Preface

The Global Domain Name System (DNS) is a fundamental and even more, an essential building block of the Internet. It can be defined as a critical information infrastructure, resolving billions of queries per day in support of global communications and commerce. The DNS is core for the correct operation of every Internet service as they strongly depend on it.

The health of the DNS is concerned with its security, stability and resiliency (SSR) as well as its performance and have a direct and strong impact on the guaranteed level of accuracy, performance and dependability of nearly all aspects of interactions on the Internet.

Under this light, The Global Cyber Security Center (GCSEC), in cooperation with the Internet Corporation for Assigned Names and Numbers (ICANN) and the DNS Operation Analysis and Research Center (DNS-OARC), organized the First Workshop on DNS health and security (DNS EASY-2011). The DNS-EASY workshop aims at bringing together researchers and professionals from academia, industry and governmental Agencies as well as representatives from across DNS ecosystem stakeholder groups (technical development, network operators, enterprise users, and security experts) to discuss all different aspects of the DNS Health and Security and its impact on the modern society.

This volume collects contributions received from academic and industrial research centers all over the world (Canada, China, Czech Republic, France, Italy, Holland, Japan, Korea, United States). Research and technical papers describe the state of the art, advancements as well address challenges in three main area:

– Models for a Secure and Robust DNS
– Maintenance and operation of DNSSEC
– Methodologies and Applications for a Secure and Robust DNS.

The workshop program, organized in three sections covering contribution in the above research area present:

October 2011

Emiliano Casalicchio
Igor Nai Favino
Program Co-Chair
DNS-EASY2011
Organization

DNS-EASY2011 is organized by Global Cyber Security Center (GCSEC) in cooperation with ICANN and DNS-OARC.

**General Co-Chairs:** Andrea Rigorni (GCSEC, IT)  
John Crain (ICANN, US)

**Technical Program Co-Chairs:** Emiliano Casalicchio (Univ. of Tor Vergata, IT)  
Igor Nai Favino (GCSEC, IT)

**Organizing & Publicity Chair:** Angelo Migliorisi (GCSEC, IT)  

**Finance Chair:** Massimo Cappelli (GCSEC, IT)

**Technical Program Committee**

Bernhard Ager, Technische Universität Berlin, FG INET, DE  
Mustaque Ahamad, Georgia Tech Information Security Center, US  
Cristina Alcaraz Tello, University of Malaga, ES  
Nevil Brownlee, The University of Auckland, US  
Emiliano Casalicchio, University of Tor Vergata, Rome, IT  
David Conrad, Virtualized LLC, US  
Steve Crocker, Shinkuro Inc., US  
Joao Damas, Bond Internet Systems, ES  
Fabrizio D’Amore, University of Rome “La Sapienza”, IT  
Enrico Gregori, IIT-CNR, IT  
Xiaodong Lee, CNNIC / Chinese Academy of Sciences, CN  
Dan Massey, Colorado State University, US  
Douglas Maughan, Dep. of Homeland Security, US  
Eric Osterweil, Verisign Inc., US  
Roberto Perdisci, University of Georgia, US  
Steve Uhlig, Technische Universität Berlin, FG INET, DE  
Paul Vixie, Internet Systems Consortium, US  
Nicholas Weaver, ICSI UC Berkeley, US  
Stephen D. Wolthusen, Royal Holloway, University of London, UK

**Organizing Committee**

Elena Agresti, GCSEC, IT  
Marco Caselli, Univ. of Rome La Sapienza, IT  
Salvatore Di Blasi, GCSEC, IT  
Maria Luisa Pagani, GCSEC, IT  
Nuria Tascon Rodriguez, GCSEC, IT
Sponsoring Institutions

The Internet Corporation for Assigned Names and Numbers, USA
The Global Cyber Security Center, Italy
Table of Contents

The 1st Workshop on DNS Health and Security

Models for a Secure and Robust DNS ........................................... 6
A Global Reference Model of the DNS ........................................... 7
  Yakup Koç, Almerima Jamakovic, Bart Gijsen
Preliminary Evaluation of Potential Impact of Failure in DNSSEC Validation ... 28
  Kensuke Fukuda, Shinta Sato, Takeshi Mitamura
A bi-objective Mixed Integer Linear Program for load balancing DNS(SEC) queries ....................................................... 41
  Stanislas Francfort, Daniel Migault, Stéphane Sénécal

DNSSEC: Maintenance and Operation ............................................ 53
Maintenance, Mishap, and Mending in DNSSEC Deployment ............... 54
  Casey Deccio
DNSSEC Automation and Monitoring ............................................. 66
  Russ Mundy, Wes Hardaker, Suresh Krishnaswamy, Wayne Morrison,
  Robert Story
DNSSEC Lives. Now what? How to avoid certain failure. .................... 78
  Richard Lamb

Methodologies and Applications for a Secure and Robust DNS .......... 92
Detecting Hidden Anomalies in DNS Communication ........................ 93
  Ondřej Mikle, Karel Slaný, Ján Veselý, Tomáš Janoušek and Ondřej Sury
What an IP-over-DNS Tunnel -A Case Study on Large Operational Network .... 104
  Ziqian Liu
Design of robust DNS adaptable to dynamic Ad hoc networks .......... 117
  Younchan Jung, J. William Atwood

Author Index ............................................................................. 130
Models for a Secure and Robust DNS
A Global Reference Model of the DNS

Yakup Koç, Almerima Jamakovic, and Bart Gijsen

1 Delft University of Technology
Department of Multi-actor Systems, Section System Engineering
P.O. Box 5015, 2600 GA Delft, The Netherlands
yakupkoc@gmail.com

2 TNO: Netherlands Organization for Applied Research
P.O. Box 5050, 2600 GB Delft, The Netherlands
almerima.jamakovic@tno.nl, bart.gijsen@tno.nl, yakupkoc@gmail.com

Abstract. The Domain Name System (DNS) is a crucial component of today’s Internet. At this point in time the DNS is facing major changes such as the introduction of DNSSEC and Internationalized Domain Name extensions (IDNs), the adoption of IPv6 and the upcoming extension of new generic Top-Level Domains. These changes can have impact on the behaviour of the DNS. In this paper we present a first global DNS reference model with the aim to predict the DNS traffic behaviour under specific conditions. In fact, this quantitative model is intended to be used for analyzing what-if scenarios. For example, how will DNS query rates at the recursive and authoritative name servers increase in case DNSSEC validation errors lead to sending more Servfail responses towards DNS clients? The DNS reference model takes into account all relevant components present in the DNS architecture. To characterize the system variables describing the query behaviour at each of these independent system components, we statistically analyze real world data from recursive resolvers. In addition, we use experimental results that characterize DNS client behaviour and data from the literature to characterize the behaviour of authoritative name servers. In order to validate our reference model we compare the model predictions to the real world data. The validation results show that the model predictions are rather accurate. At the end of the paper we present a specific what-if scenario to demonstrate the applicability of the model.

1 Introduction

In the last decade the Internet gained more and more importance such that it became an essential part of our society. As a consequence, the stability of the Internet including the Domain Name System (DNS) as a key Internet component, is crucial. The DNS is primarily used to translate the human readable domain names into the corresponding Internet protocol (IP) addresses, which are used for the routing purposes. For instance, thanks to the DNS, one just needs to recall “cnn.com” instead of “157.166.255.19”. The data for this mapping between domain names and IP addresses is stored in a tree-structured distributed database, where the mapping responsibility for each domain is assigned to designated authoritative name servers (NSs). The authoritative NSs are thus
assigned to be responsible for their particular domains which typically are the root, top-level domain (TLD) and second-level domain (SLD). This mechanism makes the DNS distributed and resilient against failure [7].

The top layer of DNS hierarchy is facing major changes: cryptographically signing the authoritative NS with DNSSEC, deploying new generic TLD names by allowing domains such as .bank as well as deploying Internationalized Domain Name extensions (IDNs), including non-ASCII characters. In addition, the uptake of IPv6 that is required to make the Internet future proof has impact on the DNS. These developments can have consequences to the stability of DNS and indirectly, to the continuity of the entire Internet. For example, the query load towards the authoritative NS is expected to increase [12, 10] and a specific type of DNS query response, i.e. Servfail responses, is expected to increase significantly [9]. All the mentioned challenges have triggered the need for public awareness and more research on proper understanding of the DNS behaviour in the increasingly evolving DNS landscape.

In this paper we present a global DNS reference model aimed at analysing what-if scenarios. For example, how will DNS query rates at the recursive and authoritative name servers increase in case DNSSEC validation errors lead to sending more Servfail responses towards DNS clients? The contribution of this paper is twofold. First, we present a global reference model taking into account the typical DNS architecture: starting from client’s OS with its stub resolver and application browser, then recursive resolver present mostly at an Internet Service Provider (ISP), to the authoritative NS which include the root, TLD and SLD servers. To characterize the system variables describing the query behaviour at each of these independent system components, we statistically analyze real-world data from recursive resolvers. The data is provided by SURFnet who serves a large number of academic customers in The Netherlands. In addition, we use a characterization of DNS client behaviour from an experimental study by TNO and SIDN, and data from the literature to characterize the DNS behaviour of authoritative name servers in more detail. Second, we validate our reference model by using Monte Carlo simulation to generate DNS behaviour predictions and compare them to the real world data. The validation results show that the model predictions are rather accurate. In addition to these main contributions we discuss shortcomings related to the real-world data and possible extensions of the model. Finally we present a specific what-if scenario to demonstrate the applicability of the model. Overall, this paper establishes a path towards the proper understanding of the DNS behaviour in the increasingly evolving DNS landscape.

The paper is organised as follows. Section 2 gives an overview of the former work on modelling efforts in the DNS community, which is followed by the description of our DNS reference model in Section 3. Section 4 explains the overall operation of the model and the model validation with real-world data is presented in Section 5. Section 6 treats the usage of the DNS reference model for the impact assessment of the increase of a specific DNS query response by a certain percentage. In Section 7 we present a summary of our results and identify further research for the DNS reference model.
2 Related work

We already mentioned several important works in the field of measurement and characterisation of DNS traffic. They all present and discuss the DNS query behaviour and corresponding data analysis tools, focussing mainly on the upper DNS hierarchy. For example, a vast majority of papers attempt to address the question of characterisation of DNS traffic at the root: CAIDA and the Measurement Factory have done numerous monitoring studies on the traffic at the root NS, among which the more recent ones are [4, 10, 13]. Besides this traffic analysis at the core component of the upper DNS hierarchy, authors attempt to address the question of characterisation of DNS traffic at the recursive resolver and at the client side. For example, in [3] authors give a statistical analysis of DNS traffic at the recursive resolver and in [15, 1] authors compare the performances of caching recursive resolvers with respect to query response time and querying behaviour towards the root, while in [2] authors attempt to characterize the querying behaviour of specific client types (e.g. a client with Linux as OS and Firefox as application browser). Equally relevant for our work are those publications that present experimental studies carried out to understand the effectiveness of DNS caching [6, 5, 16]. Furthermore, many authors point to the lack of data with which to do the long-term research and analysis in support of DNS performance, stability and security, as being one of the main concerns of the DNS community. For example, in [10] authors raise the awareness of this problem to evaluate the DNS during the expected transition phase the DNS is facing in a short time interval.

Although there is a substantial literature on the characterization of the traffic and querying behaviour of each individual hierarchical level of the DNS, we are not aware of much work that attempted to study the entire DNS. Perhaps good to mention here is that there are several works, similar to [1], which pinpoint the limitations of the current DNS deployment and its foremost influence on the performance of applications. However, to this day there is a little understanding of the way the DNS behaves as whole, especially when the expected changes are incorporated and the querying mechanism need further detailed analysis. In this respect, by introducing a reference model of the entire DNS, we make an important step in fundamentally understanding the DNS behaviour in the increasingly evolving DNS landscape.

3 The DNS reference model

3.1 General features and assumptions

Our primary concern is the scalability of the DNS system when for example redundant DNS traffic towards the recursive resolvers and authoritative NS occurs. We therefore create a reference model at the flow level, being only interested in the query flow distribution at an arbitrary point in time. Consequently, the time notion does not play a role and the distribution of the DNS queries is only dependent on the behaviour of various components of the DNS system. We therefore chose to distinguish between the following generic components in the DNS: a) client with its OS (and the corresponding stub resolver) and application browser, b) recursive resolver, and c) authoritative
NS with the root, TLD and SLD NSs. Figure 1 shows these generic components of the DNS system and also the interactions between them. In our model we assume that all clients (of the same configuration type) are independent and have identical querying behaviour, so they can be modelled as one client. The same holds for recursive resolvers and the authoritative NS being either the root, TLD or SLD. This assumption enables us to control the entire system by adjusting only input parameters for a single client, a single recursive resolver, and single root, TLD and SLD. Furthermore, we model the querying behaviour with a system variable referred to as Query Multiply Factor, which in fact reflects how many queries will be reinitiated by a component in reaction a negative query response. Then, the caching behaviour of a component: it depends strongly on TTL values and inter-arrival time of the queries, having a stochastic and state dependent behaviour [6]. However, we do not model the caching mechanism as a state, i.e. weather the domain name is in the cache or not, but rather by a probability that a queried domain name will be in the cache of the corresponding system component. We call this system variable Cache Hit Ratio. Finally, we assume that the type of a response to an initial query at the authoritative NS follows a certain distribution. This variable is referred to as Response Distribution at Authoritative Name servers. Values of these system variables are obtained by analyzing the real-world data which consists of 30,000 DNS packets, captured at an UNBOUND resolver for the duration of 14 sec.

Fig. 1: The DNS overall modeling structure

3.2 System variables

**Cache Hit Ratio** is the value which indicates the probability that a queried domain name will be in the cache of a system component under consideration. The notion of Cache Hit Ratio is different for client and recursive resolver side, therefore we treat them separately. The values for the Cache Hit Ratio at the client side indicate the probability that a query will be answered with a certain response type from the cache. Whether the received DNS data can be cached or not depends on the response type. For instance, application browsers cache only the DNS response types that provide
valid data (i.e. Valid, Valid>512B and Truncated), and the NXdomain response type. The OS and application browser Cache Hit Ratio’s are rather complicated to determine. Therefore, considering the relative scale nature of the DNS reference model, we assume "rule of thumb" values for OS and application browser Cache Hit Ratio’s. These values are given in Table 1 and Table 2. Note please that the model has the possibility of filling in the missing values, as soon as they are available to the DNS community.

### Table 1: Cache Hit Ratio values for three application browser types

<table>
<thead>
<tr>
<th>Response type</th>
<th>IE8(%)</th>
<th>Firefox(%)</th>
<th>Safari(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Valid</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Valid (&gt;512B)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NXdomain</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Truncated</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 2: Cache Hit Ratio values for four OS types

<table>
<thead>
<tr>
<th>Response type</th>
<th>Windows XP(%)</th>
<th>Windows 7(%)</th>
<th>Linux(%)</th>
<th>MAC OSX(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>25</td>
<td>25</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>Valid</td>
<td>22</td>
<td>22</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>Valid (&gt;512B)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NXdomain</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Truncated</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

In Table 1 and Table 2, "Total" stands for the amount of the total traffic which will be responded from the application browser/OS cache. Correspondingly, the percentage of 22% for example for the IE8 application browser indicates the amount of traffic that will be responded with the "Valid" response type. The Cache Hit Ratio values for OS and application browser are relatively smaller than the recursive resolver Cache Hit Ratio values since these are client specific caches.

The Cache Hit Ratio values at the recursive resolver are rather different from the Cache Hit Ratio at the client side. Queries arriving at the recursive resolver are classified into four different groups from the caching point of view: a) Non-cached queries are queries which are not in the cache. These queries have to be sent to the root directly and domain name resolution will be performed by the resolver until whole name is resolved. b) TLD-cached queries are those whose top level domain is known by caching resolver. This means that TLD-cached queries will be sent directly to TLD NS by skipping the root. c) SLD-cached queries will be directly sent to SLD NS. TLD and SLD of those queries are known by caching resolver. d) Domain-cached queries occur when the entire request is in the cache. The probability that an incoming query will be located in one
of these groups is given by the system variable Cache Hit Ratio. The Cache Hit Ratio values for the UNBOUND resolver are determined by analyzing the SURFnet data captured at an UNBOUND resolver. This data set consists of 300,000 DNS packets which we divide in 10 smaller data subsets of 30,000 DNS packets. For one of the subsets we give the Cache Hit Ratio values for UNBOUND recursive resolver in Table 3. Besides this common resolver type, we also leave the possibility of having another type of the recursive resolver, for example BIND9.

<table>
<thead>
<tr>
<th>Cached Domain</th>
<th>UNBOUND(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLD-cached</td>
<td>4.1</td>
</tr>
<tr>
<td>SLD-cached</td>
<td>41.1</td>
</tr>
<tr>
<td>Domain-cached</td>
<td>54.7</td>
</tr>
<tr>
<td>Noncached</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 3: Cache Hit Ratio values for UNBOUND recursive resolver.

<table>
<thead>
<tr>
<th>Cached Domain</th>
<th>Mean(%)</th>
<th>Variance(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLD-cached</td>
<td>4.5</td>
<td>0.51</td>
</tr>
<tr>
<td>SLD-cached</td>
<td>38.5</td>
<td>9.11</td>
</tr>
<tr>
<td>Domain-cached</td>
<td>56.9</td>
<td>12.37</td>
</tr>
<tr>
<td>Noncached</td>
<td>0.11</td>
<td>0.005</td>
</tr>
</tbody>
</table>

Table 4: Normal distribution with mean and variance of Cache Hit Ratio for UNBOUND

To bring the stochastic nature in the DNS reference model, we find distributions for each of the four caching types of the Cache Hit Ratio by analyzing the 10 data sets of 30,000 DNS packets. We test first whether the Cache Hit Ratio values, obtained from each set, are independent. The independency is tested by using Von Neumann test [8]. Distributions for each of the four Cache Hit Ratio caching types is estimated and verified by using distribution fitting techniques. It is important to note that our sample size is relatively small, i.e. n=10. However, we use Shapiro-Wilk normality test [11] for which the sample size is large enough, to conclude that the each of the four Cache Hit Ratio groups is normally distributed. Additionally, we verify this assumption by using quantile-quantile (Q-Q) plots in which we show that the plots are almost linear, pointing to the almost identical behaviour of the two compared distributions. This is given in Figure 2. Table 4 gives the mean and the variance of the estimated distributions for each of the four Cache Hit Ratio caching types.

**Response Distribution at Authoritative Name servers** is the system variable which indicates the fraction of response types that are given, in response to incoming initial
queries, at the authoritative NS. These values are different for the root, TLD and SLD NSs. Distribution values are determined by analyzing UNBOUND data sets of 30,000 DNS packets. These values are given in Table 5b while in Table 5a a detailed breakdown of response types at the authoritative NSs is given. It is interesting to see that just seven initial queries are sent to the root and all these queries are replied by NXdomain response type. The latter observation points to the proper working of the caching mechanism of UNBOUND recursive resolver while former observation is due to the fact that our data set covers just 14 seconds of DNS traffic. However, since our dataset is not large enough to determine caching mechanism of UNBOUND recursive resolver while former observation is due to the fact that our data set is not large enough to determine caching mechanism of UNBOUND recursive resolver. These values are included in Table 5b. SLD NS is the last step in the domain name resolution process. We therefore observe the diversity in SLD response types unlike it is the case for TLD responses.

