# Dealing with Lame Delegations

*Bryan Beecher* – University of Michigan

## ABSTRACT

The University of Michigan is a large, research university with over six thousand computers attached to six interconnected class B networks. The ownership and administration of these machines is widely distributed across the university, and consequently the U-M Domain Name System namespace is also quite distributed. While this keeps the workload low for any single campus hostmaster, it also introduces many possibilities for domain nameserver misconfiguration.

For some time we have run a version of the domain nameserver daemon, **named**, which includes Don Lewis' *lame delegation patch.* This patch uses syslog to generate an alert whenever *named* encounters a lame delegation, that is, an instance where a nameserver is listed as authoritative for a domain, but in fact is not performing service for that domain. We have taken this idea one step further by running a weekly job that collects these alerts, does its best to screen out any spurious ones, and then notifies the owner of the domain. This paper summarizes and discusses this work.

### Background

The University of Michigan is a large and sprawling collection of TCP/IP networks, comprising half a dozen class B networks (141.211.0.0 through 141.216.0.0). The Domain Name System (DNS) entry for the University of Michigan is **UMICH.EDU**, and like the University's IP networks, it too is large and sprawling. Each college, institute, or campus-wide entity is entitled to a domain at the **UMICH.EDU** level. For example, the College of Literature, Science, and the Arts uses the **LSA.UMICH.EDU** domain. Departments within colleges then fall under that college's domain, and so the Math Department is known as **MATH.LSA.UMICH.EDU**. All in all, there are over six thousand hosts with entries in the DNS at U-M.

It is impractical to try to centralize the domain nameservice for this many machines. Consequently, many of the sub-domains of **UMICH.EDU** have been delegated to various system administrators across campus. This makes for a very distributed system where no single hostmaster has too much work. However, it also introduces great possibilities for errors when newly hired system administrators are forced to maintain their part of the DNS when they haven't had proper training. One of the most common errors introduced is that of a *lame delegation.*

### The Problem

A *lame delegation* is an instance when a nameserver is listed as authoritative for a domain, but in fact, it is not performing service for that domain. A lame delegation is often caused at U-M when a hostmaster moves a domain nameserver from host A to host B without notifying the hostmaster of the parent domain. For example, say there are two sets of nameservers listed for **LSA.UMICH.EDU**. One set is listed in the **UMICH.EDU** domain zone file (where the domain is delegated) and the other set is listed in the **LSA.UMICH.EDU** domain zone file. Under normal operating conditions these two sets of nameservers should be identical, but if a hostmaster is careless and changes the local set without notifying his parent domain's hostmaster, they can get out of sync. In our example, if the listed nameservers for the **LSA.UMICH.EDU** domain are listed as **A.LSA.UMICH.EDU** and **B.LSA.-UMICH.EDU** in the **UMICH.EDU** zone file, but are listed as **C.LSA.UMICH.EDU** and **B.LSA.-UMICH.EDU** in the **LSA.UMICH.EDU** zone file, and the nameserver on **A.LSA.UMICH.EDU** stops serving the **LSA.UMICH.EDU** domain, then the **UMICH.EDU** zone file now contains a lame delegation.

Lame delegations are serious problems. At the very least, they cause the DNS to become much less efficient since query packets will be sent to hosts which are either not running a nameserver at all, or will be sent to nameservers which are not authoritative for a domain. In either case, the query will not be answered, and the sender will have to try another nameserver. In a more serious case, all of the nameservers listed for a domain will be lame delegations, and no queries can be answered! Interestingly, a site can be the victim of a serious lame delegation such as this without even being aware of it IF all of the resolvers on-site are configured to query a non-lame server, yet the servers listed for that site at its parent domain are all lame. Unfortunately, many excellent examples of this exist even today in the **IN-ADDR.ARPA** namespace. Here the scenario is that a site lists no nameservers for its **IN-ADDR.ARPA** domains(s) in the root servers due to ignorance or oversight, but their local nameservers do indeed serve their **IN-ADDR.ARPA** domain. And so, local queries all work since the packet is coincidentally going to a nameserver which is authoritative for that **IN-ADDR.ARPA** domain, yet all non-local queries on that **IN-ADDR.ARPA** domain fail since there are no nameservers for it listed in the root servers.

