UNIVERSITEIT VAN AMSTERDAM

MSc System and Network Engineering

Research Project II

Discovery method for a DNSSEC validating stub resolver

Author:
Xavier Torrent Gorjón
xavier.torrentgorjon@os3.nl

Supervisor:
Willem Toorop
willem@nlnetlabs.nl

July 14, 2015

Abstract

Although DNSSEC was first defined in 2005, its deployment, both at DNS servers and at DNS recursive resolvers, is still problematic. In this project, we intend to develop a discovery method to ensure DNSSEC information can be delivered to the end host. For this purpose, we used RIPE ATLAS to study the current state of DNSSEC aware and DNSSEC validating resolvers, and define a course of action from that information. Our results concluded that 64% of the recursive resolvers where able to perform basic DNSSEC queries, whereas only 56% could provide valid proofs of denial of existence, and only 40% were able to process authenticated wildcard information. Our proposed discovery method relies on using the default recursive resolvers and, in case of failure, proceed to use, in this order, the ISP recursive resolver, a public DNS resolver, or perform full resolving recursion from the host.

Contents

1 Introduction

1.1 DNS: Stub and Recursive Resolvers

Domain Name System (DNS)[10] is a distributed name system to be used by computers connected to a private network or the public Internet. It is mostly used to translate addresses easily readable and memorized by humans into IP addresses which can be used by computers. Additionally, it can also be used as an abstraction layer, as servers can switch locations or IP addresses, while maintaining their human-readable name.

DNS is based in a distributed architecture, where authoritative name servers can delegate authority of a certain subdomain to another name server. This method circumvents the need for a centralized database, providing administrative, load and geographical scalability.

As queries to the DNS system require multiple recursive queries to the root servers and the Top Level Domains (TLDs) servers, and each subdomain down until the queried domain is found, DNS uses two different resolvers: stub resolvers and recursive resolvers. Stub resolvers are hosted on each end user machine, and are used to forward queries needed from applications to a recursive resolver. Recursive resolvers receive queries from stub resolvers and proceed to do all the queries needed to find an answer, and returning it back to the clients. This separation allows for a shared caching system between many stub resolvers, effectively reducing the workload and response times of DNS queries.

1.2 DNSSEC Functionality

DNS was not, however, designed with any kind of security features. This has lead to a number of issues with the system, one of the most common being DNS cache poisoning[13].

To overcome these issues, Domain Name System Security Extensions (DNSSEC)[1, 2] was introduced. DNSSEC provides authentication of the DNS answers by adding public key cryptography features –in the form of key and signature resource records–, although the information can still be seen by unauthorized users, as it is not encrypted.

1.3 DNSSEC Limitations

Although DNSSEC fullfills its purpose, it still suffers from a number of problems:

1. The communication between the local stub resolver and the recursive resolver is still vulnerable. There have been studies[4] attempting to secure this *last mile* of the communication, but the solutions are limited.

2. The end user has no control over the configuration of the recursive resolver, so it cannot determine whether it is protected against DNS hijacking[1].

3. The adoption of DNSSEC comes with the possibility of leaving domains unavailable to users that use recursive resolvers that perform DNSSEC validation, in case there is any problem with the resource records of the domains.

   In recent years, this has been a recurring problem that has repeated a number of times. In 2012, **"nasa.gov"** was unintentionally blocked to Comcast users when they adopted DNSSEC[2]. **".gov"** zones became temporarily unavailable in 2014 because of a DNSSEC misconfiguration[3]. Similarly, HBO NOW services were recently blocked (2015) due an invalid signature at their servers[4].

   A common issue with all these cases was that the Internet Service Provider (ISP) companies were the first to receive the blame for the malfunction, even though the cause of the problems resided on these services and not the DNS resolvers. This results in an inclination of the ISPs to not perform validation at their resolvers.

4. DNS-Based Authentication of Named Entities (DANE)[8] implementations require to perform the authentication validation at the local machine, therefore colliding with the DNS recursive resolvers that perform the validation themselves.

1.4 This Document

The problems stated in Section 1.3 could be easily addressed by using a machine stub resolver as a recursive resolver. However, this solution disrupts the scalability of the DNS service, as caching mechanisms could be used only locally.

The goal of this project is to define a mechanism for a local resolver to determine the best course of action when the previous issues occur –or when the DNS queries are blocked–, trying to find the best trade-off between performance and functionality. To this end, we will study the current situation of DNS recursive resolvers through RIPE ATLAS[5], and use that information to design a system that can deliver DNSSEC data to the final user.

The document is structured as follows:

Section 2 describes the most important references and bibliography used during the course of this project.

[1] Information available at: `http://www.gohacking.com/dns-hijacking/`
[2] Source: `http://bit.ly/1GOrHxR`
[3] Source: `http://bit.ly/1gbP7aP`
[4] Source: `http://bit.ly/1GoasVi`
[5] Information available at: `https://atlas.ripe.net/`

Section 3 states the research questions this project aims to answer.

Section 4 describes the infrastructure used during this project, and how the measurements should help solve the stated research questions.

Section 5 explains how the measurements are performed, how the results were filtered and which subset of the results we focused on for our research.

Section 6 describes the results obtained from each one of our measurements.

Section 7 provides detailed information on how the results were used to design a discovery method for DNS stub resolvers.

Section 8 gathers the conclusions from this research.

Section 9 closes this document, providing insight on possible topics to extend the research performed in this project.

