Network Working Group
Request for Comments: 1912
Obsoletes: 1537
Category: Informational

D. Barr
The Pennsylvania State University
February 1996

Common DNS Operational and Configuration Errors

Status of this Memo

Abstract

This memo describes errors often found in both the operation of Domain Name System (DNS) servers, and in the data that these DNS servers contain. This memo tries to summarize current Internet requirements as well as common practice in the operation and configuration of the DNS. This memo also tries to summarize or expand upon issues raised in [RFC 1537].

1. Introduction

Running a nameserver is not a trivial task. There are many things that can go wrong, and many decisions have to be made about what data to put in the DNS and how to set up servers. This memo attempts to address many of the common mistakes and pitfalls that are made in DNS data as well as in the operation of nameservers. Discussions are also made regarding some other relevant issues such as server or resolver bugs, and a few political issues with respect to the operation of DNS on the Internet.

2. DNS Data

This section discusses problems people typically have with the DNS data in their nameserver, as found in the zone data files that the nameserver loads into memory.

2.1 Inconsistent, Missing, or Bad Data

Every Internet-reachable host should have a name. The consequences of this are becoming more and more obvious. Many services available on the Internet will not talk to you if you aren't correctly registered in the DNS.

Make sure your PTR and A records match. For every IP address, there should be a matching PTR record in the in-addr.arpa domain. If a host is multi-homed, (more than one IP address) make sure that all IP addresses have a corresponding PTR record (not just the first one). Failure to have matching PTR and A records can cause loss of Internet services similar to not being registered in the DNS at all. Also, PTR records must point back to a valid A record, not a alias defined by a CNAME. It is highly recommended that you use some software which automates this checking, or generate your DNS data from a database which automatically creates consistent data.

DNS domain names consist of "labels" separated by single dots. The DNS is very liberal in its rules for the allowable characters in a domain name. However, if a domain name is used to name a host, it should follow rules restricting host names. Further if a name is used for mail, it must follow the naming rules for names in mail addresses.

Allowable characters in a label for a host name are only ASCII letters, digits, and the ‘-’ character. Labels may not be all numbers, but may have a leading digit (e.g., 3com.com). Labels must end and begin only with a letter or digit. See [RFC 1035] and [RFC 1123]. (Labels were initially restricted in [RFC 1035] to start with a letter, and some older hosts still reportedly have problems with the relaxation in [RFC 1123].) Note there are some Internet hostnames which violate this rule (411.org, 1776.com). The presence of underscores in a label is allowed in [RFC 1033], except [RFC 1033] is informational only and was not defining a standard. There is at least one popular TCP/IP implementation which currently refuses to talk to hosts named with underscores in them. It must be noted that the language in [1035] is such that these rules are voluntary -- they are there for those who wish to minimize problems. Note that the rules for Internet host names also apply to hosts and addresses used in SMTP (See RFC 821).

If a domain name is to be used for mail (not involving SMTP), it must follow the rules for mail in [RFC 822], which is actually more liberal than the above rules. Labels for mail can be any ASCII character except "specials", control characters, and whitespace characters. "Specials" are specific symbols used in the parsing of addresses. They are the characters "()<>@,;:\".[]". (The "!" character wasn’t in [RFC 822], however it also shouldn’t be used due to the conflict with UUCP mail as defined in RFC 976) However, since today almost all names which are used for mail on the Internet are also names used for hostnames, one rarely sees addresses using these relaxed standard, but mail software should be made liberal and robust enough to accept them.

You should also be careful to not have addresses which are valid alternate syntaxes to the inet_ntoa() library call. For example 0xe is a valid name, but if you were to type "telnet 0xe", it would try to connect to IP address 0.0.0.14. It is also rumored that there exists some broken inet_ntoa() routines that treat an address like x400 as an IP address.

Certain operating systems have limitations on the length of their own hostname. While not strictly of issue to the DNS, you should be aware of your operating system's length limits before choosing the name of a host.

