computers are bad

https://computer.rip - me@computer.rip - fax: +1 (505) 926-5492

2024-02-11 the top of the DNS hierarchy

In the past (in fact two years ago, proof I have been doing this for a while now!) I wrote about the "inconvenient truth" that structural aspects of the Internet make truly decentralized systems infeasible, due to the lack of a means to perform broadcast discovery. As a result, most distributed systems rely on a set of central, semi-static nodes to perform initial introductions.

For example, Bitcoin relies on a small list of volunteer-operated domain names that resolve to known-good full nodes. Tor similarly uses a small set of central "directory servers" that provide initial node lists. Both systems have these lists hardcoded into their clients; coincidentally, both have nine trusted, central hostnames.

This sort of problem exists in basically all distributed systems that operate in environments where it is not possible to shout into the void and hope for a response. The internet, for good historic reasons, does not permit this kind of behavior. Here we should differentiate between distributed and decentralized, two terms I do not tend to select very carefully. Not all distributed systems are decentralized, indeed, many are not. One of the easiest and most practical ways to organize a distributed system is according to a hierarchy. This is a useful technique, so there are many examples, but a prominent and old one happens to also be part of the drivetrain mechanics of the internet: DNS, the domain name system.

My reader base is expanding and so I will provide a very brief bit of background. Many know that DNS is responsible for translating human-readable names like "computer.rip" into the actual numerical addresses used by the internet protocol. Perhaps a bit fewer know that DNS, as a system, is fundamentally organized around the hierarchy of these names. To examine the process of resolving a DNS name, it is sometimes more intuitive to reverse the name, and instead of "computer.rip", discuss "rip.computer" [1].

This name is hierarchical, it indicates that the record "computer" is within the zone "rip". "computer" is itself a zone and can contain yet more records, we tend to call these subdomains. But the term "subdomain" can be confusing as everything is a subdomain of something, even "rip" itself, which in a certain sense is a subdomain of the DNS root "." (which is why, of course, a stricter writing of the domain name computer.rip would be computer.rip., but as a culture we have rejected the trailing root dot).

Many of us probably know that each level of the DNS hierarchy has authoritative nameservers, operated typically by whoever controls the name (or their third-party DNS vendor). "rip" has authoritative DNS servers provided by a company called Rightside Group, a subsidiary of the operator of websites like eHow that went headfirst into the great DNS land grab and snapped up "rip" as a bit of land speculation, alongside such attractive properties as "lawyer" and "navy" and "republican" and "democrat", all of which I would like to own the "computer" subdomain of, but alas such dictionary words are usually already taken.

"computer.rip", of course, has authoritative nameservers operated by myself or my delegate. Unlike some people I know, I do not have any nostalgia for BIND, and so I pay a modest fee to a commercial DNS operator to do it for me. Some would be surprised that I pay for this; DNS is actually rather inexpensive to operate and authoritative name servers are almost universally available as a free perk from domain registrars and others. I just like to pay for this on the general feeling that companies that charge for a given service are probably more committed to its quality, and it really costs very little and changing it would take work.

To the observant reader, this might leave an interesting question. If even the top-level domains are subdomains of a secret, seldom-seen root domain ".", who operates the authoritative name servers for that zone?

And here we return to the matter of even distributed systems requiring central nodes. Bitcoin uses nine harcoded domain names for initial discovery of decentralized peers. DNS uses thirteen harcoded root servers to establish the top level of the hierarchy.

These root servers are commonly referred to as a.root-servers.net through m.root-servers.net, and indeed those are their domain names, but remember that when we need to use those root servers we have no entrypoint into the DNS hierarchy and so are not capable of resolving names. The root servers are much more meaningfully identified by their IP addresses, which are "semi-harcoded" into recursive resolves in the form of what's often called a root hints file. You can download a copy, it's a simple file in BIND zone format that BIND basically uses to bootstrap its cache.

And yes, there are other DNS implementations too, a surprising number of them, even in wide use. But when talking about DNS history we can mostly stick to BIND. BIND used to stand for Berkeley Internet Name Domain, and it is an apt rule of thumb in computer history that anything with a reference to UC Berkeley in the name is probably structurally important to the modern technology industry.

One of the things I wanted to get at, when I originally talked about central nodes in distributed systems, is the impact it has on trust and reliability. The TOR project is aware that the nine directory servers are an appealing target for attack or compromise, and technical measures have been taken to mitigate the possibility of malicious behavior. The Bitcoin project seems to mostly ignore that the DNS seeds exist, but of course the design of the Bitcoin system limits their compromise to certain types of attacks. In the case of DNS, much like most decentralized systems, there is a layer of long-lived caching for top-level domains that mitigates the impact of unavailability of the root servers, but still, in every one of these systems, there is the possibility of compromise or unavailability if the central nodes are attacked.