2 Related Work

2.1 DNS Deployment Measurements

There has been a number of measurements of the DNSSEC deployment using diverse methods[6, 11, 12]. We have used that previous research as an inspiration for our project. The measurements performed during our research update the state of these previous measurements, and we focus on analysing the responses obtained by querying wildcard and non-existing DNS domains. Additionally, we studied the differences on the results obtained by using the DNS Checking Disabled (CD) bit, used to inform the recursive resolvers that the validation of the results is to be performed locally. The option of using the DNS CD bit is a feature that was recently introduced into RIPE ATLAS during the course of this project.

2.2 DNS and DNSSEC implementations

As this project required technical understanding of the low-level specifications of DNS and DNSSEC, the original RFCs[1, 2, 10, 14] were used regularly. In addition to these documents, other information sources publicly available on the Internet were used[5, 7].

3 Research Questions

3.1 How can a stub resolver use a discovery method to process data from a recursive resolver?

The main goal of this project is to deliver a course of action for machines attempting to retrieve DNSSEC data, in such a way that the caching mechanisms of DNS are used as much as possible, only falling back to full recursion from the stub resolver in the worst-case scenario. To answer this question, we define a subquestion to provide insight on how this procedure should be implemented.

Definition *In our project we distinguish two types of DNSSEC resolvers. DNSSEC aware resolvers are those who can perform DNSSEC operations. DNSSEC validating resolvers are those that, besides being able to execute DNSSEC queries, also validate the results of these queries.*

3.1.1 What is the current state of DNSSEC aware resolvers deployment?

Measuring the status of DNSSEC from the point of view of DNS resolvers is an important step to define a course of action for DNS stub resolvers. We will study the status of DNSSEC deployment using RIPE ATLAS, which we describe in detail in Section 4.2.

3.1.2 What is the current state of DNSSEC validating resolvers deployment?

During the course of our research, RIPE implemented an option on their API to define the value of the Checking Disabled bit of DNSSEC. We used this new feature to determine how many DNS resolvers were actually performing DNSSEC validation by querying domains with bogus records.

4 Methodology

4.1 Used Infrastructure

During the course of this project, we had access to the infrastructure of NLnet Labs and the OS3 MSc. The material used included:

1. The RIPE ATLAS probe network, further discussed in Section 4.2.
2. The DNS server of NLnet Labs, to perform DNS queries directed to it from the RIPE ATLAS probes, in order to discover the recursive resolver IPs used by these probes.
3. To schedule and retrieve experiments we used the RIPE ATLAS API through Python scripts. These scripts were written in Python 2.7.9, and we used extensively the *dpkt*[6] library to parse DNS queries and answers.

4.2 RIPE ATLAS

RIPE offers to individuals and organizations the possibility to acquire *probes*, small devices which can be queried from their website and API to perform a variety of operations. During this project, there was an average of 8200 probes connected during our experiments. These probes are spread across the globe, although a significant majority is located in Europe[7].

During our project, we queried the full set of probes available at a given time in diverse occasions to perform DNS queries from them and get the corresponding answers. Although this can be considered a small set of values to determine the current state of the Internet as a whole, we deemed it to be enough to be used as a indicator of the status of DNSSEC aware resolvers, after proper filtering of the results. The details of the measurements performed and the processing of the results are gathered in Section 5.

4.3 Answering the Research Question

To answer the main research question, *How can a stub resolver use a discovery method to process data from a recursive resolver?*, we will first enumerate the different possibilities available. Then, with the help of the gathered results from the RIPE ATLAS infrastructure, we will attempt to order these possible solutions in the most effective way to achieve a good trade-off between functionality (delivering DNSSEC to the final applications) and efficiency (maintaining the DNS scalability through caching methods).

[6] Information available at: `https://pypi.python.org/pypi/dpkt`
[7] Source: `https://atlas.ripe.net/results/maps/density/`

5 Procedure Design

To obtain a better understanding about the current DNSSEC deployment situation, we designed and performed a series of measurements. In this Section, we will discuss the details of these measurements, including the specifications of the queries performed, how the results were filtered, and which subset of the retrieved data we studied.

The results of the measurements described there will be gathered and discussed in Section 6.

5.1 Types of Measurements

We prepared six different types of measurements through RIPE ATLAS, each one of them querying all the available probes. These experiments were:

1. Basic DNS resolving: As our first experiment, we wanted to determine which percentage of the probes had basic DNS capabilities. Additionally, we planed the experiment in such a way that we could get the IP address of the DNS recursive resolver used by the probe, as we discuss in Section 5.1.1.

2. Basic DNSSEC resolving: As a second step, we also wanted to determine the number of DNSSEC aware resolvers, explained in Section 5.1.2.

3. NXDOMAIN handling: Ascertaining how many recursive resolvers were capable of properly handling NXDOMAIN answers was also interesting for our research. The details of this experiment are documented in Section 5.1.3.

4. Wildcard handling: Finally, in a similar way to the NXDOMAIN queries, it was also interesting to conclude how many resolvers could properly handle wildcard domains, as discussed in Section 5.1.4.

5. Validating resolvers: Through the use of the CD bit of DNSSEC queries, we gathered information about how many recursive resolvers were validating results from queries, with the experiments described in Section 5.1.5.

6. Evaluating alternative DNS resolvers: DNS offers a redundancy mechanism, allowing users to specify more than one recursive resolver to query. We want to evaluate if, in the default configuration of the probes, backup resolvers provide different answers than the primary resolver. We develop this further in Section 5.1.6.

