The ISP Column
On Things Internet

August 2020
Geoff Huston

DNS OARC Meeting Notes

Much the Internet operations and research world has gone virtual for 2020. Meetings continue to take place and while the level of interaction in these meetings is different, many of these meetings continue to engender useful conversations. In my case I'm interested in the infrastructure that binds the network together into a coherent whole, and I don't think I'm alone in finding this topic fascinating. In the Internet's name space the DNS OARC meetings are a case where a concentrated burst of DNS tests the proposition that you just can't have too much DNS! OARC held its latest meeting on the 11th August with four presentations (https://indico.dns-oarc.net/event/36/). Here's my thoughts on the material presented at that meeting.

DNS Resolvers and RPKI

DNS security and BGP Routing security appear to address quite distinct threat realms, but the question being asked in a presentation on DNS Resolvers and RPKI ROA validation is to what extent resolvers should use ROAs to "protect" their service from being hijacked.

The threat model envisaged here is that the service IP address of a distant open recursive resolver is hijacked, and client queries are silently redirected to the attacker's service. This is not a purely hypothetical scenario and there have been persistent reports of networks in various countries and circumstances advertising a false route for Google's public DNS service (8.8.8.8). Authoritative servers are also potential targets of such routing attacks, and a successful attacker could substitute its own answers in place of the those provided by the authentic authoritative DNS server by performing a route hijack. Should DNS Recursive Resolvers and Authoritative Servers set up RPKI credentials for the route objects associated with reachability to these services and thereby invoke the awesome power of RPKI to protect the integrity of their service?

A study performed by researchers at the University of Amsterdam set up a couple of test servers, located on IP addresses that had valid and invalid RPKI ROA credentials, and then used RIPE Atlas to direct queries to these servers. The results as far as I can tell were inconclusive, and I suspect that my issues here lie in both the methodology of the experiment and the underlying nature of the supposed threat model that is proposed in this presentation (https://indico.dns-oarc.net/event/36/contributions/775/attachments/751/1281/RoV-protected-resolvers.pdf).

Let me explain.

In terms of the analytical methodology the issue stems in the use of RIPE Atlas data to make general observations about the state of the entire Internet. RIPE Atlas is a relatively small set of probes (some 11,000 in August 2020), located predominately in Western Europe in locations that are often managed by so-called 'power users' rather than commercial service defaults (https://atlas.ripe.net/results/maps/network-coverage/). Atlas' reach into the mobile Internet appears to be minimal, as is the reach into the considerable user populations of Asia, Africa, and the Americas. Atlas is an incredibly powerful and unique measurement tool for certain measurements, but to use this measurement platform to make generic "whole of Internet" conclusions does not withstand objective scrutiny in my opinion, as the data set is just too small a sample and the biases in the Atlas sample set cannot be easily corrected, if at all. The conclusions from this presentation regarding inferences for "all of Internet" behaviours seem to me to be difficult to sustain.

However, there are some deeper concerns here about the supposed threat model that this presentation sets out to investigate. In security areas there is often a knee-jerk reaction of "more is better". I'm guessing that there is a widely held belief that adding further security mechanisms makes the result more resilient, as the attacker is meant to breach all these mechanisms at the same time or in a particular sequence in order to execute a successful attack (in a so-called "defence in depth" scenario). That is often not the case in computer network scenarios, and often the system retains the risk of the weakest security measure, not the cumulative sum of all of the measures (the "weakest link" model of compounded security).

