. Ed.; RFC1862, “Network Working Group, Request For Comments: 1862.” November 1995.

24. Mockapetris, P., “Domain Names - Concepts and Facilities,” RFC 1034, November 1987.

25. Mockapetris, P., “Domain Names - Implementation and Specifications,” RFC 1035, November 1987.

26. Mullins, Michael; "How to improve DNS security;” TechRepublic (viewed 25th June 2005, Dated 20th Jan 2004); http://www.zdnet.com.au/insight/0,39023731,39115739,00.htm.

27. RSA Security site defaced. ZDNet 2000. www.zdnet.com/zdnn/stories/news/0,4586,2437384,00.html.

28. Schuba, Christoph (1993), “ADDRESSING WEAKNESSES IN THE DOMAIN NAME SYSTEM PROTOCOL,” Thesis, Purdue University.

29. Schiffman, Mike (2003) “Bound by Tradition, A Sampling of the Security Posture of the Internet’s DNS Servers,” Infonexus.

30. VeriSign, Inc. 08/2003, “DNS Assurance Solutions,” VeriSign Whitepaper.

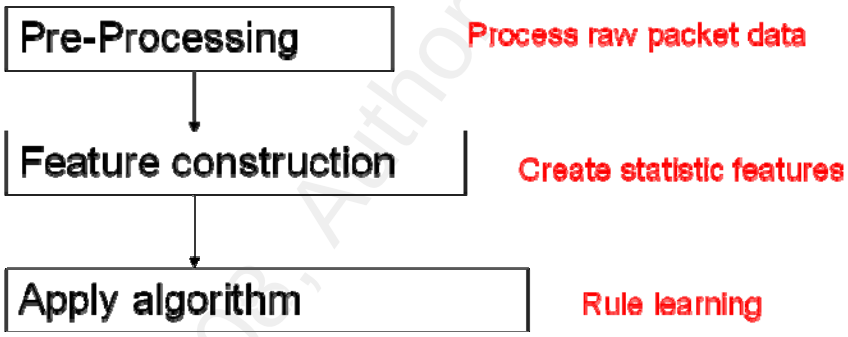
31. Vijayan, Jaikumar, 27 Oct 2003, “DNS servers prove resilient, But the core system is still weak at lower levels,” Computerworld, IDG, US.

Appendix – Determining Versions

The process involved an extensive use of data mining techniques.

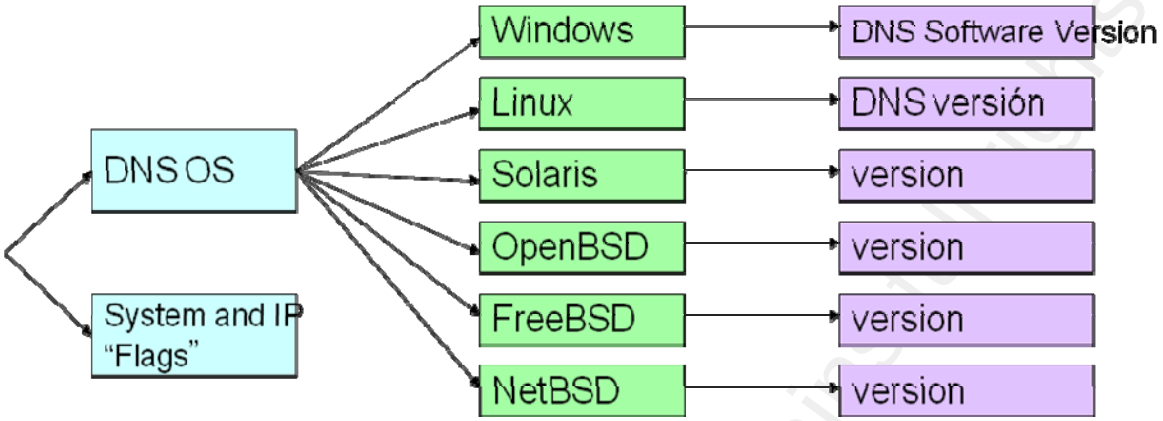
Implementing Procedure

Wenke Lee, Sal Stolfo, & Kui Mok.. (1999)
"A Data Mining Framework for Building Intrusion Detection Models",
Proceedings of the 1999 IEEE Symposium on Security and Privacy, Oakland, CA, May 1999



OS Identification = OS Detection = OS Fingerprinting

This was a critical step successfully identifying an operating system and DNS Server software type. This process utilized the Nmap (and P0F) signature databases with additions from local system testing. In this case a signature is a set of rules describing how a specific version / edition of an OS respond to the tests. This led to an analysis of differences between TCP/IP stack, an analysis of application layer data (e.g. DNS packets) and was used to refine detection of DNS Software versions.



