Monitoring DNSSEC zones: what, how and when?

Stéphane Bortzmeyer
AFNIC
Immeuble International

78181 Saint-Quentin-en-Yvelines
France
bortzmeyer@nic.fr

***Abstract*—Unlike the traditional DNS, where zones, once properly configured, stay OK forever, DNSSEC introduces much more possibilities of failure. A zone can be perfect at one time and, without human intervention, stop working later. Permanent and automatic monitoring is therefore a necessary part of DNSSEC operations. But why do we still see DNSSEC problems such as signature expirations or missing keys and why are they often detected by a post on public mailing lists rather than by monitoring? As of today, only a minority of top-level domains operators can say they never had a DNSSEC validation failure.**

Because monitoring is not perfect today, we need to think about how to monitor and what to monitor. Among various issues, there is a pressing need for a time-aware monitoring, which keeps track of the situation of the past, to be able to deduce the possible current state of DNS caches. We developed a new tool to monitor the timing of key rollover events and ran it against several domains, showing that timing errors are not an exception.

I. Introduction

The Domain Name System (DNS) allows clients, the resolvers, to query servers about resource records of various types[1] Records for a same name and of a same type make a resource record set and some operations (such as signature, see later) handle a set as atomic. DNS relies heavily on caching, for performance, and the time during which you may cache is indicated by the server, as a Time To Live (TTL).

Because DNS has several vulnerabilities, allowing to insert fake data ([AA04] and [HvM09]), the DNS SEcurity Extensions (DNSSEC) have been developed to allow cryptographic signing of resources record sets ([AAL$^+$05a], [AAL$^+$05c] and [AAL$^+$05b]). This makes modification of data easily detectable.

DNSSEC use assymetric cryptography and the public keys are published in the DNS itself. Signatures are also published as resource records, with a normal TTL. They also have an expiration date, to prevent replay. Good cryptographic practices suggest to replace the keys from time to time[2] Such key rollovers are quite delicate since old keys and signatures staying in the caches at the resolvers may easily prevent proper validation of data. Hence the importance of timing in key rollovers ([KG06]) and the existence of several tools[3] to automate this process.

As a result, breakage of domains by DNSSEC validation errors happened in many signed TLD and other domains. See for instance the use of wrong keys in .UK in september 2010[4], the expiration of signatures in .GI in january 2011[5] or the use of an invalid signature for a NSEC3 record in .FR in february 2011[6].

II. Monitoring

Network and systems monitoring is typically done today for basic services (host up/down, DNS servers replies/does not reply) but not always for more complicated things such as signature expiration. Until recently, "simple" failures like a signature expiration were still able to sneak under the radar and be undetected until an embarassing message appears on public mailing lists, such as OARC[7]'s dns-operations, asking "Does anyone know what is going on with .EXAMPLE?". Of course, we can hope that these "simple" failures will be less and less common with time, when tools get better and system administrators more aware of the issue. But we cannot be confident that this category of bugs has completely disappeared.

So, what are today the best practices in DNSSEC monitoring? Most DNS operators use a tool which perform automatically various tests at regular intervals, and raise an alarm if a test has failed repeatedly. The best known of these tools is Nagios[8] but there are many others. All these tools come with predefined tests but DNSSEC testing is not yet part of this set. In that case, the system administrator installs plugins, which perform the tests, and the tool just acts as a scheduler, running the test at specified times, and triggering alarms if needed. The Nagios API for these plugins is widely used and is recognized by many programs. Since no monitoring tool comes with "out of the box" DNSSEC tests, many plugins were developed and are available on the network such

[1] The two types most often queried are A - IPv4 addresses - and MX - mail servers for a domain.

[2] Although it is not clear that the risks and problems associated with this practice are worth it. But everyone does it.

[3] Such as OpenDNSSEC http://www.opendnssec.org/.

