Measuring the Deployment Hiccups of DNSSEC

Vasilis Pappas and Angelos D. Keromytis

Computer Science Dept.
Columbia University
New York, NY
{vpappas,angelos}@cs.columbia.edu

Abstract. On May 5, 2010 the last step of the DNSSEC deployment on the 13 root servers was completed. DNSSEC is a set of security extensions on the traditional DNS protocol, that aim in preventing attacks based on the authenticity and integrity of the messages. Although the transition was completed without major faults, it is not clear whether problems of smaller scale occurred. In this paper we try to quantify the effects of that transition, using as many vantage points as possible. In order to achieve that, we deployed a distributed DNS monitoring infrastructure over the PlanetLab and gathered periodic DNS lookups, performed from each of the roughly 300 nodes, during the DNSSEC deployment on the last root name server. In addition, in order to broaden our view, we also collected data using the Tor anonymity network. After analyzing all the gathered data, we observed that around 4% of the monitored networks had an interesting DNS query failure pattern, which, to the best of our knowledge, was due to the transition.

1 Introduction

The Domain Name System is one of the core components of the Internet, used by virtually all the network applications. It provides name to IP (network layer) address translation in a form of a distributed database. Although DNS proved to be robust and highly scalable through everyday use, later discovered vulnerabilities [5, 4] opened the doors to attacks. DNS Security Extensions (DNSSEC) [3, 9, 2, 1] were proposed in order to address these issues. These security extensions incorporated cryptographic signatures along with each DNS message sent by the servers. Replying with signed messages gives the ability to recipients to verify the authenticity of the source and also, in most of the cases, to verify the integrity of the message itself. This, not only protects against current attacks, but it could also prevent future ones.

Designing even an extension to such a vital and highly critical component of the Internet infrastructure, is not an easy task. The first RFC document about the security extensions was written in 1997, updated in 1999 and 2001 and took its final form in 2005. Even worse, the deployment proved to be harder. Although DNSSEC was first implemented in BIND9[1] in 1999, the deployment stalled until

[1] `http://www.isc.org/software/bind`

2001, when a problem in the design of key handling was found. The operational problems caused by that, rendered the deployment almost impossible at that time. The first Top Level Domain to deploy the security extension was `.SE` (Sweden) in 2005[2].

However, more recent studies during the deployment of DNSSEC [14, 12] revealed adaptation challenges that were not anticipated in the design. Middle boxes – such as firewalls, NATs, etc. – proved to be troublesome. The main reason behind that is the increase of the DNS messages size [10], in order to facilitate the security extensions' data. In this paper, we specifically study these availability problems, using data from as many vantage points as possible, during the deployment of DNSSEC on the last of the root name servers. Our results show that the rate of DNS query failures in about 4% of the monitored networks, follows an interesting pattern that we believe is most probably caused by the problems described above. To the best of knowledge, this is the most global study on the availability of DNSSEC.

2 Background

In this section, we firstly briefly describe how DNS works and then, we move one step further to also describe how the Security Extensions (DNSSEC) work in practice.

2.1 Domain Name System (DNS)

In summary, the Domain Name System (DNS) can be viewed as a large distributed database of host name to IP address pairs. Although that simplification helps in order to understand the service provided to the end users/clients, it does not give any insights about the internals of the system. DNS is comprised mainly from two components: resolvers and name servers. Name servers store information about their "zone". This information includes name to IP address pairs, as well as delegations – that is, pairs of subdomains and name servers, lower in the hierarchy. Resolvers, on the other hand, are used to query the name servers in order to find the IP address corresponding to a host name (and vice versa).

For example, suppose that an application (web browser, e-mail client, etc.) wants to find the IP address for a specific host name, e.g. `www.example.com`. The first step for the application would be to send that query to a stub resolver on its host. If the result is not previously cached locally, the stub resolver will forward the query to a configured local resolver, usually located within the user's network (ISP, corporate network, etc.). Similarly, this resolver will also try its local cache and if that fails, it will start a recursive search. Firstly, it will query one of the root name servers in order to find the name server responsible for

[2] The source of the historical data on DNSSEC is: `http://www.nlnetlabs.nl/projects/dnssec/history.html`

the top level domain, `com` in our case. Then, it will repeat the same procedure recursively, querying the `com` name server for the `example.com` name and so forth. If the resolver manages to find the corresponding pair, it will return that to the local resolver and it will be then forwarded to the application. If there is no such pair or an another error occurred, an appropriate error will be returned.

It should be mentioned here though, that even the above definition by example is still abstract. In reality, there is more complexity added by replicas and caches throughout the system, but the main functionality remains the same.