### The Solution

Don Lewis, one of the few people across the Internet contributing bug fixes and enhancements to BIND, released a patch which detects lame delegations. A lame delegation is detected when *named* forwards a query to a nameserver which was listed as an authoritative server, but the response is not marked as authoritative. A lame delegation causes a message like this to appear in the appropriate syslog log:

```
Jun 23 11:06:42 totalrecall
named[104]: Lame delegation to
'nadn.NAVY.MIL' received from
26.7.0.102 (purported server for
'NADN.NAVY.MIL') on query on name
[ward.nadn.navy.mil]
```

In this example, the namserver on totalrecall came across this lame delegation when it tried to resolve the name **WARD.NADN.NAVY.MIL**. At the time of this writing, there are three listed nameservers for the **NADN.NAVY.MIL** domains with the names: **NADN1.NADN.NAVY.MIL** (address 131.121.1.1), **USNA.USNA.NAVY.MIL** (26.7.0.102 and 128.56.1.1), and **NADN2.NADN.NAVY.MIL** (131.121.1.2). In this case, our nameserver forwarded the query on the name **WARD.NADN.NAVY.MIL** to a nameserver it was led to believe was authoritative for **NADN.NAVY.MIL**, yet in this case, that nameserver replied with non-authoritative data.

Since we started running a version of *named* with this patch, we found that we discovered hundreds of lame delegations each month. Furthermore, although some lame delegations were indeed problems with a domain nameserver at the University of Michigan, most were problems with domain nameservers located elsewhere in the Internet.

At first we ignored the non-local lame delegation messages and simply used a tool like *grep* to extract the lame delegations that were local. Yet, even this proved to be less effective than we hoped since many of the lame delegations proved to be transient problems, and there was little sense in alerting a hostmaster to a problem that no longer existed. We decided to build a tool which would screen out as many transient or spurious errors as possible,

and would then automatically alert the appropriate hostmaster via e-mail.

### The Tool

We wrote a small shell script which we run out of *cron* once per week. The script is relatively short, and we thought the best way to present it would be to simple include an annotated version of it here. It is also available via anonymous ftp from terminator.cc.umich.edu. It can be found in /dns/lamers.sh.

## The Script

The first part of the script sets up our path, identifies the location of the log file, identifies the file containing the lame delegation message we send to people, and identifies some temp files. Notice that some temp files are located on /usr/tmp rather than /tmp. Some of these files can get big as the script runs, and we found that we would fill up /tmp (which is on the / partition on our machines).

```
#!/bin/sh
PATH=:/bin:/usr/bin:/usr/ucb:/usr/local/bin
LOGFILE=/usr/spool/log/named
MAILMSG=/usr/tmp/mailmsg$$
LAMERS=/usr/tmp/lamers$$
MSGFILE=/usr/local/bin/lamer-message
LAMEREPORT=/tmp/.lamereport$$
WEEKFILE=/usr/tmp/week$$
```

The next part contains some standard information we put in anything we make available to the Internet community: A standard copyright notice, the author's name, the last change date, and some notes about the tool.

