Other Uses for Secure DNS

PAUL VIXIE

Paul Vixie took over BIND maintainance after Berkeley gave it up in 1989, rewrote it, and then hired other people to rewrite it again. He has recently hired a new team to rewrite it again. Paul is Chairman and Chief Scientist of the Internet Systems Consortium.
vixie@isc.org

Since the mid-1990s I've been a member of a distributed multi-generational team working to secure the Domain Name System. Many ideas and people have come and gone in the decade and a half since this work began, and the current work in progress now being deployed represents a dozen kinds of compromises and band-aids. Yet Secure DNS is being deployed at last, and a market for products and services in this technology is starting to appear. I think this is a good moment to try to remember why securing the DNS seemed like a good idea and to start thinking about other ways to leverage this fundamental change in Internet architecture.

DNS Itself

The original cost justification for deploying DNS itself was that the old SRI-NIC:HOSTS.TXT file was growing (hundreds of kilobytes) and updates to this file were taking a long time (several days) to propagate through the whole Internet. This file just mapped host names ↔ host addresses so that it would be possible to enter or view a host's name even though the underlying Internet architecture worked in terms of binary addresses. To put all this in context, BSD UNIX systems in the 1980s used to pull SRI-NIC:HOSTS.TXT once a week with a cron job, run the "htable" conversion utility to put it into /ETC/HOSTS format, and append a set of local host names showing local host name ↔ address assignments which weren't considered important enough to be worth sending to SRI-NIC.

When Paul Mockapetris designed DNS he gave it a much broader feature set than host name ↔ address translation. For example, DNS made the MX (mail exchanger) record possible, meaning that we could begin to send email to domains rather than to hosts and to have a domain's incoming mail services be provided by more than one host. These were exciting times since this kind of distributed autonomous reliable hierarchical database had never been done before except as proprietary single-vendor standards. Let the record show, however, that the motivation to deploy DNS was not this broader feature set but only the simple expedient of getting rid of the SRI-NIC:HOSTS.TXT file. In other words, the reason DNS was created is broader than the reason DNS was first deployed, and we only have DNS at all today because there was a reason to deploy it in the first place.

Secure DNS

While each member of the distributed and multi-generational team that developed Secure DNS can speak for him or herself as to their individual motives for participating in the effort, I believe that most of us wanted Secure DNS because it would

enable a whole new class of distributed applications that could offer enhanced behavior in the presence of crypto-authentic DNS data. We learned early on that the BSD ruserok() function and its ".rhosts" file was a terrible idea, since potential attackers were in direct control of the results of the address → name mapping—the session initiator controlled the IN-ADDR.ARPA data for their own TCP/IP source address. In first-generation DNS, all data is potentially under the indirect control of an attacker since anybody can spoof a DNS response in transit. Since an application that depended on DNS for any of its access or control plane data could be no more secure than DNS itself, no applications were allowed to depend on DNS for sensitive data; as a result, there was no sensitive data in DNS. This made early DNS a distributed hierarchical autonomous reliable database full of non-sensitive data—clearly not as useful as it could be.

But whereas the goal for many of us for Secure DNS was to enable a new class of distributed applications that would be able to depend on crypto-authentic DNS data, there were then no such applications nor any way to build them. Therefore, a short-term expedient was needed, something that would cost-justify the design and deployment of Secure DNS. Most of us realized that the short-term goal had to be to secure the DNS infrastructure itself against medium-value threats such as Web or email redirection. Even though attacks of this kind have never really been common we saw some value in ruling them out altogether. This was for a long time considered too weak a justification for the great and global expense of deploying Secure DNS, but in 2008 Dan Kaminsky showed that spoofing DNS responses was far easier than anybody had thought. After what we called the "2008 Summer of Fear," deployment of Secure DNS finally picked up steam. Note, though, that the justification was still just securing the existing DNS and its existing suite of distributed applications. We knew we couldn't sell Secure DNS based on the vaporous sounding promise of "new applications."

SSHFP

The award for "first DNSSEC-enabled application" goes to Secure Shell (SSH) for which a new record type (SSHFP) for host key fingerprints was created. Secure Shell remembers the host key for every server you've talked to in order to prevent server replacement attacks whereby someone steals a server's network traffic. Under normal conditions, when you talk to a new server Secure Shell will prompt you to verify that server's host key fingerprint just to make sure that the server is what you think it is. Many Secure Shell users do not pay much attention to this prompt and just enter "yes" or click OK or similar without ever reading or verifying the moderately long string of hexadecimal. This creates a security problem whereby users have higher trust in a Secure Shell session than they have any rational justification for, and a server traffic thief can do quite well.