Let's look at the scenario of using BGP route origination security in DNS resolution. What BGP route origination security is intended to do is to increase the assurance of IP destination address delivery. Your packets have a greater level of assurance that they will be delivered to the endpoint that has the IP address that matches the destination address in the data packet. It does not make any claim that the service host has not been compromised, nor does it make any claims at all about the integrity of the subsequent network transaction. It does not even claim to protect the ultimate delivery of the packet to its intended destination. All BGP route origination security can do is claim that the packet might be delivered to the network that is using one of a set of "authorised" Autonomous System (AS) numbers. It makes no claim about whether the network's use of that particular AS number is "authentic" or not, nor any claim as to the ultimate delivery of packets, nor any claim about the handling of the packet within the destination network, or even any claim about the handling of the packet in transit. in the inter-AS routing space. None of that. In the limited context of route origination validation, the term "routing security" is an oxymoron, even when considering the limited capability of the inter-domain routing system to provide any assurance of packet delivery and nothing else. Another issue with BGP route origination security is one shared with many security frameworks. Generating secure credentials is just not enough. Other parties have to use these credentials to validate this information. It's not just that address holders have to generate Route Origination Attestations, but routing systems need to use these attestations to validate BGP route objects. But even that is not enough. The security objective is to identify what's bad, and the way we do this is to assume that if we cannot validate the information as "good" then it is assumed to be "bad". This implies that generating routing credentials for BGP route origination is, in and of itself, largely an insubstantive action. Network operators need to deploy invalid route detection and filtering for this to have any effect at all.

But does this matter? Does the path, or even the ultimate destination of the packet provide any assurance that the subsequent transaction if genuine or not? Let's phrase this question another way. When the data is digitally signed and the data can be verified as current, entire and authentic then does it really matter as to which source you used to obtain the data or the path through the network between you and the source? When you can validate the information that you've been provided as being current, complete and authentic the manner by which you've learned this information is a far lesser concern. In many ways routing security, however poor it may be, is only really a minor factor in the larger issue of information authenticity if you cannot otherwise verify the information directly. And even then, the limited assurances of route origination are an incredibly poor substitute!

Let's look at the DNS in this light. The DNS is opaque in so many ways. Data provided from a recursive DNS resolver was itself provided to that recursive resolver in ways that are entirely opaque to the end client, and it really does not matter in the slightest what IP address was queried to retrieve the DNS data. The real concern is the authenticity and currency of the answer, irrespective of the manner of its delivery. We really cannot secure the path of information flow in the DNS as the DNS itself rejects such notions as path as an integral component of the integrity name resolution protocol. For that reason, DNSSEC was devised.

The security response for the DNS is quite simple: the greatest gains in protecting the integrity of information in the DNS is to use DNSSEC to sign that information. And the greatest gain in protecting the end user from being deceived in the DNS is for users to perform their own DNSSEC validation of signed DNS responses.

However, substitution in the DNS is not the only form of attack. There is also the issue of denial of service, where the fake route is intended to merely prevent the server from being reached at all. Is BGP route origination validation the best approach to mitigate of the risks of such denial-styled attacks? The DNS uses a number of approaches to improve the resiliency of name resolution and the mainstay here is that of diversity. DNS zone is best served by a number of name servers, not just one. These name servers are best spread across multiple networks. (This works well up to a point. Two authoritative name servers are far more resilient in combination than one, and three are likely to be more resilient than two, but a million distinct nameservers is probably detrimental in most cases. Conventional operational wisdom in the DNS points to somewhere between two and four in most cases.) Similarly, stub resolvers are best configured with more than one recursive resolver. One of these recursive resolvers should be in a local network that is not BGP routed. (This is one scenario where anycast systems are less effective than diverse unicast services. If an attacker can redirect traffic intended for the anycast address, then the attack may affect part or even all of the anycast constellation.)

Should DNS Recursive Resolvers and Authoritative Servers have their IP addresses "covered" by ROAs? My opinion is that in most cases it's a task that can be pushed to the far end of the "to do" list. DNSSEC is by far the most effective defence against substitution attacks and service diversity is the most effective response to denial of service attacks. I can think of one area where ROAs may be contemplated, and that is in the case of anycast service platform, most often seen in the "quad" anycast platforms used by some open Recursive Resolvers. In that scenario the vulnerability of the service to a routing attack is somewhat higher, and while a ROA in and of itself will not eliminate the threat of a denial attack, it can contribute to reducing the risk of such an attack to some (small) extent.

So, did I like this presentation? Yes, oddly enough considering my comments here, I did. If the intent of these presentations is to prompt a critical response and think about the assumptions behind the work and think about the analytical methodology used in the work, then this presentation clearly achieved such an intent for me!