Remember that many resource records (abbreviated RR) take on more than one argument. HINFO requires two arguments, as does RP. If you don't supply enough arguments, servers sometime return garbage for the missing fields. If you need to include whitespace within any data, you must put the string in quotes.

2.2 SOA records

In the SOA record of every zone, remember to fill in the e-mail address that will get to the person who maintains the DNS at your site (commonly referred to as "hostmaster"). The '@' in the e-mail must be replaced by a '.' first. Do not try to put an '@' sign in this address. If the local part of the address already contains a '.' (e.g., John.Smith@widget.xx), then you need to quote the '.' by preceding it with '\' character. (e.g., to become John\.Smith.widget.xx) Alternately (and preferred), you can just use the generic name 'hostmaster', and use a mail alias to redirect it to the appropriate persons. There exists software which uses this field to automatically generate the e-mail address for the zone contact. This software will break if this field is improperly formatted. It is imperative that this address get to one or more real persons, because it is often used for everything from reporting bad DNS data to reporting security incidents.

Even though some BIND versions allow you to use a decimal in a serial number, don't. A decimal serial number is converted to an unsigned 32-bit integer internally anyway. The formula for a n.m serial number is n*10^(3+int(0.9+log10(m))) + m which translates to something rather unexpected. For example it's routinely possible with a decimal serial number (perhaps automatically generated by SCCS) to be incremented such that it is numerically larger, but after the above conversion yield a serial number which is LOWER than before. Decimal serial numbers have been officially deprecated in recent BIND versions. The recommended syntax is YYYYMMDDnn (YYYY=year, MM=month, DD=day, nn=revision number. This won't overflow until the year 4294.

Choose logical values for the timer values in the SOA record (note values below must be expressed as seconds in the zone data):

Refresh: How often a secondary will poll the primary server to see if the serial number for the zone has increased (so it knows to request a new copy of the data for the zone). Set this to how long your secondaries can comfortably contain out-of-date data. You can keep it short (20 mins to 2 hours) if you aren't worried about a small increase in bandwidth used, or longer (2-12 hours) if your Internet connection is slow or is started on demand. Recent BIND versions (4.9.3) have optional code to automatically notify secondaries that data has changed, allowing you to set this TTL to a long value (one day, or more).

Retry: If a secondary was unable to contact the primary at the last refresh, wait the retry value before trying again. This value isn't as important as others, unless the secondary is on a distant network from the primary or the primary is more prone to outages. It's typically some fraction of the refresh interval.

Expire: How long a secondary will still treat its copy of the zone data as valid if it can't contact the primary. This value should be greater than how long a major outage would typically last, and must be greater than the minimum and retry intervals, to avoid having a secondary expire the data before it gets a chance to get a new copy. After a zone is expired a secondary will still continue to try to contact the primary, but it will no longer provide nameservice for the zone. 2-4 weeks are suggested values.

Minimum: The default TTL (time-to-live) for resource records -- how long data will remain in other nameservers' cache. ([RFC 1035] defines this to be the minimum value, but servers seem to always implement this as the default value) This is by far the most important timer. Set this as large as is comfortable given how often you update your nameserver. If you plan to make major changes, it's a good idea to turn this value down temporarily beforehand. Then wait the previous minimum value, make your changes, verify their correctness, and turn this value back up. 1-5 days are typical values. Remember this value can be overridden on individual resource records.

As you can see, the typical values above for the timers vary widely. Popular documentation like [RFC 1033] recommended a day for the minimum TTL, which is now considered too low except for zones with data that vary regularly. Once a DNS stabilizes, values on the order of 3 or more days are recommended. It is also recommended that you individually override the TTL on certain RRs which are often referenced and don’t often change to have very large values (1-2 weeks). Good examples of this are the MX, A, and PTR records of your mail host(s), the NS records of your zone, and the A records of your nameservers.

