acmqueue

What DNS Is Not

DNS is many things to many people—perhaps too many things to too many people.

Paul Vixie, Internet Systems Consortium

DNS (Domain Name System) is a hierarchical, distributed, autonomous, reliable database. The first and only of its kind, it offers realtime performance levels to a global audience with global contributors. Every TCP/IP traffic flow including every World Wide Web page view begins with at least one DNS transaction. DNS is, in a word, glorious.

To underline our understanding of what DNS *is*, we must differentiate it from what it *is not*. The Internet economy rewards unlimited creativity in the monetization of human action, and fairly often this takes the form of some kind of intermediation. For DNS, monetized intermediation means lying. The innovators who bring us such monetized intermediation do not call what they sell "lies," but in this case it walks like a duck and quacks like one, too.

Not all misuses of DNS take the form of lying. Another frequently seen abuse is to treat DNS as a directory system, which it is not. In a directory system one can ask approximate questions and get approximate answers. Think of a printed telephone white pages directory here: users often find what they want in the printed directory not by knowing exactly what the listing is but by starting with a guess or a general idea. DNS has nothing like that: all questions and all answers are exact. But DNS has at least two mechanisms that can be misused to support approximate matching at some considerable cost to everybody else, and a lot of that goes on.

STUPID DNS TRICKS

The first widespread form of DNS lie was to treat DNS lookups as mapping requests. CDNs (content distribution networks) such as Akamai and Web optimizer products such as Cisco Distributed Director treat incoming DNS lookups as opportunities to direct the activities of Web browsers. Using the IP source address of a DNS request, these products and services try to guess the proximity of the requester to each of many replicated content servers. Based on the measured load of each content server's system and network, and on an estimate of each content server's proximity to that requester, a DNS response is crafted to direct that requester to the closest or best content server for that URI domain.

Problems abound from this approach, but none affects the CDN operator's revenue. First and foremost it is necessary to defeat or severely limit caching and reuse of this policy-based data ("DNS lies"). Caching and reuse, which once were considered essential to the performance and scalability of DNS, would allow a policy-based response intended for requester A also to be seen by requester B, which might not otherwise receive the same answer—for example, when server loads have changed and there's a new balance. The effects of this noncaching are a higher DNS request rate (perhaps leading to higher revenue for CDNs that charge by the transaction) and more network load for access-side networks and a slightly higher floor for average transaction time.

Furthermore, it has never been wise to assume that a DNS request's IP source address gives any

hint of an end-system Web browser's network location. This is because DNS requests heard by a CDN come from recursive DNS servers as a result of cache misses; they do not come from end systems themselves. Some ISPs regionalize their recursive name servers, allowing CDNs to encode rules improving the quality of their estimates. Many recursive name servers are per-country or per-continent or even per-hemisphere, however, so it's always necessary for a CDN to deploy well-connected supernodes, and these always end up hearing a lot of out-of-region requests.

The primary benefit of a CDN is the same as gimmick-free outsourcing: it gives a content owner somebody to sue if things don't go well. That DNS system performance and stability has to pay the price for such liability shielding is at best unfortunate. Given that a CDN still requires supernodes that will hear many out-of-region requests, a gimmick-free approach here would be to answer DNS truthfully and let existing pseudorandom distribution mechanisms do their work. Noting that there is no patent on the existing pseudorandomization technologies and that nobody ever got fired for buying a CDN, we can expect to see ever-more content distributed this way in the decades to come.

NXDOMAIN REMAPPING

Fairly often, as in millions of times per second worldwide, somebody looks up a domain name in DNS that isn't there. Maybe this is a user at a Web browser making a typographical error, or maybe there's a broken link on a Web site, or maybe a hardware or software error is causing nonexistent names to go into DNS requests. One way or another, the answer is generally supposed to be NXDOMAIN (sometimes written as RCODE=3). These negative answers are cacheable, as is any other kind of DNS information, since DNS is designed to express truth, not policy. A network application (perhaps a Web browser, or mail server, or indeed anything at all that uses TCP/IP flows to do its business) that gets back one of these negative responses is supposed to treat it as an error and reject its own underlying work item that led to this lookup. For a Web browser, rejection takes the form of an "error page." For a mail server, rejection takes the place of "bounced e-mail." Every TCP/IP application, large or small, new or old, knows how to cope with NXDOMAIN.

The World Wide Web has changed the rules. Though the Web is young—and though the Internet was here before the Web and will be here after the Web and is much larger than the Web—the fact remains that the Web is what end users are looking at. Advertisers have a whole language to describe the value of end users, with words such as "impressions," "click-throughs," and "eyeballs." Why on earth, these advertisers ask, would you ever send back an NXDOMAIN if an impression was possible? So it is, increasingly, that in place of the NXDOMAIN your application knows how to handle, if you ask for a name that does not exist, you'll get a positive (deceptive; false; lying) answer that your application also knows how to handle.