2.2 DNS Security Extensions (DNSSEC)

As mention before, the Domain Name System was found to be vulnerable to a number of attacks (cache poisoning, etc.). In order to shield it against them and maybe even prevent future attack techniques, DNS Security Extensions were proposed. Basically, DNSSEC extends DNS through a hierarchal cryptographic signing of the DNS messages. Cryptographic signatures provide both authentication and integrity for the transmitted message. When a node receives a signed message, it can firstly establish the source and also verify the integrity of that message.

DNSSEC implementation added four new resource record types, namely: resource record signature (RRSIG), DNS public key (DNSKEY), delegation signer (DS) and next secure (NS). In order for a zone to enable the security extensions, it needs to generate a public/private key pair. Then, the records of each response should be signed using the private key and the RRSIG and DNSKEY records should contain the signature and the public key respectively. When a resolver receives a signed response from a DNSSEC capable server, it firstly verifies the DNSKEY, starting from the root servers and continuing recursively using the DNSKEY and DS records found in the messages. If the DNSKEY is trusted, the resolver can proceed with verifying the message itself using both the DNSKEY and RRSIG fields.

3 Methodology

The main burden of accurately examining the effect of the DNSSEC transition is the fact that it needs to be done on the end networks. Thus, the more available end points (in distinct networks), the more global the view will be and more accurate the results. For the purposes of our study, we chose to use two distributed systems, in order to maximize the coverage. These were PlanetLab [6] and Tor [8]. Although these systems have different goals – distributed testbed versus anonymity – they have one thing in common. Both of them are comprised by a (relatively large) set of distributed nodes. The size of these systems rely on the voluntariness of users to deploy more nodes and as both of them are closely related to academia, most of their nodes are hosted on academic institutions. At this point we have to mention that we do understand that having most of the end points in academia networks (and not so many in ISPs or corporate networks) could pose a limitation to our methodology. But, we believe that the coverage we have is the best possible, taking into account the readily available and freely usable distributed systems on the Internet. In sections 3.1 and 3.2 we describe in more details PlanetLab and Tor respectively and how we utilized each system to gather data.

3.1 PlanetLab

PlanetLab is a distributed testbed for networking and distributed systems research. It was established in 2003 by the Princeton University and as of May, 2010 it consists of 1,086 nodes at 506 sites[3]. To run an experiment on PlanetLab, initially, the user chooses a set of nodes and creates a "slice" containing them. Then, she is given remote access to each of the nodes in the slice, isolated by any other users.

For the purposes of our study, we created a slice consisting of 308 unique per site and alive (functional) nodes. On each of them, we periodically executed `dig`, which is a tool bundled with BIND to query DNS servers, for a list of host names, every 15 minutes. The host names we used are the top 100 form the Alexa web site(`http://www.alexa.com/topsites`), as of May, 4. In order to avoid some of the caching effects on the DNS resolver, each time we were resolving a host name, we additionally resolved another one, uniquely constructed. More precisely, we were prepending the current timestamp in seconds from the Epoch time to each host name. For example,

```
google.com -> 1273109411.google.com
```

Note that distinguishing between different kinds of resolve failures is possible. More specifically, the status field of each response is set to `NXDOMAIN`, if the host name does not exist and `SERVFAIL` if another error occurred during the resolving.

3.2 Tor

Tor is the second generation *Onion Routing* system. It provides low-latency anonymous communication between two parties, by routing traffic though multiple nodes and applying multiple layers of encryption. As an example, suppose that a user wants to anonymously browse a website using Tor. The steps she has to follow in order to achieve that would be:

1. Create a *circuit* by choosing a number of nodes (3 by default) and exchange session keys using Diffie-Hellman key exchange protocol – which provides *perfect forward secrecy.* During the creation of the circuit, the client only communicates with the first node in the circuit (*entry node*). The final node, called *exit node*, is the one who will actually communicate with the target web server.

[3] Source: `http://www.planet-lab.org/`

2. Transmit data. After the circuit is set up and the connection between the client and the web server is established, the user is able to browse the website anonymously.

We have to mention here that not all nodes support acting as *exit* ones. There are nodes that can be used only in the beginning or in the middle of circuits and never connect to hosts outside the Tor network. This behavior is based on the configuration of each node's *exit policy.* Recall that for our purposes we are only interested in utilizing hosts in as many end networks as possible. So, we are only interested in nodes that have less restrictive exit policy and support communicating with other target hosts, outside the Tor network.