Local Roots

Wes Hardaker presented on current work on a running a local root publication service (https://indico.dns-oarc.net/event/36/contributions/778/attachments/755/1278/2020-07-localroot-dns-oarc.pdf).

It's often said of any group that the conversations that they prefer to have are those which are well rehearsed within the group, as distinct from those which are more crucial for the group. It goes by various names, including "Bike Shedding" (yes, Google it!). In DNS circles the topic of the DNS root zone service is one of those conversations that has been rehearsed time after time and it continues to absorb our collective attention.

The DNS root service framework we have today is in so many ways a victim of its history. The characteristics of the root service are the result of various constraints that were applicable in an earlier age and not necessarily relevant today. Why are there 13 root server letters in a root zone priming response? History. Why are the majority of the organisations that operate these root servers domiciled in the United States? History. Why are these organisations operating a root service without formal contracts with some entity? History. Why can't [insert country name here] run a root service? History.

That last question is perhaps the one key question that has been highly contentious in political circles. Many economies recognise the critical role played by the Internet and appreciate the somewhat catastrophic consequences if the Internet simply stopped working. The DNS is an intrinsic part of the essential infrastructure of the Internet, and the observation that tends to excite some concern is that almost every Internet transaction starts with a call to the DNS, and every DNS name resolution operation notionally starts with a query to the root zone. Were the root zone servers to become inaccessible, for whatever reason, then the DNS would stop functioning as and when cached data times out. This is perhaps a dire simplification, but it's enough to get some national strategists all jittery!

The political response has been along the lines of "We need to run our own root service! America has too many, and they should be shared around" While the number 13 is perhaps an arbitrary number these days when counting the number of named root servers, then the number 220 or so too allow a distribution of a named service instance in every CC instance is way too big, and technically it's not easy to understand how we could augment the root service set by more than 200 entries and not require changes to large parts of the DNS infrastructure. Also, I'm not exactly convinced that the penguins on the Heard and McDonald Islands (country code HM) truly need to consider DNS resilience in the form of a dedicated instance of a root zone service!

The engineering response to this issue has been anycast. There may be 13 distinct root server names, and 26 distinct IP addresses (as they all now have IPv4 and IPv6 addresses), there are many more points of service delivery as every service operator now runs an anycast service constellation. Instances of the root service are now located in most economies (although country code HM is still missing out!), and in the larger economies there are multiple instances of services. These days the root service is smeared across much of the Internet with 1,097 service instances and few regions could make the case that they are disadvantaged in this respect (https://root-servers.org).

But it's still not enough.

That's not a technical or engineering statement. It's a political statement.

There's a strong reluctance to unwind history. It starts to ask some questions that are all but unanswerable. "Who's in charge?" is a good place to start. Then we can ask "Who can unravel these historical constraints?", and "Who would be responsible for setting up a new set of constraints?"

If we can't unwind these historical constraints and the anycast approach is not enough to answer some concerns with the current setup what else can we do? Here is where the concept of "Local Roots" comes in. This leverages an old technique in the DNS of so-called "unofficial secondaries" where a copy of a zone is loaded into a recursive resolver and is used to answer incoming queries. For the Root Zone this technique is described in RFC 8806. What this means is that almost any DNS resolver can be configured to operate with a local copy of the root zone and is not dependant on cache lifetimes and constant query level accessibility to the existing root servers. It creates the appearance of a local root service that is not part of the anycast service constellation of the existing root servers. It is the technical answer to the political question, according to proponents of this approach.

But it's clumsy to use.

This presentation described work that is underway to make this local root option easier to use. Dedicated services to publish the root zone via AXFR have been deployed, documentation prepared, and other refinements are proposed (https://localroot.isi.edu). It's nicely thought through and if you are thinking of running a local root zone service on your resolver this is a really good tool to use.

One aspect is still missing here, and that is the assurance of authenticity for the zone as a whole. How can a local root operator assure themselves that the copy of the root zone that they have obtained is genuine? Yes, the zone is signed, but not every element in the zone is signed (NS records, for example) and the client is left with the task of performing a validation of every digital signature in the zone, and at present there are some 1,376 of them. It would be far better to replace this tedious form of element-by-element validation with a single validation, and the work on a message digest for DNS Zones (https://tools.ietf.org/html/draft-wessels-dns-zone-digest-06) makes a whole lot of sense in this context.