For example, if I ask my own recursive name server for a name that does not exist, it will tell me NXDOMAIN. If I ask OpenDNS's recursive name server for a name that does not exist, it will send me a NOERROR response with an answer pointing at an advertising server. Note that I'm using OpenDNS as a convenient example; it did not invent this technique. Indeed, Nominum and other DNS vendors now sell an add-on to their recursive name service products to allow any ISP in the world to do this, and a growing number of ISPs are doing it. Why so many? Simply because whoever remaps these NXDOMAIN responses gets the impression revenue. There are unverified claims that some ISPs are blocking access to OpenDNS and/or all non-ISP name servers in order to force their customers to use the ISP's own name server. I say unverified, but I find the claims credible—ISPs

have wafer-thin margins and if they see this kind of manna going out the door, they can't just let it happen.

To demonstrate the extreme desire to capture this revenue, a true story: a few years ago VeriSign, which operates the .COM domain under contract to ICANN (Internet Corporation for Assigned Names and Numbers), added a "wild card" to the top of the .COM zone (*.COM) so that its authoritative name servers would no longer generate NXDOMAIN responses. Instead they generated responses containing the address of SiteFinder's Web site—an advertising server. The outcry from the community (including your humble narrator) was loud and long, and before ICANN had a chance to file a lawsuit to stop this nonsense, many people had patched their recursive name servers to remap any response from a .COM name server that was not a delegation (for example, telling how to find the GOOGLE.COM name servers) back into an NXDOMAIN. Some ISPs put logic into their policy-based routers to turn SiteFinder responses into pointers to the ISP's own advertising server instead.

DAMAGE CONTROL

NXDOMAIN wasn't designed to be a revenue hook—many applications depend on accurate error signals from DNS. For example, consider the "same origin trust model" used for Web cookies. If you're holding a cookie for GOOGLE.COM and you can be fooled into following a link to KJHSDFKJHSKJHMJHER.GOOGLE.COM, and the resulting NXDOMAIN response is remapped into a positive answer to some advertising server, then you're going to send your GOOGLE.COM cookie to that advertising server when you send your HTTP GET request there. Not such a bad thing for a GOOGLE.COM cookie, but a real problem for a BANKOFAMERICA.COM cookie. (Thanks to Dan Kaminsky for telling me about the "same origin trust model" problem.)

Remapping could also cause e-mail to be captured if an MX (mail exchanger) request is captured in this way. Many NXDOMAIN remappers try to avoid this by triggering only on A (address) requests, but to make this work they have to turn off caching, since NXDOMAINs are not type specific and since an SMTP initiator will fall back to type=A if it gets no answer from type=MX. Similar protections (designed to keep lawsuits away while still attracting revenue) include the idea of triggering the remapping logic only if the query domain begins with WWW.—but as far as I know there are a lot of typographical errors beginning WW., or ending with .CM, so I do not hold out a lot of hope for it long term. Too much money is involved, and nobody wants to leave it on the table (where in this case it belongs).

STANDARD BAD PRACTICES

There is at the time of this writing an IETF draft (think: proto-RFC) on the *Recommended Configuration and Use of DNS Redirect by Service Providers* (http://tools.ietf.org/html/draft-livingood-dns-redirect-00). The goal of this document is to present some rules for how DNS lies should be delivered in order to give all the vendors and operators in this growing market a common frame of service. Some Luddites may feel that the "standard best practice" in this area is simply not to do it at all, but this being unrealistic, we now face standards action. As a standard feature of DNS technology we can expect a day to come when all DNS services are delivered this way and our kids think of end-to-end DNS the way they think of eight-track tapes.

This document makes a substantial contribution to the debate around this feature area by suggesting that opt-out for this service should be a network layer attribute (in other words, associated

with one's Ethernet [MAC] address or equipment port number) and not a transport layer attribute. Noting that as with any other kind of information, spam opt-out is not as good for the economy as opt-in, it's valuable to the debate that this IETF draft's authors have said that Web cookies aren't good enough. Others may disagree but at least this point is now on the table. This document also talks about "legally mandated" DNS redirection, which is exactly the nightmare it sounds like it is and which we can all hope becomes a historical curiosity as rapidly as possible.

The absolutely best part of this IETF draft is Section 10 (DNSSEC Considerations), which ends as follows:

"So the only case where DNS security extensions cause problems for DNS Redirect is with a validating stub resolver. This case doesn't have widespread deployment now and could be mitigated by using trust anchor, configured by the applicable ISP or DNS ASP, that could be used to sign the redirected answers. As noted above in Section 9.7, such improper redirection of valid responses may also cause DNSSEC trust verification problems."

A RESCUE BEING THOUGHT OF

Fifteen years ago a bunch of ivory tower theoreticians got together at IETF and said, "Let's secure DNS." The threat model has evolved over time, and now this set of protocol enhancements (DNSSEC) is more or less ready for deployment and more or less allows for the possibility that DNS liars will be caught and ignored. Noting that DNSSEC has taken too long and is a committee-based horror in its inelegance and complexity, here's how it's supposed to work and how it may help curtail the current market in DNS lies.