Currently, the Tor network consists of 1536 nodes, of which 594 support acting as *exit nodes*[4]. In order to take full advantage of the Tor's exit nodes, we tried to periodically send a HTTP request to each of the 100 hosts used in 3.1, using a different exit node each time. This was repeated hourly and the result of whether the connection succeeded or not was stored. In order to implement that, we used the very convenient `TorFlow` software package (`http://fscked.org/projects/torflow`). Note though that we did not use any uniquely constructed host names in this case, as we would not be able to distinguish between failed resolves and non existent domain names.

Although using the Tor network we broaden our view with more end points in distinct networks, the level of useful information that we were able to collect was significantly lower. The interface of Tor to the client applications is through a SOCKS [11] proxy, so, the only information that we could get is whether a connection succeeded or not at the exit node.

4 Data Collection

In order to have sufficient data to examine the effects of the last step of the DNSSEC transition we monitored the DNS behavior, using the above methods, for roughly a 5 day period.

After that time period, each PlanetLab node had invoked `dig` for roughly 100,000 times. Thus, creating an equal number of files containing the output of each DNS query. In total, there were about 30 million files occupying 60 giga bytes of storage when stored as plain text files and 8.1 giga bytes when compressed using `gzip`.

On the other hand, the data collected using the Tor network were stored to a single file. Unfortunately, due to a bug in the script we used to scan the exit nodes, each iteration over them was immediately terminated after a single circuit had failed. More precisely, the total number of circuits created during the measuring period was 4,745 instead of about 9,600 – we roughly lost half of the data. Although we did lose a big portion of valuable data due to this bug, the rest of them was nonetheless important enough. Out of the 594 Tor nodes that can be *exit nodes* only 184 of them had the flags *Exit, Fast, Stable, Running*

[4] Source: `http://torstatus.kgprog.com/`

and *Valid* set to true at the same time and these were the ones we were able to effectively use.

Top Level Domains	# of nodes	Top Level Domains	# of nodes
edu	98	uk ch	9
net	50	ca	8
de	49	tw se es	7
com	34	ru	6
NA	26	pt kr br	5
pl	15	sg nz il gr cn au	4
org jp it	14	nl hu hk fi cz be at	3
fr	12	tr ro no eu dk ar	2
ve uy to th su si re jo info eg cy cx			1

Table 1. TLD distribution for both Tor and PlanetLab nodes.

Table 1 summarizes the distribution of the nodes' Top Level Domains, both for Tor and PlanetLab systems. As previously mentioned, most of the nodes are deployed in academic networks, for example the `.edu` TLD.

5 Results

After we collected the datasets described in the previous section, we analyzed them in order to extract useful insights related to the DNSSEC transition. In this section, we will thoroughly discuss our main findings from the analysis.

5.1 Node Categorization

Category	Nodes	Description
Zero	222	No failures
Few	108	A few failures
Interesting	15	Fair amount of failures, interesting patterns
Constant	7	A lot of failures, constant rate though

Table 2. Categorization of the nodes based on their failure patterns.

The question we tried to answer here was: *"Do we see any DNS query failures that could probably be due to the transition?"* We first analyzed the failures in the `dig` logs and gathered the timestamps in which each node experienced a DNS lookup failure. Based on that information we divided the PlanetLab nodes in the following four categories (summarized in Table 2):

– **Zero**. The nodes that had no DNS query failures during the monitoring period fall into this category. As expected, the majority of the nodes are in this category.

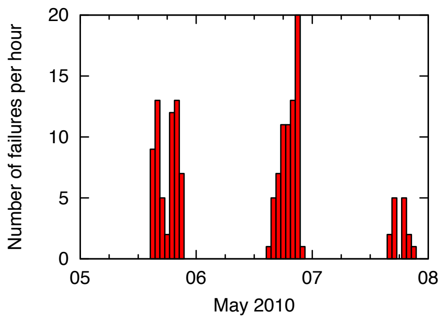

Fig. 1. Interesting failure pattern 1.

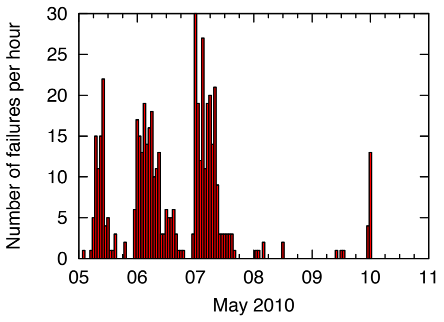

Fig. 2. Interesting failure pattern 2.