For this system variable we also find distributions for each of the four response types, again by analyzing the 10 data sets. Following the process explained previously, we first test the independency and then make sure that the obtained distributions are validated by using Shapiro-Wilk normality test and Q-Q plots. We found that the obtained distributions for each of the four response types follow normal distributions with the mean and the variance given in Table 6a and Table 6b, for TLD and SLD responses, respectively.

**Query Multiply Factor** is a system variable which indicates how many queries will be reinitiated by a component in reaction to a negative response. In other words, it...
Table 5: Values for Response Distribution at Authoritative Name servers.

<table>
<thead>
<tr>
<th>Response Type</th>
<th>Root</th>
<th>TLD</th>
<th>SLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referrals</td>
<td>0</td>
<td>377</td>
<td>741</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>1072</td>
</tr>
<tr>
<td>AAAA</td>
<td>0</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>CNAME</td>
<td>0</td>
<td>0</td>
<td>673</td>
</tr>
<tr>
<td>MX</td>
<td>0</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>PTR</td>
<td>0</td>
<td>2</td>
<td>105</td>
</tr>
<tr>
<td>NXdomain</td>
<td>7</td>
<td>31</td>
<td>510</td>
</tr>
<tr>
<td>Not Imp.</td>
<td>0</td>
<td>0</td>
<td>89</td>
</tr>
<tr>
<td>Refused</td>
<td>0</td>
<td>5</td>
<td>270</td>
</tr>
<tr>
<td>Servfail</td>
<td>0</td>
<td>2</td>
<td>199</td>
</tr>
<tr>
<td>NS</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>SOA</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>TXT</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Format Error</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>Response Type</th>
<th>Root(%)</th>
<th>TLD(%)</th>
<th>SLD(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>8.1</td>
<td>90.9</td>
<td>71.1</td>
</tr>
<tr>
<td>NXdomain</td>
<td>91.5</td>
<td>7.4</td>
<td>13.7</td>
</tr>
<tr>
<td>Servfail</td>
<td>0.4</td>
<td>0.5</td>
<td>7.9</td>
</tr>
<tr>
<td>Refused</td>
<td>0</td>
<td>1.2</td>
<td>7.3</td>
</tr>
</tbody>
</table>

(b)

Table 6: Normal distribution of responses with their mean and variance at TLD(a) and SLD(b).

<table>
<thead>
<tr>
<th>Response Type at TLD</th>
<th>Mean(%)</th>
<th>Variance(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>94.7</td>
<td>3.7</td>
</tr>
<tr>
<td>NXdomain</td>
<td>5.0</td>
<td>3.2</td>
</tr>
<tr>
<td>Servfail</td>
<td>0.2</td>
<td>0.04</td>
</tr>
<tr>
<td>Refused</td>
<td>0.05</td>
<td>0.02</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>Response Type at SLD</th>
<th>Mean(%)</th>
<th>Variance(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>80.2</td>
<td>3.23</td>
</tr>
<tr>
<td>NXdomain</td>
<td>16.9</td>
<td>2.23</td>
</tr>
<tr>
<td>Servfail</td>
<td>1.5</td>
<td>0.13</td>
</tr>
<tr>
<td>Refused</td>
<td>1.3</td>
<td>0.04</td>
</tr>
</tbody>
</table>

(b)
reflects how the component behaves when it receives a negative response to a query. Determining the values for this system variable involves the detailed characterisation of the querying behaviour at both the client and the recursive resolver.

For the client, the experiments in the lab environment have shown that when a negative response is received for an initial query, the client may automatically resend new identical repeat queries [2]. The amount of repeat queries depends strongly on the type of client’s OS and application browser: clients with various OS and application browser combinations react differently when they receive different type of responses for their initial queries. Table 7 displays Query Multiply Factors for any possible response type and for different application browsers and OS types [2]. It should be noted that the depicted numbers also include the initial query, e.g. in case of the Servfail response, a Linux-Firefox client will send in total eight queries, including the initial query.

Table 7: Query Multiply Factor values for various client’s OS and application browser types.

<table>
<thead>
<tr>
<th>Response Type</th>
<th>Windows XP</th>
<th>Windows 7</th>
<th>Linux</th>
<th>MAC OSX</th>
<th>IE8</th>
<th>Firefox</th>
<th>Safari</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NXdomain</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Partial</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Servfail</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Time-out</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Refused</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Truncated</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

To understand the querying behaviour of the recursive resolver, the analysis of the data of the two most popular resolvers is performed: UNBOUND and BIND9. The result, Query Multiply Factor for the recursive resolvers, is given in Table 8.

Table 8: Values for Query Multiply Factor for the two recursive resolvers.

<table>
<thead>
<tr>
<th>Response Type</th>
<th>UNBOUND</th>
<th>BIND9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NXdomain</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Partial</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Servfail</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Time-out</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Refused</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Truncated</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
4 Operation of the DNS model

As previously mentioned, the model considers the typical DNS architecture: a) client with its OS (and the corresponding stub resolver) and application browser, b) recursive resolver, and c) authoritative NS with the root, TLD and SLD NSs. Figure 3 depicts the DNS reference model as it is implemented in the Microsoft Excel. As seen in Figure 3, the left hand side (i.e. the client side) of the reference model is divided into three different parts: a) query to root, b) query to TLD c) query to SLD. The aim of doing this partition was to be able to determine the number and the sort of response types going back to the client from the root, TLD and SLD NSs. Following this objective, the operation of the DNS reference model will be divided in three different steps: step I) the initial queries are going from the client side to the authoritative NS side, step II) the responses to the initial queries are returned from the authoritative NS side to the client side, and step III) the repeat queries due to the negative responses are reinitiated from the client side to the authoritative NS side. In the rest of this section, the operation of the DNS reference model will be explained by considering each step separately. The model will be explained by using the following input parameters, depicted in Table 9.

<table>
<thead>
<tr>
<th>Table 9: Input parameters of reference model.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of simultaneously active DNS clients</td>
</tr>
<tr>
<td>Fraction of IPv6 clients wrt to the total number of clients</td>
</tr>
<tr>
<td>Number of simultaneously active recursive resolvers</td>
</tr>
<tr>
<td>Primary &amp; secondary NS: average number</td>
</tr>
</tbody>
</table>

**Step I** In the first step, the initial queries are generated by the client and sent to the authoritative NS via client’s OS, application browser, and recursive resolver, respectively. This process can be observed at the first row of the DNS reference model in Figure 3. In this example the client generates 1.1 qpt. It sends queries towards the application browser which forwards 0.83 qpt to the OS of the client. Note the difference between the two query rates which is due to the caching property of the Firefox application browser. As shown in Table 1, Firefox caches 25% of the total queries, meaning that it handles 25% of the incoming queries by itself and 75% of the queries are forwarded to the OS of the client. In Figure 3, the number of incoming queries at the OS, which is Linux, is equal to the outgoing query number in Linux. This is because Linux does not implement caching, as shown in Table 2. Then, at the recursive resolver there are 825 qpt which is due to the in Table 9 given number of simultaneously active clients. Afterwards, queries arriving at the recursive resolver will be classified into four different groups. The distribution of the queries over different classes is based on the system variable Cache Hit Ratio for the resolver, which is given in Table 3. According to the Table 3, 0.1% of the queries belong to the Non-cached group (i.e. the queries will be forwarded directly to the root) and 4.1%, 41.1% and 54.7% to respectively TLD, SLD
Fig. 3: The DNS reference model overview as implemented in Microsoft Excel.
and the Domain-Cached group. Consequently, 54.7% of the queries will be directly answered by the recursive resolver while 45.3% of the queries will undergo the recursive resolution process. Once the recursive resolution has been initiated at the root, it will be performed until the entire domain name is resolved. This means that the queries which are responded at the root with the Valid response type, will be sent to the TLD NS by the recursive resolver. The queries which are again qualified as valid at the TLD NS will be sent to the SLD NS. After receiving the response from the SLD NS, the domain name resolution process for the initial queries will be completed. The distribution of response types at the root, TLD and SLD can be found by using the system variable Response Distribution at Authoritative Name servers, given in Table 5b. The total number of the queries going from one particular UNBOUND resolver to the root, TLD and SLD NSs is found to be respectively 0.83, 40.49 and 368.2 qpt. Recall that 454.6 qpt will be answered by UNBOUND itself. Taking into account the input stating that there are 100 UNBOUND resolvers querying the root, TLD and SLD NSs, the total numbers of the initial queries at root, TLD and SLD NSs is 83, 4049 and 36820 respectively.

**Step II** In the second step, the responses from the authoritative NS will be sent back to the client. The response stream from the authoritative NS to the client is classified, and this classification is based on the response type. We assume that the authoritative NS will answer all the queries. As seen in Table 8, for each Servfail response, UNBOUND initiates four extra repeat query towards the authoritative NS, while for each Timeout response six new repeat queries will be initiated. In the DNS reference model, we assume that the repeat query will have the same response as the initial query. Therefore, in total five Servfail responses will be gathered at the UNBOUND although just one of them is sent back to the client. The same will be done for Timeout responses, i.e. UNBOUND initiates six extra repeat query towards the authoritative NS and only one response will be sent back to the client. Following this line of reasoning, negative responses from the root, TLD and SLD NSs are sent back from UNBOUND to the client side while positive ones are only sent after the recursive resolution process has been completed. As a consequence of the assumption that the repeat queries will have the same response as the initial queries, positive responses at SLD (resulting in a completed recursive resolution process) can only be result of the positive responses starting from the root. Recall that we assumed that each particular UNBOUND serves 1.000 identical users simultaneously, therefore the number of responses at OS (in this example Linux) can be found by simply dividing the value at the resolver by 1.000. These responses are sent from OS to the application browser (Firefox) and from the application browser to the user.

There are two important points that have to be mentioned about the transferring Valid responses to the user. The first point is about a fraction of the Valid>512B responses which leads to the Timeout responses when going from the recursive resolver to the OS. This point is included in the reference model so as to be able to analyze the effect of the residential gateways which can block the packets with size larger than 512B. In such a case, a Valid>512 response is perceived and treated as a Timeout response by the client. The second important point concerns the responses which are given by the application browser and the OS of the client. As explained in step I, a
fraction of the initial queries is immediately returned as the two mentioned components have queried domain names in their caches. Those responses, in this example from Firefox and Linux, are aggregated to the total response and seen in Figure 3 at the place between "User", "Firefox" and "Linux" by means of green arrows pointing to the Valid, Valid > 512B, NXdomain and Truncated responses. Recall that these two client’s components are caching only valid query response types (Valid, Valid > 512B and Truncated) and the NXdomain response type.

**Step III** In this step, for each negative response, there will be new reinitiated repeat queries from the client to authoritative NS. Since response streams, coming from authoritative NS, are kept separated, it is possible to determine how many new repeat queries will be reinitiated from the client to the authoritative side. The repeat queries from the application browser and OS will be reinitiated based on the values given in Table 7. Whether a repeat query is sent again towards authoritative NS depends on the type of the recursive resolver and the type of the response for which a repeat query is reinitiated. Different types of the recursive resolvers have different caching properties, to be seen in Table 8. For example, UNBOUND caches Valid and NXdomain responses. Hence, all repeat queries due to NXdomain responses will be in the cache and they will be answered by UNBOUND.

In Figure 3, for instance, we can observe that for NXdomain responses 0.061 qpt are going back from SLD NS to the client. From Table 7, the Query Multiply Factor for an NXdomain response is two for both Firefox and Linux. Therefore, 0.061 is multiplied by two when passing through Firefox and again by two when going through Linux. Consequently, 0.24 qpt will be gathered at the client’s OS to be sent to UNBOUND. As previously explained, for Query Multiply Factor, the obtained value of 2 for NXdomain response type means that one extra repeat query will be resent for each initial query. Therefore, before sending the repeat queries from OS to the recursive resolver, the number of initial NXdomain responses, which is 0.061 qpt, has to be subtracted from 0.24 qpt (sum of initial and repeat queries). Hence, 0.179 repeat queries will be sent from the OS to the recursive resolver. The same procedure will be followed for each response type and all the repeat queries will be gathered at the resolver. However, the resolver, based on its caching property, will send to the authoritative NS only those for which the caching does not play a role. For example, repeat queries due to the NXdomain responses will arrive at the UNBOUND but they will be not sent to the authoritative NS since the UNBOUND deploys negative caching. As a result, 1, 178 and 3240 repeat queries will arrive at the root, TLD and SLD, respectively.

5 Validation of the DNS model

In this section, we validate the DNS reference model by using a new data set captured also at an UNBOUND recursive resolver but in the different environmental setting. The new data set consists of 30,000 DNS packets with duration of 51 seconds. Although we are aware of the fact that validating the model with a single dataset captured at a specific time of day is a rather limited model validation, it still provides a good indication about
the capability of the DNS reference model to capture the DNS querying behaviour. To validate the model, we first analyze the data and obtain the input parameters so as to run the simulations and compare the model output to the statistics found in the real-world data. Lastly, we perform the sensitivity check of the model by using coefficient of variance indicator.

Before starting data processing, it should be ensured that all the anomalies are cleaned from the data set. For example, we observed that some misconfigured clients send lots of repeat queries for the same domain names although they receive positive answers on their queries. We determined that the most repeat queries are sent for domains "allmx.tue.nl" and "edgesmtp.uu.nl" and excluded the DNS traffic related to these domains from the dataset. Having obtained a cleaned dataset, the number of the initial queries can be determined at different point of interest (POI) in the system. The determination of the initial query numbers is crucial since the DNS reference model will be calibrated with the initial queries at the different POIs in the system. To determine the number of initial queries, first a repeat definition has to be formalized.

Repeat definition  Considering two queries, the second query will be defined as a repeat query if it has the same domain name, query type and destination level as the first query. Additionally, the time difference between two queries has to be smaller than a certain number \( \delta \). For repeats at the recursive resolvers, \( \delta \) is determined to be 13 seconds while for repeats at the authoritative NS, \( \delta \) is 3 seconds. These values are determined by analyzing the client and the recursive resolver behaviour. Recall that in the case of a Servfail response Linux client sends seven repeat queries towards the recursive resolver. The time difference between the initial query and the last repeat query is measured to be around 13 seconds. On the other hand, in the case of a Servfail response, the time difference between the initial query and the last repeat query from the UNBOUND towards the authoritative NS is measured to be 3 seconds. Having defined the repeat query notion, initial queries from a given data set of aggregated queries (i.e. initial and repeat queries) are obtained for both the recursive resolver and the root, TLD and SLD NSs. Table 10 shows these values.

<table>
<thead>
<tr>
<th>Query Type</th>
<th>Resolver</th>
<th>Root</th>
<th>TLD</th>
<th>SLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>7131</td>
<td>13</td>
<td>204</td>
<td>3414</td>
</tr>
<tr>
<td>Repeat</td>
<td>2360</td>
<td>0</td>
<td>8</td>
<td>723</td>
</tr>
</tbody>
</table>

The last model input parameter to be obtained from the real-world data is the distribution of initial queries’ OS types. In fact, this parameter indicates the fraction of the initial queries, generated by a specific client type. OS’s fingerprint on each DNS packet is found by using IP TTL values upon which different types can be distinguished. Table 11 shows the result.
Table 11: Distribution of initial queries’ OS types, based on only initial queries.

<table>
<thead>
<tr>
<th>OS</th>
<th>Linux(%)</th>
<th>Windows(%)</th>
<th>MAC(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fraction</td>
<td>60.1</td>
<td>30.2</td>
<td>9.7</td>
</tr>
</tbody>
</table>

The input parameter values obtained from the real-world data serve as a starting point for the validation process. As the data is captured at one single UNBOUND recursive resolver, the "Number of simultaneously active resolvers" will be 1. Additionally, as data is captured at one UNBOUND recursive resolver, the model will be calibrated at this POI with the number of initial queries, instead of the number of initial queries at the users, i.e. before the client’s OS and application browser. As a consequence, the "Number of simultaneously active DNS clients" is determined by trial and error method: we found that with a query rate of 1 qpt, 9700 users generate 7131 initial queries at the recursive resolver. This number of 9700 users concerns thus "Number of simultaneously active DNS clients". Furthermore, we will ignore the effect of secondary NSs by assuming that there will be no secondary NSs and assume that the fraction of IPv6 is with respect to all clients is 0. Now that we have obtained the input parameters for the model, we can run the model. Recall that we are interested in the number of initial and repeat queries at different POI in the model. The simulation is repeated (30000 times) for each of the three different combinations of the client’s OS and application browser: Windows-IE, MAC-Safari and Linux-Firefox. The outcomes of the simulations are histograms showing the distribution of the initial and the repeat queries at each POI. An example of a histogram showing the repeat query distributions at the recursive resolver can be seen in Figure 4. The most probable values from the distribution of initial and repeat queries at each POI are given in Table 12. Recall that the outcome values are weighted by the distributions of client OS as found in real-data set (given in Table 11).

Table 12: Initial and repeat queries at different POI in reference model.

<table>
<thead>
<tr>
<th>Query Type</th>
<th>Resolver</th>
<th>Root</th>
<th>TLD</th>
<th>SLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>7204</td>
<td>8</td>
<td>350</td>
<td>3030</td>
</tr>
<tr>
<td>Repeat</td>
<td>1530</td>
<td>0</td>
<td>5</td>
<td>350</td>
</tr>
</tbody>
</table>

In the following step we compare these results with the results obtained from the real-world data. We compare both, the fraction of total queries as well as the repeat-initial query ratio at POIs. Query ratio at POIs indicates the ratio between the total number of queries (i.e. initial and repeat queries) at POI and the total number of the queries in the entire system. For instance, in the DNS reference model, the fraction of total queries at the recursive resolver can be found as the ratio between the number of queries at resolver and the number of queries in the entire system, i.e.: \( \frac{(7204 + 1530)}{(7204 + 1530 + 8 + 350 + 5 + 3030 + 350)} = 70\% \).
Fig. 4: Repeat query distribution at recursive resolver.

Table 13 shows the fractions of total queries at POIs in the real-world data and the DNS reference model. It can be seen that DNS reference model predicts well the query ratio over the POIs in the system. Small errors are most probably due to the stochastic nature of the system variables.

Table 13: Query ratio at POIs, obtained from real-world data and DNS reference model.

<table>
<thead>
<tr>
<th>Query ratio</th>
<th>Resolver(%)</th>
<th>Root(%)</th>
<th>TLD(%)</th>
<th>SLD(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real world data</td>
<td>68.5</td>
<td>0.1</td>
<td>1.5</td>
<td>29.9</td>
</tr>
<tr>
<td>DNS reference model</td>
<td>70</td>
<td>0.1</td>
<td>2.8</td>
<td>27.1</td>
</tr>
</tbody>
</table>

The second test point of the DNS reference model concerns the repeat-initial query ratio at POIs. This ratio indicates the fraction of the repeat queries with respect to the total number of queries at a particular POI. For instance, in the DNS reference model, the fraction of repeat-initial queries at the resolver can be found as follows: \(1530 / (1530 + 7204) = 17.5\%\).