Neural Network Inputs

Assign a set of inputs neurons for each test

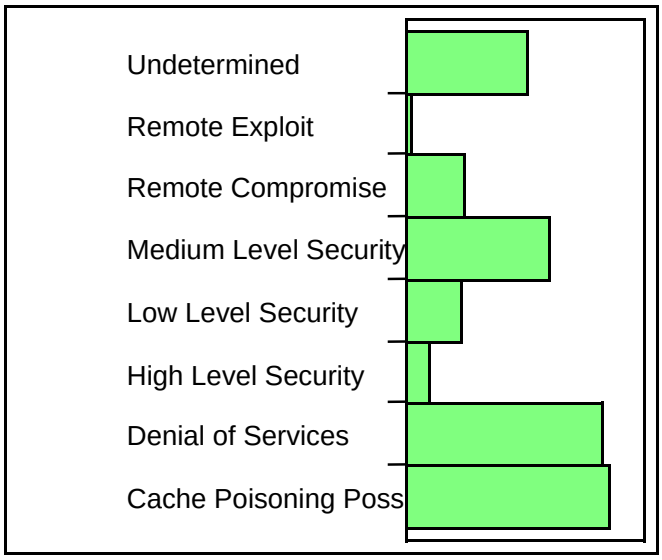
Details for tests T1 … Tx:

- one neuron for ACK flag
- one neuron per response:
- one neuron for DF flag
- one neuron per response: yes/no
- one neuron for Flags field
- one neuron for each flag: Using DNS Field options
- Multiple groups of neurons for Options fields
- one neuron in each group according to the options

  e.g.
- one neuron for W field (window size)
- One Neuron for each IP "options"

Appendix - Statistics

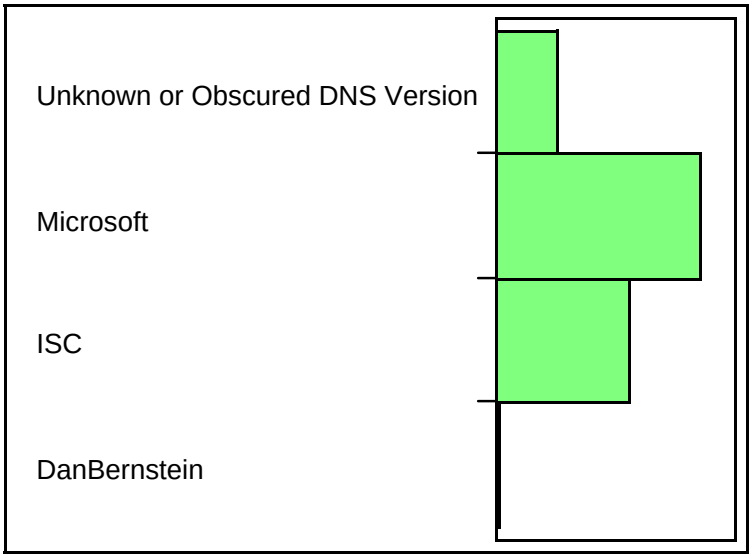
Distributions by Security Rating



Frequencies

Level	Count	Probability
Cache Poisoning Possible	1143	0.25215
Denial of Services	1098	0.24222
High Level Security	134	0.02956
Low Level Security	314	0.06927
Medium Level Security	801	0.17670
Remote Compromise	323	0.07126
Remote Exploit	39	0.00860
Undetermined	681	0.15023
Total	**4533**	**1.00000**

