Who is hijacking my NXDOMAINs

Mon, Mar 30, 2020

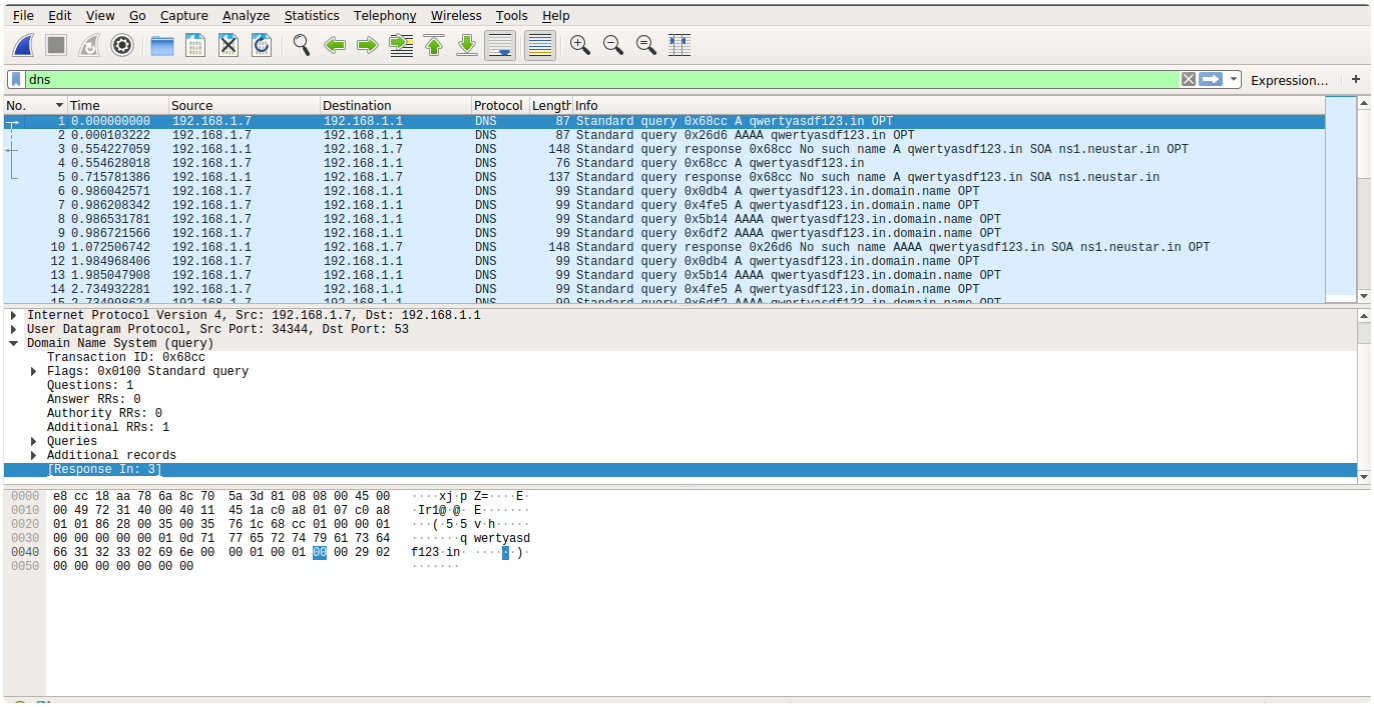
It all began a few days ago when I visited a *.in* domain that I knew didn't exist and instead of getting an error page from the browser, as one might expect, I got a blank page. Well, a blank page for me because I was using an ad-blocker(uBlock Origin) otherwise I would have seen a page filled with ads. This behavior would have been easily explained if I was using ISP provided DNS server for resolving queries. How?
Let's start with DNS(Domain Name System). DNS is responsible for mapping domain names to IP addresses and they send an NXDOMAIN(Non-Existent Domain) response if the requested domain name cannot be found and the browser will show an appropriate error page. Some DNS servers, especially the ones provided by ISPs, might resolve queries for non-existing domains to IP address of servers they control which will show ads instead of an NXDOMAIN response which will trigger the usual error page. I was aware that this is a common practice among many ISPs, but I was not using the default DNS recursive name server provided by the ISP but Cloudflare's 1.1.1.1 – which certainly does not have such malpractices.
I made sure that my OS was not overriding the DNS server set at router by checking `/etc/resolv.conf` . It had:

```
...
nameserver 127.0.0.53
options edns0
search domain.name
```

and `systemd-resolve --status` showed:

```
...
Global
          DNS Servers: 192.168.1.1
          DNS Domain: domain.name
          DNSSEC NTA: ...
...
```