And so there is always a layer of policy. A trusted operator can never guarantee the trustworthiness of a central node (the node could be compromised, or the trusted operator could turn out to be the FBI), but it sure does help. Tor's directory servers are operated by the Tor project. Bitcoin's DNS seeds are operated by individuals with a long history of involvement in the project. DNS's root nodes are operated by a hodgepodge of companies and institutions that were important to the early internet.

Verisign operates two, of course. A California university operates one, of course, but amusingly not Berkeley. Three are operated by various arms of US defense. Some internet industry associations, an NCC, another university, ICANN runs one of them themselves. It's pretty random, though, and just reflects a set of organizations prominently involved in the early internet.

Some people, even some journalists I've come across, hear that there are 13 name servers and picture 13 4U boxes with a lot of blinking lights in heavily fortified data centers. Admittedly this description was more or less accurate in the early days, and a couple of the smaller root server operators did have single machines until surprisingly recently. But today, all thirteen root server IP addresses are anycast groups.

Anycast is not a concept you run into every day, because it's not really useful on local networks where multicast can be used. But it's very important to the modern internet. The idea is this: an IP address (really a subnetwork) is advertised by multiple BGP nodes. Other BGP nodes can select the advertisement they like the best, typically based on lowest hop count. As a user, you connect to a single IP address, but based on the BGP-informed routing tables of internet service providers your traffic could be directed to any number of sites. You can think of it as a form of load balancing at the IP layer, but it also has the performance benefit of users mostly connecting to nearby nodes, so it's widely used by CDNs for multiple reasons.

For DNS, though, where we often have a bootstrapping problem to solve, anycast is extremely useful as a way to handle "special" IP addresses that are used directly. For authoritative DNS servers like 192.5.5.241 [2001:500:2f::f] [2] (root server F) or recursive resolvers like 8.8.8.8 [2001:4860:4860::8888] (Google public DNS), anycast is the secret that allows a "single" address to correspond to a distributed system of nodes.

So there are thirteen DNS root servers in the sense that there are thirteen independently administered clusters of root servers (with the partial exception of A and J, both operated by Verisign, due to their acquisition of former A operator Network Solutions). Each of the thirteen root servers is, in practice, a fairly large number of anycast sites, sometimes over 100. The root server operators don't share much information about their internal implementation, but one can assume that in most cases the anycast sites consist of multiple servers as well, fronted by some sort of redundant network appliance. There may only be thirteen of them, but each of the thirteen is quite robust. For example, the root servers typically place their anycast sites in major internet exchanges distributed across both geography and provider networks. This makes it unlikely that any small number of failures would seriously affect the number of available sites. Even if a root server were to experience a major failure due to some sort of administration problem, there are twelve more.

Why thirteen, you might ask? No good reason. The number of root servers basically grew until the answer to an NS request for "." hit the 512 byte limit on UDP DNS responses. Optimizations over time allowed this number to grow (actually using single letters to identify the servers was one of these optimizations, allowing the basic compression used in DNS responses to collapse the matching root-servers.net part). Of course IPv6 blew DNS response sizes completely out of the water, leading to the development of the EDNS extension that allows for much larger responses.

13 is no longer the practical limit, but with how large some of the 13 are, no one sees a pressing need to add more. Besides, can you imagine the political considerations in our modern internet environment? The proposed operator would probably be Cloudflare or Google or Amazon or something and their motives would never be trusted. Incidentally, many of the anycast sites for root server F (operated by ISC) are Cloudflare data centers used under agreement.

We are, of course, currently trusting the motives of Verisign. You should never do this! But it's been that way for a long time, we're already committed. At least it isn't Network Solutions any more. I kind of miss when SRI was running DNS and military remote viewing.

But still, there's something a little uncomfortable about the situation. Billions of internet hosts depend on thirteen "servers" to have any functional access to the internet.

What if someone attacked them? Could they take the internet down? Wouldn't this cause a global crisis of a type seldom before seen? Should I be stockpiling DNS records alongside my canned water and iodine pills?

Wikipedia contains a great piece of comedic encyclopedia writing. In its article on the history of attacks on DNS root servers, it mentions the time, in 2012, that some-pastebin-user-claiming-to-be-Anonymous (one of the great internet security threats of that era) threatened to "shut the Internet down". "It may only last one hour, maybe more, maybe even a few days," the statement continues. "No matter what, it will be global. It will be known."