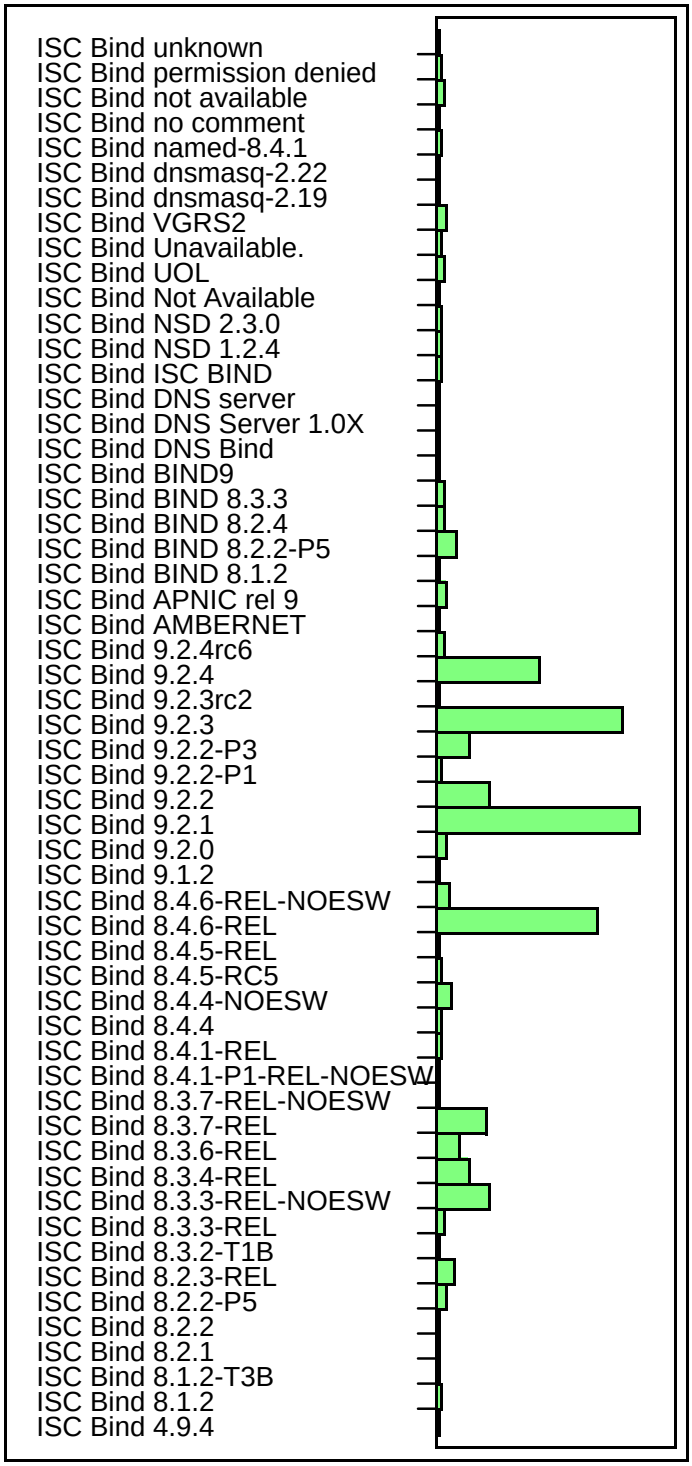
2005 Distributions by DNS Vendor



Frequencies

Level	Count	Probability
DanBernstein	17	0.00375
ISC	1518	0.33488
Microsoft	2317	0.51114
Unknown or Obscured DNS Version	681	0.15023
Total	**4533**	**1.00000**

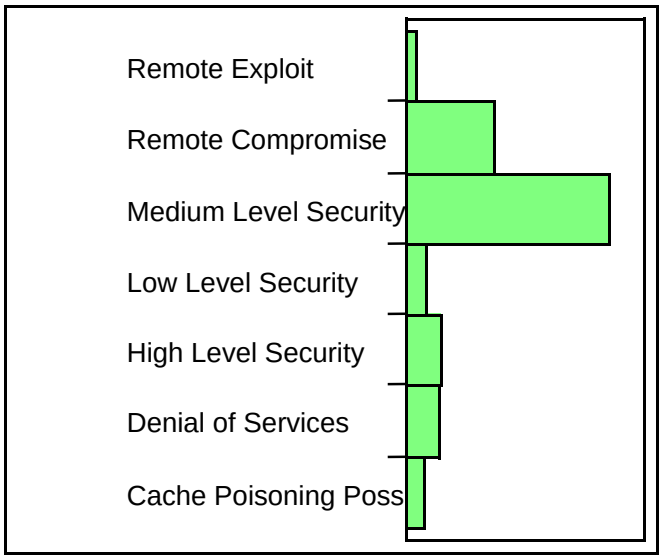
DNS Vendor = ISC (Distributions of DNS Version)