5.1.1 Basic DNS and obtaining backdoor DNS recursive resolver IP address

This first experiment served two different purposes. First, it was meant to provide insight on the amount of probes that could properly use DNS. This information was used as to filter the probes which could not get basic DNS answers from the remaining measurements. Second, we also intended to discover the IP address of the recursive resolvers used by these probes.

Our interest in the second goal for this measurement resides in the fact that many home routers –and equipment in small companies– are intended to be as cheap as possible, and are usually not up to date with recent standards, such as DNSSEC. This means that although most recursive resolvers from ISPs can properly process DNSSEC queries, home routers cannot forward that information to the end users because they are not DNSSEC aware. An example of this situation is shown in Figure 1.

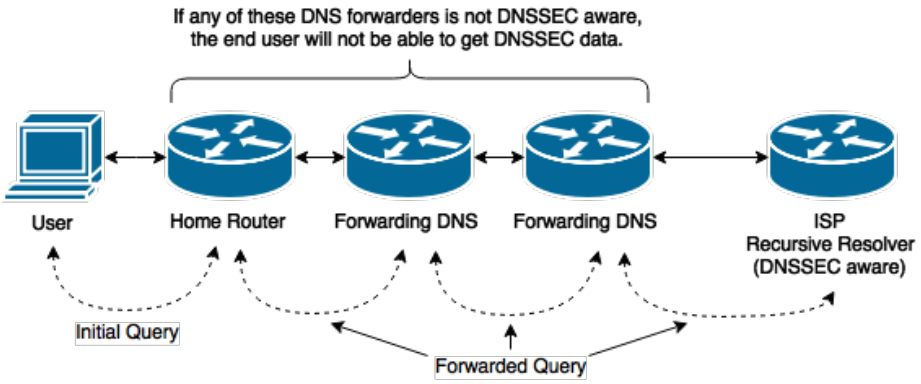

Figure 1: DNSSEC may be blocked by DNS forwarders.

If there is a DNS forwarder blocking DNSSEC, end users could still attempt to directly query their ISP recursive resolver instead of their assigned DNS resolver, effectively bypassing that limitation. Although this partially handicaps the DNS caching features, the caching mechanism at the ISP resolver remains intact. This alternative procedure is shown in Figure 2.

However, from the end user point of view, it is difficult to determine the IP address of the DNS resolver that is actually performing the recursive resolving. One way to find it is to query a DNS server that can reply customized answers. To test this, he had a special domain at the NLnet infrastructure that answered queries replying back the IP address that made the final query. The direction for this domain is ***echo.v4.nlnetlabs.nl***. There is also a similar domain name that can be queried to discover IPv6 addresses, ***echo.v6.nlnetlabs.nl***, although we

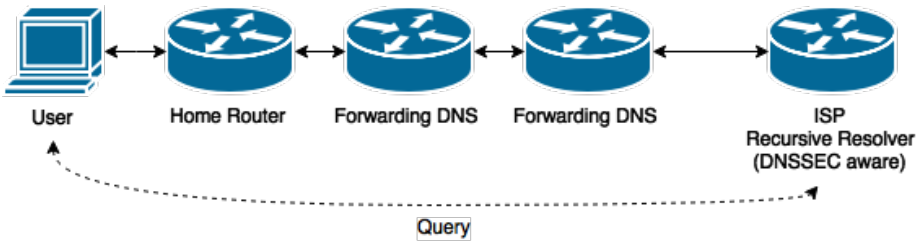

Figure 2: Bypassing DNS forwarders when they are not DNSSEC aware.

limited our experiments to IPv4 to avoid conflicts with how results might change from the deployment of IPv6, and not causes related to DNS infrastructure.

This method allows for end users to discover the IP of their DNS recursive resolver, without needing to change or extend the current DNS protocol, or having to use a different Internet protocol. Nevertheless, it has the downside that it requires a DNS server having a domain dedicated for this purpose.

The details of the query used in this measurement are described in Table 1.

Argument	**Value**
Query address	echo.v4.nlnetlabs.nl.
Record type	A (IPv4)
Use probe resolver	Yes
RD bit	Yes
DO bit	No
UDP payload size	512 bytes

Table 1: Basic DNS measurement options.

5.1.2 Basic DNSSEC

Our second experiment focuses on discovering which probes are querying recursive resolvers that can handle basic DNSSEC queries. For this purpose, the DNS queries will ask for the DNSKEY records of the root server. In this case, we set the DO bit to specify that we desire a DNSSEC-enabled answer with signature records.

The options for the queries used in this measurement were set as follows in Table 2.

Argument	Value
Query address	. (root server)
Record type	DNSKEY
Use probe resolver	Yes
RD bit	Yes
DO bit	Yes
UDP payload size	4096 bytes

Table 2: Basic DNSSEC measurement options.

5.1.3 Test NXDOMAIN Queries

Although DNS resolvers might be able to process basic DNSSEC queries, it is possible that they cannot process certain special queries because of misconfiguration, bugs on the implementations or running outdated software versions. One of these special cases are queries that target a domain that does not exist (NXDOMAIN). This is a complex situation for DNS, as to achieve authentication, the records must be signed using the key from the server. However, for such non-existent records, there is nothing to sign prior to the request, and therefore they require a special treatment.

On most implementations, this is achieved through using NSEC and NSEC3 resource records. A NSEC record marks the space covered from a certain record N to the next one, as well as stating all the types of resource records available for N. A simplified example of this is shown in Table 3. In this example, if a client asks for the IP of the domain ***g.domain.com***, which does not exist, the server would answer with the NSEC record from ***c.domain.com***, which is the responsible for the proof of denial of existence for that particular name.