But I'm left with the question of why would I want to do this? Is this any more than just a palliative technical response to what is at its heart a political question?

To describe my disquiet here let me digress for a second into delivery logistics. The two extreme approaches are "just in time" and "just in case". "Just in time" tries to deliver goods to consumers at precisely the time they are needed and in exactly the quantity needed. The benefits of this approach are reduction of standing inventory costs and improved capital efficiency, or, in other words, lower costs of production. But "just in time" is susceptible to any interruptions in supply lines. "Just in case" supply lines create a local buffer of input goods so that any interruptions in supply lines can be absorbed in the short term by the local buffer stocks. It also allows a production facility to quickly respond to increased demand without having to immediately call for instant responses in the supply channels. But "just in case" comes at a cost of increased inventory.

The DNS resolution function is operated as a hybrid of these two approaches but biased to a "just in time" model. A recursive resolver will only locally store responses once it has answered the first query by querying an authoritative server directly, and it will use this local store for a limited time "just in case" there are successive queries for the same name. The cost of obtaining the response has already been amortized in answering the initial query, so the local cache is relatively cheap to maintain. It's an approach that has proved to be easy to operate and astonishingly efficient. We've scaled the DNS by many orders of magnitude and still been able to operate the DNS infrastructure efficiently.

So why would anyone want to intrude statically configured zones into resolvers "just in case" they get a query? It has higher overheads, requires more configuration detail and introduces new elements to a DNS resolution service. Why would anyone want to incur this additional cost?

There is another way to achieve a similar technical outcome. The key observation is that the overwhelming majority of queries seen at the roots are junk domain names where the response is "no such domain" (or NXDOMAIN). Conventional caching at a recursive resolver is largely bypassed when the queries use random labels as their top-level domain, and they are passed inward to the root to resolve. Caching of DNSSEC negative response (NSEC caching), as described in RFC 8198 is a highly effective response to this. If the resolver cached the signed NSEC range response, then some 3,000 cache entries would describe the entire root zone. It would be populated on a "just in time" basis, using conventional DNS processing. And of course, it would effective for all DNSSEC-signed zones, not just the root.

If what we are after is faster "NO!" responses form the DNS, then operating a local root is a clumsy and complex technical response in my opinion. DNSSEC validation, coupled with NSEC caching will achieve a similar result with a lot less manual configuration of the resolver.

If what we are after is a technical solution to a political desire to unravel the complex set of interdependencies that make each national realm of the Internet totally dependent on external elements, then the root zone of the DNS is a tiny part of a much larger set of interwoven dependencies.

DNS and UDP Fragmentation

For decades UDP transport has been the heart of the DNS resolution protocol. This lightweight transaction protocol with no session overheads has allowed the DNS to scale without imposing a major cost on the network, on servers nor on clients, and do so while keeping within performance parameters that make DNS name resolution times largely invisible to most users most of the time. However, UDP's strength lies in small payloads. Once the payload is larger than a single IP packet then IP fragmentation is needed. In both IPv4 and IPv6 packet fragmentation is a vexed topic. It's a security nightmare and firewalls often default to "when in doubt, discard!" rules for packet fragments. There is also the issue of IPv6 extension header processing, path MTU discovery and management, ICMP message integrity, to name a few. It's not that IP fragmentation doesn't work all of the time. It can work. Some of the time. The problem is detecting when it doesn't work and figuring out what to do next.

In IPv4 the protocol specification still states that a host is not required to accept an IP datagram larger than 576 octets in size. These days we use a de facto assumption that the maximum Ethernet packet size of 1,500 octets is a universal constraint and our servers are commonly configured with a 1,500 octet MTU. The assumption being that if the system is located behind a constrained network where the MTU is smaller (tunnels are a good example) then it's up to the hosts behind the tunnel to adapt. The rest of the Internet just doesn't care.