2005 Frequencies

Level	Count	Probability
ISC Bind 4.9.4	3	0.00198
ISC Bind 8.1.2	9	0.00593
ISC Bind 8.1.2-T3B	5	0.00329
ISC Bind 8.2.1	6	0.00395
ISC Bind 8.2.2	5	0.00329
ISC Bind 8.2.2-P5	14	0.00922
ISC Bind 8.2.3-REL	24	0.01581
ISC Bind 8.3.2-T1B	2	0.00132
ISC Bind 8.3.3-REL	10	0.00659
ISC Bind 8.3.3-REL-NOESW	69	0.04545
ISC Bind 8.3.4-REL	41	0.02701
ISC Bind 8.3.6-REL	29	0.01910
ISC Bind 8.3.7-REL	63	0.04150
ISC Bind 8.3.7-REL-NOESW	2	0.00132
ISC Bind 8.4.1-P1-REL-NOESW	2	0.00132
ISC Bind 8.4.1-REL	7	0.00461
ISC Bind 8.4.4	7	0.00461
ISC Bind 8.4.4-NOESW	21	0.01383
ISC Bind 8.4.5-RC5	9	0.00593
ISC Bind 8.4.5-REL	5	0.00329
ISC Bind 8.4.6-REL	204	0.13439
ISC Bind 8.4.6-REL-NOESW	16	0.01054
ISC Bind 9.1.2	2	0.00132
ISC Bind 9.2.0	14	0.00922
ISC Bind 9.2.1	257	0.16930
ISC Bind 9.2.2	67	0.04414
ISC Bind 9.2.2-P1	7	0.00461
ISC Bind 9.2.2-P3	42	0.02767
ISC Bind 9.2.3	235	0.15481
ISC Bind 9.2.3rc2	5	0.00329
ISC Bind 9.2.4	131	0.08630
ISC Bind 9.2.4rc6	12	0.00791
ISC Bind AMBERNET	2	0.00132
ISC Bind APNIC rel 9	14	0.00922
ISC Bind BIND 8.1.2	5	0.00329
ISC Bind BIND 8.2.2-P5	27	0.01779
ISC Bind BIND 8.2.4	10	0.00659
ISC Bind BIND 8.3.3	10	0.00659
ISC Bind BIND9	5	0.00329
ISC Bind DNS Bind	5	0.00329
ISC Bind DNS Server 1.0X	5	0.00329
ISC Bind DNS server	2	0.00132
ISC Bind ISC BIND	9	0.00593
ISC Bind NSD 1.2.4	7	0.00461
ISC Bind NSD 2.3.0	7	0.00461
ISC Bind Not Available	5	0.00329
ISC Bind UOL	10	0.00659
ISC Bind Unavailable.	7	0.00461
ISC Bind VGRS2	14	0.00922
ISC Bind dnsmasq-2.19	6	0.00395
ISC Bind dnsmasq-2.22	5	0.00329
ISC Bind named-8.4.1	7	0.00461
ISC Bind no comment	5	0.00329
ISC Bind not available	12	0.00791
ISC Bind permission denied	9	0.00593
ISC Bind unknown	5	0.00329
Total	**1518**	**1.00000**

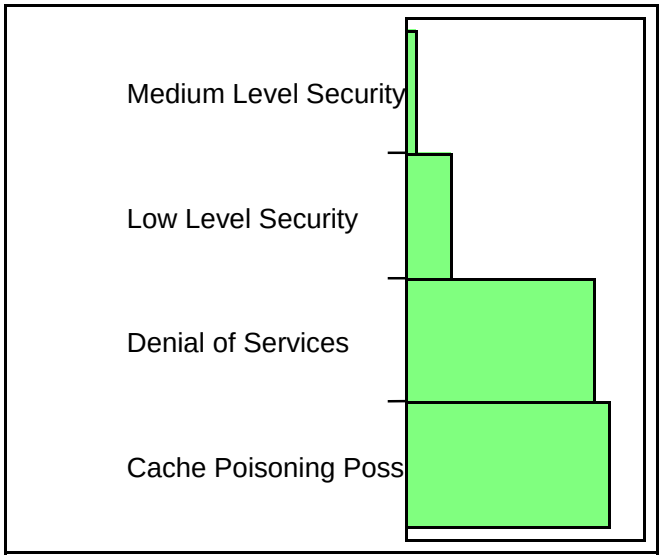
2005 Distributions by Security Rating (where DNS Vendor = ISC)