[4] http://blog.nominet.org.uk/tech/2010/09/24/dnssec-incident-report/

[5] http://dnssec-deployment.org/pipermail/dnssec-deployment/2011-January/004719.html

[6] https://www.dns-oarc.net/files/workshop-201103/DNSSEC_Key_Deletion_Issue-Vincent_Levigneron-afnic.pdf

[7] DNS Operations, Analysis and Research Center https://www.dns-oarc.net/

[8] http://www.nagios.org/

as http://dns.measurement-factory.com/tools/nagios-plugins/check_zone_rrsig_expiration.html, http://dave.knig.ht/check_full_zone_rrsig_expiration/, https://dnssec.surfnet.nl/?p=562, http://svn.durchmesser.ch/trac/check_dnssec/, etc.

What are the specific DNSSEC tests to perform? The most obvious is that the authoritative name servers all serve the DNSKEY and transmit signed data. As far as we know, no TLD ever went the DNSSEC way with a missing name server but anecdotal evidence indicates that it happened with lower zones.

After that, of course, the first test which comes to mind is signature expiration which is still, at the date of the paper, the most common failure. But testing that the authoritative name servers does not serve expired signatures, while a first step, is not sufficient: when the signature is expired, it is already too late! Even if some resolvers (for instance Unbound) accept signatures for some time after the expiration, you cannot rely on it, specially since caching means that the expired signatures may stay for some time in the resolvers. You therefore need to be proactive, something that most of the above tools do, by testing that a signature is "about" to expire ("about" being configurable).

In the spirit of "you can debug DNSSEC with only dig and date", the Unix shell script at http://www.bortzmeyer.org/files/check-sig.sh illustrates this method: it adds a given amount of days to the current time, retrieves the signatures, and test if their expiration time is below the above time. If not, it warns that the signature is too close from expiration. Such a very simple tool can easily be integrated into a monitoring solution.

Other common tests are general consistency (that the key used for signing is in the keyset, that the signatures are valid, etc), and, for some tools, that the chain of trust from a given trust anchor (such as the key of the DNS root) can be followed.

These tools are developed for the specific purpose of monitoring a DNSSEC server. But the simplest of all tests is to set up a validating resolver and to test your zones through it. Validating resolvers, by default, returns an error code (SERVFAIL, for Server Failure) is validation fails. You then just have to test for this code. To be sure that the trust anchor has not been deleted from the configuration, you can even test that the reply comes with the AD (Authentic Data) bit set, indicating a positive validation[9]. The strength of this method is its simplicity: no programming, except a very small wrapper which sends the request and test the result.

While it is a very useful method (and we use it to monitor continously the .FR name servers, using the Unbound resolver), its main weakness is that the result depend on the state of the resolver's cache. For instance, if the key used for the signatures is no longer in the authoritative name servers but is still in the cache, thanks to its TTL, this test will wrongly report a success. Addressing this weakness is covered in the next section.

Some tools integrate many different tests and therefore try to offer a comprehensive testing of the DNSSEC setup. They often provide a Web interface and sometimes a command-line interface, suitable for automatic periodic testing. The best known tools today are:

- IIS' DNScheck http://dnscheck.iis.se/
- Verisign's DNSSEC debugger http://dnssec-debugger.verisignlabs.com/
- Sandia Laboratories' DNSviz http://dnsviz.net/
- AFNIC's Zonecheck http://www.zonecheck.fr/

Some can be used for monitoring (such as Zonecheck at AFNIC which is run periodically against .FR).

III. Time-aware monitoring

There is apparently today no tool that can check that a DNSSEC zone is always consistent. By "consistent", we mean:

1) every RRSIG still in the caches of the resolvers is made by a key which is currently in the zone,
2) and that the key used for the current signatures is in all the key sets still cached.

As a result, it is quite possible that some DNSSEC zones have sometimes "windows of vulnerability" where, for instance, a cached RRSIG has no key in the authoritative servers, but these windows are not detected because there are very few validating caches: if they are all lucky, nobody will notice the problem. .FR had the problem once[10]. It was detected because it lasted too long but a similar problem of a short duration could have been unnoticed. To see if these problems occur, we need a new tool.

Such a tool cannot be a one-shot testing tool like are most of the DNSSEC Nagios plugins discussed in the previous section. It needs to have a memory, to remember the signatures that were present before, as long as their TTL is not over.