- **Few**. Some of the nodes sporadically encountered a small number of failures. These were most of the times less than 10, with just a few cases that had up to almost a hundred. Still, we consider these failures not caused by the DNSSEC transition since they were few in number and occurred randomly.
- **Interesting**. This category contains the nodes that had a fair amount of failures and most importantly, their failure rates were forming an interesting pattern. In average, these nodes failed for a few hundreds of queries.
- **Constant**. Finally, there were some nodes that were constantly failing for a number of queries. Clearly that could not be due to the DNSSEC transition as this failure rate was constant during the whole monitoring period. Probably, that should be the result of a misconfiguration or blacklisting – the top 100 hostnames we used contained some adult oriented sites too.

5.2 Failure Patterns

In this section we examine a few specific failure patterns as observed on individual PlanetLab nodes. More precisely, we closely examine three failure patterns of what we called *Interesting* in the previous section and one example of a *Constant* failure pattern. Although, we are more concerned to the first ones, as the failures in these cases were probably be caused by the transition, we include the last example (constant failure rate) for completeness.

Figures 1,2 and 3 show the failure patterns for three nodes that we categorized as *Interesting.*

In the first case (Figure 1) we see that the failures started on the 5th of May in the afternoon. Same time as the transition of the last root server to DNSSEC was performed. This is a strong evidence that that could be their cause. It is also interesting that we see a similar failure pattern on the next day (6th of May) that could be due to caching effects. One common Time-to-Live (TTL) value for the host name to IP address pairs is 86,400 seconds, which equals to one day. The same was observed on the third day too, but with lower magnitude. Finally, the most interesting thing is that this node had no more failures for the next two days, until 10th of May, when we stopped the monitoring. Our explanation

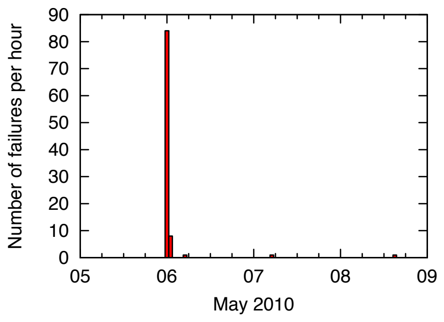

Fig. 3. Interesting failure pattern 3.

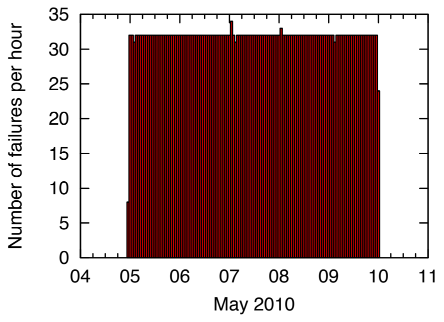

Fig. 4. Constant failure rate.

for that is that there could be indeed an error that was fixed by the network administrators in the first couple of days.

The next one, Figure 2, follows almost the same pattern as the previous one. We see that the failures start on the day of the transition and they have a diurnal pattern. Most likely again due to the fact that the TTL set by the name servers is one day. The interesting thing here though is that although the problem seemed to have been fixed on the fourth day (9th of May), we do see more failures on the fifth day. That is hard to explain by this figure alone, but, correlating more diverse data from the same network could reveal more insights.

The third *Interesting* node we chose to discuss follows a different failure pattern than the previous two (Figure 3). In this case, (i) the failures do not start on the 5th of May and (ii) there is only a tiny diurnal pattern. This is again a both interesting and hard to explain situation without having more details. We speculate that it could have been indeed caused by the the DNSSEC transition and the failures were reported later either due to a wrong configured clock on the node or due to an uncommon caching policy on the local resolver.

We should mention here that out of the fifteen *Interesting* cases, most of them follow the pattern of the first two. Which is what we believe was caused by the transition. But, we decided to show this case too, because of its uniqueness.

Finally, Figure 4 shows a case of a node that had a constant DNS query failure rate (i.e., example of a *Constant* node). As shown in the figure, this particular node fails to resolve about 32 host names per hour. Recall from Section 3.1 that each node was periodically trying to resolve 100 host names every fifteen minutes. In addition, each of these queries was followed by another query to a uniquely constructed hostname, under the same domain. So, each domain name was queried eight times per hour. Dividing the failure rate with the number of queries for each individual domain per hour, results in the number four. This constant failure rate of 32 queries per hour could mean that there were just four domain names that always failed to resolve. After investigating the log files of that specific case, we confirmed that that was the case. This node was always failing to resolve four domains, of which one of them was an adult oriented site, another was a webmail service and the other two, social networking websites. In conclusion, this confirms our first intuition about the nodes that had constant failure rate, which was blacklisting.