Frequencies

Level	Count	Probability
Cache Poisoning Possible	71	0.04677
Denial of Services	125	0.08235
High Level Security	134	0.08827
Low Level Security	80	0.05270
Medium Level Security	746	0.49144
Remote Compromise	323	0.21278
Remote Exploit	39	0.02569
Total	**1518**	**1.00000**

Distributions by Security Rating (where DNS Vendor = Microsoft)



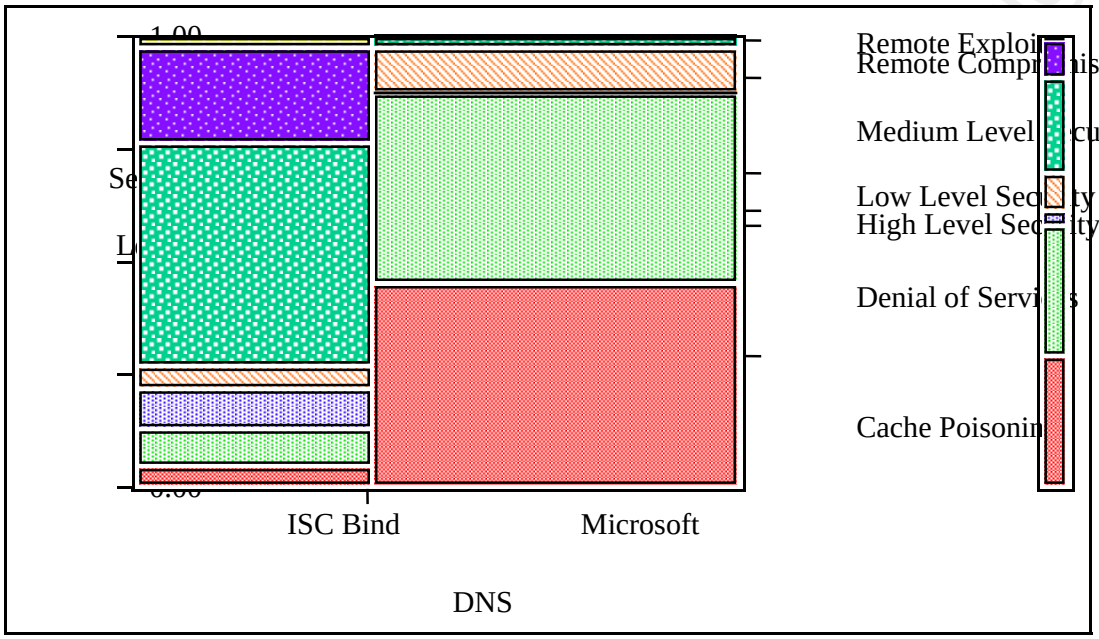
Frequencies

Level	Count	Probability
Cache Poisoning Possible	1055	0.45533
Denial of Services	973	0.41994
Low Level Security	234	0.10099
Medium Level Security	55	0.02374
Total	**2317**	**1.00000**

2005 Detailed Analysis of Security Level by DNS Vendor

Weight: No. of Servers

Mosaic Plot



Contingency Table

DNS Vendor By Security Level

Count Total % Col % Row %	Cache Poisoning	Denial of Services	High Level Security	Low Level Security	Medium Level Security	Remote Compromise	Remote Exploit	
ISC Bind	71	125	134	80	746	323	39	1518
	1.85	3.26	3.49	2.09	19.45	8.42	1.02	39.58
	6.31	11.38	100.00	25.48	93.13	100.00	100.00	
	4.68	8.23	8.83	5.27	49.14	21.28	2.57	
Microsoft	1055	973	0	234	55	0	0	2317
	27.51	25.37	0.00	6.10	1.43	0.00	0.00	60.42
	93.69	88.62	0.00	74.52	6.87	0.00	0.00	
	45.53	41.99	0.00	10.10	2.37	0.00	0.00	
	1126	1098	134	314	801	323	39	3835
	29.36	28.63	3.49	8.19	20.89	8.42	1.02	