Table 14 shows the repeat-initial ratio at POIs in the real-world data and in the DNS reference model. At recursive resolver a difference of 7.3\% is observed. We expect this error occurs due to effect of IPv6 clients. IPv6 clients send two queries in pair for address resolution: A and AAAA query. When they receive a negative response from the recursive resolver, then they resend repeat queries also in pair meaning that they send more repeat queries than the Query Multiply Factor values given in Table 7. The error at the authoritative NS might be due to the effect of secondary NSs. For example, we observed that in case of Servfail response, each additional NS causes five extra queries: at
Table 14: Initial-repeat query ratio at different POI obtained from real-world data and reference model.

<table>
<thead>
<tr>
<th>Initial-repeat ratio</th>
<th>Resolver(%)</th>
<th>TLD(%)</th>
<th>SLD(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real world data</td>
<td>24.8</td>
<td>3.8</td>
<td>17.5</td>
</tr>
<tr>
<td>DNS reference model</td>
<td>17.5</td>
<td>1.4</td>
<td>10.4</td>
</tr>
</tbody>
</table>

first, two repeat queries are sent to primary NS. If it again receives a Servfail response, then it queries the secondary NS. If secondary NS also returns Servfail responses, then UNBOUND will again query the primary NS. This querying pattern continues until each NS is queried five times. In this way, a Servfail response causes in total ten queries instead of five as in the case of only one NS.

As a last step we answered the question of how the variation in the system variables affects the outcome of the DNS reference model. In other words, how sensitive is the model output with respect to the stochastic system variables. This question is answered by using coefficient of variance (CoV) metric. CoV is a statistical measure of dispersion around the mean in a probability distribution. We found that CoVs at the output are smaller than 1, meaning that the dispersion in the distributions is small and all the values are concentrated around the mean. Then, CoVs of system variables and output values are comparable, which implies that the DNS reference model does not amplify the uncertainty due to the random system variables.

6 Case study

In order to illustrate the value of our reference model we briefly indicate how the model can be applied to a specific case. As stated earlier, the DNS is facing several major changes, among which the introduction of DNSSEC. Potentially DNSSEC introduces the risk of an increase in Servfail responses due to validation errors or other factors. For example, any error made in DNSSEC signatures at the authoritative side will result in a validation error at the recursive NS. And by default the recursive NS will feed back the validation error to the client side as a Servfail response. We evaluate the impact of this potential increase in Servfail responses and quantify the increase of DNS traffic towards the recursive resolver, the TLD and SLD NSs as a function of the relative increase of Servfail responses.

First, we investigate the impact of an increase of responses (in %) from the TLD that trigger the resolver to return a Servfail response to the client. In this case we keep the fraction of Servfail responses at the root and SLD constant. In particular, we vary the value of our model input parameter Response Distribution for Servfail at the TLD (see Table 6a) and observe the values predicted by our model for the DNS traffic increase from clients towards the resolver, and from the resolver towards the TLD NS. For the other model input parameters we use the default parameter values presented in the tables in Section 3.2. In this case we focussed on UNBOUND resolver behavior. For the
DNS query volumes and the distribution of traffic per OS we used the real-world data as described in the previous section. The results are presented in Figure 5. The x-axies represent the percentage of increase in Servfail response at the authoritative NS (TLD and SLD). For example, 2% implies that additional 2% of Servfails are added to the percentage of Servfail responses specified in Table 6a. On the y-axis the predicted, relative increase of DNS traffic towards the recursive resolver, respectively towards the authoritative NS is plotted. The figure shows a more or less linear relation between the Servfail increase and the DNS traffic. However, the DNS traffic increase towards the TLD is much stronger, than from the client towards the resolver. More detailed analysis of the model results (not presented here) can explain these results. Not explicitly shown in Figure 5 is the observation that additional Servfail responses triggered by responses from the TLD does not increase DNS traffic between any other POIs.

Similarly, we investigate the traffic increase towards the recursive resolver or towards the authoritative NS as a consequence of the Servfail increase at SLD. This is given in Figure 6. We observe that the DNS traffic towards the authoritative NS increases significantly by the increase of Servfail responses. For example, in case of 10% of additional Servfail response at SLD, the DNS traffic would increase for almost 35% towards SLD NS. We further observe that the increase in Servfail at SLD NS has larger impact on the DNS traffic than the increase in Servfail at TLD NS.

![Fig. 5: Impact of increase in Servfail responses at TLD NS on DNS traffic.](image)

### 7 Conclusion and further research

In this paper we have introduced a global DNS reference model with the aim to assess the scalability of the DNS system in case of certain what-if scenarios. The DNS reference model consists of components that model the DNS behaviour of client OS and application browser, the recursive resolver and the authoritative name servers. The
values of the parameters for these components are obtained from real-world data (captured on the operational SURFnet DNS infrastructure), results from experiments with DNS clients, and complemented with results from data analyses published by other researchers. We validated the model by comparing the DNS model output to the statistics found in the real-world data and conclude that the model predictions are rather accurate. In addition, we discussed the shortcomings and possible extensions of the model and how additional analysis based on real-world data can be done to further increase the accuracy of three model variables that we introduced:

- Cache Hit Ratio, used to characterize the caching property of the client and the recursive resolver,
- Response Distribution at Authoritative Name servers, used to characterize the response behaviour of the authoritative NSs, i.e. the root, TLD and SLD,
- Query Multiply Factor, used to characterise the query behaviour, in reaction to negative query response, of the client and the recursive resolver.

For the first two system variables, the probabilistic distributions are found by analyzing real-world data. We have shown that these system variables can be approximated by a Gaussian distribution. For the Query Multiply Factor, we relied on the lab experiments but also on results published in [2]. Having determined the probabilistic distributions for the system variables we have accounted for the stochastic behaviour of the DNS. For the validation of the model, we relied on the approach of Monte Carlo simulation. We have compared the results from the real-world data and the results from the DNS reference model, and shown that the DNS reference model captures the DNS behaviour properly. We have tested the model performance based on the two test points: the fraction of total queries and the initial-repeat queries ratio at various POIs in the system. We have observed a negligible error at the first test point while the error in the second test point was relatively small. We attributed the error in the second test point to the effects of IPv6 enabled clients and the secondary NS, possibly present in the real-world data set. After validating the model, we have used CoV metric to show that
the output of the DNS reference model output is not sensitive to the variations in the system variables. Finally, we demonstrated the applicability of the model by evaluating the impact of a potential increase in Servfail responses.

For future work we propose to validate the model with data from different DNS environmental settings e.g. a different UNBOUND recursive resolver. Additionally, although we used a data set consisting of 300,000 DNS packets, analysis of a larger data set will be needed in order to determine the response distribution at the root. Remark- ing that the system variable Response Distribution at Authoritative Name servers has a crucial importance for the initial-repeat query ratio, we recommend to extend the data analysis with more and larger data sets to obtain representative numbers for all the system variables. Furthermore, extending the model with the effects of secondary NS and IPv6 enabled hosts, belongs to the possible ways this work could be extended. We expect that modelling these factors would reduce the error and leads to a more accurate DNS reference model prediction of DNS query behaviour.

Previous publications have mentioned that a significant part of the DNS traffic in the real-world data sets is generated by a very limited number of clients. In our analysis we confirmed this effect, that may be resulting from e.g. misconfigured name servers or client. Detecting this kind of behaviour requires some data engineering in order to obtain clear interpretation of the DNS data. More advanced algorithms for detecting these DNS anomalies would contribute significantly to better understand DNS behaviour and would help improving DNS research results.

References


Author Biographies

Yakup Koç received his B.Sc and M.Sc in Electrical Engineering at the Delft University of Technology (TU Delft), Delft, the Netherlands, in 2009 and 2011, respectively. He is currently pursuing the Ph. D. degree in the Section of System Engineering, Department of Multi-Actor Systems, TU Delft. His Ph.D. work focuses on designing smart energy systems, so called SmartGrids. His research interests include network reliability and robustness of complex networks.

Almerima Jamakovic graduated and received a M.Sc. degree in Electrical Engineering at the faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) at Delft University of Technology (TU Delft), the Netherlands, in 2004. After finishing her study Almerima joined the Network Architecture and Services (NAS) Group of TU Delft to work towards a PhD degree, which she obtained in the third quarter of 2008. During her PhD she performed research in the field of complex networks, focusing on the quantitative characterization of such complex structures and its application to robustness analysis. From June 2008 till present day Almerima is employed at TNO, the Netherlands institute for applied research. As a member of the department of Performance of Networks and Systems she works on the topic of quantitative analysis of networks and systems in a wide range of performance- and optimization-related projects in the field of Information systems and Telecom networks.

Bart Gijsen After receiving his Msc. degree in both computing science and mathematics Bart joined KPN Research in 1996 as a researcher in the field of performance analysis of Internet technology based information and communication systems. From 2003 till present day Bart is employed at TNO as a senior innovator in the expertise group Performance of Networks and Systems. One of his focus areas is quantitative modelling and impact prediction of Internet security and stability. Since 2010 Bart is also part-time director of the Dutch ICT Innovation Platform "Critical ICT infrastructures".
Preliminary Evaluation of Potential Impact of Failure in DNSSEC Validation

Kensuke Fukuda\textsuperscript{1}, Shinta Sato\textsuperscript{2}, and Takeshi Mitamura\textsuperscript{2}

\textsuperscript{1} National Institute of Informatics, Japan
\textsuperscript{2} Japan Registry Service, Japan
kensuke@nii.ac.jp, \{shinta,mitamura\}@jprs.co.jp

Abstract. We evaluate what impact of failure in the operation of DNSSEC would have on the top level domain (TLD)-level and registered domain-level authoritative servers by using a one-day long whole DNS queries to/from \texttt{.jp} TLD servers. Our preliminary results reveal that (1) the increase in affected cache servers and ASes due to a failure in DNSSEC operations is characterized by a log-type function. More specifically, 18\% of cache servers and 70\% of ASes accessing TLD servers fail the validation of DNSSEC due to failure at TLD servers in the first 10 minutes, in a scenario with full deployment of DNSSEC. Additionally, 50\% of cache servers are affected within 6 hours. (2) This effect is still limited in the current status of deploying DNSSEC in \texttt{.jp}. Only 0.8\% of cache servers fail to validate a query to current DNSSEC enabled domains in the first 10 minutes. (3) The effect of failure related to popular registered domains (e.g., portals, and shopping sites) is modeled as a stretched distribution close to power-law. Thus, the effect of failure at a few popular domains is significantly larger than that at most of typical domains.

1 Introduction

The Domain Name Service (DNS) plays one of the most essential roles – name resolution – in the Internet. It provides a highly distributed, reliable, and lightweight service to users. However, the original DNS has no authentication mechanism to validate the appropriateness of server responses. The DNS Security Extension (DNSSEC) is an extension of the DNS to provide an authentication mechanism, which consists of a chain of trust from the root server to authoritative servers being responsible for the final name resolution through the DNS hierarchy. To guarantee the chain of trust, DNSSEC mainly relies on two types of resource records, i.e., DNSKEY and DS. The DNSKEY record is the public key of a registered domain maintained by its authoritative server, and The DS record is a key registered to the parent authoritative server in the DNS hierarchy to ensure a chain of trust. The consistent management of two keys by authoritative servers is the most crucial part of DNSSEC to provide a reliable DNS service.

The root and some TLD servers have officially launched the service of DNSSEC (i.e., the registration of DS and DNSKEY) officially, and this is now on the way toward actually being deployed in many TLDs. However, several failures in DNSSEC validation have been recently reported despite the careful operations, causing corruption in the chain of trust in the validation phase [16, 10, 1, 9, 8]. The current impact of
Evaluation of Impact of Failure in DNSSEC Validation

such failures on Internet users is still limited because of the early stages of deployment. However, the impact of failures will become more and more serious on the Internet as DNSSEC becomes more commonly used.

This paper discusses our preliminary evaluation of the spatial and temporal impact of failure in DNSSEC validation at an authoritative server. More specifically, we intend to estimate how many cache servers and ASes are affected by failure over time by using a one-day long DNS traffic trace taken at all ".jp" TLD servers (hereafter, JP-DNS servers). We will first overview failures that have occurred in the real world. Then, we will evaluate how such failures temporally and spatially affects cache servers all over the world, with the dataset. The main findings of our preliminary analysis are as follows; (1) The increase in affected cache servers and its ASes is characterized by a log-type curve. For example, we find that 18% of cache servers and 70% of ASes accessing JP-DNS servers fail the DNSSEC validation due to a failure of them in the first 10 minutes, in a scenario of full deployment of DNSSEC in ".jp", Additionally, 50% of cache servers are affected within 6 hours. (2) This effect is still limited in the current status of deploying DNSSEC in ".jp". Only 0.8% of cache servers fail to validate a query to current DNSSEC enabled domains in the first 10 minutes. However, this is significantly high if we consider the current number of the enabled domains. (3) The effect of failure related to a popular registered domain is modeled as a stretched distribution close to a power law. Thus, the effect of failure at a few popular domains is significantly larger than that at most of typical domains.

2 Failure of DNSSEC operation

We introduces five real examples of failures related to DNSSEC operations.

– RIPE NCC (21th, Sep, 2010) [16]. The affected range was all signed zones administrated by RIPE NCC. The timestamps of the signature for both start and end were set to 0, and they were distributed. The reason for failure was due to trouble in the registry system in the rollover of KSK.

– Nominet UK (11th, Sep, 2010) [10]. The affected range was whole ".uk". There was a contradiction between ZSK and signature. This occurred when a stand-by system did not use the same ZSK key of the primary system.

– AFNIC (12th Feb, 2011) [1]. The affected range was the whole ".fr". The correct signature for the .fr zone was not generated because of bugs in DNS software. It took 3 hours for the recovery.

– mozilla.org (16th, Sep, 2010) [9]. The affected range was ".mozilla.org". Validation of mozilla.org failed because registering and enabling DS keys to its parent server was earlier than enabling its own keys at mozilla.org.

– iab.org (31st, Aug, 2010) [8]. The affected range was ".iab.org" due to the expiration of the signature of the key by accident.

One of the most important lessons is that we could not avoid failures even though we implemented operations carefully. Such failures not only affected to the name resolution of a single domain but also that of a wide range of domains, depending on the location in
the DNS hierarchy. It should also be emphasized that none of them were second/minute-order failures to be recovered. Thus, misoperation related to DNSSEC turned out to be long-time failures in DNS.

From these facts, we could categorize what-if scenarios in failures of DNSSEC validation into three types. (1) root server level failure, (2) TLD level failure, (3) registered domain level failure. Clearly, they are in an inclusive relationship if there is one trust anchor. A root server level failure affects to all queries, and a TLD-level failure affects to all domains to be registered by the TLD. In this paper, we focus on scenarios 2 and 3 mainly because of the availability of the dataset. The failure at authoritative servers we assume is general, which causes the validation error in DNSSEC at cache servers (or validators) that includes the previously mentioned cases.

3 Dataset

We collected all DNS traffic to/from JP-DNS servers from 9am on 31st Mar to 9am on 1st Apr, 2009 (24 hours) with a tcpdump command. The total number of queries to the servers was 1,468,593,954 (17kqps). The servers consisted of seven servers ([a-g].dns.jp) and their instances distributed all over the world. Figure 1 indicates the variations in the number of A record queries received at instances in both domestic and international regions. Different from the root server behavior, there is a clear diurnal trend based on the Japanese standard time (+9 UTC), while this diurnal trend does not fit into the variation in the traffic volume in the residential broadband users as shown in [5]. We can see that the domestic DNS queries have two specific peaks near 0am and 4am, likely related to a logging process (e.g., syslog). However, the peak hour for the international queries shifted by a few hours, corresponding to the daytime in the U.S. and Europe. Moreover, we confirmed that the traffic load on servers differed greatly because of the location of the servers and routing. Using the queries of a single server or instance may have added unexpected bias to the analysis. Thus, we analyze all of the queries to/from the servers as a whole.
The purpose of the study is to demonstrate the impact of failure at authoritative servers on cache servers. We extracted the source IP address of a query packet as the IP address of a cache server. Moreover, we relied on BGP prefix information to relate the IP addresses of cache servers to corresponding AS numbers, served by the routeviews project [17]. The total number of the distinct source IP addresses corresponding to cache servers was 1.7 millions and that of the distinct ASes was about 24,000. Considering a typical value of cache TTL (86400 sec) in JP-DNS servers and the measurement period in our experiment, the number of cache servers will increase more in total.

The data set had no real DNSSEC traffic because “.jp” zone launched the DNSSEC service to the registered domains (i.e., registration of DS keys for them) on January 2011. Consequently, we simulate the impact of failure in DNSSEC validation with this dataset.

4 Results

4.1 Number of cache servers and queries

![Figure 2: Number of appeared IP addresses in AS](image2)

![Figure 3: Number of IP addresses and queries from AS](image3)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Type</th>
<th>Value</th>
<th>Rank</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>isp (jp)</td>
<td>1</td>
<td>11</td>
<td>isp* (vn)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>isp* (tr)</td>
<td>12</td>
<td>12</td>
<td>isp (us)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>isp* (br)</td>
<td>13</td>
<td>13</td>
<td>isp (us)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>isp* (nl)</td>
<td>14</td>
<td>14</td>
<td>isp (us)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>isp* (br)</td>
<td>15</td>
<td>15</td>
<td>isp (jp)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>isp (pt)</td>
<td>16</td>
<td>16</td>
<td>isp (it)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>isp (de)</td>
<td>17</td>
<td>17</td>
<td>isp* (in)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>isp (cn)</td>
<td>18</td>
<td>18</td>
<td>isp (tw)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>isp* (br)</td>
<td>19</td>
<td>19</td>
<td>isp (jp)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>isp (us)</td>
<td>20</td>
<td>20</td>
<td>isp* (fr)</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Rank of ASes (#IP addresses)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Type</th>
<th>Value</th>
<th>Rank</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>isp (jp)</td>
<td>1</td>
<td>11</td>
<td>isp (jp)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>isp (jp)</td>
<td>12</td>
<td>12</td>
<td>isp (jp)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>isp (jp)</td>
<td>13</td>
<td>13</td>
<td>se (jp)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>isp (jp)</td>
<td>14</td>
<td>14</td>
<td>isp (jp)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>isp (jp)</td>
<td>15</td>
<td>15</td>
<td>isp (us)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>se (us)</td>
<td>16</td>
<td>16</td>
<td>isp (jp)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>isp (jp)</td>
<td>17</td>
<td>17</td>
<td>se (us)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>isp (jp)</td>
<td>18</td>
<td>18</td>
<td>isp (jp)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>isp (us)</td>
<td>19</td>
<td>19</td>
<td>se (us)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>isp (tw)</td>
<td>20</td>
<td>20</td>
<td>ac (jp)</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Rank of ASes (#queries)
First of all, we characterize basic statistics on cache servers and corresponding ASes in the dataset. Figure 2 plots the cumulative distribution of the number of distinct IP addresses (i.e., cache servers) per an AS (on log-log scale). The fit in the figure indicates a power law distribution with a slope of 0.65. The plot has a stretched distribution, i.e., a small number of IP addresses appeared in most of ASes, while a huge number of addresses appeared in a few ASes. For example, 90% of ASes had less than 55 IP addresses, but we still observed some ASes with more than 30,000 IP addresses. ASes with more than 10,000 IP addresses were mostly commercial ISPs. Table 1 lists the top 20 ASes in terms of the number of IP addresses. The types of the ASes are all commercial ISPs over the world, even though we have only shown the type of ASes instead of the actual AS number or their name due to privacy issues. However, we could not find any ASes corresponding to academic institutions. By manually checking host names of sampled IP addresses belonging to commercial ISPs, we confirmed that many of them were related to those assigned to their customers (residential users and small businesses). This suggests that each customer uses own cache servers rather than cache servers provided by ISPs in such ASes. Also, those with asterisks “*” were listed in the “Top50: Bad Hosts and Networks” by HostExploit [7], and such cache servers are likely related to spam activities.

