Data Distribution With DNSSEC

BY MICHAEL W. LUCAS

One of the hard problems on a public network is trusting identity. How do you know a server is what it claims to be? Within an organization you can have many different solutions, but on the public Internet you have the choice of trusting assorted external entities to verify identity (Certificate Authorities) or hand-crafting your own network of identification (OpenPGP's web of trust). DNS Security Extensions, or DNSSEC, is changing that.

What Is DNSSEC?

The Domain Name Service (DNS) maps hostnames to IP addresses and vice versa. It's why you can type http://www.FreeBSD.org into your web browser, rather than http://8.8.178.110. But DNS was designed back in the early 1980s, and was intended for a much smaller network with a much more limited range of users. At the time, people were just happy to have a protocol that let system administrators enter a hostname and get an IP back. They didn't have to worry about spoofed addresses, malware, and financially motivated intruders. Today, on an Internet riddled with criminals and malware, we need a way to verify that DNS information is accurate.

DNS Security Extensions add cryptographic verification to DNS information. Each stage of the DNS lookup has a digital signature, and local clients can validate those signatures. DNSSEC does not provide confidentiality, as DNS is public data. But DNSSEC makes changes obvious—it makes DNS tamper-evident to anyone who cares to look.

In some ways, DNSSEC combines the Web of Trust with Certificate Authorities. The root zone has a digital signature. Clients must trust that signature. (Yes, there are protocols to replace that signature as in case of compromise.) The root zone gives digital signatures to records for the zones beneath it, and those zones sign the records for zones beneath them. So when you use DNSSEC to look up the IP address for www.FreeBSD.org, the client can get the digital signature on the host www.freebsd.org, and then use the signature inside .org to validate the domain's signature, and finally use the signature on the root zone to validate .org. The client builds a chain of trust, using the trusted key on the root zone to trust the identity of the end site.

If you have to trust an external entity, how does DNSSEC differ from a certificate authority? Go look in your web browser. You'll see dozens of certificate authorities. Some of these organizations are more careful and reputable than others. Many of them have issued certificates to organizations not eligible for them – for example, more than one CA has issued certificates for companies like Google, Apple, and Microsoft to organizations or individuals that shouldn't have them.

Organizations, even careful CAs, fail. With DNSSEC, you don't rely on a third party to authenticate identity.

Also, certificate authorities expect payment for validating identity. That payment might not be much, but when you have dozens or hundreds of servers it can become rather pricey. Many organizations choose to not secure data in transit rather than spend money securing every little connection.

Today, DNS is used for many things besides host and IP address information. DNS offers configuration information to anything from servers to phones, lists valid mail senders for a domain, and more. Once this channel becomes tamper-evident, though, you can start to put public security information in DNS. We'll consider two types of data commonly distributed via DNSSEC: fingerprints for SSH public keys and SSL certificate.

SSH Host Key Fingerprints

SSH uses public key cryptography to verify that the server you're connecting to is actually the server you think it is, and to secure the data exchange between client and server. Correct use of SSH requires that when a user first connects to a server, he must examine the offered public key and compare it to an accurate copy of the public key. This is tedious and annoying, and most SSH users don't bother. Worse, most SSH users quickly decide to ignore public key warnings from SSH, reducing the protocol's security.

Comparing two cryptographic fingerprints is exactly the sort of thing a computer is good at, but until now there was no standard way to securely transmit these fingerprints online. DNSSEC provides that channel through SSH Fingerprint (SSHFP) records. Newer versions of FreeBSD include an OpenSSH client that automatically checks for SSHFP records.

Start by creating SSHFP records for your hosts and insert them into the zone's record for that host.

Run `ssh-keygen -r` to generate SSHFP records from the public key files in /etc/ssh. Use the hostname, as you want it to appear in the SSHFP record, as an argument. Here I create SSHFP records for my web server, www.michaelwlucas.com.

```
$ ssh-keygen -r www
www IN SSHFP 1 1 f44d08efc159…
www IN SSHFP 1 2 86c744ce05ba…
www IN SSHFP 2 1 9af675d68969…
www IN SSHFP 2 2 7914036e9053e14db552…
www IN SSHFP 3 1 0f7c928e3954c54f0b32…
www IN SSHFP 3 2 5c5192e78de10…
```

This displays the SSHFP records for all host keys on the local machine. These records are specific to the local machine—I could run the exact same command on www.FreeBSD.org and would get records with completely different fingerprints.

If a host has multiple names and you might use any of those names to connect to the host, then each host name needs SSHFP records. For example, my web server is also known as pestilence.michaelwlucas.com. I'll need two copies

of these records in my zone, one for each hostname, creating something like this.

```
www IN A 192.0.2.33
www IN SSHFP 1 1 f44d08efc159…
www IN SSHFP 1 2 86c744ce05ba…
www IN SSHFP 2 1 9af675d68969…
www IN SSHFP 2 2 7914036e9053e14db552…
www IN SSHFP 3 1 0f7c928e3954c54f0b32…
www IN SSHFP 3 2 5c5192e78de10…
pestilence IN A 192.0.2.33
pestilence IN SSHFP 1 1 f44d08efc159…
pestilence IN SSHFP 1 2 86c744ce05ba…
```

The hashes are identical for each variant of the hostname.

