

## Re: FreeBSD 7.1 and BIND exploit

---

*Source:* <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/stable/2008-07/msg00439.html>

---

- *From:* Doug Barton <[dougb@xxxxxxxxxxxx](mailto:dougb@xxxxxxxxxxxx)>
  - *Date:* Tue, 22 Jul 2008 22:40:03 -0700
- 

Lots of good discussion on this thread, I'm going to cherry-pick some things to respond to.

Kevin Oberman wrote:

And, if you are not sure how good a job it does (and I am not), you should use the OARC test to check how well it works: `dig +short porttest.dns-oarc.net TXT`

If the result is not "GOOD", it's not good enough.

You can test a remote server by adding "@remote-server" to the dig command. The server may be specified by name or IP address.

This is good, I would not specify a name server to test by hostname though, you're hitting a chicken/egg problem if you do. There is also <https://www.dns-oarc.net/oarc/services/dnsentropy> if your desktop is behind the resolver you want to test.

Paul Schmehl wrote:

I just played around with it recently. It's not that easy to understand initially \*and\* the trust anchors thing is a royal PITA.

The trust anchors thing is made mega easier if you use the DLV service provided by ISC. If you want to configure specific trust anchors yourself, you really need to know what you're doing first.

Once you implement DNSSEC you \*must\* generate keys every 30 days.

You are talking about signing a zone you are authoritative for. I am talking about enabling DNSSEC for the named that comes with FreeBSD and by default is configured as a resolver only.

## Re: FreeBSD 7.1 and BIND exploit

So, I think, if you're going to enable it by default, there needs to be a script in periodic that will do all the magic to change keys every 30 days.

I would certainly never write, and I can't imagine that I would ever approve such a script. There are too many moving parts to setting good DNSSEC key policy for me to be comfortable with a cookie-cutter approach.

Jeremy Chadwick wrote:

For what it's worth, I went looking into DNSSEC last week, and after a few hours of skimming then reading, I concluded it's over-engineered and adds too much hassle for it to be considered a worthwhile "upgrade".

I could certainly sympathize with an argument that parts of the DNSSEC spec are over-engineered. The problem however is that the DNS, more than any other Internet system that I can think of, has so many corner cases that they almost seem to be the norm sometimes, and more than any other service it has been engineered to accommodate them. Personally I was in favor of the "flag day" approach to DNSSEC deployment 7 years ago, and was told "that's not how we do things in DNS." The corner cases have gotten worse since then, not better.

DNS, for most people, is expected to be a "simple thing".

And back in "the good old days" it could be. Those days are over. (Arguably they ended in 1995, but I digress.)

There also appears to be an assumption made throughout all of the documentation that I've read: that your recursive and non-recursive DNS servers are separate. I'm still trying to understand why that assumption is made;

Because we've been telling people to do that since, oh, say, the '90's? There are very good reasons to do that, I'm not going to belabor them here. The simplest way to understand the justification is that by their very nature authoritative service operates in the UNtrusted world, and you would like your name resolution to happen in a more or less TRUSTED way. The two goals are antithetical.

Matthew Seaman wrote:

Forgive me for being obtuse. What I meant was the capability to enable checking signatures on DNS RRs as a routine effect of

## Re: FreeBSD 7.1 and BIND exploit

getnameinfo() etc. by modifying resolver(3) routines or similar locally, without needing a DNSSEC enabled recursive resolver listed in resolv.conf? I've a feeling the answer is no, but I haven't been able to find anything definitive.

Ok, how is this: no. :) But seriously folks, the problem here is that you're talking about 3 completely separate spheres of operation, which for 99.9% of Internet users are all operating under the control of different people. I'll take it as a given that the members of this list understand what authoritative name service is, and that in order for DNSSEC to work you have to sign the zone that is on the authoritative name server. Interestingly enough, that is the easy part.

The second "sphere" is the resolving name server (often erroneously referred to as a caching server) that goes out into the wide world and gathers answers to queries generated by users on its network.

The third "sphere" is the stub resolver located on your little computer. What you're asking is, can we have a stub resolver in FreeBSD that will "do the DNSSEC thing" regardless of what kind of resolving name server it's sitting behind? (Please note that I and others have made the argument for a long time now that there is actually a fourth sphere, the application itself. More people agree with that idea now than did in the past, but the question is still what to do about it.)

So, here is the problem (actually, problemS) with that. First, the authoritative server is just spitting out bits, it really has no knowledge of what it's sending (let's leave nasty stuff like NSEC out of the picture). The way things stand now, it's the resolver's job to validate that the signature on the data is good, similar to doing "gpg --verify \$blah." If the signature validates, then there are no "issues," you just pass the answer back to the stub and you're done.

But (and here is where the corner cases start to come in), what if the signature is valid, but the key is expired? Or, what if the signature doesn't validate at all? How much do I care? If you're talking about me surfing over to my bank's website, I care quite a bit! If you're talking about surfing to my webcomic sites, it would be nice if the signatures were good, but it might also be fun to go to a pseudo-random site that is supposed to be my favorite webcomic. (Or, I might know that the admin of the webcomic site is a doofus, and regularly screws up his zone editing process thereby making the signatures invalid for N period of time, and I can live with that.)

In a world where the root and \$YourFavoriteTLD are both signed, it is theoretically possible that you could engineer a system where your stub gets passed the data it needs (RRs, signatures, keys, etc.) AND is capable of doing the validation, but that's really, really unlikely to ever happen. (Thus my "no" answer above.)

Now, if you'd really like your head to hurt, start thinking about ways that the resolving name server can signal the stub about the relative validity of the data it's passing (signed/unsigned/signed but bad sig/signed w/expired key ...), and then if your head doesn't hurt enough already, start thinking about the fourth sphere above, what is the application going to do with that knowledge? (Forget what the user is going to do with it, they already blow right past the SSL warnings, no reason to believe that this will be any different.)

For now, the "solution" is to configure the resolver to know about zones that must be signed in order to pass answers back to the stubs. I don't think anyone thinks that's even the tip of this particular iceberg though.

hope this helps,

Re: FreeBSD 7.1 and BIND exploit

Doug

--

This .signature sanitized for your protection

---

freebsd-stable@xxxxxxxxxxx mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-stable>

To unsubscribe, send any mail to "freebsd-stable-unsubscribe@xxxxxxxxxxx"