That's the end of the section. Some Wikipedia editor, no doubt familiar with the activities of Anonymous in 2012, apparently considered it self-evident that the attack never happened.

Anonymous may not have put in the effort, but others have. There have been several apparent DDoS attacks on the root DNS servers. One, in 2007, was significant enough that four of the root servers suffered---but there were nine more, and no serious impact was felt by internet users. This attack, like most meaningful DDoS, originated with a botnet. It had its footprint primarily in Korea, but C2 in the United States. The motivation for the attack, and who launched it, remains unknown.

There is a surprisingly large industry of "booters," commercial services that, for a fee, will DDoS a target of your choice. These tend to be operated by criminal groups with access to large botnets; the botnets are sometimes bought and sold and get their tasking from a network of resellers. It's a competitive industry. In the past, booters and botnet operators have sometimes been observed announcing a somewhat random target and taking it offline as, essentially, a sales demonstration. Since these demonstrations are a known behavior, any time a botnet targets something important for no discernible reason, analysts have a tendency to attribute it to a "show of force." I have little doubt that this is sometimes true, but as with the tendency to attribute monumental architecture to deity worship, it might be an overgeneralization of the motivations of botnet operators. Sometimes I wonder if they made a mistake, or maybe they were just a little drunk and a lot bored, who is to say?

The problem with this kind of attribution is evident in the case of the other significant attack on the DNS root servers, in 2015. Once again, some root servers were impacted badly enough that they became unreliable, but other root servers held on and there was little or even no impact to the public. This attack, though, had some interesting properties.

In the 2007 incident, the abnormal traffic to the root servers consisted of large, mostly-random DNS requests. This is basically the expected behavior of a DNS attack; using randomly generated hostnames in requests ensures that the responses won't be cached, making the DNS server exert more effort. Several major botnet clients have this "random subdomain request" functionality built in, normally used for attacks on specific authoritative DNS servers as a way to take the operator's website offline. Chinese security firm Qihoo 360, based on a large botnet honeypot they operate, reports that this type of DNS attack was very popular at the time.

The 2015 attack was different, though! Wikipedia, like many other websites, describes the attack as "valid queries for a single undisclosed domain name and then a different domain the next day." In fact, the domain names were disclosed, by at least 2016. The attack happened on two days. On the first day, all requests were for 336901.com. The second day, all requests were for 916yy.com.

Contemporaneous reporting is remarkably confused on the topic of these domain names, perhaps because they were not widely known, perhaps because few reporters bothered to check up on them thoroughly. Many sources make it sound like they were random domain names perhaps operated by the attacker, one goes so far as to say that they were registered with fake identities.

Well, my Mandarin isn't great, and I think the language barrier is a big part of the confusion. No doubt another part is a Western lack of familiarity with Chinese internet culture. To an American in the security industry, 336901.com would probably look at first like the result of a DGA or domain generation algorithm. A randomly-generated domain used specifically to be evasive. In China, though, numeric names like this are quite popular. Qihoo 360 is, after all, domestically branded as just 360---360.cn.

As far as I can tell, both domains were pretty normal Chinese websites related to mobile games. It's difficult or maybe impossible to tell now, but it seems reasonable to speculate that they were operated by the same company. I would assume they were something of a gray market operation, as there's a huge intersection between "mobile games," "gambling," and "target of DDoS attacks." For a long time, perhaps still today in the right corners of the industry, it was pretty routine for gray-market gambling websites to pay booters to DDoS each other.

In a 2016 presentation, security researchers from Verisign (Weinberg and Wessels) reported on their analysis of the attack based on traffic observed at Verisign root servers. They conclude that the traffic likely originated from multiple botnets or at least botnet clients with different configurations, since the attack traffic can be categorized into several apparently different types [3]. Based on command and control traffic from a source they don't disclose (perhaps from a Verisign honeynet?), they link the attack to the common "BillGates" [4] botnet. Most interestingly, they conclude that it was probably not intended as an attack on the DNS root: the choice of fixed domain names just doesn't make sense, and the traffic wasn't targeted at all root servers.

Instead, they suspect it was just what it looks like: an attack on the two websites the packets queried for, that for some reason was directed at the root servers instead of the authoritative servers for that second-level domain. This isn't a good strategy; the root servers are a far harder target than your average web hosting company's authoritative servers. But perhaps it was a mistake? An experiment to see if the root server operators might mitigate the DDoS by dropping requests for those two domains, incidentally taking the websites offline?