To make an OpenSSH client check for SSHFP records, set VerifyHostKeyDNS to yes in ssh_config or ~/.ssh/config. ssh(1) will use the SSHFP records to validate host keys without prompting the user. Computers are good at comparison. You aren't. Let them do the work.

DANE and TLSA

DANE, or DNS-based Authentication of Named Entities, is a protocol for stuffing public key and or public key signatures into DNS. As standard DNS is forged easily, you can't safely do this without DNSSEC. With DNSSEC, however, you now have an alternative way to verify public keys. We'll use DNSSEC-secured DNS to verify web site SSL certificates (sometimes called DNSSEC-stapled SSL certificates).

In DNSSEC Mastery I predicted that someone would release a browser plug-in to support validation of DNSSEC-staples SSL certificates. This wasn't terribly prophetic, as several different groups had already started down that road. I'm pleased to report that the fine folks at http://dnssec-validator.cz have completed their TLSA verification plugin. I'm using it in Firefox, Chrome, and IE. One day browsers will support DANE automatically, but until then, we need a plug-in.

DNS provides SSL certificate fingerprints with a TLSA record. (TLSA isn't an acronym; it's just a TLS record, type A. Presumably we'll move on to TLSB at some point.) A TLSA record looks like this:

```
_port._protocol.hostname TLSA ( 3 0 1 hash...)
```

If you've worked with services like VOIP, this should look pretty familiar. For example, the TLSA

record for port 443 on the host dnssec.michaelwlucas.com looks like this: long one

```
_443._tcp.dnssec TLSA ( 3 0 1
4CB0F4E1136D86A6813EA4164F19D294005EBFC02F10CC400F1776C45A97F16C)
```

Where do we get the hash? Run openssl(1) on your certificate file. Here I generate the SHA256 hash of my certificate file, dnssec.mwl.com.crt.

```
# openssl x509 -noout -fingerprint -sha256 < dnssec.mwl.com.crt
SHA256 Fingerprint=4C:B0:F4:E1:13:6D:86:A6:81:3E:A4:16:4F:19:D2:94:00:5E:BF:C0:2F:10
:CC:40:0F:17:76:C4:5A:97:F1:6C
```

Copy the fingerprint into the TLSA record. Remove the colons. That's it.

Interestingly, you can also use TLSA records to validate CA-signed certificates. Generate the hash the same way, but change the leading string to 1 0 1. I'm using a CA-signed certificate for https://www.michaelwlucas.com, but I also validate it via DNSSEC with a record like this.

```
_443._tcp.www TLSA ( 1 0 1
DBB17D0DE507BB4DE09180C6FE12BBEE20B96F2EF764D8A3E28EED45EBCCD6BA )
```

So: if you go to the trouble of setting this up, what does the client see?

Start by installing the DNSSEC/TLSA Validator (https://www.dnssec-validator.cz/) plugin in your browser. Hopefully, by the time this article reaches you it will be an official FreeBSD port. In the meantime, Peter Wemm has built the Firefox version of the plugin on FreeBSD, and he has a patch (http://people.freebsd.org/~peter/MF-dnssec-tlsa_validator-2.1.1-freebsd-x64.diff.txt) and a 64-bit binary. (http://people.freebsd.org/~peter/MF-dnssec-tlsa_validator-2.1.1-freebsd-x64.xpi) If you're looking for a way to contribute to FreeBSD, porting this would be very useful.

The plugin adds two new status icons. One turns green if the site's DNS uses DNSSEC, and has a small gray-with-a-touch-of-red logo if the site does not. Not having DNSSEC is not cause for alarm. The second icon turns green if the SSL certificate matches a TLSA record, gray if there is no TLSA record, and red if the certificate does not match the TLSA record.

Should you worry about that self-signed certificate? Check the TLSA record status. If the domain owner says "Yes, I created this cert," it's probably okay. If the self-signed cert fails TLSA validation, it's a self-signed certificate: probably okay on a mailing list archive, not okay for your bank.

You can use a variety of hashes with TLSA, and you can set a variety of conditions as well. Should all certificates in your company be signed with RapidSSL certs? You can specify that in a TLSA record. Do you have a private CA? Give its fingerprint in a TLSA record. If you want to play with these things, check out RFC 6698 or my book "DNSSEC Mastery."

I have had some issues with the plugin on my laptop after suspending it. My home and office both perform DNSSEC validation. When I travel to the coffee shop and resume, the plug-in causes performance issues. Restarting the browser solves them. I expect this to improve quickly, however, and it might be a non-issue before you read this article.

DNSSEC gives you an alternate avenue of trust, outside of the traditional and expensive CA model. Spreading TLSA more widely means that you can protect more services with SSL without additional financial expenses.

Of course, you cannot deploy either of these services without working DNSSEC. DNSSEC isn't that hard these days—if I can do it, you can too.

Michael W. Lucas is the author of *Absolute FreeBSD, Absolute OpenBSD,* and *DNSSEC Mastery,* among others. He lives in Detroit, Michigan, with his wife and a whole mess of rats. Visit his website at https://www.michaelwlucas.com.