This shows that the OS is indeed using the DNS server provided by the router, which uses 1.1.1.1 and 1.0.0.1 as resolvers.
There was no point in blindly trying any further to figure out what was wrong. Time to *see* what is actually happening using packet sniffing. I used Wireshark to capture DNS query and response packets when visiting qwertyasdf123[.]in, which is a non-existent domain. And this is what it captured:



The *packet no. 1* above is a DNS query asking for DNS A record (which will point to the IPv4 address of the server hosting the domain), and got the response(*packet no. 3*) as `No such name` (i.e NXDOMAIN as expected). Similarly, we can see that a query is made for DNS AAA record(similar to A record but points to IPv6 address) and got the response(*packet no. 10*) as `No such name` . So far, so good. But *packet no. 6* is a DNS query asking for the server IP of **qwertyasdf123.in.domain[.]name** which got a no error response that the domain name resolves to `78.47.226.171` . On visiting that IP address, I got redirected to the same website that I was being redirected to when I was visiting non-existent .in domains – parrz[.]com. I repeated these steps a few more times with different domains and every time an NXDOMAIN response is received for both A and AAA record queries, a third query for A record of `<domain>.domain.name` is sent, where `<domain>` is the domain name I typed like *qwertyasdf123.in* or *qwertyasdf123.com*.
Does `search domain.name` line in `/etc/resolv.conf` has got anything to do with it? Seems like it does. The resolv.conf man page says this about the search configuration:

```
...
Resolver queries having fewer than ndots dots (default is 1) in them will be attempted using each component of the search path in turn until a match is found.
...
ndots:n
Sets a threshold for the number of dots which must appear in a name given to res_query(3) (see resolver(3)) before an initial absolute query will be made.  The default for n is 1, meaning t
hat if there are any dots in a name, the name will be tried first as an absolute name before any search list elements are appended to it.  The value for this option is silently capped to 1
5.
```

After a couple of searches on the web, I figured out what that means – under the default case(when ndots option is not set), for resolving `example.com` , the resolver will try to resolve `example.com` first and if it fails, it'll append search list domain(in this case *domain.name*) to it and try to resolve it. So in our case, when a query for the host address of *qwertyasdf123.in* results in an NXDOMAIN, the resolver will try to resolve *qwertyasdf123.in.domain.name*. .name is indeed a top-level domain extension just like .com or .org and someone might have brought domain.name and configured a wildcard DNS record to redirect all `<anything>.domain.name` queries to a server filled with ads.
I decided to figure out more about *domain.name* and confirm that it had a wildcard DNS record pointing to `78.47.226.171` before finding out how the `search domain.name` line got into my resolv.conf file. I ruled it out as something I might have carelessly configured during OS installation. I did `dig *.domain.name` and got:

```
...
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 8453
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
...
```

`status: NXDOMAIN` – that was an unexpected response, which means **.domain.name* is a non-existent domain name. Well, it seems like .name allows third-level registrations(i.e *in.domain.name* and say, *com.domain.name* are separately registered) and *domain.name* is shared and cannot be registered individually. And `dig *.in.domain.name` returned:

```
...
;; ANSWER SECTION:
*.in.domain.name.       898     IN      CNAME   in.domain.name.
in.domain.name.         897     IN      A       78.47.226.171
...
```