We developed such a tool. Its algorithm is:

1) Runs the tool periodically (a good interval is a small fraction of the TTL of RRSIG and DNSKEY RRsets).
2) When the tool runs for the first time, it stores in a database various RRSIGs, the keys they use and the TTL. Same thing with the DNSKEY record.
3) Each time the tool runs, it:
   - checks that, for every RRSIG still possibly in the caches, the authoritative name servers *currently have* the key (or the complete key chain, if we want to test from the DS in the parent),
   - checks that, for every DNSKEY still possibly in the caches, the *current* signatures use only keys in this set.
   - updates the database

What resource record type to use to get the signatures? For a general-use tool, SOA, NS and DNSKEY are the only one we are sure they exist. But signatures on DNSKEY are special (they are often done with two keys or, of only one, by the KSK, not the ZSK). Currently, we record signatures on

[9] With the NSEC3 and opt-out option, it does not work for negative answers, such as "this domain does not exist", for which the AD bit is not set.

[10] http://operations.afnic.fr/en/2010/11/22/dnssec-validating-resolution-problem.html

both the SOA and the DNSKEY but, for the analysis, we use only the signatures on the SOA. It may not be sufficient (for instance, if every resource record set is signed independently, some signatures may be right for one type but not for the other) but there is little choice. If you accept to modify the code for specific domains, you can use a record which is present in these domains[11]

It would be interesting too to have the signatures of the "non-existing domain or data" records, NSEC or NSEC3. Again, there is no perfectly sure way to get them for any domain: you cannot query them directly so you have to query a domain that you believe does not exist (for instance by using a long random name[12]) or a type which is so uncommon that it probably does not exist. It is not perfect but it would extend the monitoring to these NSEC* records.

The security of a delegation, with DNSSEC, is done with a DS[13] resource record in the parent zone. This DS is signed by the parent. The same issues may occur when replacing it with a new one and it would be interesting to monitor it as well.

One may wonder if we should use the TTL or the signature expiration date in the calculations. [AAL+05a], sections 6 and 8.1 say the resolver must use the first value (either the expiration of the TTL or the expiration of the signature). In practice, signature durations are almost always much larger than TTL so the current version of the tool takes only TTL into account.

The current tool was developed in Python[14], using the excellent dnspython package[15] to retrieve and parse DNS resource records. The signatures and keys are stored in a SQLite[16] database. In the database schema, there is one table for the signatures and two for the keys, one for the keys themselves and one for the key sets. That is because some information is attached to the key, such as the cryptographic algorithm which was used, and some is attached to the key set, such as the TTL. The caching works per set, not per record, hence the need to store the set itself.

Two instances of the program are running, one for monitoring of the keys (addition and deletion), one as a Nagios plugin, to be used from a monitoring system (most of existing monitoring tools can use plugins written for Nagios). Various utility programs allow to display things such as the history of keysets for a given zone:

```
% ./keyset-zone.py cz
#1 of cz.: ['34702', '14568*']
         (first 2011-01-31 21:23:50Z,
          last 2011-03-08 07:16:47Z)
#2 of cz.: ['34702', '14568*', '400']
         (first 2011-03-08 08:14:50Z,
          last 2011-03-15 16:14:57Z)
#3 of cz.: ['14568*', '400']
         (first 2011-03-15 17:16:28Z,
          last 2011-03-22 14:16:44Z)
```

You can see above a key rollover in two steps. On March 8th, the ZSK 400 is introduced, on March 15th, the ZSK 34702 is deleted.

Since it relies on a SQL database, you can also explore the tables with regular SQL commands, here for the signatures:

```
sqlite> SELECT first_seen,last_seen,ttl FROM Signatures
          WHERE type=6 AND name='192.in-addr.arpa.'
                AND key_tag=20918 ORDER BY last_seen DESC;
2011-03-28 17:29:30|2011-03-28 20:17:31|86400
2011-03-28 13:22:23|2011-03-28 16:25:05|86400
2011-03-28 09:19:59|2011-03-28 12:28:09|86400
```

And here for the keysets, then for the keys:

```
sqlite> SELECT first_seen,last_seen,ttl,id FROM Keysets
          WHERE  name='192.in-addr.arpa.'
          ORDER BY last_seen DESC;
2011-03-29 09:38:45|2011-03-31 08:30:30|14400|J/dCsFib6kxRer/
2011-03-21 21:39:09|2011-03-29 08:38:16|14400|NgM4JKT7QacTgX+

sqlite> SELECT first_seen,last_seen,key_tag FROM Keys
          WHERE  name='192.in-addr.arpa.'
          ORDER BY last_seen DESC;
2011-03-01 15:34:17|2011-03-31 08:30:30|39318
2011-03-21 21:39:09|2011-03-31 08:30:30|60494
2011-03-01 15:34:17|2011-03-29 08:38:16|20918
```

The tool is published under a free software licence[17] at https://github.com/bortzmeyer/key-checker.