2.3 Glue A Records

Glue records are A records that are associated with NS records to provide "bootstrapping" information to the nameserver. For example:

```
        podunk.xx.       in      ns      ns1.podunk.xx.
                         in      ns      ns2.podunk.xx.
        ns1.podunk.xx.   in      a       1.2.3.4
        ns2.podunk.xx.   in      a       1.2.3.5
```

Here, the A records are referred to as "Glue records".

Glue records are required only in forward zone files for nameservers that are located in the subdomain of the current zone that is being delegated. You shouldn’t have any A records in an in-addr.arpa zone file (unless you’re using RFC 1101-style encoding of subnet masks).

If your nameserver is multi-homed (has more than one IP address), you must list all of its addresses in the glue to avoid cache inconsistency due to differing TTL values, causing some lookups to not find all addresses for your nameserver.

Some people get in the bad habit of putting in a glue record whenever they add an NS record "just to make sure". Having duplicate glue records in your zone files just makes it harder when a nameserver moves to a new IP address, or is removed. You’ll spend hours trying to figure out why random people still see the old IP address for some host, because someone forgot to change or remove a glue record in some other file. Newer BIND versions will ignore these extra glue records in local zone files.

Older BIND versions (4.8.3 and previous) have a problem where it inserts these extra glue records in the zone transfer data to secondaries. If one of these glues is wrong, the error can be propagated to other nameservers. If two nameservers are secondaries for other zones of each other, it’s possible for one to continually pass old glue records back to the other. The only way to get rid of the old data is to kill both of them, remove the saved backup files, and restart them. Combined with that those same versions also tend to become infected more easily with bogus data found in other non-secondary nameservers (like the root zone data).

2.4 CNAME records

A CNAME record is not allowed to coexist with any other data. In other words, if suzy.podunk.xx is an alias for sue.podunk.xx, you can’t also have an MX record for suzy.podunk.edu, or an A record, or even a TXT record. Especially do not try to combine CNAMEs and NS records like this!:

```
        podunk.xx.      IN      NS      ns1
                        IN      NS      ns2
                        IN      CNAME   mary
        mary            IN      A       1.2.3.4
```

This is often attempted by inexperienced administrators as an obvious way to allow your domain name to also be a host. However, DNS servers like BIND will see the CNAME and refuse to add any other resources for that name. Since no other records are allowed to coexist with a CNAME, the NS entries are ignored. Therefore all the hosts in the podunk.xx domain are ignored as well!

If you want to have your domain also be a host, do the following:

```
        podunk.xx.      IN      NS      ns1
                        IN      NS      ns2
                        IN      A       1.2.3.4
        mary            IN      A       1.2.3.4
```

Don’t go overboard with CNAMEs. Use them when renaming hosts, but plan to get rid of them (and inform your users). However CNAMEs are useful (and encouraged) for generalized names for servers -- ‘ftp’ for your ftp server, ‘www’ for your Web server, ‘gopher’ for your Gopher server, ‘news’ for your Usenet news server, etc.

Don’t forget to delete the CNAMEs associated with a host if you delete the host it is an alias for. Such "stale CNAMEs" are a waste of resources.

Don’t use CNAMEs in combination with RRs which point to other names like MX, CNAME, PTR and NS. (PTR is an exception if you want to implement classless in-addr delegation.) For example, this is strongly discouraged:

```
        podunk.xx.      IN      MX      mailhost
        mailhost        IN      CNAME   mary
        mary            IN      A       1.2.3.4
```

[RFC 1034] in section 3.6.2 says this should not be done, and [RFC 974] explicitly states that MX records shall not point to an alias defined by a CNAME. This results in unnecessary indirection in accessing the data, and DNS resolvers and servers need to work more to get the answer. If you really want to do this, you can accomplish the same thing by using a preprocessor such as m4 on your host files.

Also, having chained records such as CNAMEs pointing to CNAMEs may make administration issues easier, but is known to tickle bugs in some resolvers that fail to check loops correctly. As a result some hosts may not be able to resolve such names.