NAME	TYPE	RDATA
a.domain.com.	AAAA	1234::4321
a.domain.com.	RRSIG	Digital Signature for the Resource Record Set
a.domain.com.	NSEC	c.domain.com. [AAAA, NSEC, RRSIG]
c.domain.com.	A	1.3.3.7
c.domain.com.	RRSIG	Digital Signature for the Resource Record Set
c.domain.com.	NSEC	j.domain.com. [A, NSEC, RRSIG]
j.domain.com.	A	145.100.104.171
j.domain.com.	TXT	"Hello World"
j.domain.com.	RRSIG	Digital Signature for the Resource Record Set
j.domain.com.	NSEC	a.domain.com. [A, TXT, NSEC, RRSIG]

Table 3: NSEC records example.

This method, however, allows malicious users to obtain the entire dataset of domain names available on a specific server, a technique commonly known as *zone walking*. For this reason, many servers use NSEC3 records instead of NSEC, which work in the same way but use hash the names of the domains instead of having them in plain text. This defense mechanism, however, can be broken, and it is still possible to enumerate DNS zones[8] that use NSEC3. Nevertheless, zones that have a large amount of delegations (such as TLDs) still favour NSEC3 because of its OPT-OUT flag functionality, which reduces the amount of signatures needed for a zone, hence making the zone file smaller[9].

In this case, we made two different experiments, to test the ability of recursive resolvers to process both NSEC and NSEC3 records. To test NSEC records we queried the root domain, and we queried the Netherlands domain (*.nl*) to check the NSEC3 handling. Table 4 summarizes the details of these queries.

Argument	**Value (NSEC test)**	**Value (NSEC3 test)**
Query address	nonexistingdomain.	nonexistingdomain.nl.
Record type	A (IPv4)	A (IPv4)
Use probe resolver	Yes	Yes
RD bit	Yes	Yes
DO bit	Yes	Yes
UDP payload size	4096 bytes	4096 bytes

Table 4: DNSSEC NXDOMAIN handling measurement options.

5.1.4 Test Wildcard Queries

Similar to NXDOMAIN answers, another common source of issues are wildcard records. This is a result of different DNS implementations handling them differently. Wildcard records are records that have a single asterisk (*****) character as their leftmost label, and that are intended to match non-existent records that would match the wildcard. As an example, ****.domain.com.*** should act as a wildcard record, and match queries that do not have an associated resource record, such as ***xyz.domain.com.*** (in our Table 3 example). Wildcard answers must provide proper proof of the existence of the wildcard record, as well as proof that the queried domain does indeed not exist.

To test this DNS functionality, we created wildcard records in the nlnetlabs and nlnet domains, as the root servers and TLDs –at least the most common ones, such as ***.nl*** and ***.com***– did not have wildcard records. This wildcard records were introduced as ****.xavier.nlnetlabs.nl.*** (NSEC) and ****.xavier.nlnet.nl.*** (NSEC3). Table 11 includes a description of the queries performed in this measurement.

[8]Source: `http://dnscurve.org/nsec3walker.html`

Argument	Value (NSEC test)	Value (NSEC3 test)
Query address	wildcardtest.xavier.nlnetlabs.nl.	wildcardtest.xavier.nlnet.nl.
Record type	A (IPv4)	A (IPv4)
Use probe resolver	Yes	Yes
RD bit	Yes	Yes
DO bit	Yes	Yes
UDP payload size	4096 bytes	4096 bytes

Table 5: DNSSEC NXDOMAIN handling measurement options.

5.1.5 DNSSEC validation test with CD bit

In addition to the previous tests, we defined a measurement to check the number of recursive resolvers performing DNSSEC data validation, using the CD bit of DNSSEC. Querying a domain with bogus DNS records, such as ***dnssec-failed.org***, allows us to determine how many recursive resolvers properly validate the data they receive. The details of this measurement are as described in Table 6.

Definition *The DO bit in DNS is used to ask the recursive resolver to retrieve and validate –if possible– resource records corresponding to DNSSEC data. In addition to this, the CD bit can be also set to specify that the host wants to validate the data itself, instead of delegating that responsibility to the recursive resolver.*

Argument	Value
Query address	dnssec-failed.org.
Record type	A
Use probe resolver	Yes
RD bit	Yes
DO bit	Yes
CD bit	Yes/No
UDP payload size	4096 bytes

Table 6: Recursive resolvers DNSSEC validation measurement options.

5.1.6 Evaluating alternative DNS resolvers

Most DNS resolving implementations include one or more backup recursive resolver addresses in case the default resolver does not return an answer. It

is interesting for our research to determine whether these alternative servers provide significantly different answers. If the answers are mostly the same, then these alternative resolvers could not be efficiently used for our discovery method. Otherwise, if a probe can get different answers from each one of its alternative recursive resolvers, it would be interesting to study this line of work further, and examine the validity of these answers.

To study this we will individually process the answers of each node using the data from RIPE ATLAS, and examine interesting parameters, such as how many of these probes do use backup DNS resolvers, and whether these secondary resolvers return different answers for the same query.

5.2 Filtering Results

In order to reduce the bias of the measurements, we did some minor filtering of the results. In this Section we document which data was filtered and the reasoning behind the filter.