IV. Results

The program started to run at the end of January 2011. 54 domains are monitored, both top-level domains and "important" domains at the second or third level. As suspected, several of them experienced rollover problems, which apparently have not been detected by other means. It seems to indicate that the tools currently use to manage the keys are far from perfect.

Here are examples of the two possible problems, new keys used too soon and old keys deleted too soon, with the guinea pig domain office--enregistrement.fr. Here are the keysets before and after a careless key rollover:

```
% ./keyset-zone.py office--enregistrement.fr
...
#3 of office--enregistrement.fr.:
          ['15856*', '1611', '56550', '36216']
          (first 2011-02-08 11:33:00Z,
           last 2011-02-27 14:28:42Z)
#4 of office--enregistrement.fr.:
          ['15856*', '37155']
          (first 2011-02-27 15:22:41Z,
           last 2011-03-22 13:21:35Z)
```

The KSK 15856 stayed but the entire set of three ZSK was replaced by a new ZSK, without concern for the proper key rollover timing. And here is the analysis:

```
% ./examine-history.py office--enregistrement.fr
ERROR: signature of zone office--enregistrement.fr.
       last seen at 2011-02-27 14:28:42
       (with a TTL of 3600)
```

[11] For instance there is often a TXT record at the apex, with various human-readable information, to help debugging.

[12] It will not work if the zone contain wildcards.

[13] "Delegation Signer", a cryptographic hash of the public key of the child zone

[14] http://www.python.org/

[15] http://www.dnspython.org/

[16] http://www.sqlite.org/

[17] BSD three-clause, allowing pretty much every use.

```
       while the key 1611 was retired
       at 2011-02-27 15:22:41
...
```

Here, the signatures were made by the key 1611 and the key was deliberately withdrawn fom the DNSKEY set[18] without waiting for the signatures and the caches to expire.

Second example:

```
% ./examine-history.py office--enregistrement.fr
...
ERROR: signature of zone office--enregistrement.fr.
       first seen at 2011-02-27 15:22:41
       while the last keyset before key 37155
       was last seen at 2011-02-27 14:28:42
       and its TTL was 3600
```

Here, in the same rollover, the new key was immediately used for signatures, while the former key set was still possibly in some caches.

Do this sort of problem actually occur in the wild? Unfortunately, yes. See for instance this rollover:

```
% ./examine-history.py noaa.gov
ERROR: signature of zone noaa.gov.
         last seen at 2011-02-18 11:17:47
         (with a TTL of 86400)
         while the key 23826 was retired
         at 2011-02-18 21:26:20
```

Because of the TTL, the key should have been published until February 19th at noon. Because of the early retirment of key 23826, unvalidatable signatures have been available in caches for up to 14 hours.

In the same rollover, another problem occured:

```
ERROR: signature of zone noaa.gov.
      first seen at 2011-02-18 21:26:20
      while the last keyset before key 52668
      was last seen at 2011-02-18 20:27:06
      and its TTL was 86400
```

The new key 52668 was immediately used while it should have been published but not used for 24 hours[19]

From the sample of tested domains, it seems that the first error (key retired while signatures were still in some caches) is the most common. Here is another example:

```
% ./examine-history.py isoc.org
ERROR: signature of zone isoc.org.
        last seen at 2011-03-01 21:31:08
        (with a TTL of 86400)
        while the key 41414 was retired
        at 2011-03-02 10:30:13
```

The tool was again too eager to delete the key (12 hours while the TTL was 24 hours).

The problem can also happen to TLD. Here, with the ccTLD of Bulgaria:

```
% ./examine-history.py bg
ERROR: signature of zone bg.
       last seen at 2011-03-19 18:14:39
       (with a TTL of 345600)
```

[18]The monitor program runs once every hour so the times are approximate.