The note below also lies a little bit. You can make use of this script even if you do not have either *query* or *host*. *query* is a simple program which queries nameservers using the resolver routines available in the C library. Unlike some other packages, it does not include external resolver routines to link in; it uses the same code that the other software on the machine uses. *host* is a short shell script around *query* which, when given an IP address as an argument, returns the associated host name. Tools such as *dig* and *nslookup* would also do just fine, although the shell script would have to be modified appropriately to handle their output instead.

```
# ------------------------------------------------------------
```

```
#  Copyright (c) 1991 Regents of the University of Michigan.
#  All rights reserved.
#
#  Redistribution and use is permitted provided that this notice
#  is preserved and that due credit is given to the University of
#  Michigan. The name of the University may not be used to endorse
#  or promote products derived from this software without specific
#  prior written permission. This software is provided "as is"
#  without express or implied warranty.
#
#  Lame delegation notifier
#  Author:  Bryan Beecher
#  Last Modified:   6/25/92
#
#  To make use of this software, you need to be running the
#  University of Michigan release of BIND 4.8.3, or any version
#  of named that supports the LAME_DELEGATION patches posted to
#  USENET.  The U-M release is available via anonymous ftp from
#  terminator.cc.umich.edu:/unix/dns/bind4.8.3.tar.Z.
#
#  You must also have a copy of query(1) and host(1).  These
#  are also available via anonymous ftp in the aforementioned
#  place.
# --------------------------------------------------------------
#  handle arguments
# --------------------------------------------------------------
#       -d <day>
#       This flag is used to append a dot-day suffix to the LOGFILE.
#       Handy where log files are kept around for the last week
#       and contain a day suffix.

#       -f <logfile>
#       Change the LOGFILE value altogether.

#       -w
#       Count up all of the DNS statistics for the whole week.

#       -v
#       Be verbose.

#       -t
#       Test mode.  Do not send mail to the lame delegation
#       hostmasters.
# --------------------------------------------------------------
```

For a given service that we provide on one of our machines, we maintain one week's worth of log files, broken into eight log files: one for each previous day of the week plus one for the current day. For example, the current log file for *named* would be located in /var/log/named. And if it happens to be Tuesday, then yesterday's log file would be located in /var/log/named.Mon, the previous day's would be located in /var/log/named.Sun, and last week's log file would be in /var/log/named.Tue .

The -f and -t flags exist mainly for debugging. Before we unleashed this upon the Internet community, we wanted to make sure that we wouldn't be generating tons of unwanted mail. The -f flag is handy when you'd like to hand-generate your own lame delegation data and then use it with this script. The -t flag performs all the usual work, except it does NOT notify the hostmaster via e-mail. It does, however, still build a list of who it would have mailed had the -t flag not been specified.

```
VERBOSE=0
TESTMODE=0
while [ $# != 0 ] ; do
  case "$1" in
    -d)
    LOGFILE=$LOGFILE"."$2
```

```
      shift
      ;;
      -f)
      LOGFILE=$2
      shift
      ;;
      -w)
      cat $LOGFILE* > $WEEKFILE
      LOGFILE=$WEEKFILE
      ;;
      -v)
      VERBOSE=1
      ;;
      -t)
      TESTMODE=1
      ;;
   esac
   shift
done
```

We added the following line so that the script would clean up after itself if it was killed during a run.

```
#---------------------------------------------------------------------------
#  Clean up and exit on a HUP, INT or QUIT
#---------------------------------------------------------------------------
trap "rm -f $LAMERS $MAILMSG $LAMEREPORT $WEEKFILE ; exit" 1 2 3
```

The first thing we do is search the log to see if any lame delegations were detected. We toss out lines with an asterisk on them since those tended to be lame delegations of the form "server xxx.xxx.xxx.xxx is a lame delegation for domain *". We really aren't able to do anything with a message like that, and it isn't clear to us exactly how those are getting generated either. We also down-case everything at this point so its easier to parse and handle later.

After the initial pruning we strip off the domain name and nameserver's IP address from the line in the log file. We sort those, toss out duplicates, and write the results to a temp file. If the temp file is non-empty, we know that we found some lame delegations to handle.

```
#---------------------------------------------------------------------------
#  See if there are any lamers
#---------------------------------------------------------------------------
grep "Lame" $LOGFILE | tr A-Z a-z | grep -v "*" | awk '{
        print substr($16, 2, length($16) - 3), $12 }' |
        sort | uniq | awk '{
                printf("%s %s\n", $1, $2)
}' > $LAMERS

if [ ! -s $LAMERS ] ; then
        exit 0
fi

if [ $VERBOSE -eq 1 ] ; then
        echo "Found" `awk 'END { print NR }' $LAMERS` "lame delegations"
fi
```

The following message mentions *potential* lame delegation because we've often found that there are cases when the lame delegation patch flags a server as lame even though it does not appear to be lame by the time this script runs. It isn't clear to us if the culprit here is the lame delegation patches (which are flagging innocent nameservers as lame), *named* itself (which is giving bogus information to the lame delegation code), or if there are simply many transient errors in the domain name system.