Having NS records pointing to a CNAME is bad and may conflict badly with current BIND servers. In fact, current BIND implementations will ignore such records, possibly leading to a lame delegation. There is a certain amount of security checking done in BIND to prevent spoofing DNS NS records. Also, older BIND servers reportedly will get caught in an infinite query loop trying to figure out the address for the aliased nameserver, causing a continuous stream of DNS requests to be sent.

2.5 MX records

It is a good idea to give every host an MX record, even if it points to itself! Some mailers will cache MX records, but will always need to check for an MX before sending mail. If a site does not have an MX, then every piece of mail may result in one more resolver query, since the answer to the MX query often also contains the IP addresses of the MX hosts. Internet SMTP mailers are required by [RFC 1123] to support the MX mechanism.

Put MX records even on hosts that aren’t intended to send or receive e-mail. If there is a security problem involving one of these hosts, some people will mistakenly send mail to postmaster or root at the site without checking first to see if it is a "real" host or just a terminal or personal computer that’s not set up to accept e-mail. If you give it an MX record, then the e-mail can be redirected to a real person. Otherwise mail can just sit in a queue for hours or days until the mailer gives up trying to send it.

Don’t forget that whenever you add an MX record, you need to inform the target mailer if it is to treat the first host as "local". (The "Cw" flag in sendmail, for example)

If you add an MX record which points to an external host (e.g., for the purposes of backup mail routing) be sure to ask permission from that site first. Otherwise that site could get rather upset and take action (like throw your mail away, or appeal to higher authorities like your parent DNS administrator or network provider.)

2.6 Other Resource Records

2.6.1 WKS

WKS records are deprecated in [RFC 1123]. They serve no known useful function, except internally among LISP machines. Don’t use them.

2.6.2 HINFO

On the issue HINFO records, some will argue that these is a security problem (by broadcasting what vendor hardware and operating system you so people can run systematic attacks on known vendor security holes). If you do use them, you should keep up to date with known vendor security problems. However, they serve a useful purpose. Don’t forget that HINFO requires two arguments, the hardware type, and the operating system.

HINFO is sometimes abused to provide other information. The record is meant to provide specific information about the machine itself. If you need to express other information about the host in the DNS, use TXT.

2.6.3 TXT

TXT records have no specific definition. You can put most anything in them. Some use it for a generic description of the host, some put specific information like its location, primary user, or maybe even a phone number.

2.6.4 RP

RP records are relatively new. They are used to specify an e-mail address (see first paragraph of section 2.2) of the "Responsible Person" of the host, and the name of a TXT record where you can get more information. See [RFC 1183].

2.7 Wildcard records

Wildcard MXs are useful mostly for non IP-connected sites. A common mistake is thinking that a wildcard MX for a zone will apply to all hosts in the zone. A wildcard MX will apply only to names in the zone which aren't listed in the DNS at all. e.g.,

```
        podunk.xx.      IN      NS      ns1
                        IN      NS      ns2
        mary            IN      A       1.2.3.4
        *.podunk.xx.    IN      MX      5 sue
```

Mail for mary.podunk.xx will be sent to itself for delivery. Only mail for jane.podunk.xx or any hosts you don't see above will be sent to the MX. For most Internet sites, wildcard MX records are not useful. You need to put explicit MX records on every host.

Wildcard MXs can be bad, because they make some operations succeed when they should fail instead. Consider the case where someone in the domain "widget.com" tries to send mail to "joe@larry". If the host "larry" doesn't actually exist, the mail should in fact bounce immediately. But because of domain searching the address gets resolved to "larry.widget.com", and because of the wildcard MX this is a valid address according to DNS. Or perhaps someone simply made a typo in the hostname portion of the address. The mail message then gets routed to the mail host, which then rejects the mail with strange error messages like "I refuse to talk to myself" or "Local configuration error".