We have also summarized the top 20 ASes in terms of the number of queries in Table 2. The types of ASes have been categorized into commercial ISPs (isp), search engines (se) and academic networks (ac). Different from the number of IP addresses in AS, the most listed ASes are commercial ISPs in Japan. Also note that the effect of the search engine sites is not negligible.

Furthermore, Fig. 3 is a scatter plot of the number of IP addresses per AS and the number of queries from ASes in 24 hours (on log-log scale). The number of queries from ASes is widely distributed over eight orders of magnitude, and the ASes with numerous queries correspond to commercial ISPs in Japan, as was expected. Similarly, we also confirmed that the number of queries from an IP address and AS were followed by a stretched distribution, though they are not shown for brevity. Furthermore, the plots indicate a positive correlation between the number of IP addresses and the number of queries. The difference in regions in the scatter plot roughly reflects the type of ASes, even though it might be biased by country and language. ASes with a few IP addresses with a small number of queries correspond to small size ASes. Those with a few IP addresses with a large number of queries suggest large size ASes with cache servers provided by ASes. Those with many IP addresses with a large number of queries indicate large size ASes with mixtures of cache servers organized by customers and ISPs. These rough categories are useful for taking into consideration what effect propagation has on validation failure. Note that we ignored the effect of the cache servers behind a NAT for the sake of simplicity in our analysis.

4.2 Effect of TLD level failure

Now, we investigate into the effect of TLD-level DNSSEC failure causing validation errors in a whole TLD, which is a similar type of failure that occurred in the RIPE, AFNIC, and Nominet explained in section 2. The metric here we used is the propagation time and space from failure at a TLD-level server to cache servers. The original packet
traces collected at all instances were split into subtraces whose time bin was 10 minutes. Then, these subtraces for all instances with the same time (in 10 min bins) were merged into one. Finally, we obtained a list of IP addresses, corresponding ASes, and query names per time period.

The propagation effect is measured by the cumulative number of IP addresses or ASes to be affected by a failure over time. Thus, it approximately corresponds to the number of unique IP addresses/ASes from the beginning \((t = 0)\) to a given propagation time \((t = n; 0 \leq n < 144)\). Furthermore, we focus on the propagation time when all of the IP addresses in an AS appeared in the subtraces. This indicates the time that all cache servers in an AS failed in DNSSEC validation due to failure at the TLD level failure.

The data were collected for 24 hours starting from 9am (JST). One method of estimating the effect of the propagation is to replay the subtraces from the beginning \((t = 0)\) to the end \((t = 143)\) (hereafter \textit{replayed}). However, this may have been biased by the daily DNS traffic trend shown in Fig.1 as well as residential user traffic trend [5]; In order to remove such the trends, we also use shuffled data that consist of the same data but the time index in subtraces is randomly shuffled (hereafter \textit{shuffled}). In other words, the shuffled data preserves the time correlation of the queries in a subtrace (i.e., within 10min), though they were non-correlated between subtraces. We calculated the average metrics of the shuffled data with 10 different random seeds.

Figures 4 represents the cumulative number of newly appeared IP addresses of cache servers to access JP-DNS servers. The two plots correspond to the result for replayed traces and that of shuffled ones. The shapes of both curves resemble a log-type function that rapidly increases and then saturates over time. The shuffled data are more smoother than the replayed ones because the trend has been removed. The total number of unique IP addresses in the dataset was about 1.7 M. In the first 10 minutes \((t = 0)\), the number of accessed IP addresses reaches about 250,000 (18% of the total accessed IP addresses). This means that 18% of the cache servers will fail to validate DNSSEC

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig4.png}
\caption{Newly appeared IP addresses over time.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig5.png}
\caption{Newly appeared ASes over time.}
\end{figure}
with JP-DNS servers in the first 10 minute if a single failure occurs at the JP-DNS servers.

Note that the replayed data offer more grounds for optimism than the shuffled data. For instance, the propagation time that affected to 1M cache servers is 10 hours for the replayed one, but only 4 hours for the shuffled data. It should also be emphasized that the IP addresses that appeared in the first 10 minute are likely cache servers that serve more clients because we expect that these servers would have a higher probability of frequently appearing in subtraces (hereafter we refer those to core cache servers). On the other hand, those that newly appeared in the rest of the time would likely be cache servers at residential or small business users in a large ISP (hereafter edge cache servers.)

Similarly, Fig. 5 displays the number of newly accessed ASes, which are mapped with BGP information. Not surprisingly, in the first 10 minute, 15,000 ASes will experience the validation failure more than once, because of the affection to core cache servers in the ASes. We can see that the curves rapidly saturate in a first few hours, meaning that there are a small number of ASes sending a query to TLD servers spontaneously.

Related to the effect to ASes, Fig. 6 plots the number of ASes in which access to all IP addresses were completed over time. In other words, this corresponds to the elapsed time in which a given AS has been completely affected by failure at JP-DNS servers. As expected, the number of affected ASes is smaller than that in the previous figure. We confirmed that approximately 2,500 ASes are affected in the first 10 min; there are a small number of cache servers (i.e., mainly core cache servers) in such ASes. The gradual increase in the curves suggests that there were a large number of leaf cache servers.

Finally, we normalize the number of IP addresses and ASes to compare them each other as plotted in Fig. 7. We can see that 60% of ASes (red), 18% of IP addresses (green), and 10% of ASes with all cache servers (blue) directly suffered by a failure in
the first 10 minute. By the first 2 hours, 85% of ASes, 35% of IP addresses, and 25% of ASes with the all cache servers will fail to the validation.

Considering the current status of deploying DNSSEC in “.jp” zone, we evaluate the impact of failure at JP-DNS servers in currently DNSSEC enabled domains (309 domains as of July 2011). Figures 8 and 9 display the number of newly appeared IP addresses and ASes that queried to DNSSEC enabled domains. That is, they correspond to the number of cache servers and their ASes to encounter a DNSSEC validation error when they resolve domain names with DNSSEC through the chain of trust. The shapes of the curves resemble those in the previous corresponding figures. In the first 10 minute, 13,000 IP addresses appeared out of 1.6 millions. The impact of the failure in the current situation would be 0.8% (= 13,000/1,700,000) of the total number of cache servers; the percentage is up to 12% of the ASes. Finally, we confirmed that 34% of the addresses and 75% of the ASes are affected after 1 day. This impact is sufficiently significant, when the ratio of DNSSEC enabled domains (30%) in the total number of JP domains (> 1.2 millions) is considered. One reason for this is the earlier deployment of DNSSEC in some nation-wide ISPs.

Fig. 8: Newly appeared IP addresses that queried DNSSEC enabled zones

Fig. 9: Newly appeared ASes that queried DNSSEC enabled zones

4.3 Effect of registered domain level failure

Here, we will provide more details on what effects failure had on popular domain names. The failures at mozilla.org and iab.org that were discussed in section 2 are included in this category. Table 3 lists the top-50 queried A records in the whole trace. Again due to privacy issues, we only show the types of queried names instead of the actual queries. dns corresponds to DNS servers that have been deduced from the prefix name of the query such as dns or ns. portal indicates the name of popular web sites (e.g., news, and shopping sites). anomaly was labeled by manual inspection. misc refers to the rest of them. The bracket indicates suffix of gTLD and ccTLD.

The most queried name was the name of a secondary DNS server of a server hosting provider due to some errors. The second most queried name was a no-existent domain.
Table 3: Commonly queried types of A records

<table>
<thead>
<tr>
<th>rank</th>
<th>type</th>
<th>rank</th>
<th>type</th>
<th>rank</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>anomaly (co.jp)</td>
<td>22</td>
<td>anomaly (jp)</td>
<td>39</td>
<td>portal (co.jp)</td>
</tr>
<tr>
<td>2</td>
<td>anomaly (jp)</td>
<td>23-36</td>
<td>dns (jp)</td>
<td>40</td>
<td>portal (jp)</td>
</tr>
<tr>
<td>3</td>
<td>dns (ad.jp)</td>
<td>27</td>
<td>portal (jp)</td>
<td>41</td>
<td>dns (ne.jp)</td>
</tr>
<tr>
<td>4-12</td>
<td>dns (dns.jp)</td>
<td>28</td>
<td>dns (ne.jp)</td>
<td>42</td>
<td>dns (ne.jp)</td>
</tr>
<tr>
<td>10-11</td>
<td>dns (ne.jp)</td>
<td>29</td>
<td>portal (co.jp)</td>
<td>43</td>
<td>misc (co.jp)</td>
</tr>
<tr>
<td>13</td>
<td>portal</td>
<td>30</td>
<td>portal</td>
<td>44</td>
<td>portal</td>
</tr>
<tr>
<td>14-25</td>
<td>dns (ad.jp)</td>
<td>31</td>
<td>dns (jp)</td>
<td>45</td>
<td>misc (ne.jp)</td>
</tr>
<tr>
<td>15</td>
<td>misc (co.jp)</td>
<td>32</td>
<td>local.status</td>
<td>46</td>
<td>dns (ad.jp)</td>
</tr>
<tr>
<td>16</td>
<td>dns (jp)</td>
<td>33</td>
<td>dns (co.jp)</td>
<td>47</td>
<td>portal (jp)</td>
</tr>
<tr>
<td>17-30</td>
<td>dns (ad.jp)</td>
<td>34</td>
<td>portal (co.jp)</td>
<td>48</td>
<td>portal (jp)</td>
</tr>
<tr>
<td>19</td>
<td>portal (jp)</td>
<td>35</td>
<td>portal (jp)</td>
<td>49</td>
<td>portal (ne.jp)</td>
</tr>
<tr>
<td>20-22</td>
<td>dns (ad.jp)</td>
<td>36</td>
<td>dns (jp)</td>
<td>50</td>
<td>misc (ne.jp)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>37</td>
<td>dns (jp)</td>
<td>51</td>
<td>portal (co.jp)</td>
</tr>
</tbody>
</table>

The third one was a secondary name server of an ISP. The fourth to eleventh names corresponded to the secondary name servers of JP-DNS (*.dns.jp). Thus, the top 10 queries should be considered as special queries. We observed various DNS servers as well as popular portal sites in the rest of queries.

![Fig. 10: Number of affected IP addresses for top 1000 queries. (a) 10min bin, and (b) 1 hour bin.](image)

Now, we evaluate what effect failure had on DNSSEC operations at a well-known site. For example, what would happen if popular domains (e.g., portal sites) wrongly registered their DS or DNSKEY. Figure 10 indicates the number of IP addresses of cache servers that sent queries of commonly appeared (top1000) names. The x-axis indicates the rank of the queried domain names in ascending order. The number of top-10 queries are distributed around 1000-10000 IP addresses, though there are special queries like anomalies and dns.jp servers as shown in Tab. 3. However, the plots from 10th to 1000th are followed by a stretched distribution. The guideline in the figures indicates a power-law fit with a slope of 0.65. Thus, the impact of failure in commonly queried names is more and more significant. Even for a 10 minute failure in the top1000 names, more than 1000 cache servers will fail to resolve the names. The number of cache servers increases to 4000-40000 for 1 hour failure (Fig.10(b)). Also note that our...
manually inspected and found that most outliers from the power-law fit were queries to
non-existent domains especially for smaller numbers of IP addresses.

In summary, the impact of failure in DNSSEC operation on a popular domain name
is followed by a heavy-tailed distribution, and more popular sites affects more cache
servers.

5 Discussion

We will now discuss the implications of our simulation results on DNSSEC operations.
Two issues need to be considered to minimize the impact of failure. One is how rapidly
operators find failures (early detection), and the other is how quickly they mitigate the
effect of the failure (early mitigation).

In general, the TLD authoritative servers hardly identify a failure of validation with-
out the information from the cache servers because the validation is performed by cache
servers. One potential way of the early detection is to collect statistical information re-
lated to the validation from cache servers. This approach – cache servers as sensors –
is attractive, but requires collaboration of TLD authoritative servers and cache servers
with a secure protocol for the data reliability. As another approach, operators at the
TLD servers could monitor the validity of own TLD servers with active probes from
their cache servers. In this case, our results suggest that the frequency and the order
of domain names to probe are important for effective detections, because the popular-
ity of the domain name to be accessed is followed by a power-law decay. Thus, more
frequently asked domain names should be probed more than others even for monitoring.

A more problematic issue is likely to be early mitigation. It is not easy to identify
the root cause of failure and to fix it even if operators notice the failure earlier. As
explained in section 2, the time required for the recovery ranges from hours to days.
The operational time period for adding/removing DS records to the parent authoritative
server must be short to reduce failures from spreading. This period is currently expected
to be less than 48 hours in root servers (ICANN) and 15-30 minutes in JP-DNS.
However, our simulation results indicated that the current setting in the root servers
would have a huge impact on the users. Thus, a more effective strategy is required for
quickly blocking the diffusion of the failure, though the current solution would be only
removal of DS records at the parent authoritative servers.

6 Related works

There have been many studies that have characterized DNS traffic behavior for the
past few decades due to the importance of DNS in the Internet. They have broadly
been classified into (1) passive or active measurements, and (2) authoritative servers
(including root servers) and cache servers.

The behavior of the root servers has been well analyzed with passive measurements
[3, 2, 18, 4]. These studies mainly clarified the quality of the queries. For example, 95%
of the queries to the root servers are invalid. However, it is hard to obtain all queries
heading to the root servers because of the highly distributed instances, while the data
at most of the root servers have been collected by the DITL activity. Instead, we only
used JP-DNS data, but these completely covered all queries to the servers, enabling us to estimate the impact of failure. We expect that the results of our analysis can be applied to other TLDs, depending on their size. As for load of DNS servers, Pang et al. investigated what impact failure had authoritative DNS servers and cache servers [12] with DNS access logs and CDN logs. However, they did not focus on characterizing DNSSEC behaviors.

Recently, in terms of troubleshooting, active measurements on authoritative servers and its analysis have been performed [14, 13, 15, 6]. They basically investigated the dependencies of authoritative DNS servers with graph structures. Their earlier paper showed the lame delegations and the cyclic anomalies in DNS. Regarding the impact of the DNSSEC, Osterweil et al. have been analyzing the deployment of DNSSEC with active measurements [11]. A passive measurement work on a cache server-side focused on anomalies of queries [19], characterizing the types of non-existent domain queries and fast flux servers.

7 Conclusion

We investigated the temporal and spatial impact of failure in DNSSEC validation on cache servers and corresponding ASes, with a one-day long dataset covering all queries to/from JP-DNS servers. Our main findings in analysis were: (1) The increase in affected cache servers and its ASes is characterized by a log-type curve. More specifically, 18% of cache servers and 70% of ASes accessing to JP-DNS servers will fail the validation of DNSSEC by a failure at TLD servers in the first 10 minutes in a scenario of full deployment of DNSSEC in ".jp". Additionally, 38% of cache servers were affected within 2 hours. (2) This effect was still limited under the current status of deploying DNSSEC. 0.8% of cache servers failed to validate a query to DNSSEC enabled domains in the first 10 minutes. However, this ratio is significantly higher considering the number of the currently available DNSSEC enabled domains in ".jp". (3) The effect of a failure related to a popular registered domain was modeled as a stretched distribution close to a power-law. Thus, the effect of a failure at a few number of popular domains was significantly greater than that at most of typical domains.

Our preliminary results suggest that more specific analysis need to be done. In particular, we will perform what impact failure has in one of the instances and server selection, as well as analyze impact that depends on the status of DNSSEC deployed at registered domains.

Acknowledgments

A part of this work is supported by a Grant-in-Aid for Young Scientists (A), JSPS, Japan.

References


Author Biographies

Kensuke Fukuda is an associate professor at the National Institute of Informatics (NII), Japan. He received his Ph.D degree in computer science from Keio University at 1999. He worked in NTT laborratories from 1999 to 2005, and joined NII in 2006. In 2002, he was a visiting scholar at Boston University. Currently, he is also a researcher of PRESTO JST (Sakigake) since 2008. His research interests are Internet traffic measurement and analysis, intelligent network control architectures, and the scientific aspects of networks.
Shinta Sato is a research engineer of Research and Development Dept., at Japan Registry Services Co., Ltd. (JPRS) in Tokyo, Japan. His experience includes the administrations and operations of DNS and Registry Systems of .JP Top Level Domain (TLD). In his current position, he is involved in the technology strategy planning and research related to DNS.

Takeshi Mitamura is the manager of Research and Development Dept. and Systems Management Dept., at Japan Registry Services Co., Ltd. (JPRS) in Tokyo, Japan. He is involved in the technology strategy planning and management/operation related to DNS and Internet Domain Names infrastructures.
A bi-objective Mixed Integer Linear Program for load balancing DNS(SEC) queries

Stanislas Francfort, Daniel Migault, Stéphane Sénécal
Orange Labs
38-40 rue du Général Leclerc, 92794 Issy-les-Moulineaux CEDEX 9, FRANCE
{stanislas.francfort, daniel.migault, stephane.senecal}@orange-ftgroup.com

Abstract. This paper addresses the problem of efficiently load balancing domain name queries on DNS resolution platform. Efficiently means that resource required for the resolution MUST be optimized compared to existing architectures and equally shared among the nodes of the resolving platform. This optimization leads to well balance the load, which reduces drastically the number of servers to be deployed as well as to significantly save power consumption. In order to achieve this goal, the paper considers splitting DNS traffic according to the queried FQDN – rather than IP addresses. Then, based on the specific statistical distribution of the DNS queries, we define an efficient method so that nodes deal with the roughly same number of FQDN, as well as to deal with the same number of DNS queries. We formulate this bi-objective problem under the framework of a Mixed Integer Linear Programming model and use a solver to process the resulting optimization problem. The model we developed is very efficient on real data, and provides a promising offline process for load balancing DNS queries on a large resolution platform.

Keywords: DNS, DNSSEC, load balancer, Integer Programming.

1 Introduction

Domain Name System (DNS) [1, 2] is the protocol used to bind a Fully Qualified Domain Name (FQDN) like www.google.com to an IP address. Thus every time end users are typing an URL in their web browsers, a DNS resolution is performed in order to find where the web server is located.