In DNS, data producers are the authoritative name servers, each of which is the delegated authority for one or more zones. For today, think of a DNS zone as everything at or below a certain name, so, for example, WWW.GOOGLE.COM is in the GOOGLE.COM zone. DNSSEC allows these zones to be signed and verified using public-key cryptography. The private (signing) key is used by the editor of the zone to generate signature records for each set of real records. The public key is used by recursive name servers to verify that the data they receive was signed by the holder of the corresponding private key. Public (verification) keys are published using DNS itself, by including each zone's key in the zone's parent zone—so the public key for the GOOGLE.COM zone is published in the COM zone and so on. I'm deliberately skipping a long and unpleasant story about where the public key for COM is supposed to be published, since it's not germane to this article.

In theory, an end-system owner who doesn't like being lied to can work cooperatively with zone editors who don't like their zones getting lied about if each of them deploys DNSSEC. An application that is supposed to receive an NXDOMAIN but that today receives a pointer to an advertising server would in a DNSSEC world receive a "signature not present" error. This would be an error because DNSSEC has a way to inform a validator that a signature *should have been* present. Note that in the vast majority of cases zone editors don't care whether their zones are being lied about, and, therefore, DNSSEC will remain silent most of the time. Consider also that this was not the original DNSSEC threat model; we really thought we had to stop on-the-wire corruption such as that discovered by Dan Kaminsky in 2008, and the idea of stopping in-the-middlebox corruption such as NXDOMAIN remapping really is just gravy.

DNSSEC will complicate life for CDN providers using Stupid DNS Tricks, but it won't end that war since it's still possible to sign every policy-based answer and keep all the answers and signatures

available, and still send different answers to the same question based on requester identity and policy, and have the signatures all be perfectly valid.

DNSSEC will also complicate life for sysadmins and application developers. We (ISC—the BIND people) are doing what we can to improve on that in BIND9 9.7 (due out by December 2009), and there are plenty of other service and technology providers in the space as well. The killer app for DNSSEC will be a Web browser and Web server that can authenticate to each other without using X.509 (volunteers are hereby encouraged to get together and try to make that happen).

DIRECTORY SERVICES

Browser implementers including Microsoft and Mozilla have begun doing DNS queries while collecting URIs from their graphical front end in order to do fancy "auto-completion." This means that during the typing time of a URI such as http://www.cnn.com/, the browser will have asked questions such as W, WW, WWW, WWW.C, WWW.CN, WWW.CNN, and so on. It's not *quite* that bad, since the browsers have a precompiled idea of what the top-level domains are. They won't actually ask for WWW.C, for example, but they are now asking for WWW.CN, which is in China, and WWW.CNN.CO, which is in Colombia.

Although one simple-sounding solution is for Microsoft and Firefox to buy some name-server hardware and network links for China and Colombia (and no doubt many other affected top-level domain operators), that won't stop the information leak or remove this stupid and useless traffic from the rest of the network. Since the truly best solution is, as usual, *stop doing this stupid thing*—and we all know that isn't going to happen—perhaps this behavior can be made optional, and then we can just argue about what the default (opt-in vs. opt-out) should be. This is the first time in the history of DNS that someone has used it prospectively, to find out if what has been typed is or isn't a valid domain name, in order to support something like auto-completion. As in so many other novel uses of DNS, this is not what it was designed for.

Had DNS been designed with this kind of thing in mind, one of the ways we would be able to tell is that domain names would be written from highest- to lowest-order term (COM.CNN.WWW). This would allow partial name completion just as happens in graphical file system browsers. Absent a complete redesign, which won't happen in our lifetime because of the size and usefulness of the installed base, all we can do is ask browser implementers to be smarter and prepare for more DNS traffic on our networks.

CONCLUSION

What DNS is not is a mapping service or a mechanism for delivering policy-based information. DNS was designed to express facts, not policies. Because it works so well and is ubiquitous, however, it's all too common for entrepreneurs to see it as a greenfield opportunity. Those of us who work to implement, enhance, and deploy DNS and to keep the global system of name servers operating will continue to find ways to keep the thing alive even with all these innovators taking their little bites out of it.

These are unhappy observations and there is no solution within reach because of the extraordinary size of the installed base. The tasks where DNS falls short, but that people nevertheless want it to be able to do, are in most cases fundamental to the current design. What will play out now will be an information war in which innovators who muscle in early enough and gain enough

market share will prevent others from doing likewise—DNS lies vs. DNS security is only one example. Q

LOVE IT, HATE IT? LET US KNOW

feedback@queue.acm.org

PAUL VIXIE is president of Internet Systems Consortium (ISC), a nonprofit company that operates the DNS F root name server and that publishes the BIND software used by 80% of the Internet for DNS publication. He is also chairman of American Registry for Internet Numbers (ARIN), a nonprofit company that allocates Internet number resources in the North American and Caribbean regions. Previously, Mr. Vixie was a founder and president of PAIX, the first neutral commercial Internet exchange; SVP/CTO of AboveNet; and founder of the first anti-spam company (MAPS LLC) in 1996.