Checking for expired/changed DNS records

One of the things that happens with DNS is, over the years, you don't clean up when a domain goes to a different DNS provider, or the domain just expires. Both of these can happen through oversight, or when a client decides to move their service without telling you.

In 2020, we found almost 1/3 or the domains we had in our Bind server were no longer pointed to us by the registrars. The authority on where a domain gets its name services from is found with whois, so I wrote some scripts that grab the information, then process it.

Getting the whois information

On unix machines, the *whois* command does the lookup for you, and retrieves a bunch of text designed to be read by a human. Perl also has a set of libraries that provide the same function, but it will error out under certain conditions, such as when a TLD (Top Level Domain) does not have a whois function. When calling the Unix whois command, the command returns an error (which I ignore), but also displays a message on STDOUT which we can use.

The following script accepts a list of domain names on STDIN, calls whois (either through a local command or via the Perl library), then writes the results to a file. The files are written to a subdirectory (./whois/ by default), and the file names are the domain name, followed by the suffix .whois. Thus, if you tell it to retrieve the entry for example.com, the output will be written to ./whois/example.com.whois

The script pauses 1 second between entries so the whois providers don't get mad at us, but it pauses 16 seconds after asking for a .org TLD, since they have rate limits of 4/min.

Assuming you have all your domains listed in /etc/bind/SEC, and the zone files are just the names of the domains with no suffix, you can run the following command:

```
ls /etc/bind/SEC/* | rev | cut -d'/' -f1 | rev | ./getWhoisRecord.pl
```

Explanation

1. ls /etc/bind/SEC/* gets a list of all files in that directory
2. We need to get rid of the directories. Could use basename, but rev | cut -d'/' -f1 | rev does it for us
    1. the first 'rev' does a complete reversal of the text, ie catdog becomes godtac
    2. We use cut, on / (the directory delimiter), to grab only the first part
    3. do another rev, and godtac becomes catdog again.
3. we now have a list of domain names, which we pass to getWhoisRecord.pl

getWhoisRecord.pl

```
#! /usr/bin/env perl

# https://perlmaven.com/checking-the-whois-record-of-many-domains
```

```
use strict;
use warnings;

# allow the 'which' command for any OS
use File::Which; # apt-get install libfile-which-perl

# check if whois is installed on this machine
my $whois = which 'whois';
# write the files to ./whois/
my $outPath = 'whois';

unless ( defined $whois ) { # we don't have it, so use Perl's
implementation
   use Net::Whois::Raw; # apt-get install libnet-whois-raw-perl
   $Net::Whois::Raw::CHECK_FAIL = 1;
   $Net::Whois::Raw::OMIT_MSG = 1;
}

mkdir $outPath unless -d $outPath;

while ( my $domain = <> ) {
  chomp $domain;
  next if $domain =~ m/arpa$/; # skip rdns entries
  warn "ERROR in [$domain], not processing\n" if $domain !~ m/^[a-z0-9-
]+\.[a-z0-9-]+$/;
  my $outFile = "$outPath/$domain.whois";
  print STDERR "$domain\t";
  if ( -e $outFile ) {
     print STDERR "Already exists, skipping\n";
     next;
  }
  if ( defined $whois ) {
     `$whois $domain > $outFile`;
  } else {
     my $data = whois($domain);
     open OUT,">$outFile" or die "Could not write to $outFile: $!\n";
     print OUT $data;
     close OUT;
  }
  if ($domain =~ /\.org$/) {
      print STDERR "Waiting 16 seconds to avoid rate limit\n";
      sleep 16;
  } else {
     print STDERR "Waiting 1 second before next one\n";
     sleep 1;
  }
}
```

Finding the NS records

Well, now we have a directory full of .whois files, and we want to process them. Unfortunately, it seems that everyone want's to do it differently. Gabor Szabo wrote a script that takes this into account and posted it at https://perlmaven.com/checking-the-whois-record-of-many-domains, and I have shamelessly stolen his code, made some minor changes, then included it here.

This script takes a list of files on the cli, processes each in turn, building a data structure (hash %results) to hold the relationship between name servers and domain names. It then dumps the results in a tab delimited text file to STDOUT.

The function get_ns is Mr. Szabo's, with a few modifications by me to add the “name server's”

- Expired Domain – the whois server responded, telling us it could not find an entry
- Invalid TLD – We got the message that no whois server is known for this kind of object
- Empty Record – our file is empty? Not sure why, but you need to check it out

This can be run (assuming you're still in the same directory as the above script) as

```
./getWhoisNS.pl whois/* > getWhoisNS.csv
```

which will read all the files, and send the output to getWhoisNS.csv. Hint, you can send the info to another program just as easily.