As previously stated in Sections 3.2 and 4, the information we gathered from the Tor network was not that comprehensive. In addition to that, we also lost a big portion of the data due to a programming bug in our scripts. Being not inclusive enough the dataset, we were not able to draw any conclusions from it alone. But, we did use this data in order to help us verify some moot cases.

6 Related Work

To the best of our knowledge, the most relevant work to our is Osterweil's et al. study on the deployment of DNSSEC[14]. The authors gathered historical data over 2 years – since the initial deployment of the security extensions. After analyzing their dataset, they derived three basic metrics to quantify the effectiveness of the DNSSEC's deployment, namely *Availability*, *Verifiability* and *Validity*. In addition, their results exposed previously undocumented open issues during the deployment, such as lack of Availability due to middle boxes (NATs, firewalls, etc.), poor Verifiability as a result of not having an external key authentication and finally they showed that cryptographic verification is not necessarily equivalent to validation.

Although in terms of their work we only looked at *Availability*, our dataset was gathered from hundreds of distinct networks whereas they used five locations. We believe that using such a geographically diverse monitoring infrastructure gave us a more global view of the problem. And most importantly, our results do confirm what was anticipated by the authors about the Availability metric, in a small scale though.

The implementation of the system used in the above study, namely *SecSpider*, is described in detail in [13]. Some early observations on the data gathered by SecSpider are reported in [12]. One of their observations, which was the need to reduce signatures lifetime, is addressed in [15].

Finally, Curtmola et al. compared the performance of PK-DNSSEC versus SK-DNSSEC, arguing that a hybrid approach would leverage the benefits from both worlds [7].

7 Conclusion

DNSSEC's goal is to shield the domain name system to any attacks based on the authenticity and integrity of the messages. This is achieved by the incorporation of cryptographic signatures along with each message sent by the name servers. Although the deployment of the security extensions started after years of designing and testing, studies showed that problems related to availability could still be experienced in some networks.

Our study revealed that in a relative small number of the networks we monitored, availability issues were evident. We consider these issues to have been most probably caused by the DNSSEC deployment on the last root name server.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. CNS-09-14312. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol modifications for the dns security extensions. RFC 4035, March 2005.
2. R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource records for the dns security extensions. RFC 4034, March 2005.
3. G. Ateniese and S. Mangard. A new approach to dns security (dnssec). In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 86–95, New York, NY, USA, 2001. ACM.
4. D. Atkins and R. Austein. Threat analysis of the domain name system (dns). RFC 3833, August 2004.
5. S. M. Bellovin. Using the domain name system for system break-ins. In *SSYM'95: Proceedings of the 5th conference on USENIX UNIX Security Symposium*, pages 18–18, Berkeley, CA, USA, 1995. USENIX Association.
6. B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, 2003.
7. R. Curtmola, A. del Sorbo, and G. Ateniese. On the performance and analysis of dns security extensions. In Y. Desmedt, H. Wang, Y. Mu, and Y. Li, editors, *The Fourth International Conference on Cryptology and Network Security (CANS)*, number 3810 in LNCS, pages 288–303. Springer-Verlag, December 2005.
8. R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.
9. A. Friedlander, A. Mankin, W. D. Maughan, and S. D. Crocker. Dnssec: a protocol toward securing the internet infrastructure. *Commun. ACM*, 50(6):44–50, 2007.
10. O. Gudmundsson. Dnssec and ipv6 a6 aware server/resolver message size requirements. ftp://ftp.rfc-editor.org/in-notes/rfc3226.txt, 2001.
11. M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. Socks protocol version 5. RFC 1928, March 1996.
12. E. Osterweil, D. Massey, and L. Zhang. Observations from the dnssec deployment. In *NPSEC '07: Proceedings of the 2007 3rd IEEE Workshop on Secure Network Protocols*, pages 1–6, Washington, DC, USA, 2007. IEEE Computer Society.
13. E. Osterweil, D. Massey, and L. Zhang. Deploying and monitoring dns security (dnssec). In *ACSAC '09: Proc. of the 2009 Annual Computer Security Applications Conference*, pages 429–438, Washington, DC, USA, 2009. IEEE Computer Society.
14. E. Osterweil, M. Ryan, D. Massey, and L. Zhang. Quantifying the operational status of the dnssec deployment. In *IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 231–242, New York, NY, USA, 2008. ACM.
15. H. Yan, E. Osterweil, J. Hajdu, J. Acres, and D. Massey. Limiting replay vulnerabilities in dnssec. In *Secure Network Protocols, 2008. NPSec 2008. 4th Workshop on*, pages 3 –8, 19-19 2008.