Recent Secure Shell versions now look for an SSHFP record in Secure DNS corresponding to the server's host name. If the result is crypto-authentic in Secure DNS and matches the server's offered key, then Secure Shell need not prompt its user to verify this fingerprint. This may seem like a small thing, especially if it had to carry the full cost of designing and deploying Secure DNS, but it is an example of the kind of things that are possible when we can trust the data we get back from DNS. The full cost of Secure DNS need not be justified by any single new application or new feature, and this fingerprint click-through was a legitimate security concern that could only have been fixed by utilizing a secure global public key infrastructure such as Secure DNS.

X.509 and TLS

Transport Layer Security (TLS) is a way to encrypt TCP/IP session data and possibly also verify the identity of the host or user at the other end of a TCP/IP session. Some sessions start out encrypted (as in HTTPS and IMAPS), in which case it's called Secure Sockets Layer (SSL). Other protocols can switch from clear text to encrypted in a negotiated manner, in which case it's called TLS. As usual in such systems, each side has a persistent host or user key pair (called a "certificate") whose public half is sent to the other side during crypto-negotiation so that the private half can be used for generating secure session keys or signatures. The format of the keying information transmitted during TLS negotiation or SSL startup is called "X.509" and it contains, among other things, a signature on the certificate itself by some outside authority. This signature is used to validate the certificate as belonging to the given host or user. And that's where it all goes off the rails.

If you buy a certificate from an authority known to the other parties to whom you wish to speak securely, they can verify the "certificate authority signature" on your certificate and thus decide to trust the certificate—your certificate—that you're presenting to them. The problem is that the "other end" is usually a Web browser and the maker of that Web browser doesn't necessarily know which certificate authorities they should trust, so pretty much (with a few exceptions) everybody just trusts everybody. Noting the low utility of many certificate authorities, quite a few Web and mail server operators decide to just use a "self-signed certificate," where no certificate authority is involved at all. This results in browser popup messages warning of self-signed certificates which browser operators (that is, end users) usually just click through and ignore. To round things out, some recent incidents have shown lax security or lax verification by certificate authorities such that a lot of certificates out there probably should not have been issued but will nonetheless be universally trusted.

The IETF DANE working group has taken on the task of defining a Secure DNS schema for certificate verification. This will be similar to the Secure Shell SSHFP record, where the operator of the Web or mail server generates a certificate and puts the fingerprint of this certificate into Secure DNS, from where it can be fetched and crypto-authenticated by the other end during TLS negotiation or SSL startup. Some important questions remain, such as whether this will someday enable universal self-signed certificates or whether there will always be a market for "certificate authority" services. What's absolutely certain is that there is value in this approach and that Secure DNS—as the first hierarchical autonomous reliable distributed public key infrastructure—is what's going to make it possible.

User Certificates

The Internet has made connectivity almost universal, but there is nothing like a universal identity system. I don't mean in an Orwellian "big brother" sense, don't worry, I don't want that either. I'm simply noting that passwords don't work well at scale—between one set of people forgetting them and resetting them and another set of people guessing and leaking them, we know that a system with hundreds of millions of passwords is inherently not secure and cannot be made secure. In addition, most of us possess dozens of passwords for different online resources and we either write them down or make them easy to remember or use the same password everywhere or never change them or perhaps all of the above. I cannot imagine a

more fruitful electronic crime environment than one in which a billion people do their online buying and selling and banking using passwords.

Happily, Secure DNS will make it possible for any user to create a crypto-authentic anchor for their online identity which could then be the basis for a unified, open, and secure identity system that might in some cases (or on some days or in some places) use passwords, or fingerprint readers, or signature scanners, or near field communications readers, or PIN codes, or challenge and response systems, or whatever those crazy kids in the future will think of. We can't build a system like that without a hierarchical autonomous reliable distributed public key infrastructure. Fortunately, with Secure DNS we now have one of those. In this article, I'm not describing the specifics of what a unified open and secure identity system might look like, merely noting that the first task for the designers of such a system would be to design and deploy something very much like Secure DNS to anchor it all—unless Secure DNS already exists, in which case they can leverage it.

Your Idea Here

DNS in both its original and its new secure form is like a large whiteboard waiting for someone to walk by with a compelling idea. I've told you mine, but I'm actually much more interested in hearing what the rest of the distributed systems community (that is, Internet application developers and creative investors) can think of. Before the Internet the world did not have, and no one really imagined the impact of, universal reachability. Now look. Before Secure DNS the world did not have, and I think no one really imagines the impact of, universal public key infrastructure. Let's find out.

References

RFC 952: "DoD Internet Host Table Specification."

RFC 4255: "Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints."

draft-ietf-dane-protocol-10: "Using Secure DNS to Associate Certificates with Domain Names for TLS."

BIND 9.7: "DNSSEC for Humans," http://www.isc.org/.