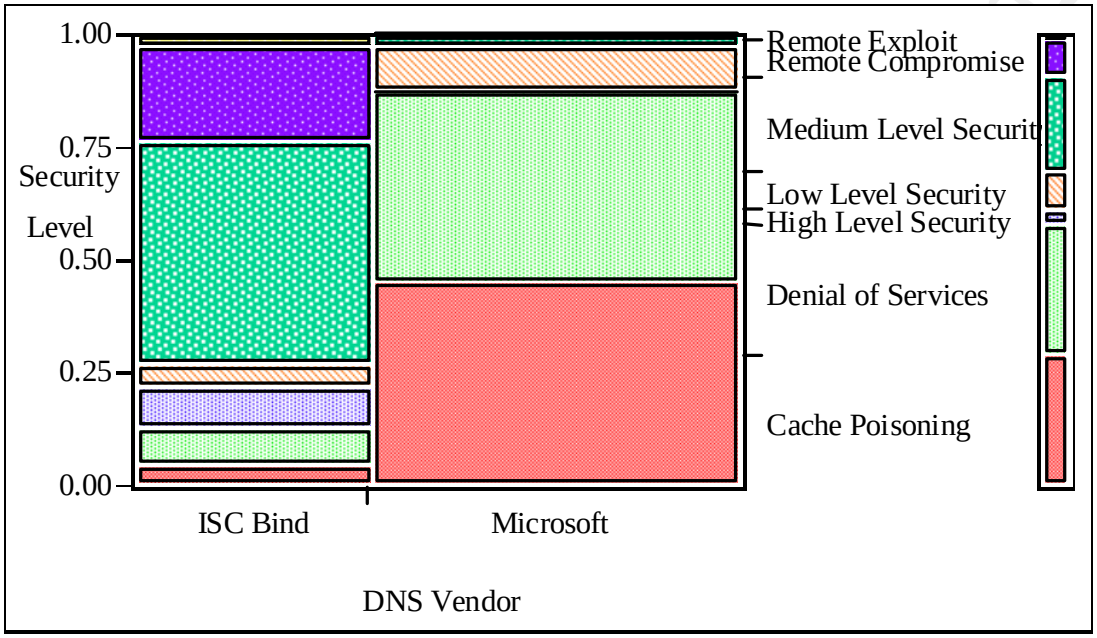
Tests

Source	DF	-Log Like	R Square (U)
Model	6	1541.6312	0.2478
Error	-1	4679.3206	
C. Total	5	6220.9518	
N	3835		

Test	Chi Square	Probability > Chi Squared
Likelihood Ratio	3083.262	0.0000
Pearson	2630.165	0.0000

2005 Contingency Analysis of Security Level by DNS Vendor

Mosaic Plot



Contingency Table

DNS Vendor by Security Level

Count Total % Col % Row %	Insecure	Secure	
ISC	558 14.55 21.58 36.76	960 25.03 76.86 63.24	1518 39.58
Microsoft	2028 52.88 78.42 87.53	289 7.54 23.14 12.47	2317 60.42
	2586 67.43	1249 32.57	3835

Tests

Source	DF	-Log Like	R Square (U)
Model	1	550.1054	0.2273
Error	3833	1870.0874	
C. Total	3834	2420.1927	
N	3835		

Test	Chi Square	Probability>Chi Square
Likelihood Ratio	1100.211	<.0001
Pearson	1076.350	<.0001

Test	Chi Square	Probability>Chi Square

Fisher's Exact Test	Probability of Null Hypothesis	Alternative Hypothesis
Left	P <.0001	Probability(Security Level = Secure) is greater for DNS Vendor (**ISC**) than (**Microsoft**)
Right	P =1.0000	Probability (Security Level = Secure) is greater for DNS Vendor (**Microsoft**) than (**ISC**)

Confidence Intervals

Level	**Count**	**Probability**	**Lower CI**	**Upper CI**	**1-Alpha**
Cache Poisoning Possible	1143	0.25215	0.235908	0.269118	0.990
Denial of Services	1098	0.24222	0.226217	0.258984	
High Level Security	134	0.02956	0.023737	0.03676	
Low Level Security	314	0.06927	0.060172	0.079627	
Medium Level Security	801	0.17670	0.162587	0.191766	
Remote Compromise	323	0.07126	0.062027	0.081737	
Remote Exploit	39	0.00860	0.005719	0.012925	
Undetermined	681	0.15023	0.137074	0.164412	

Level	Count	Probability	Lower CI	Upper CI	1-Alpha
DanBernstein	17	0.00375	0.002029	0.006922	0.990
ISC	1518	0.33488	0.317075	0.353163	
Microsoft	2317	0.51114	0.492014	0.530235	
Unknown or Obscured DNS Version	681	0.15023	0.137074	0.164412	