getWhoisNS.pl

```
#! /usr/bin/env perl

# based in part on code from
# https://perlmaven.com/checking-the-whois-record-of-many-domains

use strict;
use warnings;

use Data::Dumper;
use File::Basename;

my %results;

for ( my $i = 0; $i < scalar( @ARGV ); $i++ ) {
  my $data = '';
  my $domain = basename( $ARGV[$i], ( '.whois' ) );
  if ( open DATA,"<$ARGV[$i]" ) {
     $data = join( '', <DATA> );
  } else {
     warn "Could not read $ARGV[$i]: $!\n";
     next;
  }
  my @ns = get_ns($data);
```

```
  if ( @ns ) {
     foreach my $thisNS ( @ns ) {
        push @{ $results{ $thisNS } }, $domain;
     }
   } else {
      push  @{ $results{ 'Unknown' } }, $domain;
   }
}

foreach my $ns ( sort keys %results ) {
   print "$ns\t", join( "\n$ns\t", @{ $results{$ns} } ) . "\n";
}


sub get_ns {
  my ($data) = @_;

  my @ns;

  return ('Empty Record') unless $data; # this is a bad domain?
  return ('Invalid TLD' ) if $data =~ m/No whois server is known for
this kind of object/;

  @ns = map { uc $_ } $data =~ /^\s*Name Server:\s*(\S+)\s*$/mg;

  if (not @ns) {
      @ns = map { uc $_ } $data =~ /^nserver:\s*(\S+)\s*$/mg;
  }

  if (not @ns) {
      my @lines = split /\n/, $data;
      return ('Expired Domain') if $lines[0] =~ m/^No Data Found/ ||
$lines[0] =~ m/^No match for/ || $lines[0] =~ m/^NOT FOUND/;
      my $in_ns = 0;
      for my $line (@lines) {
          if ($line =~ /^\s*Domain servers in listed order:\s*$/) {
              $in_ns = 1;
              next;
          }
          if ($line =~ /^\s*$/) {
              $in_ns = 0;
          }
          if ($in_ns) {
              $line =~ s/^\s+|\s+$//g;
              push @ns, uc $line;
          }
      }
      @ns = sort @ns;
  }

  return @ns;
```

```
}
```

We Don't Host it

Finally, we want a little script to run through the CSV we just created and see if our name server has anything in it that whois says we don't host.

This has a (very small) complication because it is perfectly legitimate to have your domain listed on several DNS servers, and there is no reason to have them from the same provider. As a matter of fact, we provide DNS hosting for a couple of clients who have their own DNS servers, they just want their records on a third and fourth place. So, we take one pass to find out the domains we are listed on for whois, then we take a second pass to see if there is anyone not in that list.

Since the script has no way of knowing what our name servers are, we pass them in as command line arguments, so

```
./weDontHost.pl ns1.example.com ns2.example.com < getWhoisNS.csv
```

Or, if you don't care about having the CSV for any other purpose, you can combine the previous step with this one by using the command

```
./getWhoisNS.pl whois/* | ./weDontHost.pl ns1.example.com ns2.example.com
```

which just passes the output of getWhoIsNS.pl to weDontHost.pl

weDontHost.pl

```
#! /usr/bin/env perl

use strict;
use warnings;

# we will list our name servers as CLI arguments
my %ourNS = map { lc $_ => 1 } @ARGV;
# if we don't clear ARGV, perl thinks we're passing filenames
# to open for STDIN
@ARGV = ();

# The tab delimited file of nameserver\tdomain are on STDIN
my @domainList = <>;
chomp @domainList;

my %weHost;
my %dontHost;

# get the ones we DO host
for ( my $i = 0; $i < @domainList; $i++ ) {
   my ($ns,$domain) = split( "\t", $domainList[$i] );
```

```
   $weHost{$domain} = 1 if $ourNS{lc $ns} || defined( $weHost{$domain}
);
}

# get the ones we DON'T host. NOTE: we do it this way because a user
may have
# us as a primary or a secondary, and have an entry with some other
entity
for ( my $i = 0; $i < @domainList; $i++ ) {
   my ($ns,$domain) = split( "\t", $domainList[$i] );
   $dontHost{$domain} = 1 unless defined( $weHost{$domain} );
}

# dump the output
print "We don't do DNS for these domains\n" . join( "\n", keys
%dontHost ) . "\n";

1;
```

What do I do with this?

Well, you can run this manually whenever you want. Or, do what we do, which is script the run and send the output to an admin every quarter or so. The first script, getWhoisRecord.pl, is kind of noisy. I like to see I didn't mess up, so I told it to tell me which domains it is working on on STDERR, so you might want to comment those prints out (you don't want to redirect STDERR since it may have valuable info you'd miss). Once you've done that, you can create a bash (or perl) script, like the following:

checkNSValid.sh

```
#! /usr/bin/env bash

# simple script customized for our DNS server
# runs the necessary scripts to check for NS validity on our server

# first, refresh the whois records for all domains in the /etc/bind/SEC
# directory. Places by default into ./whois/ directory
ls /etc/bind/SEC/* | rev | cut -d'/' -f1 | rev | ./getWhoisRecord.pl
# now, get the NS records for whois dump in the ./whois/ directory
# create a tab delimited file for the output
./getWhoisNS.pl whois/* > getWhoisNS.csv

# finally, process the csv and see if there is anything there that we
# don't host
./weDontHost.pl ns1.example.com ns2.example.com < getWhoisNS.csv
```

Caveats

- First and foremost, Mr. Szabo and I have different coding styles, so this is a mishmash of them. One of these days, if I care enough, I'll go in and clean it up so it is the “right way” (ie, mine, not his...). Until then, live with it. Or download, change it to your “right way”. If you do that and sent it back to me, I'll replace the stuff here and give you credit.
- This is almost a “one off”, so I didn't spend a lot of time on pretty. All three perl scripts could easily have cli parameter processing to override the constants I used. Anybody want to turn off the print STDERR without commenting them out?
- Needs more comments.
- whois/ directory doesn't get cleaned out. I created that so I was not getting fresh copies every time I ran it during testing, and just left it in.
- puts whois/ directory as a subdir of your current directory. Works for me.

  ```
  cd /path/to/script && ./checkNSValid.sh
  ```

  is fugly, but it works.

But, bottom line is, scripts work for my purposes now, and with some minor modifications, might work for you. Better still, go to Mr. Szabo's site and download his code. For a cron job, it is probably much better (single program, does it all). I want the intermediate files so I can later write a different filter to see when domains are coming up for renewal, for one thing.

Links

- https://perlmaven.com/checking-the-whois-record-of-many-domains

From:
https://kb.unixservertech.com/ - **Unix Server Tech Knowledge Base**

Permanent link:
https://kb.unixservertech.com/dns/checkexpired

Last update: **2020/09/29 05:32**