5.2.1 Exclude public DNS servers

During our first testing measurements, we noticed that a considerable amount of the probes –approximately 15% of the answers– were querying the Google DNS server, as its IP address is well-known and easy to remember (*8.8.8.8* in IPV4). As having such amount of queries to the same public DNS resolver could distort the results of the measurements, we decided to identify the most commonly used resolvers, and exclude those who belonged to a public DNS resolver.

Table 7 summarizes the addresses we excluded from our measurements.

IP Addresses	**Owner**
8.8.8.8, 8.8.4.4	Google DNS
2001:4860:4860::8888, 2001:4860:4860::8844	Google DNS
208.67.220.220, 208.67.222.222	OpenDNS (non-DNSSEC)

Table 7: Filtered DNS server addresses.

5.2.2 Exclude loopback addresses

Similarly to the public DNS servers, we also noticed a significant number of probes using a loopback address (**127.0.0.1**) for their queries, meaning their stub resolver acted as a recursive resolver. As that is undesirable due the lack of scalability, and we intend to use the stub resolver as a recursive resolver only as a last resort mechanism in our discovery method, we excluded probes performing queries directed to these addresses as well.

5.2.3 Handling timeouts and format errors

Besides the filtering of addresses, another problem that required to be fixed were the nodes that were not behaving as expected. From these, the majority (over 99%) corresponded to recursive resolvers not answering, so the probes that made these queries returned this timeout error. A minority of errors were classified as *Unreachable Network* and *Invalid Argument* in the response retrieved from RIPE ATLAS.

Nodes answering with these error messages were excluded from the measurement. However, other errors, such as returning an unexpected RCODE or returning an invalid set of resource records, were not filtered, and used as part of the study.

5.3 Studied Data

After filtering the list of probes, we needed to choose which subset of the data retrieved from the remaining valid probes we wanted to focus on. Generally, for all the experiments we analysed the returned RCODEs and resource records.

5.3.1 RCODEs

The Response CODE is a field of the DNS header set to 0 on queries, and changed by the DNS server on the reply to summarize the result of the requested query to the client. Generally, the expected –or desired– RCODE is 0. This is the case for most of our measurements, with the exception of the NXDOMAIN measurements, that are supposed to return a RCODE value of 3, indicating that the requested name does not exist.

Additionally, it is to be expected that there will be a number of RCODEs with a value of 2, indicating a Server Failure, in our measurement of the DNSSEC validating resolvers. When using the CD bit, recursive resolvers should always return the result of the queries, whether they are valid or not (with a RCODE of 0). When the CD bit is not used, recursive resolvers that perform validation should return such Server Failure answer if the data is invalid, whereas resolvers not performing validation will keep sending answers with the 0 RCODE.

Appendix A summarizes the most used RCODEs.

5.3.2 Returned Resource Records

Besides the basic DNS measurement, all the other measurements are expected to return a number of additional resource records with the answer. These include the DNSSKEY resource record for the basic DNSSEC measurement, as well as its corresponding RRSIG signatures. Similarly, we expect a number of NSEC and NSEC3 records for the NXDOMAIN and wildcard handling measurements, paired with their corresponding signature and Start of Authority (SOA) records.

Appendix B includes a summary of the resource records that were relevant for this study.

6 Measurement Results

In this Section we discuss the results from our measurements. We first present a number of considerations common to all measurements, and afterwards each Subsection will discuss the results related to each one of them.

All measurements were performed at least three times, and all of them queried the entire set of probes available at a certain moment, which always yield, approximately, 14.000 results. From these 14.000 results, we discarded approximately 2.800 with the filters explained in Section 5.2. From the remaining results, we also discarded those that gave errors, which left the valid dataset that we studied with approximately 10.000 results (it should be noted that this is not a 1:1 probe - result ratio, as some probes had more than one resolver).

From the valid dataset, there was always a number of probes (approximately 450) returning RCODE values of 1, indicating a Format Error, and others (approximately 80) returning values of 5, indicating a denied request. As these numbers were constant and originating from the same probes, we deemed that they probably had errors in their configuration. Additionally, for the sake of simplicity we did also not include the A records –along with their corresponding RRSIG records, in DNSSEC queries– in the tables summarizing the results of our experiments.

All these numbers remained constant during the course of this project, with minor differences –never exceeding 5%– that we attribute to probes going online and offline.

6.1 Basic DNS and obtaining backdoor DNS recursive resolver

As expected, our first measurement returned a majority of No Error RCODEs. The only erratic RCODEs received were, as we stated before in this Section, part of a series of probes that do not properly process DNS queries. Being a simple DNS query, there were no additional resource records to study.

We successfully retrieved the IPs of the ISP recursive resolvers during this experiment, using the method described in Section 5.1.1.

6.2 Basic DNSSEC

As with the basic DNS measurement, the basic DNSSEC measurement did not yield any unexpected results regarding RCODEs, as almost the entirety of the returned values was again No Error.

However, the obtained resource records showed a significant discrepancy. There were a number of queries that did not return a resource record at all, and queries that returned only two DNSKEY records –one for the Key Signing Key and another one for the Zone Signing Key–, without a signature record. The remaining queries returned the correct set of two DNSKEY records and one RRSIG record. This information is showcased in Table 8.

Definition *To simplify key management, most DNS servers use two different keys to sign information. The Key Signing Keys are used to validate the Zone Signing Keys. In turn, Zone Signing Keys are used to sign the different resource records present in a DNS zone, and provide that data in the form of RRSIG resource records. This method permits the renewal of keys and signatures on a zone without the need of changing the DS records on the parent zones.*

Received RR	**Percentage**
No RR	7.94%
DNSKEY(x2)	28.34%
DNSKEY(x2) + RRSIG	63.71%

Table 8: Basic DNSSEC test returned resource records. Valid answers correspond to the the last row.