Today, DNS resolving servers deal with traffic that has a daily mean of 40 000 queries per second, with flash crowds up to 120 000 which corresponds to the DNS traffic Orange has with its residential End Users. On the other hand, the DNS traffic keeps on increasing and roughly double each year. DNS has been designed in the eighties so that resolution could be efficient and involve the least possible resources. As such, the DNS resolving platforms are usually splitting traffic between the servers, thanks to load balancers that only consider networks layers, i.e. IP addresses. As a result, the traffic is uniformly split among the servers, and each server deals with the same amount of queries. The advantage of such architecture is that it is very scalable, and when the traffic increases, network administrators only need to add more servers. The drawback of
this architecture is that there is no synchronization between the servers’ caches and that servers perform parallel DNS resolutions for popular FQDN. This has not been an issue with DNS since resolutions are quite straightforward. However, with DNSSEC, the DNS SECurity extension, cf. [3–5], resolution requires one or more signature checks as well as longer datagrams. In fact such resolutions require much more resources than DNS’s resolution and [6, 7] show that, depending on the software implementation, the platform requires between 2 and 4.25 times more resource with DNSSEC than with DNS.

A key point is to observe that if the same DNSSEC query has been addressed previously to the same server within a given period of time, defined by the variable “Time To leave” (TTL), the server does not perform another resolution over internet and sends back a response already stored in cache. In order to save computational resources, we aim at optimizing the number of queries for which the response is stored in cache. By affecting every query of a given FQDN to the same server (or node), we achieve for each FQDN only one cryptographic resolution every TTL seconds. On the other hand traditional architectures that splits DNS traffic without considering the FQDN, can perform up to \( J \) cryptographic resolutions every TTL where \( J \) is the number of nodes of the resolving platform.

In our solution, the distribution, that is to say the definition of which FQDN goes on which server is defined by an offline process (Fig. 1a). Once computed on a captured data, the table is stored and used to load balance the queries online. The table has to be computed for each given period of time. This split is performed according to the requested FQDN and does not consider the network layer parameters such as IP addresses. In this case, each server of the platform is responsible for a set of FQDN, and all queries for a given FQDN will be resolved by a specific server. The matching FQDN/Server is stored in the table. This maximizes the probability that the response is already cached, however, by doing so, there is no guarantee that servers will have to deal with an equal number of queries.

In this paper, we are looking how the traffic can be distributed so that the two following criteria can be satisfied:

1. The number of FQDN is well balanced between the different servers of the platform.
2. The number of DNS queries is well balanced between the different servers of the platform.

The DNS traffic considered in this paper is 5 min DNS capture from our residential End Users at the rush hour 19:33 – 19:38 on October 19 2009.

Criterion 1 ensures that each server will store approximately the same amount of data in its cache, aiming to perform approximately the same number of cryptographic computations while criterion 2 ensures that each server resolves approximately the same number of queries. Optimizing along those two objective functions turns the problem into a bi-objective well-balancing integer program on a huge data set captured from real DNS traffic.

As far as we know, no such study on minimizing cryptographic DNSSEC computations by optimizing the queries to the cache have been conducted, nor any solutions addressing the large scale problem of well balancing a huge number of integers into a small
number of boxes\footnote{Note that this problem is different from the Bin-Packing problem or the Knapsack problem. Our problem could be reduced to those ones if we had to fit exactly the same amount of queries (resp. signature checks) to each server.}. Section 2 introduces the problem modeled as a Mixed Integer Linear Program (MIP). We also point out the fact that the size of the instances is so huge that it can not be solved in such a way. In section 3 we explain how the very specific statistical distribution of the DNS queries allows us to split the data into two parts, one solved with an efficient MIP, the other balanced with a fast heuristic. Experimental results, and comparisons to existing solutions such as round-robin, showing the efficiency of our solution, are presented in section 4. Finally, section 5 concludes the paper and describes future work.

2 Modeling the problem as a single objective Mixed Integer Linear Program

The problem \( (P) \) we have to solve, in order to fill the table, can be formulated as follows: “minimizing the difference of the number of queries received by each server while minimizing the number of FQDN received by each servers with respect to the constraint: each query of a given FQDN has to be sent to the same server”.

2.1 Aggregating the two objectives

First, note that this is a bi-objective problem (i.e. which optimizes along two objective functions), and in order to manage it, we construct a single aggregate objective function. We aggregate them in a standard way, by a linear combination of the two objective functions:

\[
\min \sum_{i=1}^{n} w_i x_i + \sum_{j=1}^{m} y_j,
\]

where \( w_i \) and \( y_j \) are the weights of the objectives, and \( x_i \) and \( y_j \) are the decision variables.
functions. Let $k \in \mathbb{R}$ be an aggregating parameter. In order to compare the respective computing resources of a unique standard DNS query $C_{\text{req}}$ and a unique cryptographic signature check $C_{\text{sig}}$, we can define $k$ to be $\frac{C_{\text{sig}}}{C_{\text{req}}}$. This aggregation is modeled by inequation (9) and objective function (10).

[9] shows that if the checks of the signature computation are stored in cache, then the servers can be loaded by 3.33 times more queries than if the checks are not in cache. This leads us to a good approximation of $\frac{C_{\text{sig}}}{C_{\text{req}}}$ and we will set the aggregating parameter $k$ to be equal to 3.33 in the remaining of the paper. However as [7] shows such values may vary with the used implementation and its configuration.

### 2.2 Modeling the problem as a Mixed Integer Linear Program

In the following we denote by $i \in I$ the FQDN, and by $N_i$ the number of queries whose object is FQDN $i$.

In a previous work, we modeled the problem into four different MIP models, and we compared them. One of them performed far more efficiently than the three others. We present the four models in appendix as well as the comparison of how fast they converge. In the present paper, we only present the most efficient one, and we will refer to it as “the” MIP model. Note that this result is conform to the studies we can find in [13].

Consider the following constraints and variables:

$$
I \quad \text{set of FQDN} \quad (1)
$$

$$
J \quad \text{set of servers} \quad (2)
$$

$$
x_{i,j} = \begin{cases} 1 & \text{if FQDN } i \text{ is assigned to server } j \\ 0 & \text{else} \end{cases} \quad (3)
$$

$$
k \in \mathbb{R} \quad \text{an aggregating parameter} \quad (4)
$$

$$
N_{i \in I} \quad \text{the number of query for FQDN } i \quad (5)
$$

We can now introduce the following variables: $\forall j \in J$ a server, let $S_j$ (resp. $T_j$) be the total amount of queries (resp. FQDN) for server $j$.

It is then possible to express problem $(P)$ as a MIP with 3 sets of constraints and a single objective function:

$$
S_{j \in J} = \sum_{i \in I} N_i \times x_{i,j} \quad \forall j \in J \quad (6)
$$

$$
T_{j \in J} = \sum_{i \in I} x_{i,j} \quad \forall j \in J \quad (7)
$$

$$
\sum_{j \in J} x_{i,j} \geq 1 \quad \forall i \in I \quad (8)
$$

$$
S_{j_1} + k \times T_{j_2} \leq M \quad \forall j_1, j_2 \in J \quad (9)
$$

Objective function: minimize $M \quad (10)$
3 DNS queries data

In this section we show that our MIP model does not behave very well on real data. Actually, it is not possible to solve a MIP on such a huge data. But thanks to the specific statistical distribution of the data, we are going to derive another far more efficient model.

In order to guarantee that data for this study are realistic, we considered DNS queries captures from operational DNS servers of a telecommunication operator company measured in 2009. The main file is a brute capture of 5 mn of typical traffic, which represents about 800 MB of raw data.

3.1 DNS statistical distribution

The table 2a lists the ten most required FQDN during the 5 minutes of capture of the network traffic flow. Table 2a clearly shows that the 10th requested FQDN is half less popular than the most popular FQDN. Indeed, the DNS queries repartition is very imbalanced. Thus, within 5 mn, we captured 17 299 154 queries distributed in 1 211 880 FQDN which leads to an average of 14.27 queries per FQDN.

However, this average value is not very meaningful. In fact, the minimum of this repartition is 1 query, but the median is also equal to 1.

There are actually 837 154 FQDN requested a single time and 374 726 FQDN requested strictly more than once. As a counterpart, the maximum values 271 586 and is obtained for an unique FQDN.

This observation on the data points the fact that the statistical distribution is very specific and could be used to derive a more efficient model than solving a MIP on the
whole data. Let us study in details this very imbalanced repartition in the remaining of the section and take it into account in order to decompose our problem into two sub-
problems. From [9] which studies clustering DNS data, it is noticeable that the different FQDN have quite different query frequencies, which are distributed like a power law. In fact, some phenomena in which the notoriety increases because of notoriety (Matthew Effect) give birth to power law distribution. [10] describes an analysis of the queries distribution with respect to domain names on the AOL servers during one day. The conclusion deduced from the analysis is that the repartition of queries with respect to domain is indeed distributed as a power law distribution. The case of popularity in DNS analysis is similar.

However, we do not have to prove that the repartition of FQDN queries follows a power law distribution. Instead, our interest is to investigate the global behavior and to note that we can estimate it roughly by a power law. Under the power law, there is a great imbalance between the most frequent events and the rare ones, while the rare events are numerous. This observation allows us to design an efficient method to resolve the problem.

The Fig. 1b represents the distribution of $N_i$ from the most frequent to the rarest. Depicted in the graph with a log-log scale, this distribution is roughly linear, which is one of the properties of power law distributions.

The quantiles of this distribution are shown in Table 2b. They are interesting for demonstrating the imbalance characteristics of queries repartition $N_i$ among the different FQDN $i$.

The frequencies associated to FQDN definitely follows a largely non-Gaussian distribution law. These characteristics make uniform repartition among the servers of the platform a very inefficient way to split the DNS traffic. Simulations confirmed this fact as we can see in the first line of table 3. Indeed, the query rate associated to each FQDN is very imbalanced, then, affecting too many popular FQDN to a single server would result in a very imbalanced distribution of the incoming traffic. The following section explain why the proposed solution works well if the data is distributed as a power law.

### 3.2 Decomposition into two subproblems

In order to make DNSSEC traffic well balanced among the different servers, a strategy resulting from this very imbalance distribution is to share the problem into two sub-
problems, i.e. defining the set of FQDN whose distribution is defined using MIP and the set of FQDN whose distribution can be defined with a uniform stateless function like a hash function.

Let $s$ be a threshold, and $I_s$ the set of FQDN receiving more than $s$ queries. From now on, we are no longer trying to calibrate $s$, but instead, the linked number $||I_s||$, which is the one we want to choose in order to separate efficiently the two subproblems.

The first subproblem is to assign carefully the $||I_s||$ FQDN that are frequently requested to servers, by solving the MIP on this subset of FQDN.

The second one is to assign uniformly the $||I - I_s||$ others. Observe that since DNS

---

2 However, claiming without ambiguity this is indeed a power law would require more than the arguments presented in this paper, as shown in [11, 12].
queries are temporally uncorrelated, this uniform repartition is similar to a round-robin repartition on servers.

We are clearly taking advantage of the imbalanced distribution. Table 3 shows indeed that the number of queries managed by solving a MIP increases very fast when \( \|I_s\| \) increases, thanks to the very imbalance distribution of \( N_i \). Moreover, simulations on real data shows that when a uniform distribution is performed over the remaining \( I - I_s \) FQDN, the imbalance due to those FQDN not performed by the MIP decreases very fast. Let us call “residual imbalance” this imbalance, calculated as the difference between the load of most loaded server and the load of the less loaded one. The third column in the table 3 shows how this residual imbalance decreases very fast with respect to \( \|I_s\| \).

The portion of the most frequently requested FQDN we have to consider in the MIP

\[
\|I_s\| \quad \sum_{N_i > s} N_i \quad \text{Residual imbalance} \quad S_{\text{max}} - S_{\text{min}}
\]

<table>
<thead>
<tr>
<th>|I_s|</th>
<th>\sum_{N_i &gt; s} N_i</th>
<th>Residual imbalance</th>
<th>|I_s|</th>
<th>\sum_{N_i &gt; s} N_i</th>
<th>Residual imbalance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>805 329</td>
<td>1 000</td>
<td>10 990 719</td>
<td>50 768</td>
</tr>
<tr>
<td>100</td>
<td>5 524 529</td>
<td>233 180</td>
<td>2 000</td>
<td>11 623 826</td>
<td>27 468</td>
</tr>
<tr>
<td>200</td>
<td>7 043 815</td>
<td>141 988</td>
<td>4 000</td>
<td>12 683 199</td>
<td>15 579</td>
</tr>
<tr>
<td>300</td>
<td>8 030 281</td>
<td>111 583</td>
<td>8 000</td>
<td>13 562 536</td>
<td>10 414</td>
</tr>
<tr>
<td>400</td>
<td>8 656 926</td>
<td>88 818</td>
<td>16 000</td>
<td>14 280 865</td>
<td>6 202</td>
</tr>
<tr>
<td>500</td>
<td>9 105 475</td>
<td>77 275</td>
<td>32 000</td>
<td>14 885 176</td>
<td>3 799</td>
</tr>
<tr>
<td>700</td>
<td>9 734 240</td>
<td>63 588</td>
<td>48 583</td>
<td>15 193 650</td>
<td>2 625</td>
</tr>
<tr>
<td>1000</td>
<td>10 390 719</td>
<td>50 768</td>
<td>374 726</td>
<td>16 462 000</td>
<td>1 009</td>
</tr>
</tbody>
</table>

(a) 1 - 700 FQDNs (b) 1000 - 374 726 FQDNs

Fig. 3: Number of FQDN managed by MIP and residual imbalance w.r.t. \( I_s \)

results from a trade-off. In fact the more FQDN we consider in the MIP, the better the traffic will be balanced among the servers. However the more FQDN we consider, the bigger the MIP will be, and hence the more time, the more resources are required. Note that if \( \|I_s\| \) increases too much, we can expect the MIP to become intractable. On the contrary, when \( \|I_s\| \) decreases, the optimization problem may become easy, but the imbalance increases very fast (cf. Table 3).

The power-law like distribution described in section 3 ensures on one hand that the imbalance decreases very fast, and on the other hand that most of the queries will be affected to servers by the MIP, even if \( \|I_s\| \) is very small which makes our solution far more efficient that round-robin. As a consequence, due to the power law like distribution of the data, we can expect to find an \( \|I_s\| \) small enough for the MIP to be tractable, but big enough to lead to an acceptable small residual imbalance.

Note that the number of variables of the MIP is \( \|I_s\| \times \|J\| \) where \( \|I_s\| \) is the number FQDN balanced by the MIP, and \( \|J\| \) is the number of servers.

As examples, choosing the threshold \( s \) to be the median = 1 (resp. the mean = 14.27) of the \( N_i \)'s leads to an optimization problem to solve with 374 726 \times \|J\| \) (resp.
On the other hand, in [9] is described an innovative clustering methodology which leads to 4 clusters where 3 of them are very different from the fourth one. Those 3 clusters are the more frequently asked FQDN found on DNS data. More precisely, from 30 seconds of DNS capture, which gave 167 793 queries the “adaptative k-means” (resp. “k-means”) algorithm gave 84 (resp. 143) FQDN in the 3 clusters. In this case, we can choose $I_s$ in order its cardinal $\|I_s\|$ to be 200.

The proposed platform sketched in the introduction can be now presented: to solve the bi-objective load balancing problem on a huge data, this data is split into two subsets and then a repartition on one of the subsets is performed by solving a MIP. This repartition is stored in a table. Once this table is full, the load balancer is able to well balance very efficiently the whole incoming DNS traffic, whether by a call to the table for the most frequent FQDN, or by a uniform repartition for the remaining FQDN.

## 4 Numerical experiments

Once the model has been defined, we solve it by writing it into a modeling language GNU MathProg and by using the open source GLPKv4 LP-MIP solver. Computations have been conducted on an AMD Athlon II X3 powered by a Linux 2.6 kernel. From now on, we fix the number of servers to be $\|J\| = 10$.

Fig. 4a shows that GLPK converges quite quickly when solving the MIP model on 200 FQDN. We can notice as well that the remaining balance achieved, once past the 200th second, is very small.

Fig. 4b and 4c show how the imbalance increases when the number of FQDN managed by the MIP varies. This imbalance is computed as the difference between the maximum and minimum number of queries sent to the server ($\max(S_{j_1}) - \min(S_{j_2})$ for $j_1$ and $j_2$ in $J$). Computations are respectively done for 200 and 3 600 seconds, which are good trade-off between the number of resolutions avoided and computing time to full the table.

In Fig. 4b GLPK has been stopped after 200 seconds. It appears a steep slope between 200 and 300 FQDN. This is the range where the MIP becomes too large to solve for GLPK. On this size of instance, it has not had time enough to branch efficiently. The solution found on instances greater than 300 is so bad than we had to represent the imbalance logarithmically on the figure.

The maximum number of FQDN GLPK can solve on this MIP in less than 200 seconds is 400. Beyond this point, no feasible solution is found. This leads us to an efficient scenario, called scenario MIP-200, which consist on balancing 200 FQDN by solving a MIP in 200 seconds.

The total imbalance is then the sum of the imbalance due to the MIP and the residual imbalance, that is $14 397 + 141 988 = 156 386$; and the number of signature checks is equal to the number of servers $\|I\| = 1 211 880$. In this case the difference between the maximum and the minimum number of FQDN sent to the servers is 4.

Fig. 4c shows the imbalance when the size of the problem varies. GLPK has been stopped after 3 600 seconds. It appears a peak at 200 FQDN. This unexpected behavior
Fig. 4: Performance Evaluation (with $\Delta = S_{\text{max}} - S_{\text{min}}$)

shows how difficult it is for GLPK to solve the problem when there is only the most frequent FQDN to manage; very large and very different numbers are hard to balance. However, when the number of FQDN grows, the number of small $N_i$ grows as well, and the problem becomes easier to solve as it is easier to fill differences with small numbers.

The maximum number of FQDN that GLPK is able to handle efficiently on this MIP in less than 3 600 seconds is 700. This leads us to the scenario MIP-700, which consist on balancing 700 FQDN by solving a MIP in 3600 seconds.

The total imbalance is then the sum of the imbalance due to the MIP and the residual imbalance, that is $8\,865 + 63\,588 = 72\,453$; and the number of signature checks is equal to the number of servers $\|I\| = 1\,211\,880$. In this case, the difference between the number of FQDN assigned to the server overloaded and the server underloaded is 19, which is very few compared to the size of the problem.

In order to get an evaluation of the gains provided by scenarios MIP-200 and MIP-700, we can compare them to two other scenarios, namely scenarios RR-FQDN and RR-Req.

Scenario RR-FQDN (round-robin on FQDN): each query of a given FQDN is sent to a unique server, and the set of FQDN is uniformly distributed among the servers, without considering the frequency associated to each FQDN. This uniform distribution of the FQDN on the servers can be achieved with a round-robin.

In scenario RR-Req (round-robin on queries): DNS queries are uniformly distributed among the servers without considering FQDN. This solution is the classic DNS solution as well as the way load balancing is performed in practical DNSSEC architectures.

Table 4d summaries the results. Solving a MIP on 200 to 3 600 seconds or performing a round-robin on FQDN can save half of the number of cryptographic computations.
(and corresponding power consumption) involved in the DNSSEC protocol than with the classic DNS solution RR-Req. Note as well that MIP-200 (resp. MIP-700) divide by 5 (resp. 11) the difference between the less loaded server and the most loaded one compares to RR-FQDN. This means that the platforms could be designed for a lower charge.

In other words, if we consider that the bottleneck of DNSSEC migration is due to CPU, and as shown by [6], that migration to DNSSEC requires between 2 and 4.25 times the number of servers, then MIP based distribution requires from 62% to 75% of servers.