Wildcard MX records are good for when you have a large number of hosts which are not directly Internet-connected (for example, behind a firewall) and for administrative or political reasons it is too difficult to have individual MX records for every host, or to force all e-mail addresses to be "hidden" behind one or more domain names. In that case, you must divide your DNS into two parts, an internal DNS, and an external DNS. The external DNS will have only a few hosts and explicit MX records, and one or more wildcard MXs for each internal domain. Internally the DNS will be complete, with all explicit MX records and no wildcards.

Wildcard As and CNAMEs are possible too, and are really confusing to users, and a potential nightmare if used without thinking first. It could result (due again to domain searching) in any telnet/ftp attempts from within the domain to unknown hosts to be directed to one address. One such wildcard CNAME (in *.edu.com) caused Internet-wide loss of services and potential security nightmares due to unexpected interactions with domain searching. It resulted in swift fixes, and even an RFC ([RFC 1535]) documenting the problem.

2.8 Authority and Delegation Errors (NS records)

You are required to have at least two nameservers for every domain, though more is preferred. Have secondaries outside your network. If the secondary isn't under your control, periodically check up on them and make sure they're getting current zone data from you. Queries to their nameserver about your hosts should always result in an "authoritative" response. If not, this is called a "lame delegation". A lame delegations exists when a nameserver is delegated responsibility for providing nameservice for a zone (via NS records) but is not performing nameservice for that zone (usually because it is not set up as a primary or secondary for the zone).

The "classic" lame delegation can be illustrated in this example:

```
        podunk.xx.      IN      NS      ns1.podunk.xx.
                        IN      NS      ns0.widget.com.
```

"podunk.xx" is a new domain which has recently been created, and "ns1.podunk.xx" has been set up to perform nameservice for the zone. They haven't quite finished everything yet and haven't made sure that the hostmaster at "ns0.widget.com" has set up to be a proper secondary, and thus has no information about the podunk.xx domain, even though the DNS says it is supposed to. Various things can happen depending on which nameserver is used. At best, extra DNS traffic will result from a lame delegation. At worst, you can get unresolved hosts and bounced e-mail.

Also, sometimes a nameserver is moved to another host or removed from the list of secondaries. Unfortunately due to caching of NS records, many sites will still think that a host is a secondary after that host has stopped providing nameservice. In order to prevent lame delegations while the cache is being aged, continue to provide nameservice on the old nameserver for the length of the maximum of the minimum plus refresh times for the zone and the parent zone. (See section 2.2)

Whenever a primary or secondary is removed or changed, it takes a fair amount of human coordination among the parties involved. (The site itself, it's parent, and the site hosting the secondary) When a primary moves, make sure all secondaries have their named.boot files updated and their servers reloaded. When a secondary moves, make sure the address records at both the primary and parent level are changed.

It's also been reported that some distant sites like to pick popular nameservers like "ns.uu.net" and just add it to their list of NS records in hopes that they will magically perform additional nameservice for them. This is an even worse form of lame delegation, since this adds traffic to an already busy nameserver. Please contact the hostmasters of sites which have lame delegations. Various tools can be used to detect or actively find lame delegations. See the list of contributed software in the BIND distribution.

Make sure your parent domain has the same NS records for your zone as you do. (Don't forget your in-addr.arpa zones too!). Do not list too many (7 is the recommended maximum), as this just makes things harder to manage and is only really necessary for very popular top-level or root zones. You also run the risk of overflowing the 512-byte limit of a UDP packet in the response to an NS query. If this happens, resolvers will "fall back" to using TCP requests, resulting in increased load on your nameserver.

It's important when picking geographic locations for secondary nameservers to minimize latency as well as increase reliability. Keep in mind network topologies. For example if your site is on the other end of a slow local or international link, consider a secondary on the other side of the link to decrease average latency. Contact your Internet service provider or parent domain contact for more information about secondaries which may be available to you.