An A record pointing to `78.47.226.171` as expected. When I visited that address on my browser, the server responded with a 302 redirect to http://parrz[.]com/?dn=226.171 which contains some obfuscated JavaScript code that will either show ads or redirect the user to shady betting and trading websites. I decided to do a quick search on parrz[.]com and found a similar case on a Spanish ISP forum – in which the user posted that visiting non-existing .es domains on both their laptop and android smartphone are redirected to parrz[.]com. They finally resolved the problem by replacing their router. With this new knowledge, I tried visiting *qwertyasdf123.in* on my android smartphone and it got redirected to the same parrz[.]com, hinting that the `domain.name` in resolv.conf is probably coming from my router. Furthermore, it showed that *es.domain.name* is also registered and redirects to parrz[.]com. I got a list of all ccTLDs on a text file and ran the following bash commands to see how many such domains are registered:

```
cat ccTLDs.txt | while read line; do dig "$line.domain.name" | tail -7 | head -1 | awk '{ print $1 "\t" $5 }'; done > resolvedDomains.txt
```

which will try to resolve each domain extension in the list suffixed with `domain.name` and stores resolved IP address along with it into a file (resolvedDomains.txt) like this:

```
AC.domain.name. 78.47.226.171
AD.domain.name. 78.47.226.171
AE.domain.name. 78.47.226.171
AF.domain.name. 78.47.226.171
...
```

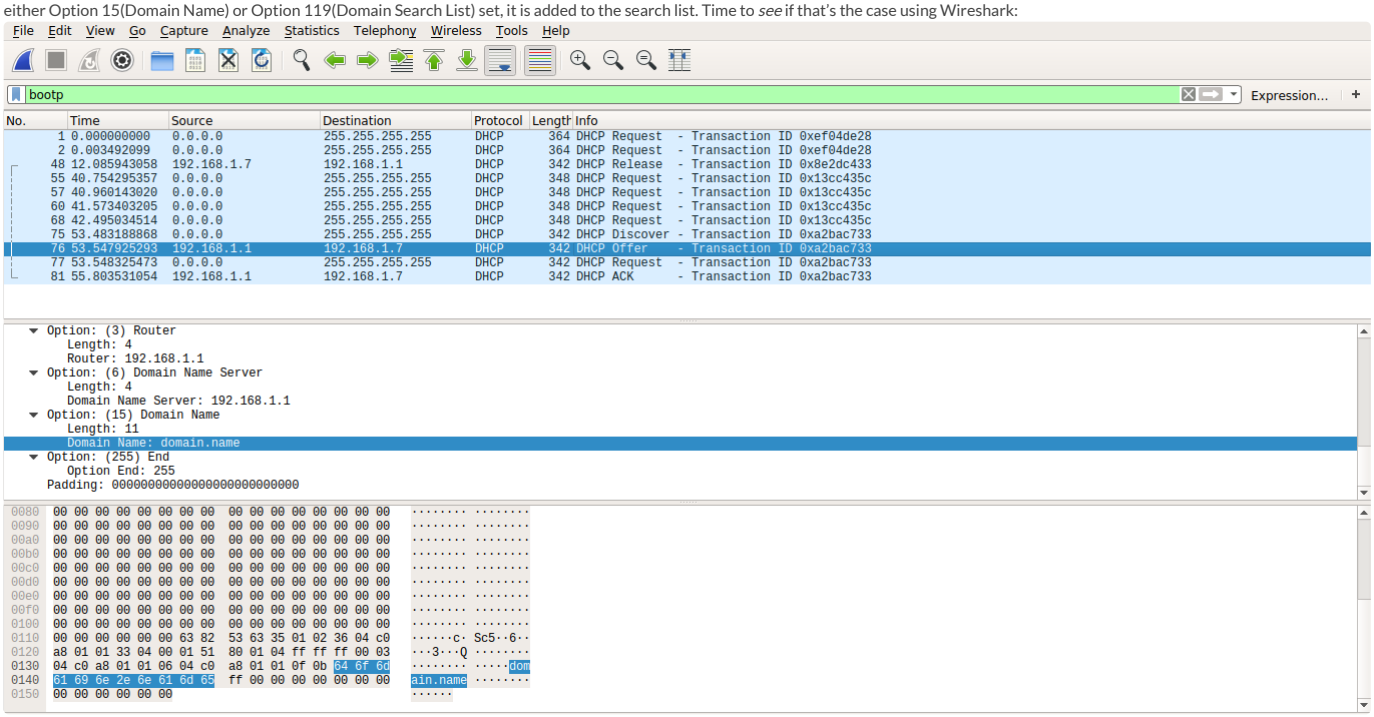
Many domains are mapped to a few IP addresses, let's see how many domains resolve to each using:

```
cat resolvedDomains.txt | awk '{ print $2  }' | sort | uniq -c
```