So if 1,500 octets is the defacto maximum packet size in today's public Internet, what's the smallest size? In IPv4 its 28 octets, while Ipv6 defines a minimum size of 1,280 octets? (Why 1,280 and not any other number is because of an arbitrary choice in the IPv6 design process. They wanted a number less than 1,500 because of the issue of packet encapsulation, but they wanted it to be substantially larger than IPv4's setting of 28. The number 1,280 came about because it's the sum of 1,024 and 256. Nothing much deeper than that!)

The question is: "What's a reliable setting for an IP packet size in the DNS that maximises the likelihood of delivery?" Or, in other words, What's an EDNS(0) default buffer size setting that can be used with some confidence if we want to avoid packet-size related loss in the network?" The values they tested range from 1,204 to 1472 in IPv4 and 1,220 to 1,452 in IPv6. They tested both a path from the Atlas probe to their test serv er, testing the behaviour of the end host environment and the path from some open recursive resolvers to the authoritative server.

This study used the RIPE Atlas measurement platform, and the previous methodology comments apply here as well.

The results from this study were certainly surprising. They found a drop rate of 19% for 1,500 octet IPv4 DNS packets on the server to probe path, and a 25% drop rate for IPv6 packets of the same size. In IPv4 the drop rate tapered off to a little over 1% at the 1,428 octet size, while the drop rate remained close to 4% for Ipv6 packets greater than 1,280 octets in size. For 1,280 octet IPv6 packets the drop rate was 1.7%. This is a very high drop rate, but it does need to be put into context. There are few DNS transactions outside of DNSSEC-enabled responses where the DNS response size exceeds 1,000 octets. If the responses from the root servers are any guide here, the are remarkably few responses from the root servers that exceed 1,184 octets of payload (https://stats.dns.icann.org/stats/d/wom-ext-5minagg-rssac/rssac?orgId=1). So while the UDP packet drop rate for larger DNS responses being set to stub resolvers is very high according to this study, the number of such responses that would reasonably be expected to occur in the conventional use of the DNS is extremely low. Where this might become a consideration is the scenario where DNSSEC validation is moved from recursive resolvers out to the stub resolvers in large volumes.

When the study tested the path between the authoritative server and open recursive resolvers the picture was significantly improved. The failure rate for 1,500 and 1,492 octet IPv4 packets was around 2.7%. This halved for packets does to 1,260 octets in size and halved again to some 0.7% for packets of size 1,260 or less. Curiously, the IPv6 results showed a similar pattern, but with lower drop drats. Larger (1,492 – 1500 octets) had a 1.4% drop rate, dropping to around 0.7% for sizes larger than 1,280 octets and then 0.4% for smaller packets.

It's challenging to place these results into context. The relatively small set of Atlas probes opens the results up to experimental uncertainty, and it is challenging to calculate this uncertainty level in the absence of much larger scale comparable measurements. It does call into question our defacto assumption that packets of 1,500 octets in size is a robust choice of the default IP packet size for the public Internet, but the limited scope of the experiment sample size makes this conclusion hard to put into context.

The "Core" of the DNS

I recall a presentation I heard some years ago that looked at the resilience of a country's public network services. It pointed out that while there were many agencies and many channels of publication, many of these publication channels used the same hosting provider, the same DNS provider and the same access provider. The underlying infrastructure was in fact built on a restricted set of platforms and the entire public system was more vulnerable than many had supposed.

Ed Lewis is performing a comparable form of study of the DNS top level zone infrastructure. He is assembling a database of elements of this space, including the registries used, the nameservers used and the IP addresses and routing data. He also envisages collecting information on DNSSEC crypto algorithms used to sign the second level zones.

It's all a work-in-progress at this stage so the presentation looked at the methodology of data collection. The interesting aspect of this work lies in the future phase of data analysis. Its abundantly clear that aggregation and the hollowing out of diversity in service provision is happening in many aspects of today's Internet. Without studies such as the one proposed here it is not so clear if the same consolidation is happening in the upper layers of the DNS.

Disclaimer

Author

Geoff Huston AM, B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net