3. BIND operation

This section discusses common problems people have in the actual operation of the nameserver (specifically, BIND). Not only must the data be correct as explained above, but the nameserver must be operated correctly for the data to be made available.

3.1 Serial numbers

Each zone has a serial number associated with it. Its use is for keeping track of who has the most current data. If and only if the primary's serial number of the zone is greater will the secondary ask the primary for a copy of the new zone data (see special case below).

Don't forget to change the serial number when you change data! If you don't, your secondaries will not transfer the new zone information. Automating the incrementing of the serial number with software is also a good idea.

If you make a mistake and increment the serial number too high, and you want to reset the serial number to a lower value, use the following procedure:

Take the ‘incorrect’ serial number and add 2147483647 to it. If the number exceeds 4294967296, subtract 4294967296. Load the resulting number. Then wait 2 refresh periods to allow the zone to propagate to all servers.

Repeat above until the resulting serial number is less than the target serial number.

Up the serial number to the target serial number.

This procedure won’t work if one of your secondaries is running an old version of BIND (4.8.3 or earlier). In this case you’ll have to contact the hostmaster for that secondary and have them kill the secondary servers, remove the saved backup file, and restart the server. Be careful when editing the serial number -- DNS admins don’t like to kill and restart nameservers because you lose all that cached data.

3.2 Zone file style guide

Here are some useful tips in structuring your zone files. Following these will help you spot mistakes, and avoid making more.

Be consistent with the style of entries in your DNS files. If your $ORIGIN is podunk.xx., try not to write entries like:

```
        mary            IN      A       1.2.3.1
        sue.podunk.xx.  IN      A       1.2.3.2
```

or:

```
        bobbi           IN      A       1.2.3.2
                        IN      MX      mary.podunk.xx.
```

Either use all FQDNs (Fully Qualified Domain Names) everywhere or used unqualified names everywhere. Or have FQDNs all on the right-hand side but unqualified names on the left. Above all, be consistent.

Use tabs between fields, and try to keep columns lined up. It makes it easier to spot missing fields (note some fields such as "IN" are inherited from the previous record and may be left out in certain circumstances.)

Remember you don’t need to repeat the name of the host when you are defining multiple records for one host. Be sure also to keep all records associated with a host together in the file. It will make things more straightforward when it comes time to remove or rename a host.

Always remember your $ORIGIN. If you don’t put a ‘.’ at the end of an FQDN, it’s not recognized as an FQDN. If it is not an FQDN, then the nameserver will append $ORIGIN to the name. Double check, triple check, those trailing dots, especially in in-addr.arpa zone files, where they are needed the most.

Be careful with the syntax of the SOA and WKS records (the records which use parentheses). BIND is not very flexible in how it parses these records. See the documentation for BIND.

3.3 Verifying data

Verify the data you just entered or changed by querying the resolver with dig (or your favorite DNS tool, many are included in the BIND distribution) after a change. A few seconds spent double checking can save hours of trouble, lost mail, and general headaches. Also be sure to check syslog output when you reload the nameserver. If you have grievous errors in your DNS data or boot file, named will report it via syslog.

It is also highly recommended that you automate this checking, either with software which runs sanity checks on the data files before they are loaded into the nameserver, or with software which checks the data already loaded in the nameserver. Some contributed software to do this is included in the BIND distribution.

4. Miscellaneous Topics

4.1 Boot file setup

Certain zones should always be present in nameserver configurations:

```
        primary         localhost               localhost
        primary         0.0.127.in-addr.arpa    127.0
        primary         255.in-addr.arpa        255
        primary         0.in-addr.arpa          0
```

These are set up to either provide nameservice for "special" addresses, or to help eliminate accidental queries for broadcast or local address to be sent off to the root nameservers. All of these files will contain NS and SOA records just like the other zone files you maintain, the exception being that you can probably make the SOA timers very long, since this data will never change.