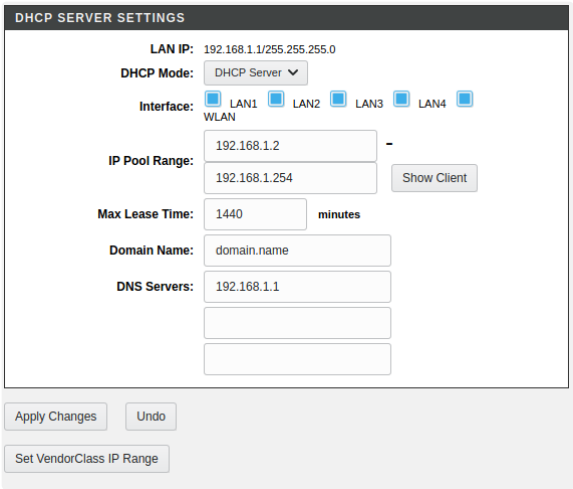
which returned:

```
11
1 174.139.160.58
1 217.70.184.38
1 77.78.104.3
237 78.47.226.171
```

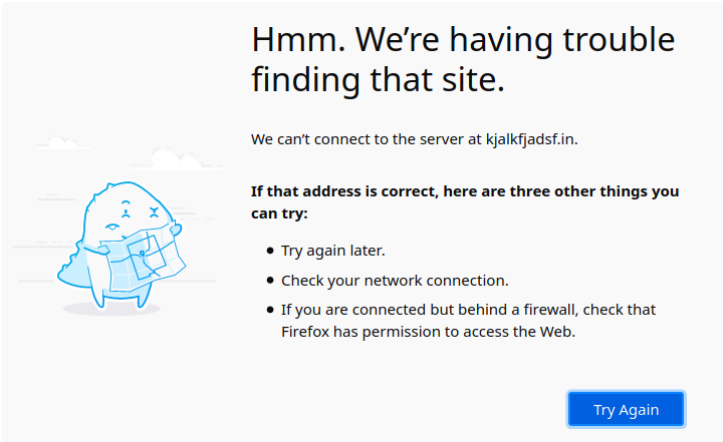
Almost all of them are pointed to `78.47.226.171` which redirects to parrz[.]com. All of the domains which points to it have *Gransy, s.r.o.* as the domain registrar so it is not clear whether these domains are parked or not. If not, *.domain.name* registration costs 14.70 USD on regtons[.]com(Gransy's "domain solution" according to their website) and for 237 domain names, it would have cost them 3,483 USD per year.
Parrz[.]com has an Alexa rank of around 40,000 in India, which is too much for a website that just shows shady advertisements. If so, I'm probably not alone in having this problem.
Now the only question that remains is how `domain.name` got added in the search option of `/etc/resolv.conf`. After searching for a while, turns out that if the DHCP server(in this case my router) responds with either Option 15(Domain Name) or Option 119(Domain Search List) set, it is added to the search list. Time to *see* if that's the case using Wireshark:



Packet no. 76 is a DHCP Offer packet from the router and, as expected, has Option 15(Domain Name) set as `domain.name`. So it is indeed coming from the router and I found it in `Setup > Local Network > DHCP Server` as:



I replaced it with `dlink-home-network`, restarted the router and then visited some non-existent .in domain and instead of being redirected to ads about trading and betting, I got:



Problem solved! I tried to find out if `domain.name` was the default value for that field for my router but didn't find anything on the web. I asked some of my friends who have D-Link routers and while the newer ones had that field blank or the names `dlinkrouter` or just `dlink` some of the older models had `domain.name` just like me. These models were DSL 2730-U(firmware version IN_1.11 and IN_1.10) and DIR-600M(firmware version 3.04). Do check your router settings and if you have `domain.name` or anything ending with a valid tld, replace it with something like `my-home-network`.

---

cybersecurity

dns network security