6.3 Test NXDOMAIN Queries

The majority of the RCODEs obtained in the NXDOMAIN test measurements had a value of 3, equivalent to a Name Error, as it was expected from this query. However, a small amount (between 2 and 3.5%) of the probes returned a RCODE value of 0, indicating No Error, which is incorrect for this specific of query.

Both NSEC and NSEC3 measurements produced a similar percentage of valid results, as can be seen in Tables 9 and 10. The difference between both valid percentages is negligible, being slightly above 0.5%. In contrast, that success rate of approximately 56% is noticeably below the 63.71% observed from the previous experiment, implying that not all DNSSEC aware resolvers are able to forward the entirety of the DNSSEC data to the probe that performed the query.

When using the ISP resolvers as the recursive resolvers for our queries, instead of the probe defined resolvers, the amount of successful DNSSEC queries went

Received Resource Records	Percentage
No RR	22.27%
Only SOA	21.49%
SOA + NSEC + RSIG(x2)	56.23%

Table 9: NSEC test returned resource records. Valid answers correspond to the the last row.

Received Resource Records	Percentage
No RR	12.44%
Only SOA	27.68%
SOA + RRSIG	3.62%
SOA + NSEC3(x2) + RSIG(x3)	0.58%
SOA + NSEC3(x3) + RSIG(x3)	55.67%

Table 10: NSEC3 test returned resource records. Valid answers correspond to the the last row.

up to approximately 75%, which represents a 20 points increase over the original values.

6.4 Test Wildcard Queries

Besides a very minor number of SERVFAIL RCODEs (below 0.25%), all queries returned the expected No Error RCODE (0). As with the NSEC3 measurement, this query returned a high variety of different resource records, as shown in Table 11.

However, the valid answer for this query should return, at least, the IP address of the domain and a NSEC resource record to provide proof that there was no more specific result, both of them paired with RRSIG records. This minimum valid answer was found in approximately 10% of the queries, while a considerable amount of them –approximately 29%– returned, in addition to these, NS records paired with a RRSIG record.

Ultimately, approximately 40% of the queries returned a valid answer. Compared to the NXDOMAIN results, this percentage is considerably lower than the previous 56%. We attribute this reduction in the amount of valid responses to the fact that wildcard entries are complex and have a huge diversity of different implementations, both in DNS servers and DNS resolvers.

Another interesting result from this measurement is that there are a number of resolvers –approximately 18.7%– that could retrieve RRSIG resource records for the A and NS records on their answers, but did not send the necessary

NSEC/NSEC3 records for the complete answer. This means that these resolvers actually DNSSEC aware, as they were able to retrieve RRSIG records, but could not successfully retrieve the complete set of resource records required for a valid DNSSEC wildcard query (or that DNS forwarders between the stub resolver and the recursive resolver lost that information along the way).

Received Resource Records	Percentage (NSEC3)	Percentage (NSEC)
No RR	31.31%	29.85%
NS(x3)	7.19%	6.94%
NS(x3) + RRSIG	18.6%	18.7%
RRSIG + NSEC/NSEC3	8.88%	10.65%
NS(x3) + RRSIG(x2) + NSEC/NSEC3	29.22%	29.51%

Table 11: DNSSEC wildcard handling measurement options. The last two rows correspond to the valid answers.

When attempting to run the same experiment using the ISP resolvers, the amount of successful authenticated wildcard answers went up to roughtly 60% of the answers. Similar to the NXDOMAIN measurement, this is a quite significant increase in the number of valid responses.

6.5 DNSSEC validation test with CD bit

In the two measurements performed –using the bogus domain **dnssec-failed.org.**– for this test, the returned RCODEs were enough to spot the differences on the resulting datasets. With the CD bit enabled, recursive resolvers were asked to not perform any kind of validation on the obtained DNSSEC data. As such, all the received RCODEs had a value of 0, indicating No Error.

Setting the CD bit to zero, the returned RCODEs were gathered in two different groups. The first group was still answering with a No Error RCODE, and it accounted for approximately 74% of the results. The remaining 26% was composed by responses with a RCODE of 2, indicating a server failure.

From these results we can distinguish the portion of DNS recursive resolvers that actually validated the data they received, formed by the group of resolvers included in the 26% that did answer with a server failure RCODE.

6.6 Evaluating alternative DNS resolvers

From the valid answers we retrieved from the previous measurements, we studied the composition of these results on a per-probe basis. Over half of the probes (51.40%) did not have an alternative DNS server configured. 41.41% of the probes had one or more backup recursive resolvers, and they received

the same answer for all their recursive resolvers. Finally, the remaining 7.17% corresponds to probes who had one or more backup resolvers, and they did provide different answers for the same query. The details of that small amount of probes that had different answers are gathered in Table 12.

Used recursive resolvers	**Number of valid answers**	**Amout of probes**
2	0	8.46%
2	1	54.22%
2	2	4.99%
3	0	2.39%
3	1	12.80%
3	2	15.84%
3	3	1.30%

Table 12: Overview of the number of used resolvers compared to the amount of them providing valid DNSSEC data (over the 7.17% of probes that had multiple resolvers and received different answers from them)

From these results we can conclude that approximately 6.30% of the probes that had different results on their answers had different resource records on these answers, but were all valid for DNSSEC validation. Almost 10% of these probes had answers with different resource records, but none of them provided enough information to perform DNSSEC validation. The remaining 83.86% of the probes had answers with different results, and from these only a part of them contained valid DNSSEC proofs. This last group is the most interesting for our research, as it contains the set of probes that could retrieve valid DNSSEC information by querying their pre-defined alternative DNSSEC resolvers.