5 Conclusion

DNS resolving platforms that are looking to optimize their cache needs to split the traffic between the different servers by considering the FQDN of the queries. By doing so we need to find a way so that the traffic as well as the resources required for the resolution is well balanced between the different servers.

This paper shows how to balance the DNS traffic in a platform composed of multiple servers. The traffic is considered as balanced if every server performs the same number of signature checks and furthermore, if servers also have to deal with the same amount of incoming traffic.

The paper solves this problem by splitting the data into two subsets and then performs a repartition on one of the subsets by solving a MIP. This repartition is stored in a table. Once this table is set, the load balancer redirects any DNS query to the defined server by looking into this table. If the FQDN is not in the table, then a uniform repartition is used like a hash function performed on the FQDN, or a round-robin.

Based on real DNS capture, we show that this solution can be deployed and improve the platform’s performance, that reduces by up to 75% the platform’s number of servers needed compared to traditional existing platforms.

In order to deploy an operational solution, it could be interesting to investigate more specifically how the frequency of DNS queries is varying over time. This would give an estimation of the latency we should wait before to re-run the MIP on a data set in order to recalibrate it.

Another perspective would be to improve the models and to study the possibilities of combinatorial algorithms taking into account the power law like repartition of DNS queries in order to optimize more efficiently and thus reduce computation resources needed to administrate DNSSEC platforms.

References


**Appendix**

As usual in Mixed Integer Programming, several models (sets of inequalities) describing one problem are not equivalent to each other. Some of them are far more efficient. By describing the four models in the appendix, we want to permit the interested reader to compare efficiencies of different MIP models solving problem \((P)\). The constraints and variables (1) to (7) on page 44 are common to the four models. For the second model, constraints and objective function (6) to (10) are replace by (11) to (13). It introduces two slack variables \(u_j, l_j \geq 0\), and aims at minimizing sum of those two slack variables:

\[
\sum_{j \in J} x_{i,j} \geq 1 \forall i \in I
\]

\[
S_j - u_j + l_j = \frac{1}{\|I\|} \sum_{i \in I} (N_i) \forall j \in J
\]

\[
\text{minimize} \sum_{j \in J} u_j + l_j
\]

For the third model, constraints and objective function (6) to (10) are replaced by (14) to (17). It introduces four slack variables \(S_{\text{min}}, S_{\text{max}}, T_{\text{min}}, T_{\text{max}}\):

\[
\sum_{j \in J} x_{i,j} = 1 \forall i \in I
\]

\[
S_{\text{min}} \leq S_j \leq S_{\text{max}} \forall j \in J
\]

\[
T_{\text{min}} \leq T_j \leq T_{\text{max}} \forall j \in J
\]
minimize \( S_{\text{max}} - S_{\text{min}} + k \ast (T_{\text{max}} - T_{\text{min}}) \) \hspace{1cm} (17)

For the fourth model, constraints and objective function (6) to (10) are replaced by (18) to (20). It introduces only one slack variable \( m \) and tries to maximize it:

\[
\sum_{j \in J} x_{i,j} \leq 1 \quad \forall i \in I \hspace{1cm} (18)
\]

\[
S_{j_1} + k \ast T_{j_2} \geq m \quad \forall j_1, j_2 \in J \hspace{1cm} (19)
\]

\[
\text{maximize} \ m \hspace{1cm} (20)
\]

Efficiency of the four models is shown in Fig. 5 on a 200 seconds example. The most efficient model, say model F2, is the one presented in the present article.

Fig. 5: Convergence of the 4 models.

Author Biographies

**Daniel Migault** received a MS degree from Institut Polytechnique de Grenoble (INPG). He is currently working on DNSSEC and IPsec in Mobile and Multihomed environment toward its PhD. His PhD is funded by the Security Department of Orange Labs in collaboration with Institut Telecom Sud Paris and Université Pierre et Marie Curie (UPMC).

**Stephane Senecal** received a MS in mathematics in 1998 from Joseph Fourier University, a MEng in electrical engineering in 1999 and a PhD in signal processing and telecommunications in 2002, both from Institut National Polytechnique, Grenoble, France. He worked at the Institute of Statistical Mathematics in Tokyo and at France Telecom Japan from 2003 to 2007. Since then, he has joined Orange Labs to develop network optimization methods and IP data processing solutions. His research interests span from machine learning and data mining to statistical simulation methods.

**Stanislas Francfort** received a MS in discrete mathematics from Université Pierre et Marie Curie (UPMC) in Paris, France. He applied his mathematical skills in cryptography, network security and finance until 2009. Since then, he has joined an Operations Research team at Orange Labs to enhance optimization methods applied to telecommunications companies’ topics.
DNSSEC: Maintenance and Operation
Maintenance, Mishap, and Mending in DNSSEC Deployment

Casey Deccio
Sandia National Laboratories*, Livermore, CA, USA
cddeci@sandia.gov

Abstract. The Domain Name System Security Extensions (DNSSEC) add an element of authentication to the DNS, which is a foundational component of today’s Internet. However, maintenance of a DNSSEC deployment is more complex than that of its insecure counterpart. In this paper we identify some of the specific misconfigurations impacting DNSSEC deployments, analyze their pervasiveness from an extended survey of production DNS, and assess maintenance and corrective behaviors. Over half the zones we analyzed have been affected by misconfiguration. We also observed a significant presence of repeat occurrence and average correction times of up to two weeks. We summarize our findings and suggest insights towards improving the quality of DNSSEC deployment.

1 Introduction

The Domain Name System (DNS) [14, 15] is a distributed database for looking up data based on domain name and query type and is one of the foundational components of the Internet. The most common use is mapping domain names to Internet addresses.

The DNS Security Extensions (DNSSEC) [5–7] were introduced to protect the integrity of DNS responses. DNSSEC allows DNS administrators to cryptographically sign and validate DNS data. The number of DNSSEC-signed zones has increased significantly in the last year, including the DNS root zone and a large number of top-level domains (TLDs) [2, 10, 3]. However, in order to achieve its security benefits, DNSSEC adds non-trivial complexity to the DNS. This increases the chances of a DNS outage if not properly deployed or maintained. The effects of misconfiguration have been felt at various levels in the DNS hierarchy, including TLDs. An understanding of DNSSEC components, their relationship, and the protocol itself are essential for proper deployment.

In this paper we analyze a portion of the DNSSEC deployment over roughly a year’s time frame (June 2010 – July 2011) to answer the following questions:

– What DNSSEC maintenance practices are being employed?
– What is the prevalence of misconfiguration among DNSSEC deployments?

* Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.
– How are operators addressing broken DNSSEC deployments?

We base our analysis on a survey of a sample of DNSSEC-signed zones polled over an extended period of time. We use the results of our analysis to suggest tools whose functionality would improve the quality of DNSSEC deployment.

The remainder of our paper is organized as follows. In Section 2 we provide a brief review of DNS and DNSSEC, and in Section 3 we outline challenges associated with DNSSEC. In Section 4 we describe our survey of DNSSEC deployment and analyze the results. Section 5 contains suggestions for improved DNSSEC deployment. We refer to previous work in Section 6 and conclude in Section 7.

2 DNS Background

In the DNS [14, 15] a resolver queries authoritative servers to receive answers. It learns authoritative servers for a DNS zone by starting at the root zone and following referrals downward in delegated DNS namespace until it receives an authoritative response. Queries include a name and type, and answers are comprised of resource records (RRs), which have a name, type, and record data. Resource records are grouped by name and type into RRsets.

DNSSEC [5–7] adds authentication to the DNS. RRsets are signed on a per-zone basis, and each signature is contained in an RRSIG RR. Authoritative servers return RRSIGs with any RRsets they cover. A zone’s public keys are published in the zone’s DNSKEY RRset. Having an RRSIG and the corresponding DNSKEY a validating resolver can verify the integrity of the RRset it covers.

DNSSEC scales by establishing a chain of trust upwards through the namespace hierarchy, and anchoring with the DNSKEY of a common ancestor zone, typically the root. The link between zones is accomplished by the introduction of DS (delegation signer) RRs in the parent zone. A DS includes the cryptographic digest of a DNSKEY in the child zone of the same name. When the DNSKEY corresponding to an authenticated DS or trust anchor is used to sign the the zone’s DNSKEY RRset, it becomes a secure entry point (SEP), and all DNSKEYs in the RRset are authenticated. A common setup is for a zone to sign only its DNSKEY RRset with the SEP key (a key signing key or KSK) and sign other zone data with a second key (a zone signing key or ZSK). The DNSSEC chain of trust is illustrated in Figure 1.

DNSSEC also provides authenticated denial of existence—an assurance that an RRset with queried name and type does not exist. This is accomplished using NSEC (next-secure) RRs, which are provided in a response to show a validator where the nonexistent RRset would appear (i.e., in a canonical ordering of the names in the zone) if it did exist. Hashed authenticated denial of existence using NSEC3 RRs is a newer protocol introduced to address challenges inherent in the use of NSEC [13].

3 DNSSEC Challenges

DNSSEC carries additional maintenance considerations, and negligence or misconfiguration may result in validation failures. We briefly discuss DNSSEC maintenance and misconfiguration in this section.
Since RRSIGs have a limited lifetime, the RRsets they cover must be periodically re-signed to replace RRSIGs that would otherwise go stale. While DNSKEYs technically do not expire, it is recommended that they periodically be replaced through a process called a key rollover [12]. Non-SEP DNSKEYs can be rolled without involving third-parties and are thus self-contained. However, when a SEP DNSKEY is rolled, the parent zone must be involved to handle the change in DS RRs. Likewise, a validator must be engaged when a configured trust anchor is rolled [22].

Misconfiguration of a DNSSEC deployment results in a break in the chain of trust and a failed validation for the RRsets involved. We enumerate six specific misconfigurations, to which we refer in the remainder of this paper.

### DS Mismatch
If DS RRs are present in a parent zone but none correspond to any self-signing DNSKEYs in the child zone, the chain of trust is broken, and RRsets in the child zone and below are deemed bogus. Such is the case with broken.com in Figure 1.

### DNSKEY Missing
If a DNSKEY referenced in an RRSIG or DS is necessary to complete a chain of trust, but not included in the DNSKEY RRset, then the chain is broken.

### NSEC Missing
The lack of NSEC RRs in a negative response (e.g., non-existent domain name) results in failed validation of that response. Validated negative responses are particularly critical to insecure delegations for proving that no DS RRs exist for a child zone and thus that there is no secure link from parent to child zone, as in the case of insecure.com in Figure 1.

### RRSIG Missing
If an authoritative server does not provide the RRSIGs necessary to complete a chain of trust for a given RRset, then the chain is broken.

### RRSIG Bogus
The signature in the record data of an RRSIG must validate against the RRset it covers, or it is invalid.
If an RRSIG is allowed to expire, or is published before its inception date, then it fails to validate.

4 DNSSEC Deployment Survey and Analysis

We describe in this section a survey of DNSSEC deployment and analyze results. Our survey consisted of periodic polling of production DNS zones signed with DNSSEC during a timespan of over one year—June 2010 to July 2011. We analyzed each signed zone several times daily, querying each authoritative server to elicit various DNSSEC-related responses. Our zones came from three sources: hostnames extracted from URLs indexed by the Open Directory Project (ODP) [17]; names queried to recursive resolvers at the SC08 conference [21]; and names submitted by third parties to the Web-based DNSViz analysis tool [20].

We identified production signed zones in our data set by considering only those signaling their public intent to be validated by resolvers—those with an authentication chain to the root zone trust anchor (after the July 2010 signing of the root [2]) or to the trust anchor at ISC’s DNSSEC Look-aside Validation (DLV) service [11]. DLV [24] was introduced to allow an arbitrary zone to be securely linked to a zone other than its hierarchical parent for trust anchor scalability prior to the root signing. We also excluded zones apparently not set up for production DNSSEC—those containing the names “test”, “bogus”, “bad”, and “fail”, and those that were subdomains of known DNSSEC test namespaces (e.g., dnsops.gov and dnsops.biz, of the Secure Naming Infrastructure Pilot [16]). The total number of production signed zones analyzed was 2,242, though the total number analyzed during any given polling period varied as new zones were added or as monitored zones entered or left production DNSSEC.

The breakdown of analyzed zones by TLD is shown in Figure 2. For signed zones under the .gov TLD, which made up the largest contingency of those analyzed, over 40% of zones experienced some type of misconfiguration. For nearly all TLDs shown, at least 30% experienced some type of misconfiguration.

4.1 DNSSEC Maintenance Observations

We first report on maintenance practices observed during our survey. We present in Figure 3 the average lifetime of RRSIGs covering the DNSKEY and SOA (start of authority) RRsets, made by the KSK and ZSK respectively, as a cumulative distribution function (CDF). This gives us an idea of the frequency at which administrators are required to re-sign their zone data. About 44% of zones had an average RRSIG lifetime of 30 days—the large vertical jump in the CDF—and 25% had an average RRSIG lifetime of 10 days or less. Nearly 5% of zones maintained RRSIGs with lifetimes over a year. The distributions for RRSIG lifetimes of each key type mirror each other for all but about 2% of cases, in which the lifetime of RRSIGs made by a KSK approach 400 days, contrasted with ZSK RRSIGs with 30-day lifetimes.

The average lifetime of RRSIGs (made by KSKs) for the subset of zones that allowed their RRSIGs to expire is also represented in Figure 3. This shows that zones maintaining RRSIGs with longer lifetimes and lower re-signing frequencies have a
Fig. 2: TLD breakdown of production signed zones analyzed, and zones that exhibited misconfiguration during our survey.

Fig. 3: CDF for average lifetimes of RRSIGs covering DNSKEY and SOA RRsets of production signed zones, made respectively by the zones’ KSKs and ZSKs.
larger showing of RRSIG expiration. Roughly 32% of the zones experiencing expirations have RRSIG lifetimes greater than 30 days, although only about 24% of all zones have RRSIG lifetimes of similar magnitude.

During our survey we observed key rollovers to analyze the lifetimes of ZSKs and KSKs. Only zones for which we observed two or more rollovers could we accurately determine the average lifetime of keys of either role. Although, for zones that experienced a single rollover we estimated lifetime using the maximum of the time monitoring the zone prior to the rollover and the time from the rollover to the completion of our survey.

We can’t judge the significance of the fact that 90% of zones performed fewer than two KSK rollovers during our survey given that some best practice documents recommend rollovers every 1 – 2 years [8]. However, we might sense some hesitancy to implement the potentially troublesome KSK rollover in the 72% of zones for which no rollover was detected. While ZSK rollovers are less complex and require no coordination with the parent zone, we find it interesting that 11% performed only a single ZSK rollover, and 37% performed no ZSK rollovers during our survey.

The CDF of the average ZSK and KSK lifetimes for zones having performed one or more rollovers is shown in Figure 4. Half of the sample zones that performed two KSK rollovers did so on average over 75 days apart, and 55% of zones that performed two ZSK rollovers did so in 30 days or less, on average.

![CDF for average lifetimes of ZSKs and KSKs for zones having performed one or more rollovers during our survey](image)

**Figure 4:** CDF for average lifetimes of ZSKs and KSKs for zones having performed one or more rollovers during our survey.

Figure 4 also plots the average KSK lifetime for zones that experienced bad KSK rollovers (i.e., resulting in DS mismatch) during our survey. The distribution of KSK lifetimes for these zones is fairly even, indicating there is no significant correlation between KSK rollover frequency and DS mismatch, based on our observation. Half of
the zones that experienced bad KSK rollovers averaged a KSK lifetime of more than 200 days, while half of zones rolled their KSKs within 200 days.

### 4.2 DNSSEC Misconfiguration Pervasiveness

Throughout our survey we identified the DNSSEC misconfigurations described in Section 3 to determine their pervasiveness in production. Our analysis was based on authentication of the SOA RRset for each zone. We identified 6,484 unique instances of misconfiguration in 1,240 zones over the duration of our survey. The breakdown of misconfigurations observed is shown in Figure 5 and summarized in Table 1.

![Fig. 5: Instances of DNSSEC misconfiguration observed, grouped by type and whether zones were completely affected, partially affected, or corrected incrementally](image)

<table>
<thead>
<tr>
<th>Misconfiguration Type</th>
<th>Total instances</th>
<th>TLDs Affected</th>
<th>Complete or Incremental</th>
<th>Avg duration (days)</th>
<th>Repetition rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS Mismatch</td>
<td>392</td>
<td>0</td>
<td>392 (100%)</td>
<td>12</td>
<td>22%</td>
</tr>
<tr>
<td>DNSKEY Missing</td>
<td>1,250</td>
<td>1</td>
<td>98 (8%)</td>
<td>6.8</td>
<td>40%</td>
</tr>
<tr>
<td>NSEC Missing</td>
<td>449</td>
<td>3</td>
<td>21 (5%)</td>
<td>14</td>
<td>39%</td>
</tr>
<tr>
<td>RRSIG Missing</td>
<td>2,418</td>
<td>6</td>
<td>0 (0%)</td>
<td>6.3</td>
<td>52%</td>
</tr>
<tr>
<td>RRSIG Bogus</td>
<td>284</td>
<td>0</td>
<td>210 (74%)</td>
<td>5.7</td>
<td>42%</td>
</tr>
<tr>
<td>RRSIG Dates</td>
<td>1,691</td>
<td>4</td>
<td>1,280 (76%)</td>
<td>7.7</td>
<td>45%</td>
</tr>
</tbody>
</table>
For each misconfiguration type, the instances are subdivided based on whether they affected all authoritative servers (complete), a subset of servers only (partial), or were resolved iteratively (incremental). Incremental means that a misconfiguration affecting all authoritative servers was resolved on a subset of servers before it was fixed universally.

The largest contributor to certain DNSSEC validation failure was RRSIGs with invalid dates. This typically refers to expired RRSIGs, although we identified several misconfigured zones that were repeatedly signed with future inception dates, never quite allowing their RRSIGs to reach the validity period. In the partial and incremental occurrences of RRSIG expiration, it is likely that the zone was re-signed (possibly before RRSIG expiration, in the case of partial misconfiguration), yet some of the authoritative servers did not transfer the most recent version of the fresh zone. The next largest contingency of complete misconfiguration is attributed to DS mismatches, largely attributed to bad KSK rollovers.

Missing RRSIGs are the leading contributor to possible failure partial misconfiguration. We attribute this to zones that were signed but for which one or more authoritative servers do not support DNSSEC and therefore do not return RRSIGs appropriately. We observed several zones with servers that “flapped” between returning RRSIGs and not.

4.3 Misconfiguration Resolution

We analyze several aspects of the misconfiguration instances we observed to understand the resolution path taken by administrators to correct problems. Two paths can be taken to appease resolvers validating misconfigured zones: correct the configuration; or remove the DS RRs for the zone, creating an insecure delegation. Misconfiguration instances are categorized by corrective action in Figure 6. Unresolved events persisted beyond the lifetime of our survey, so their resolution is unknown. The misconfiguration that most frequently saw anchor removal relative to proper correction was DS mismatches, at 20%.

We next examine the duration of each misconfiguration instance to understand the responsiveness of DNS administrators in identifying and taking corrective action for a DNSSEC-related problem, whether by removing its anchor or properly correcting the problem. Figure 7 plots the lifetime of each event as a cumulative distribution function (CDF), and the averages are shown in Table 1. Note that we exclude from our event duration results from events which were unresolved at the conclusion of our survey.