The "localhost" address is a "special" address which always refers to the local host. It should contain the following line:

```
        localhost.      IN      A       127.0.0.1
```

The "127.0" file should contain the line:

```
        1    PTR     localhost.
```

There has been some extensive discussion about whether or not to append the local domain to it. The conclusion is that "localhost." would be the best solution. The reasons given include:

"localhost" by itself is used and expected to work in some systems.

Translating 127.0.0.1 into "localhost.dom.ain" can cause some software to connect back to the loopback interface when it didn’t want to because "localhost" is not equal to "localhost.dom.ain".

The "255" and "0" files should not contain any additional data beyond the NS and SOA records.

Note that future BIND versions may include all or some of this data automatically without additional configuration.

4.2 Other Resolver and Server bugs

Very old versions of the DNS resolver have a bug that cause queries for names that look like IP addresses to go out, because the user supplied an IP address and the software didn’t realize that it didn’t need to be resolved. This has been fixed but occasionally it still pops up. It’s important because this bug means that these queries will be sent directly to the root nameservers, adding to an already heavy DNS load.

While running a secondary nameserver off another secondary nameserver is possible, it is not recommended unless necessary due to network topologies. There are known cases where it has led to problems like bogus TTL values. While this may be caused by older or flawed DNS implementations, you should not chain secondaries off of one another since this builds up additional reliability dependencies as well as adds additional delays in updates of new zone data.

4.3 Server issues

DNS operates primarily via UDP (User Datagram Protocol) messages. Some UNIX operating systems, in an effort to save CPU cycles, run with UDP checksums turned off. The relative merits of this have long been debated. However, with the increase in CPU speeds, the performance considerations become less and less important. It is strongly encouraged that you turn on UDP checksumming to avoid corrupted data not only with DNS but with other services that use UDP (like NFS). Check with your operating system documentation to verify that UDP checksumming is enabled.

References

[RFC 974] Partridge, C., "Mail routing and the domain system", STD 14, RFC 974, CSNET CIC BBN Laboratories Inc, January 1986.

[RFC 1033] Lottor, M, "Domain Administrators Operations Guide", RFC 1033, USC/Information Sciences Institute, November 1987.

[RFC 1034] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, USC/Information Sciences Institute, November 1987.

[RFC 1035] Mockapetris, P., "Domain Names - Implementation and Specification", STD 13, RFC 1035, USC/Information Sciences Institute, November 1987.

[RFC 1123] Braden, R., "Requirements for Internet Hosts -- Application and Support", STD 3, RFC 1123, IETF, October 1989.

[RFC 1178] Libes, D., "Choosing a Name for Your Computer", FYI 5, RFC 1178, Integrated Systems Group/NIST, August 1990.

[RFC 1183] Ullman, R., Mockapetris, P., Mamakos, L, and C. Everhart, "New DNS RR Definitions", RFC 1183, October 1990.

[RFC 1535] Gavron, E., "A Security Problem and Proposed Correction With Widely Deployed DNS Software", RFC 1535, ACES Research Inc., October 1993.

[RFC 1536] Kumar, A., Postel, J., Neuman, C., Danzig, P., and S. Miller, "Common DNS Implementation Errors and Suggested Fixes", RFC 1536, USC/Information Sciences Institute, USC, October 1993.

[RFC 1537] Beertema, P., "Common DNS Data File Configuration Errors", RFC 1537, CWI, October 1993.

[RFC 1713] A. Romao, "Tools for DNS debugging", RFC 1713, FCCN, November 1994.

[BOG] Vixie, P, et. al., "Name Server Operations Guide for BIND", Vixie Enterprises, July 1994.

5. Security Considerations

Security issues are not discussed in this memo.

6. Author's Address

David Barr
The Pennsylvania State University
Department of Mathematics
334 Whitmore Building
University Park, PA 16802

Voice: +1 814 863 7374
Fax: +1 814 863-8311
EMail: barr@math.psu.edu