7 Discovery Method

From the results gathered from our measurements, we can now design a discovery method for a DNSSEC validating stub resolver to query, retrieve and process DNSSEC data. We first introduce some considerations that will impact on the format of that method, and afterwards we proceed to describe the mechanism in detail.

7.1 Initial considerations

First, depending on the needs of the end user application, the stub resolver should determine whether it is necessary or not to properly handle authenticated NXDOMAIN and wildcard queries. Some applications might not need these features, and be able to work as intended only with basic DNSSEC (or any subset of these query types). If the application can work without these additional, more complex features, it may require less steps to successfully deliver valid data to the application.

Second, from the results of our last measurement we concluded that, in most configurations, alternative recursive resolvers do not provide different answers. This implies that if the primary recursive resolver is unable to answer a query, the backup resolvers will most likely not be able to do so either. In conclusion, although resolvers are still necessary to provide redundancy to the DNS infrastructure, we will not consider them for our discovery method.

As a final consideration, it is important to note that for the validation to be performed at the machine running the stub resolver, this resolver should set up the CD bit on the DNS queries, to indicate to the recursive resolvers that the validation will be performed at the host.

7.2 Method Development

In the best case scenario, the default configuration of a host will give him access to DNSSEC data, without having to use any additional mechanism. According to our results, approximately 40% of the hosts are in this situation (which raises up to 64% if the application does not require to process NXDOMAIN or wildcard queries).

If the default configuration is not enough to obtain DNSSEC information, the host should first attempt to query directly the recursive resolver from its ISP, avoiding intermediate forwarding. Our measurements showed that this effectively increases the success rate of DNSSEC availability, increasing NXDOMAIN queries from a 56% to a 75% success rate and wildcard queries from 40% to 60%.

If the ISP recursive resolver is also unable to provide the host with valid DNSSEC data, the next step for the host should be to query a public DNS server that can handle DNSSEC. One well known example for this could be the Google DNS servers, available at **8.8.8.8** and **8.8.4.4**. At this step, almost the entirety of all hosts should be able to obtain DNSSEC data, as long as the chosen public DNS server is capable of processing DNSSEC information.

As a last resort mechanism, the stub resolver can be used as a recursive resolver itself to gather DNSSEC information. This is an undesirable scenario, as the data collected by this stub resolver will only be cached for itself, which limits the scalability of the DNS infrastructure as a whole.

There is the possibility that users would desire to not use public DNS servers for their queries, as that could provide sensitive data directly to the companies running them. In that case, the step of using a public DNS should be skipped, and proceed instead to do full recursive queries from the host itself.

8 Conclusions

In this Section we discuss the conclusions gathered from our work.

From the results of our measurements, we determine that DNSSEC adoption in recursive resolvers is still scarce. Furthermore, DNSSEC queries that involve more complex proofs, such as authenticated denial of existence and wildcard resolution are even more problematic. Basic DNSSEC queries were properly answered –with correct authentication proofs– by 64% of the resolvers, whereas denial of existence proofs and wildcard resolutions were properly handled in only 56% and 40% of the resolvers, respectively.

To overcome these limitations, we have successfully defined a discovery method to deliver DNSSEC information to the end users. Besides the last resort solution of performing recursive queries from the end host, other solutions in our discovery method are able to offer a shared cache mechanism for multiple users, which helps maintaining the scalability of DNS.

This discovery mechanism is based on performing DNS queries through alternative recursive resolvers, other than the originally defined by the system, either by user specification or DHCP assignment. The first alternative is to directly query the resolver from the user ISP. Initially that is also the resolver that is being used, but the DNSSEC information is lost somewhere along the way because of intermediate DNS forwarders not being able to handle DNSSEC data. When skipping these intermediate forwarders, we noticed a significant raise on the success rate of the DNSSEC queries.

The second alternative involves using a public DNS server, capable of performing DNSSEC operations, to retrieve DNSSEC data. The only requirement of this step is to use a public DNS server that is capable of processing DNSSEC queries.

Finally, the worst-case scenario involves using the stub resolver as a recursive resolver. Although it is possible to do that, the caching mechanisms of DNS are lost when performing queries that way, and that makes this solution undesirable unless there is no alternative.

Additional conclusions from our measurements are:

1. Only 26% of the DNS resolvers do actually perform validation of DNSSEC data (which is a small valued compared to the 64% of resolvers that could perform basic DNSSEC queries).
2. Even though usually hosts have at least two defined DNS servers, in most configurations that only helps achieve a higher degree of redundancy and availability, but does not help in getting proper DNSSEC answers.

9 Future Work

As a follow up work to the results of this project, we propose two lines of work that could extend and validate the work done:

1. First, it would be interesting to use an alternative infrastructure or method, other than RIPE ATLAS, and compare the corresponding results to the ones gathered in this project. This would provide a better insight on the current status of DNSSEC resolvers, and perhaps introduce new ways of dealing with the problems stated in this document.

2. Additionally, to validate the conclusions from this project, a proof of concept of the method discussed could be developed. This application could be used to determine the accuracy of our method, and change it to achieve better performance if necessary.