On average, events involving RRSIGs with bogus signatures were corrected within the least amount of time, followed by missing RRSIGs and RRSIGs with invalid dates. This is not surprising since the remedy for each of these is simply re-signing the of the zone which replaces the affected RRSIGs. At least half of each of these misconfigurations were corrected within two days.

Events involving missing NSEC RRs averaged the largest correction time at two weeks, likely because the misconfiguration is more subtle in most cases. DS mismatches took 12 days to correct on average, and one in four took over a week to correct.
Fig. 6: Breakdown of corrective action taken for resolution of DNSSEC misconfiguration instances

Fig. 7: CDF of the duration of DNSSEC misconfiguration instances of each type
4.4 Repeated Misconfiguration

Figure 8 shows the number of events per zone, classified by type, and Table 1 includes the repetition rate for different misconfigurations.

![Cumulative Distribution](image.jpg)

**Fig. 8: Number of events per zone, by type.**

DS mismatches were repeated by one in five affected zones, a significant proportion, but the smallest in comparison to other misconfigurations. The relative infrequency could be because KSK rollovers generally happen less frequently than RRSIG renewals, so there is less opportunity for error. Missing RRSIGs and RRSIGs with bad dates account for the highest occurrence of repeat offense, averaging 46% and 51%. Almost 30% of zones affected by missing RRSIGs experienced this misconfiguration more than three times.

5 Suggestions for Improved Deployment

We attribute much of the misconfiguration experienced in our survey to the novelty of DNSSEC deployment. DNS administrators are learning the protocol and how to deploy, maintain, and troubleshoot problems, and DNSSEC functionality is still fairly new to many name server implementations.

We advocate development and use of analysis tools to help administrators understand their DNSSEC deployments and verify that they are properly configured. Such analysis should not only be performed at the time of deployment, but should be a matter of periodic evaluation because of the dynamics of DNSSEC. The fact that it takes nearly a week on average to fix DNSSEC-related misconfigurations suggests that administrators are either slow to detect the problems or slow to implement solutions. Also,
over 20% of all misconfiguration types have affected zones more than once. Regular monitoring can help decrease the detection time and help administrators prevent future occurrences of misconfiguration.

Several tools have been developed to help administrators analyze their DNSSEC deployments. DNSViz [20] provides comprehensive analysis of domain names via a Web interface by analyzing the chain of trust from RRset to anchor, and it checks authoritative servers for consistency. Results are presented to users in a graphical format. The DNSSEC Debugger [23] also traces the chain of trust and provides a detailed textual output of the results. Other online tools examine only the name itself for configuration correctness [1, 4]. DNS administrators can leverage tools such as these to better manage their DNS deployments.

6 Previous Work

Previous studies have analyzed DNSSEC deployment from various perspectives. SecSpider [18, 3] and IKS Jena [10] discover signed zones and maintain an ongoing status of DNSSEC deployment in terms of pervasiveness and configuration issues. SecSpider monitors from different world-wide locations, assessing consistency of results from different vantage points, and its data was the basis for an assessment of DNSSEC availability, verifiability, and validity [19]. Another DNSSEC availability study has focused on misconfiguration and offered protocol extensions to mitigate their effects [9]. Our work does not seek to examine the breadth of DNSSEC deployment, but focuses on analyzing related patterns of maintenance, misconfiguration, and corrective action in DNSSEC deployment.

7 Conclusion

The DNS is an essential component of the Internet. DNSSEC was designed to protect the integrity of DNS responses, but its deployment has been wrought with challenges. In this paper we have presented an analysis of DNSSEC deployment based on a survey spanning over one year. We observed maintenance behaviors and identified and analyzed configuration errors affecting DNSSEC validation. We analyzed patterns of corrective action and the time taken to correct misconfiguration. We found that misconfiguration affected DNSSEC deployment at even the highest levels and took up to two weeks on average to correct, depending its nature. We recommended tools to aid administrators in properly deploying and maintaining their DNSSEC deployments.

If DNSSEC deployment is to be successful, then the challenges posed by its administrative and protocol complexity must be met. Growing experience should be coupled with increased protocol awareness, facilitated by functionality provided by tools and name server implementations to achieve a reliable and trustworthy DNS.

References

1. DNSCheck by .SE. http://dnscheck.iis.se/.

Author Biography

Dr. Casey Deccio is a Senior Member of Technical Staff at Sandia National Laboratories in Livermore, CA. He joined Sandia in 2004 after receiving his BS and MS degrees in Computer Science from Brigham Young University, and he earned his doctoral degree from the University of California, Davis, in 2010. Dr. Deccio’s research interests lie primarily in modeling and availability analysis of DNS and DNSSEC, and he is the author of DNSViz, the Web-based tool for DNS analysis.
DNSSEC Automation and Monitoring

Russ Mundy, Wes Hardaker, Suresh Krishnaswamy, Wayne Morrison, Robert Story

SPARTA (dba Cobham Analytic Solutions)
http://www.sparta.com

Abstract. This paper discusses the need for automating various DNSSEC operations and describes some of the tools available in the DNSSEC-Tools suite that ease DNSSEC management. ¹

1 Introduction

The Domain Name System (DNS) provides the binding between a name and some information about that name, such as an IP address. It is one of the core components of the Internet infrastructure. DNSSEC adds cryptographic protection to DNS responses, making it possible to detect when a response has been modified.

Managing the correctness and availability of DNS has always been essential for normal use of the Internet. IP addresses are difficult to memorize and can change over time. In many systems, automation is often a critical part of the DNS management system. When adding DNSSEC, automation becomes more important, even for smaller systems.

This paper outlines a number of considerations related to automating DNSSEC operations. In Section 2 we discuss the need for DNSSEC automation and monitoring. In Section 3 we describe some of the tools that we developed in the DNSSEC-Tools [4] suite in order to support the various DNSSEC automation needs, while in Section 4 we outline some of the monitoring capabilities that we provide in the DNSSEC-Tools suite. We provide conclusions and future directions in Section 5.

1.1 Simple Overview of Pre-DNSSEC Maintenance

For the simplest scenario, DNS is essentially zero-maintenance. Once a zone is configured and running, it can continue to run indefinitely. (Barring such changes as adding, deleting, or changing a node or address.)

Automation for this scenario should similarly be fairly simple. When any data in the zone changes, the management system should check the zone for validity (e.g., named-checkzone), and then load the new configuration into the active name server (e.g., rndc-reload).

Even in the simplest scenario, monitoring the zone(s) and server(s) periodically is important. At the service level, the operator would want to monitor any changes to response time for the servers from various network vantage points, the number of

¹
unanswered queries, and issues that might prevent a resolver from being able to contact the authoritative name server. At the server level, the server load, bandwidth consumed, and the different name server error conditions are generally of interest. Finally from a zone point of view, the integrity and consistency of zone data are important. The zone data must match delegation information in the parent zone and across different authoritative instances of the zone.

1.2 What Changed with DNSSEC

DNSSEC is a larger beast than DNS alone and, as you might expect, requires a little more care in its handling and feeding.

Every DNSSEC-enabled zone has a set of cryptographic keys. These keys are used to cryptographically sign each record in the zone, and the resulting signatures are included in the zone.Cryptographic best practices require that periodically new keys be created and the zone be re-signed with the new keys to generate new signature records. This process, called Key Rollover, is described in RFC 4641 [2].

Thus, even if no zone data have been changed, new signatures must be generated periodically, before existing signatures expire. While key rollovers aren’t terribly complicated, timing is critical. An error while managing the keys (e.g., publishing new keys too late or removing old keys too early) will cause some or all of the domain to become BOGUS and thus unreachable by hosts or applications using DNSSEC-validating resolvers.

While it is essential to monitor all aspects of DNS mentioned in the previous section, with the introduction of DNSSEC certain aspects require special attention. First, the expiration time stamp on signatures introduces a reliance on absolute time and an accurate system clock. Recursive name servers still use a given resource record’s Time To Live (TTL) field to keep its cache up to date; however, with DNSSEC the TTL values also affect the timing parameters associated with various key rollover operations and therefore require special consideration.

Next, the consistency of the secure delegation information published at the parent zone is as important as the consistency of the delegated zone itself. However, since DNSSEC deployment does not always occur in a top-down fashion, and because the root has only recently been signed, sites which were early adopters of DNSSEC may have configured these Secure Entry Points (SEPs, aka trust anchors) in various supporting infrastructures (e.g., BIND, DNSSEC Lookaside Validation or DLV). Each of these trust anchors must be monitored for rollover, so that new keys can be updated in the configuration for each system.

DNS response sizes are another factor that deserve attention. Typically, DNSSEC responses are much larger than in unsecured DNS for the simple reason that they now carry the additional security meta-data that is necessary for validating answers. The larger response size may not conform to the preconceived notions of some operators and device vendors who believe that DNS response will never exceed 512 bytes and

---

\[2\] The TTL field provides the relative measure of time from the moment the record was fetched from an authoritative name server.
will always be UDP-based. While this assumption would have resulted in operational problems even without DNSSEC, DNSSEC reveals such problems in their full glory.

Finally, cryptographic key strength and good cryptographic key-handling practices are essential to maintain the assurances provided by DNSSEC. Certain cryptographic algorithms may lose favor over time, certain parameters used while generating keys may result in weaker keys, and cryptographic libraries may have to be updated. However, notwithstanding the need to protect the confidentiality of private key material, it is useful to note that the main purpose of DNSSEC is to protect the DNS data. The cryptographic keys are only as important as the data they protect. In other words, the same degree of security that ensures the DNS zone content is correct and protected should be provided for the private key that is used to sign the zone contents.

2 Automation Overview

It became very apparent during initial DNSSEC fielding operations that automation was going to be necessary in order to offset some of the additional DNSSEC operational complexities.

2.1 DNSSEC Operations that Can Be Automated

This section discusses some of the different DNSSEC operations that lend themselves to being automated. With automated solutions, it is very important for operators to understand what operations are being automated on their behalf in case they ever have to recover their system from an error.

**Zone Re-signing and Key Management** While signing a single zone requires nothing special other than to run the appropriate zone-signing command with the appropriate arguments, this process can easily get unwieldy as the number of managed zones grows. Each zone may have different key size and algorithm requirements, different key-rollover intervals, and different zone-signing intervals. With different key-rollover intervals, remembering the specific arguments to invoke for different zones at different times, and the specific keys to use during each zone-signing phase, easily becomes onerous. An automated zone-signing solution is therefore crucial for environments that involve multiple zones.

**Key Generation** Key generation relies on a good source of entropy in order to generate good keys. If keys need to be generated often, such as the case where a single operator serving a large number of zones suddenly needs to rollover multiple keys, the system may be unable to keep up with these demands – unless it relies on cryptographic hardware that can provide it with this capability or it has a ready pool of keys available at its disposal. The process of generating keys ahead of time is one that may be automated.
Algorithm Rollover Algorithm agility in protocols refers to a protocol’s ability to easily embrace and incorporate new cryptographic algorithms into the running system, without any updates to the protocol itself, thus being agile in its response to new cryptographic advances. DNSSEC was designed with algorithm agility in mind; however, the need to balance the agility in the adoption of new algorithms versus being resilient in the face of algorithm “downgrade” attacks means that the algorithm rollover process must be carefully coordinated through different stages. The constraints imposed by various TTLs must also be kept in mind. The complexity of this operation alone would warrant an automated solution.

Key Rollover As alluded to multiple times already, key rollover is an operation that involves multiple steps and that must be performed by operators at specific intervals. Even in the best circumstances, such as in a scheduled key rollover, key rollover requires a minimum of three stages with each stage having a zone-dependent wait time before it can commence.

There are also certain special cases for key rollover: in certain instances the key-rollover operations may be performed on a machine that is normally “offline” in order to keep the private keys secure. Key-rollover operations may also occur across different entities, such as the case where a secure domain is being transferred from one operator to another. With multiple zones, this entire set of operations can be very cumbersome to manage, but automated tools make this process very straightforward.

DS Transfer to Parent The Delegation Signer (DS) record in the parent zone provides the cryptographic binding from the parent zone to the child zones in the DNS delegation hierarchy. Initial transfer of the DS record must rely on established out-of-band channels between the parent and child, but once this is set up further updates can easily leverage the existing secure delegation structure. Since updates occur within the DNS protocol itself and do not necessarily follow contractual relationships between the registry, registrar, and registrant, it may not be viable in many instances. However, in cases where it can be used this in-band approach offers the potential to significantly reduce inconsistencies between parent and child zone data.

Trust Anchor Rollover Software which performs DNSSEC validation must be configured with one or more keys which are considered trusted (trust anchors). As mentioned previously, accepted cryptographic practices normally include periodic replacement of the keys; thus, trust anchors will need to be replaced periodically. In the case of an emergency rollover (e.g., a compromised key), the key may change much earlier than expected. Therefore any key which is configured as a trust anchor must be monitored for rollover so that configuration data can be updated. The requirement for an automated trust anchor solution was founded in an Internet Engineering Task Force (IETF) protocol working group and a standardized solution to the problem has already been defined in RFC 5011 [3].
2.2 Evolution of DNSSEC Automation

The first step towards DNSSEC automation was to identify the low-hanging fruit associated with reducing overhead of DNSSEC management. Wrapper scripts were created to remember the specific arguments that had to be passed to the key-generation and zone-signing commands, *cron* scripts were used to automate the periodic operations such as zone re-signing, and cheat-sheets [1] were used to mechanize some of the state-based operations involved in key-rollover. While these were perfectly reasonable tools to accomplish the given tasks at hand, these operations were still being performed by operators who were heavily involved in the initial DNSSEC adoption process. As the level of DNSSEC adoption grew, so also did the need for better tools that would work for zone administrators with all levels of DNSSEC experience.

It is almost certain that no two randomly selected environments will be alike. Similarly every operator will have his or her own preference of one tool over another to perform a given task, a preference of what tasks to automate versus ones to perform by hand, and what events to monitor at what granularity. In addition, the zone-generation process may vary across the board – some environments may use a static zone while others may use a dynamically updated zone; some zones may be embodied in a single zone file, while others may be generated from databases and scripts that have been carefully cared for through generations of sysadmins who would be fearful of making any change to this delicately balanced system. While building automation tools, it is very important to keep any automation components as modular and flexible as possible so that they can accommodate different operator needs and enable both stand-alone operation and integration into existing zone-management systems.

The current lay of the land for automated DNSSEC solutions provides a number of choices for the DNS operator. These include the following:

**Using Existing Software** Often it may be impractical to modify the existing DNS infrastructure beyond simple upgrades of the name server software. Over the years, the different flavors of DNS server software have seen vast improvements in their ability to automate various DNS functions, and many now handle the basic DNSSEC functions in a very user-friendly manner out of the box. However, many of the key rollover operations are still left to the operator to perform using hand-rolled scripts, which presents the opportunity for errors.

**Using Wrapper Scripts** Wrapper scripts around the various name server programs provide a simple way to augment the default name server capability with support for operations such as key management, zone re-signing, and key rollover. The benefit of using existing wrapper scripts over using hand-rolled ones is that most of the heavy lifting has already been done, most likely by folks who live and breathe DNSSEC. Further, with open source solutions, the code is available for review and customizations, which enables the operators to tailor the tools to their environments without having to start from scratch.

**DNSSEC Appliances** These boxes are often plug-and-play, requiring little to no knowledge of DNSSEC. One common option sits between an existing hidden master and the
public slaves. The appliance receives updates of the unsigned DNS zone from the master, signs the zone, and pushes the signed zone out to the public slaves. The appliance could also replace the hidden master. In both cases, the appliance handles signing the zone and key rollover.

Using a Registrar that Supports Signing and Maintenance More and more registrars are offering solutions for managing DNSSEC zone data. These services are an excellent option for organizations without the time or expertise for managing their own DNS infrastructure. The DNS data can be managed via a web interface, and most also offer an API that can be integrated with in-house management systems.

2.3 Why Automation Does Not Replace the Need for Monitoring
Automation systems are very useful in their ability to prevent simple (but costly!) errors. Human error can never be eliminated and one can be reasonably sure to find real-world examples of various DNS-related outages resulting from simple operator errors [5]. However, inasmuch as tools and automation can reduce the occurrences of errors, they do not replace the role of the operator itself.

While automation can help reduce the chances that a step will be missed or performed too late, there’s always a chance that some unexpected condition could result in undesired behavior. In addition to good ol’ human error, there is still the possibility for software, network, and hardware errors. As DNS environments grow larger and more complex, with multiple slaves, hidden masters, and remote key storage, more potential points of failure are introduced.

For this reason, it is important, if not critical, to have monitoring in place so that any service-affecting issues can be detected as soon as possible. Moreover, automation tools can also fail; thus, it is important that any DNSSEC automation system also provide the necessary hooks that allow the operator to integrate error reporting into their normal workflow.

2.4 Real-World Examples of DNSSEC Management Errors
Some real-world examples that highlight the importance of complementing DNSSEC automation with good monitoring practices are given below.

– In October, 2009, an incorrect software update resulted in all name resolution for Swedish (.se) domains to fail for over an hour [8]. When the error was detected, the zone was republished, but without DNSSEC signature. While this resolved issues that non-validating resolvers were having, it meant that validating resolvers continued to have issues until a new signed zone was published over an hour later. Comprehensive testing of validation status for the zone before publishing could have mitigated this issue.

– In September, 2010, Nominet had a hardware failure which caused a system to failover to a backup system. Unfortunately, there was an inconsistency in the key database and configuration file on the backup system [7]. Automated synchronization and monitoring the status of the synchronization process could have mitigated this issue.
In November, 2010, a network issue within the French signing system caused two systems to become out of sync. As a result, dynamic updates to the zone began failing [11]. Comprehensive error checking of the synchronization process could have mitigated this issue.

In February, 2011, the automated signer system used by RIPE did not include the DS record for the (then) current KSK key [9]. It was thought that a high system load caused the error. However, in April, 2011, the system failed again in the same way, while the system was under a normal load [10]. Comprehensive testing of validation status for the zone before publishing could have mitigated this issue.

In February, 2011, the entire French zone (.fr) was unavailable to validating resolvers for almost four hours ([12] and [11]). Although the .fr zone operators had a monitoring system in place, the monitoring system was not monitoring all record types. As (bad) luck would have it, the error affected a record type that was not being monitored and the error went undetected for over an hour before users started reporting problems. Comprehensive testing of validation status for the zone before publishing could have mitigated this issue.

3 Automation with DNSSEC-Tools

DNSSEC-Tools [4] is an open source software suite that includes various utilities for managing a DNSSEC-enabled zone. Tools are available for signing zones, validating signed zones, automatically managing key rollovers, automatically managing trust anchors, checking zones for warning and error conditions, watching and summarizing log files, and for debugging deployments and operational environments. The tool suite also includes a validator library, a command-line debugging tool and software patches to enable DNSSEC in a number of common applications.

This section describes some of the tools available in DNSSEC-Tools that enable automation of DNSSEC operations.

3.1 zonesigner

The DNSSEC-Tools zonesigner utility generates keys and signs a zone with those keys. It uses BIND's basic building-block tools for key generation and zone signing, and provides wrappers around these tools to make them easier to use.