[19]Contacted, the NOAA DNS administrator diagnosed a bug in the recent version 1.6 of the program they use, rollerd https://www.dnssec-tools.org/wiki/index.php/Rollerd. Previous version had no such issues.

TABLE I
SUMMARY OF ALL THE SIGNIFICATIVE INCIDENTS DETECTED WITH THE FIRST TESTS OF THIS TOOL

Zone	Date	Glitch	Window of vulnerability
weather.gov	2011-04-07	used too early	18h
isoc.org	2011-03-29	retired too early	11h
192.in-addr.arpa	2011-03-28	retired too early	14h
my	2011-03-26	retired too early	24h
bg	2011-03-19	retired too early	72h
isoc.org	2011-03-01	retired too early	11h
noaa.gov	2011-02-18	used too early	24h
noaa.gov	2011-02-18	retired too early	24h

```
       while the key 2817 was retired
       at 2011-03-20 20:12:40
```

The TTL was huge (345600 seconds) and therefore the signature could have been in the caches until March 23rd. But the key 2817 was retired before.

Some cases are more complicated and more difficult to analyze. For instance, this reported problem is questionable:

```
% ./examine-history.py isc.org
ERROR: signature of zone isc.org.
        first seen at 2011-02-27 01:08:42
        while the last keyset before key 21693 was
        last seen at 2011-02-27 00:06:07
        and its TTL was 7200
```

That's because isc.org[20] uses a rare technique, double signature where every resource record is signed with two keys. The key 21693 was published on February 27th and immediately used for actual signatures. Since the TTL of the DNSKEY set is two hours, it seems to mean that the signatures may have been unvalidatable for up to two hours, if the DNSKEY RRset was already in the cache. But another key was in the old keyset, 26982, and this key was also used for signing. It means that every record had a correct signature, with 26982, and a possibly invalid one, with 21693.

What should be the behaviour of a validating DNS resolver in such a case? [AAL+05b], section 5.3.3 ("Checking the Signature") says a validating resolver is free to choose its policy when there are multiple signatures so, in theory, an unlucky resolver could pick the wrong one. But this rule seems to be applicable only when the two signatures have a proper chain of trust from a trust anchor. If a resolver misses key 21693, it must tries the signatures made by the other key so the warning of our tool was probably spurious in that case.

V. CONCLUSION AND FUTURE WORK

This short run of the new key rollover analysis tool showed that the problem is still, at the beginning of 2011, a reality. There is not yet an indication that DNSSEC problems are entirely behind us. While we can hope that, in five or ten years, these problems will be no only solved but also forgotten, except by a few old timers, DNSSEC today's situation forces

[20]Warning: the previous example was isoc.org, this one is isc.org, just one letter is different.

us to do a lot of monitoring to catch the issues before someone else spot them.

With time, new and better key management tools will be developed, and tested, but it is a long-term project.

To improve the monitoring, we hope to develop of a vizualisation system of the database, interactive, able to show the various periods, allowing to zoom to see discrepancies, etc. It would make analysis easier and more spectacular.

ACKNOWLEDGMENT

Thanks to the DNS administrators who, warned of the issue, took the time to discuss it with me. For tool development, thanks to Joe Abley. For discussions and brainstorming about the new tool, thanks to Kim-Minh Kaplan, Patrik Wallström and Duane Wessels.

Presented at the SATIN conference[21] on April 4th, 2011.

REFERENCES

[AA04] D. Atkins and R. Austein. Threat Analysis of the Domain Name System (DNS). RFC 3833, RFC Editor, August 2004.

[AAL+05a] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033, RFC Editor, March 2005.

[AAL+05b] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. RFC 4035, RFC Editor, March 2005.

[AAL+05c] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. RFC 4034, RFC Editor, March 2005.

[HvM09] A. Hubert and R. van Mook. Measures for Making DNS More Resilient against Forged Answers. RFC 5452, RFC Editor, January 2009.

[KG06] O. Kolkman and R. Gieben. DNSSEC Operational Practices. RFC 4641, RFC Editor, September 2006.

[21] http://conferences.npl.co.uk/satin/