10 Ethical considerations

As this project is strictly academical and does not use any sort of personal or sensitive information in any way, we did not find any ethical issues during its progress.

11 Acknowledgements

First, I would like to express my gratitude to NLnet Labs, and specially to my supervisor, W. Toorop, for the opportunity of doing this project at their office in Science Park, and for all the help provided during the course of it. These thanks also extends to the developers at RIPE, for their effort on implementing new functionalities to their API for this project in such a short notice. I would also like to thank A. Stavroulakis and N. Triantafyllidis for their feedback and suggestions on the project report.

Additional thanks go for my family and friends for their support during my studies. Namely, X. Torrent, A. Gorjón, Q. Torrent, N. Matabosch, E. Valverde, M. Mora, E. Cuspinera, C. Pardo, F. Garí and X. Muñoz.

This project is dedicated to Mery and Antonia Guirado.

References

[1] R. Arends et al. *DNS Security Introduction and Requirements*. RFC 4033. http://www.rfc-editor.org/rfc/rfc4033.txt. RFC Editor, 2005.

[2] R. Arends et al. *Protocol Modifications for the DNS Security Extensions*. RFC 4035. http://www.rfc-editor.org/rfc/rfc4035.txt. RFC Editor, 2005.

[3] R. Arends et al. *Resource Records for the DNS Security Extensions*. RFC 4034. http://www.rfc-editor.org/rfc/rfc4034.txt. RFC Editor, 2005.

[4] Marc Buijsman, Matthijs Mekking, and Jeroen van der Ham. *Securing the last mile of DNS with CGA-TSIG*. 2014.

[5] CC BY.SA. "Authenticated Denial of Existence in the DNS". In: (2012).

[6] Nicolas Canceill. *Measuring the deployment of DNSSEC over the Internet*. 2014.

[7] The TCPIP guide. *DNS Message Header and Question Section Format*. http://www.tcpipguide.com/free/t_DNSMessageHeaderandQuestionSectionFormat.htm.

[8] P. Hoffman and J. Schlyter. *The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA*. RFC 6698. http://www.rfc-editor.org/rfc/rfc6698.txt. RFC Editor, 2012.

[9] B. Laurie et al. *DNS Security (DNSSEC) Hashed Authenticated Denial of Existence*. RFC 5155. http://www.rfc-editor.org/rfc/rfc5155.txt. RFC Editor, 2008.

[10] P. Mockapetris. *Domain names - implementation and specification*. STD 13. http://www.rfc-editor.org/rfc/rfc1035.txt. RFC Editor, 1987.

[11] Anastasios Poulidis and Hoda Rohani. *Research Project 2 DNSsec Revisited*. 2014.

[12] Internet Society. *DNSSEC deployment maps*. http://www.internetsociety.org/deploy360/dnssec/maps/. 2015.

[13] Sooel Son and Vitaly Shmatikov. "The hitchhikers guide to DNS cache poisoning". In: *Security and Privacy in Communication Networks*. Springer, 2010, pp. 466–483.

[14] Paul Vixie. *Extension Mechanisms for DNS (EDNS0)*. RFC 2671. http://www.rfc-editor.org/rfc/rfc2671.txt. RFC Editor, 1999.

A List of Response Codes (RCODE)

This Appendix summarizes the different response codes of DNS answers[10]. This Appendix does not include all the available RCODE values, only those who are most widely used and that were relevant in our measurements.

RCODE Value	Description
0	No Error
1	Format Error
2	Server Failure
3	Name Error
4	Not Implemented
5	Query Refused

Table 13: Possible RCODE values on DNS answers.

No Error (0): This is the standard RCODE answer for successful queries.

Format Error (1): The name server was unable to interpret the query.

Server Failure (2): The name server was unable to provide an answer due an internal error.

Name Error (3): This RCODE states that the domain name asked in the query does not exist.

Not Implemented (4): The name server indicates that the requested operation is not implemented.

Query Refused (5): The name server refused to process the query. This is usually a result of attempting to query for a zone transfer, although it can also happen when a server only accepts requests from certain users.

B List of Resource Records (RR)

This Appendix showcases the resource records that were more relevant during the course of our project.

RR Value	RR Type	Description
1	A	IPv4 Address record[10]
2	NS	Name Server record[10]
6	SOA	Start of Authority[10]
43	DS	Delegation Signer[3]
47	NSEC	Next Secure Record[3]
46	RRSIG	IPv4 DNSSEC Signature[3]
48	DNSKEY	DNS key record[3]
50	NSEC3	NSEC version 3[9]

Table 14: Summary of resource records.

A Resource Record: A resource records are used to map hostnames into IPv4 addresses.

NS Resource Record: Name Server records are used to delegate a DNS zone to the marked authoritative name server in the record.

SOA Resource Record: SOA resource records specify diverse parameters regarding the configuration of an authoritative zone. These parameters include a contact email, the domain name of the zone and several configuration timers.

DS Resource Record: This resource record is used to identify the DNSKEY RR of a delegated zone.

NSEC Resource Record: NSEC is a resource record introduced with DNSSEC that is used to provide authenticated denial of service of names.

RRSIG Resource Record: RRSIG resource records include digital signatures of other resource records present on a DNSSEC-secured zone. Each record set has a a RRSIG resource record.

DNSKEY Resource Record: This resource record contains the key to be used on DNSSEC queries for a specific zone.

NSEC3 Resource Record: NSEC3 is a parameter similar to NSEC that utilizes hashes instead of the real names to prevent zone-walking attacks.