At this point each lame delegation is identified by a unique (nameserver, domain) pair.

```
#  There were lamers; send them mail

touch $LAMEREPORT
NAME=""
while read DOMAIN IPADDR ; do
        #-------------------------------------------------------------
        # Echo args if verbose
        #-------------------------------------------------------------
        if [ $VERBOSE -eq 1 ] ; then
                echo $IPADDR "is a potential lame delegation for" $DOMAIN
        fi
```

The next thing we do is lookup the SOA record for the domain. We do this so that we can fetch an ostensibly official e-mail address to which to send the mail. In our experience the e-mail address listed in an SOA record often is syntactically incorrect, or contains some unusable address. The script isn't too careful about this, and so we end up seeing a fair number of bounces which we then handle ourselves.

```
#------------------------------------------------------------
# Lookup the SOA record form $DOMAIN.  A really broken name
# server many have more than one SOA for a domain, so exit
# after finding the first one.  Send it to the local hostmaster
# if we cannot find the proper one.
#------------------------------------------------------------
if [ $VERBOSE -eq 1 ] ; then
        echo "Looking up the hostmaster for $DOMAIN"
fi
HOSTMASTER=`query -h $DOMAIN -t SOA 2> /dev/null | \
                   awk '/mail addr/ { print $4 ; exit }' | sed -e 's/./@/'`
NAME=`host $IPADDR 2> /dev/null`
if [ -z ""$HOSTMASTER ] ; then
        if [ -z ""$NAME ] ; then
                HOSTMASTER="hostmaster"
        else
                HOSTMASTER="postmaster@"$NAME
        fi
fi
```

This is one of the tests we make to weed out the spurious lame delegations. There have been cases where a parent server has listed another nameserver as authoritative for a domain, yet that actual server does not report itself as authoritative for a domain. This is still a problem, and in the future we should do something better here than just continue (e.g., send mail to the parent domain telling them about the problem).

```
#------------------------------------------------------------
# Find the name associated with IP address $IPADDR.  Query
# the nameserver at that address:  If it responds listing
# itself as a domain namserver, then it is lame; if it isn't
# in the list, then perhaps the lame delegation alert was
# spurious.
#------------------------------------------------------------
if [ $VERBOSE -eq 1 ] ; then
        echo "Making sure that $IPADDR is listed as a NS for $DOMAIN"
fi
if [ -n ""$NAME ] ; then
        query -n $IPADDR -h $DOMAIN 2>&1 | grep "domain name" | \
                                              grep -i $NAME > /dev/null
        if [ $? -eq 1 -a $VERBOSE -eq 1 ] ; then
                echo $NAME does not seem to be a nameserver for $DOMAIN
                continue
        fi
fi
```

We query the listed nameserver twice. Even in the case of a lame delegation, it may return with authoritative data on the first query since it may have just made the query to an authoritative nameserver. If it is a lame delegation, then the second query will be from the nameserver's cache rather than from its authoritative data, and so the **aa** (authoritative answer) header flag will be missing. Some really malformed answers set all of the flags, and so if an unusual one is set, like **tc** we consider the answer to be invalid.

```
    #----------------------------------------------------------
    # If the delegation is no longer lame, don't send mail.
    # We do the query twice; the first answer could be authori-
    # tative even if the nameserver is not performing service
    # for the domain.  If this is the case, then the second
    # query will come from cached data, and will be exposed
    # on the second query.  If the resolver returns trash, the
    # entire set of flags will be set.  In this case, don't
    # count the answer as authoritative.
    #----------------------------------------------------------
    if [ $VERBOSE -eq 1 ] ; then
            echo "Making sure that $IPADDR is not providing authoritative data now"
    fi
    query -n $IPADDR -h $DOMAIN > /dev/null 2>&1
    query -n $IPADDR -h $DOMAIN 2>&1 | grep header | grep aa | \
                                                   grep -v tc > /dev/null
    if [ $? -eq 0 ] ; then
      if [ $VERBOSE -eq 1 ] ; then
        if [ -n ""$NAME ] ; then
                echo $NAME seems to be serving $DOMAIN OK now
        else
                echo $I seems to be serving $DOMAIN OK now
        fi
      fi
      continue
    fi
```