Remember that Qihoo 360 operates a large honeynet and was kind enough to publish a presentation on their analysis of root server attacks. Matching Verisign's conclusions, they link the attack to the BillGates botnet, and also note that they often observe multiple separate botnet C2 servers send tasks targeting the same domain names. This probably reflects the commercialized nature of modern botnets, with booters "subcontracting" operations to multiple botnet operators. It also handily explains Verisign's observation that the 2015 attack traffic seems to have come from more than one implementation a DNS DDoS.

360 reports that, on the first day, five different C2 servers tasked bots with attacking 336901.com. On the second day, three C2 servers tasked for 916yy.com. But they also have a much bigger revelation: throughout the time period of the attacks, they observed multiple tasks to attack 916yy.com using several different methods.

360 concludes that the 2015 DNS attack was most likely the result of a commodity DDoS operation that decided to experiment, directing traffic at the DNS roots instead of the authoritative server for the target to see what would happen. I doubt they thought they'd take down the root servers, but it seems totally reasonable that they might have wondered if the root server operators would filter DDoS traffic based on the domain name appearing in the requests.

Intriguingly, they note that some of the traffic originated with a DNS attack tool that had significant similarities to BillGates but didn't produce quite the same packets. Likely we will never know, but a likely explanation is that some group modified the BillGates DNS attack module or implemented a new one based on the method used by BillGates.

Tracking botnets gets very confusing very fast, there are just so many different variants of any major botnet client! BillGates originated, for example, as a Linux botnet. It was distributed to servers, not only through SSH but through vulnerabilities in MySQL and ElasticSearch. It was unusual, for a time, in being a major botnet that skipped over the most common desktop operating system. But ports of BillGates to Windows were later observed, distributed through an Internet Explorer vulnerability---classic Windows. Why someone chose to port a Linux botnet to Windows instead of using one of the several popular Windows botnets (Conficker, for example) is a mystery. Perhaps they had spent a lot of time building out BillGates C2 infrastructure and, like any good IT operation, wanted to simplify their cloud footprint.

High in the wizard's tower of the internet, thirteen elders are responsible for starting every recursive resolver on its own path to truth. There's a whole Neal Stephenson for Wired article there. But in practice it's a large and robust system. The extent of anycast routing used for the root DNS servers, to say nothing of CDNs, is one of those things that challenges are typical stacked view of the internet. Geographic load balancing is something we think of at high layers of the system, it's surprising to encounter it as a core part of a very low level process.

That's why we need to keep our thinking flexible: computers are towers of abstraction, and complexity can be added at nearly any level, as needed or convenient. Seldom is this more apparent than it is in any process called "bootstrapping." Some seemingly simpler parts of the internet, like DNS, rely on a great deal of complexity within other parts of the system, like BGP.

Now I'm just complaining about pedagogical use of the OSI model again.

[1] The fact that the DNS hierarchy is written from right-to-left while it's routinely used in URIs that are otherwise read left-to-right is one of those quirks of computer history. Basically an endianness inconsistency. Like American date order, to strictly interpret a URI you have to stop and reverse your analysis part way through. There's no particular reason that DNS is like that, there was just less consistency over most significant first/least significant first hierarchical ordering at the time and contemporaneous network protocols (consider the OSI stack) actually had a tendency towards least significant first.

[2] The IPv4 addresses of the root servers are ages old and mostly just a matter of chance, but the IPv6 addresses were assigned more recently and allowed an opportunity for something more meaningful. Reflecting the long tradition of identifying the root servers by their letter, many root server operators use IPv6 addresses where the host part can be written as the single letter of the server (i.e. root server C at [2001:500:2::c]). Others chose a host part of "53," a gesture at the port number used for DNS (i.e. root server J, [2001:7fe::53]). Others seem more random, Verisign uses 2:30 for both of their root servers (i.e. root server A, [2001:503:ba3e::2:30]), so maybe that means something to them, or maybe it was just convenient. Amusingly, the only operator that went for what I would call an address pun is the Defense Information Systems Agency, which put root server G at [2001:500:12::d0d].

[3] It really dates this story that there was some controversy around the source IPs of the attack, originating with none other than deceased security industry personality John McAfee. He angrily insisted that it was not plausible that the source IPs were spoofed. Of course botnets conducting DDoS attacks via DNS virtually always spoof the source IP, as there are few protections in place (at the time almost none at all) to prevent it. But John McAfee has always had a way of ginning up controversy where none was needed.

[4] Botnets are often bought, modified, and sold. They tend to go by various names from different security researchers and different variants. I'm calling this one "BillGates" because that's the funniest of the several names used for it.