

DNSSEC primer

Answers You Can Trust

One of the largest holes in Internet security is finally plugged. *By Kurt Seifried*



I hope by the time you are reading this article that the DNS root domain (.) will include signed data for all the top-level domains (.nl, .org, etc.). Why? Because once that is done, you will finally be able make DNS queries and actually validate that the answer you received is correct.

As I wrote back in 2008 [1], DNS has some pretty severe and unfixable flaws (e.g., the transaction IDs are too short) that make it easy for attackers to inject malicious data into DNS servers (“cache poisoning”). If you are using a wireless network and the attacker is local, he or she can simply view your DNS requests and send a response before the server does. With London about to be covered in WiFi for the 2012 Olympics, more and more WiFi usage is inevitable.

Situations like this are exactly where DNSSEC comes into play. The primary goal of DNSSEC is to authenticate data and provide proof that the data are correct and have not been spoofed, tampered with, or otherwise manipulated while in transit. DNSSEC does this through classic private/public key cryptography. The premise is simple: Given a public key, you can verify the correctness of data signed with the corresponding private key. This works especially well

with DNS, because we rely on a chain of data that ultimately originates from the root (.) DNS zone. Note that I’m not going to cover how to set up DNSSEC for a domain. That has been documented to death already [2]; instead, I’m going to focus on the client side.

How DNSSEC Works

DNSSEC is simple in theory and a bit messy in practice (like most things). If you request DNS information for *www.example.org*, the DNS server handling your query will first consult its (.) zone file, which contains static data for the root servers. Using this, it will then contact a root server and ask for information on the .org domain. If the DNSSEC OK bit (00) is set in the OPT record portion of the query, then the server should reply with signed .org data. But how does your DNS server verify these data? How does it get the public key for the root domain? Much like the (.) zone file, it has DNS servers that will include a copy of the key (on OS install media, etc.), solving the chicken and egg problem of how to get a copy of the public root key securely (in practice, more than one key is likely).

Given this root public key, your DNS server can now verify that the data sent are correct. Now you can make a request to the *example.org* DNS servers knowing that you have the correct location for them. When the *example.org* DNS server replies with the signed information for *www.example.org*, you can then verify it using the public key for *example.org*. But how does your DNS server get the *example.org* public key? Just request it from the .org name servers; again, because it is signed, you can verify that the key is correct. The end result is a chain of verification and trust that resides in the root key installed on your system (which presumably got there in a secure manner – e.g., from your install CD).

What DNSSEC Won’t Do

Because of its name, people often expect DNSSEC to secure DNS completely. Un-

KURT SEIFRIED

Kurt Seifried is an Information Security Consultant specializing in Linux and networks since 1996. He often wonders how it is that technology works on a large scale but often fails on a small scale.

fortunately, DNSSEC doesn't do this; it only allows you to check the correctness of answers – nothing more, nothing less. For example, attackers can still view DNS traffic (it is only signed, not encrypted). Attackers can also register domains that look like legitimate domains (e.g., *www.yourbank-newsite.com*) and send email claiming that you need to log in and verify all your account information.

Things DNSSEC Will Make Possible

When I said that DNSSEC only allows you to verify that answers are correct, I lied a little bit. DNSSEC also will provide the ability to bootstrap other services that rely on public/private key encryption to provide authentication and encryption.

CERT records can be used to store PKIX, PGP, and other types of certificates; IPSECKEY records can be used to store private IPSEC public keys, and so on. All this can be done currently, but the risk of an attacker manipulating these records and intercepting sensitive information is significant without DNSSEC.

SSH server key management, for example, can finally be addressed securely and easily by publishing the public SSH key in DNS with the SSHFP record type and enabling `VerifyHostKeyDNS` in your `ssh_config` file.

Also, many ISPs take failed DNS queries and send a response to the client anyway (often directing them to a generic search page). In these cases, DNSSEC will prevent such action by allowing a domain to send an authenticated response indicating there is no such record (so if the ISP decides to send a fake record, you will know and your client can ignore it).

Checking Your ISP for DNSSEC Support

On May 19, when I last checked, both of my ISP's resolving name servers failed to support the large reply sizes needed by DNSSEC. Wait a second. Large reply sizes? Because DNSSEC needs to move around signed records and signed public

keys, a DNS response can easily be several kilobytes in size.

Many DNS servers only support smaller DNS reply sizes (because of

"The primary goal of DNSSEC is to authenticate data and provide proof that the data are correct."

badly configured servers, firewalls, etc.). If a DNS server truncates replies, then when you make a DNSSEC request, it could get chopped off, which will render it useless. Fortunately, RIPE has set up a test you can use to check. With the use of either the `dig` utility or a Java application [3], do:

```
dig txt test.rs.ripe.net ➤
@your.dns.ser.ver
```

If you see a response like this,

```
rst.x1399.x1218.x1003.rs.ripe.net. ➤
48 IN TXT
"10.2.3.4 DNS reply size limit is ➤
at least 1399 bytes"
```

then your DNS resolving server will have trouble supporting DNSSEC. However, if you see

```
rst.x3839.x3833.x3828.rs.ripe.net. ➤
54 IN TXT
"10.1.2.3 summary bs=4096,➤
rs=3839,edns=1,do=1"
```

followed by several pages of records, then you will be able to use DNSSEC with no problems.

Setting Up DNSSEC Support

So, what if you want to use DNSSEC but your ISP isn't supporting it yet? You can set up your own DNS resolver. On most Linux systems, this consists of installing a package such as `caching-nameserver` or `BIND`.

On most recent versions of Linux (e.g., Fedora 12), configuration for DNSSEC support is enabled by default. Assuming the root key is signed by the time you read this, you will need to modify `named.`

`conf` to include the following in the options:

```
dnssec-enable yes;
dnssec-validation yes;
```

Also, you'll need the root-signing key, which will probably look something like:

```
trusted-keys {
"." 257 3 5 "AwEDHFA234...";
};
```

The `dnssec-validation yes;` directive will cause BIND to check for DNS signatures, and, should a signature fail or not be appended properly, the server will reply to the client with a `SERVFAIL` (server failed) response. This will prevent the client from connecting to a potentially hostile site.

Checking dig Output

Once DNSSEC is enabled, how do you test it? With `dig`, of course:

```
dig +dnssec example.org
```

Check the output flags, and, if the `ad` flag is present, that means you have authentic data and DNSSEC is indeed working.

The Future

Now that DNSSEC finally (or shortly) will be rolled out for the root zones [4] and several of the top-level domains (like `.org`), you need client software and libraries to set the DNSSEC OK bit in outgoing requests. Or, configure your DNS servers to use the `dnssec-must-be-secure` setting to require DNSSEC for specific domains that must be secure. In the short term, this is probably the best approach (because DNSSEC currently is enabled on very few domains). ■■■

INFO

- [1] "DNS Attacks" by Kurt Seifried, *Linux Magazine*, October 2008, pg. 60, <http://www.linuxpromagazine.com/Issues/2008/95/DNS-ATTACKS>
- [2] DNSSEC and BIND: <http://www.isc.org/software/bind/dnssec>
- [3] DNS reply size issues: <http://labs.ripe.net/content/testing-your-resolver-dns-reply-size-issues>
- [4] Root DNSSEC: <http://www.root-dnssec.org/>