If we've reached this point, then we probably have a genuine lame delegation. We then use *sed* to do a quick substitution on a copy of the message we send out to mark in the proper domain name and namserver IP address. We then mail off the message using the name "dns-maintenance@umich.edu" in the "From:" line. That way if it bounces (or gets a reply), the message will not go to root's mailbox.

```
    #----------------------------------------------------------
    # Notify the owner of the lame delegation, and also notify
    # the local hostmaster.
    #----------------------------------------------------------
    if [ $TESTMODE -eq 0 ] ; then
      if [ $VERBOSE -eq 1 ] ; then
        echo "Sending to $HOSTMASTER about lame server $IPADDR for domain $DOMAIN"
      fi
      echo "To: " $HOSTMASTER > $MAILMSG
      echo "Subject: $IPADDR seems to be a lame delegation for $DOMAIN" >> $MAILMSG
      cat $MSGFILE >> $MAILMSG
      sed -e "s|%DOMAIN%|$DOMAIN|" -e "s|%SERVER%|$IPADDR|" $MSGFILE |
                    /usr/lib/sendmail -t -fdns-maintenance
    fi
    echo $HOSTMASTER $DOMAIN $IPADDR >> $LAMEREPORT
  done < $LAMERS
```

Now that we have processed all of the lame delegation messages, we're just about done. The last action we take is to send a single message to the local hostmaster at U-M listing all of the lame delegations. The message - a short lame delegation report - lists one line per lame delegation found. On that line are the domain, the nameserver, and the e-mail address that we used when sending the message.

```
  #----------------------------------------------------------
  # No news is good news
```

```
#------------------------------------------------------------
if [ -s $LAMEREPORT ] ; then
        Mail -s "Lame report" hostmaster@umich.edu < $LAMEREPORT
fi
rm -f $LAMERS $MAILMSG $LAMEREPORT $WEEKFILE
```

## Conclusions

We've been using this tool to notify hostmasters of lame delegations for about one year now. So far the results have been very encouraging. Except for some really hard-core lamers, most sites who are flagged one week as having a lame delegation do not get flagged the following week. Unfortunately, the number of lame delegations discovered each week doesn't seem to be going down either. We've been told by the folks at NSFNET that domain packets account for about 10% of the backbone T3 traffic; it would be interesting to conduct an experiment to try to discover exactly how much of that is due to misconfigured nameservers.

Response from hostmasters receiving mail from this script has generally been very favorable. Comments run the gamut from "Thanks for the heads up. We don't mind at all getting the mail." to "Quit sending us this message...". Most of the unfavorable replies come from people who don't understand why they're getting the message, and after a friendly phone call or follow-up message, they are usually eager to cooperate.

All of the software mentioned above and the e-mail template that we use are available via anonymous ftp from terminator.cc.umich.edu:/dns.

## Author Information

After receiving his MS in Computer Engineering from the University of Michigan in 1988, Bryan Beecher could not bring himself to leave academic life for the hardships of a real job, and so has spent the past four years handling domain nameservice and teaching UNIX system administration courses for U-M. The U-M distribution of BIND 4.8.3 contains many features he has added, such as Shuffle Address (SA) records, query logging and report generation, and, of course, the lame delegation tools. Most recently, he has begun running U-M's USENET news server *(destroyer),* and is part of the team creating better tools for X.500-based directory services. He can be reached via e-mail at bryan@umich.edu.