The goal for zonesigner was to minimize the number of steps that would be required in order to take a zone's state from unsigned to signed, while also simplifying future re-sign and key rollover operations. zonesigner accomplishes this by maintaining state information related to zone keys and zones in a keyrec file and supplying the correct set of arguments to the BIND zone-signing programs as necessary. zonesigner uses the best-practice recommended defaults for arguments when they are not specified, thus invoking zonesigner for a zone with no additional arguments will typically result in zonesigner doing the "right" thing.

With the exception of auto-incrementing the serial number in the zonefile, zone-signer keeps the zonefile largely untouched. Thus, the operators can continue to operate on the older zonefiles as usual, while all DNSSEC-related processing happens behind
the scenes. This is useful, since when editing a zonefile (or even using programs that act as front-ends to zone-editing functions) operators do not have to be concerned with the large quantities of DNSSEC metadata that get added as part of a signing operation.

3.2 rollerd

rollerd manages the key rollover process, keeping track of a zone’s rollover phase and the time until the next rollover action. When required, rollerd runs zonesigner to generate new keys for a zone and then signs the zone with those keys.

rollerd stores various state data pertaining to key rollover in a rollrec file. An operator operating an unsigned zone would create the rollerd configuration file, adding each zone to be managed by rollerd, and then sign the zones.

While it would be highly desirable to reduce all key rollover operations to a single step, currently the process of KSK rollover isn’t completely automatable, since the administrator of the parent zone has to be involved in publishing the DS record for the child zone. However, rollerd will send notify the administrator of the child zone a newly generated DS record must be transferred to the parent.

4 Monitoring with DNSSEC-Tools

DNSSEC-Tools include several tools for monitoring zones, trust anchors, and the “user experience” of DNS as a service. Tools include stand-alone scripts and plugins for well-known monitoring software. These are further described below.

4.1 Server-End Monitoring Tools

donuts donuts is a lint-like checking tool for analyzing DNS zone files. It reports any errors it finds, either as text output if run as a command line application or in a GUI window if an error browser interface is desired.

Since different operators may have different views on interesting events to monitor for their zones, donuts was designed to be extremely flexible in the types of errors it can detect. Rules are defined in donuts rule configuration files using a given syntax. For example, a local policy could dictate that all zones must have a TTL of at least 5 minutes. donuts can be easily configured to support such checks.

blinkenlights, bubbles, lights These are stand-alone GUI tools that display information on the current rollover state of the zones rollerd is managing. The tools provide various level of detail. Given a rollrec file, these tools can very quickly give the operator a visual snapshot of the rollover status for any associated zones and flag any problems. Additionally, some aspects of rollerd execution and zone rollover control are available through blinkenlights.
**Nagios Plugin**  Large IT operations typically include a centralized system for monitoring the health of various network services on an ongoing basis. *Nagios* [6] is a very popular open source solution for IT infrastructure monitoring. The DNSSEC-Tools distribution contains a *Nagios* plugin to gather rollover status, a program for generating rollover-specific *Nagios* configuration objects, a software patch for generating prettier rollover data display in the *Nagios* GUI, a plugin for checking zonefiles for errors, and several sample files. The result is a set of *Nagios* features that enable the operator to determine any zone errors and the current rollover status for various zones.

### 4.2 Validating-End Monitoring Tools

**logwatch Extensions**  *logwatch* is a system log analyzer and watcher. It is comprised of a number of modules that look for specific patterns in system log files and then summarizes that information to the system administrator, typically via email. *logwatch* is not itself part of DNSSEC-Tools distribution, but the DNSSEC-Tools project has contributed of *logwatch* modules that parse DNSSEC-related log messages generated by the ISC BIND name server. These modules enable the system administrator to obtain an overview of DNSSEC-related events at the recursive name server and take any action that may be necessary. The DNSSEC-Tools extensions for *logwatch* (*dnssec* and *resolver*) are included in the mainline distribution of *logwatch*.

**trustman**  This tool is an implementation of the RFC 5011 [3] approach for automated trust anchor management. As a provisioning-end monitoring tool, *trustman* enables a zone owner to check if the KSK rollover operation was done in compliance with RFC 5011.

**Visualization Tools**  The DNSSEC-Tools Project has recently developed some tools to provide operators and users with visualizations of DNSSEC activities. Of particular note is the *dnssec-nodes* tool which provides near real-time visualization of nodes being resolved and validated. A video of this tool can be seen at: http://www.dnssec-tools.org/videos/DNSSEC-Tools-QT.mp4.

### 4.3 User Experience Monitor (UEM)

The primary users of DNS information are “Internet users”, not the operators of authoritative DNS servers. Since DNS server operators normally cannot “see” what “Internet users” of their DNS information “see” in terms of service quality, DNS server operators monitor and measure their service quality by instrumenting their name servers. Although this does provide valuable information about server performance, it does not monitor the quality of DNS service a user receives. The UEM system provides a DNS operator with the ability to measure the quality of DNS service from the perspective of the users of their service.
Monitoring the DNS User Experience  To provide this “Internet user” perspective, the UEM system monitors response time of queries to name servers of “zones of interest”, which can be any zone in the DNS. To obtain these response times, UEM sensors collect their data from “topologically interesting spots” in the Internet selected by the UEM operator. The response times for the individual servers for the zone are logged to provide a view of the topological location of that sensor. The response time logs are then transferred to the UEM manager for display. The manager adds the response data to a set of databases, and the response results may be viewed through a Nagios interface. The Nagios interface can be configured to provide individual or aggregated views of the response time data for name servers and zones. Through the use of multiple sensors, the service quality for a zone can be seen from a broad user perspective. Nagios provides monitoring of current results and graphs historical results.

Our particular interest has been the DNSSEC impact on root name servers, and our operation has a set of UEM sensors that sends queries to all root name servers the operators have configured for monitoring.

Anycast Response Times  Anycast addressing is used by a number of large and popular DNS zones (such as the root zone) to improve query resolution time by spreading query load over a larger number of hosts than what is possible using the DNS referral alone. It also allows geographical dispersion of the servers, which will benefit from continued connectivity in times of disaster or network outages. By the nature of anycast
addressing, an any-casted server with N instances will appear to just have a single IP address.

The UEM sensors can send the appropriate query to determine which any-cast instance of a name server has responded to the query. The UEM manager will record this data and track the response times for each name server’s “any-cast” instances. (This assumes that name servers respond to these queries with a distinct label for each anycast instance.) As with any user-centric DNS monitoring, the UEM operator can collect information about the quality of DNS service being provided to “Internet users” as well as having a data set that can be used to identify changes in service quality over time that can, in most cases, be traced back to the basic cause such as individual name server problems.

5 Conclusions and Future Work

In this paper we outlined some of the operational changes imposed by DNSSEC to vanilla management of DNS and the implications of automation and monitoring of DNS operations. We presented some of the tools available in the DNSSEC-Tools suite for facilitating automated zone signing and key rollover and for monitoring various DNSSEC events at the server-end and at the validating end.

The DNSSEC-Tools effort was undertaken to fill a void in the available solution space for automated DNSSEC solutions. Since the tools were designed with modularity in mind, they easily lend themselves to being integrated with other frameworks where DNSSEC needs to be retrofitted. As future work, we plan on integrating some of our tools with existing DNS hosting control panel software and other server management consoles, which we see as an important step in order to making DNSSEC use more widespread. Providing support for algorithm rollover, integration with Hardware Security Modules (HSMs), supporting environments with hundreds or thousands of zones being managed, and providing a unified logging framework for the various tool components are other improvements planned for the tool suite.

The Nagios plugins distributed with DNSSEC-Tools currently require that rollerd perform its rollover operations on the same host as the Nagios host. As part of future work we also plan on developing plugins that can determine various DNSSEC status conditions by polling the zone remotely and feeding the results through some of our troubleshooting tools.

DNSSEC is real and it is here to stay. In order to maximize the benefits of this technology it is important that the quality of DNSSEC operations mature to a level that operational errors are very rare and are resolved expeditiously when they do arise. Tools that enable the operator to hide the complexities of DNSSEC are necessary in this respect and a number of tools, including many from the DNSSEC-Tools suite, provide the building blocks towards fulfilling this goal.

Acknowledgments

DNSSEC Tools development and this paper are supported by the Dept. of Homeland Security, Science & Technology, under Contract No. FA8750-11-C-0088.
References


4. DNSSEC-Tools: DNSSEC Software Libraries and Tools,
   http://www.dnssec-tools.org/


7. Simon. DNSSEC Incident report. [Internet] Blog posting.


12. Renaut, Philippe. Study and action plan following the incident with validating resolvers on 12 Feb 2011 [Internet]. 2011 Feb 18.

Richard Lamb
ICANN, Marina del Rey CA 90292, USA,
richard.lamb@icann.org

Abstract. With the healthy deployment of DNSSEC well on its way and serious efforts to make use of the resulting global PKI to expand the benefits of cryptographic security to the masses, DNSSEC has the potential of becoming a critical link for a wide range of industry applications. However, many of the current practices employed and mindset in the current chain from registrant to root are inadequate and need to improve if the Internet is to reap the full benefits from DNSSEC. This paper will seek to identify the weak links in this chain and to outline approaches various entities can take to strengthen them.

Disclaimer
The views expressed in this paper are those of the authors and do not represent those of their employer.

1 Introduction

At the time of this writing, 72 out of 310 TLDs have deployed DNSSEC. Due to the popularity of some of those TLDs, over 80 percent of the domain names on the Internet have the ability to deploy DNSSEC. However, since it is still a relatively new technology, less than 1 percent of domain names have DNSSEC deployed on them. This is both a drawback and an opportunity.

Our increasing reliance on the Internet, and its protocols, for critical applications encompassing everything from financial and health care systems to new applications, such as smart grid in the electric utility space, make it imperative that the Internet’s infrastructure needs to be improved. DNSSEC is a big step in that direction.

Unfortunately, many of the current practices employed in the chain of trust (Registrant to end user) are inadequate. Reaping the full benefits of DNSSEC will require enhancement. This paper will identify the weak links in this chain and outline approaches entities can take to strengthen them.

Given the early stages of overall deployment, the goal of this paper is to encourage a race to the top with the establishment of best practices instead of one to the bottom (untrustworthy deployment leading to little value and minimal adoption) by raising awareness and learning from and building on existing sources of trust on the Internet.
2 DNSSEC Lives! - Nirvana?

With the root signed [1] and a healthy implementation rate on TLDs, we believe full DNSSEC deployment, at least at the top level, is an eventuality and the benefits from a trusted DNS will be realized.

Future innovative efforts to develop a range of global security and authentication solutions based on a DNSSEC secured DNS will increase security and improve the overall Internet experience [2, 3].

Although the idea of using the DNS to provide more than just IP addresses and domain names has been around for some time, the concept of using the DNS to deliver key material was cited as early as [4]. The expansion to other cryptographic applications [5, 6] quickly followed with the view that DNSSEC would someday become a reality.

It should be noted that the desperate need for a global source of identity on the Internet (e.g., e-mail addresses) has had vendors relying on the unsecure DNS for some time as a validation mechanism for creating accounts on web sites and even as part of acquiring digital certificates attesting to the owner of a web site. As the only source of globally unique identifying material on the Internet, the vendors had little choice but to attempt to make use of the DNS in this way.

Once DNSSEC is fully deployed, the technical underpinnings will finally be in place to trust the DNS for such transactions – but this requires that DNSSEC be deployed and managed carefully.

2.1 SSL

Given the availability of this newfound secure, global database in the DNS, the first natural step is to find ways to extend and improve upon existing sources of trust on the Internet. Currently the only widespread source of such trust is Certificate Authorities (CA) providing SSL digital certificates.

Digital certificates provide a mechanism for a CA to cryptographically attest to the identity of the certificate’s owner. The certificate and associated key data is deployed by a web site owner to provide a way for the end user’s browser to validate the site’s owner.

While the inclusion of a CA’s public key or "root key" to validate certificates in an operating system (OS), or its removal in the case of CA compromise, is often a difficult and lengthy process, the addition, update, or removal of records from the DNS is easy and can be dynamic.

Problems with OS distribution include, but are not limited to, dependence on update release cycles, expensive audit requirements (not necessarily bad), inability to reach static installations, and flawed certificate revocation systems.

Placing CA root key material in a DNSSEC secured DNS provides the ability to easily update key material quickly and at little or no expense. This not only improves CA key management but also has the potential of greatly increasing the number of SSL protected sites (currently only 4M [7, 8] out of over 200M) by reducing certificate distribution costs.

Interest in being able to indicate to OS or browsers which CA root key to trust for a particular domain has recently increased with the spate of CA compromises [9–11].
Competition among the large number of CAs (1482 [7]) has often resulted in the use of inadequate methodologies to verify the identity of domain name holders. This has, at times, reduced the quality of a CA’s attestation ¹ and hence certificates issued by it, making it hard to distinguish which CA to trust. Using data from DNS to decide which CA root key to validate against provides a level of “trust agility” in the face of changing trust models and threats.

Linking DNSSEC and SSL CA services and having combined operations (e.g., a Registrar that offers DNS hosting and is a CA) can lead to synergies such as increased assurance that a certificate is associated with the actual domain name owner; reduced costs using shared facilities and procedures between DNSSEC and CA; reduced certificate management costs via DNSSEC distribution; and marketing and additional product opportunities for an entity.

Standards, that are in the final stages of approval, have been collaboratively developed between Internet standards organizations such as the IETF and CA organizations for DNSSEC based certificate distribution and management [12].

Although support for SSL certificates drawn from or validated in DNS is not yet supported in popular operating systems, demonstration software [13] and experimental efforts building DNSSEC SSL support directly into browsers [14] has shown promise.

### 2.2 SMIME

SMIME is a mature secure email system supported by many email packages. Although it has been in existence for some time, difficulties in distributing and managing the PKI key material needed to support SMIME, not to mention being able to exchange such material across different organizations, have kept widespread SMIME use out of reach.

Just like SSL, SMIME uses certificates with a chain of trust to “root” keys. Each email sender has its own certificate that is used by recipients to validate the sender. If a sender can look up a recipient’s certificate, it can also encrypt email destined for that recipient.

The difficulty here is how to easily lookup the certificate of a recipient or sender across disparate systems on the Internet. Currently enterprise certificate distribution mechanisms vary between systems and have unique access control regimes that make interoperability difficult. A single common secure mechanism might finally let SMIME reach its full potential.

DNSSEC to the rescue! Similar to the SSL efforts above, work is being done to place SMIME certificates (or cryptographic hashes uniquely identifying them) into the DNSSEC secured DNS in a standardized format [15]. This would then remove technical barriers to a truly cross-organizational, trans-national secure email system built on an installed base of existing products.

As with SSL certificates, the ability to draw certificate information from the DNS is not part of currently popular operating systems, but demonstrations of this approach have been successful and show promise [13].

¹ A popular mechanism used by many CAs to validate the identity of a domain name holder for Domain Validated (DV) certificates relies on the unsecured DNS as part of email exchanges.
2.3 Other applications

If DNSSEC is widely deployed in a trustworthy manner, we believe many other applications, in addition to SSL and SMIME, will "hang their keys in the DNS. Standards for IPSEC [5], DKIM [6], and discussions on how to secure the ever expanding percentage of VoIP phone calls are either already in place or in process.

On a broader scale, DNSSEC may provide support to the quest for improved identity mechanisms in cyberspace that many governments are taking on in response to public cybersecurity concerns.

Regardless of the focus we believe the open, hierarchical, bottom-up, multi-stakeholder nature of DNS and DNSSEC lends itself to becoming a platform for innovative security solutions that will go far beyond simply securing DNS lookups.

3 Reality Check – Failure?

Every one of the above applications relies on being able to trust responses from the DNS thus placing ever greater pressure on its various entities to operate in a trustworthy manner.

The cryptographic algorithms used in DNSSEC provide mechanisms to ensure this technically. However, the management of the key material associated with DNSSEC is, by its nature, one with opportunities for missed updates and compromise which could lead to failure and mistrust.

The combination of this added complexity with the additional material (e.g., keys) that must be accurately exchanged via administrative/out-of band interfaces between entities in the hierarchy makes the nirvana described above an empty hope if these issues are not given adequate consideration.

We believe the current early stage of DNSSEC deployment at the Registrant level is an opportunity to develop an environment to encourage the implementation of secure processes and practices before DNSSEC comes to be widely deployed and critically relied upon.

With the potential for such high levels of reliance on a DNSSEC secured DNS, it is better to work to set the bar high from the start.

3.1 Chain of Trust

When an end user goes to a web page he/she begins a series of actions that rely on many entities in the Internet ecosystem. The first step typically involves querying the ISP’s local DNS resolvers. The DNS resolvers will retrieve the requested information on behalf of the user. If DNSSEC is enabled on the resolver, it will also attempt to cryptographically validate the response before returning it to the user.

If this is a new request for the resolver, it will then query root, Registry and Registrant DNS servers until it has tracked down the requested information and, for the DNSSEC, validate each intermediate response against prior ones in the DNS hierarchy and terminating with the root key. Since Registrars are responsible for processing Registrant requests and data into Registries, they too are in the path.
From the perspective of DNSSEC validation, the end user relies on the ISP’s resolver to lookup DNS records. The resolver ultimately relies on the root key to validate the DNS result. The root uses its key to attest to the Registry keys. The Registry uses its key to attest to key material received from the Registrar who handles keys controlled by the Registrant. This forms a chain of trust from Registrant to end user.

Registrant $\rightarrow$ Registrar $\rightarrow$ Registry $\rightarrow$ Root $\rightarrow$ ISP/resolver $\rightarrow$ End User

or by way of example:

mybank.se $\rightarrow$ GoDaddy $\rightarrow$ IIS .SE Registry $\rightarrow$ ICANN $\rightarrow$ City DSL $\rightarrow$ mybank account holder

In order for a DNSSEC response to be trusted, each entity along this chain of trust must not only support DNSSEC, but do so in a trustworthy manner. As a chain of trust, the level of trust placed by the end user in a response is set by the weakest link in the chain.

What follows is an overview of possible weaknesses at each link in the chain in the context of DNSSEC.

### 3.2 Registrant

Each Registrant is responsible for deploying DNSSEC on its domain name. This means either putting together its own DNSSEC signing system, complete with key generation and management systems, or outsourcing these operations to a third party.

The current DNS mentality of “set and forget” for what has been effectively a static file does not work for DNSSEC. Specifically, the time dependent nature of DNSSEC signatures requires regular updates and key rollover. Failing to do so can leave the domain name unreachable. For many Registrants, the skills for these additional tasks may not exist or exceed either current human or financial resources. This can lead to inadequate or insecure implementations that may lead to the eventual removal of DNSSEC functionality due to cost or potential reputational harm.

Alternatively, for the vast majority of Registrants, the complexity of DNSSEC key maintenance and other duties will relegate DNSSEC operations to a third party just as Web and DNS hosting is done now.

However, this simply moves the problem to the DNS provider who may have only a limited interest in trustworthy DNSSEC operations while the Registrant still bears the bulk of its own reputational responsibility. The Registrant should therefore carefully consider its agreement with and the reputation of the DNS provider.

Unfortunately, without building awareness in the end user and Registrant communities, providing such services quickly becomes a race to the bottom for the DNS operator who would only be driven to provide a minimal implementation with little concern for trustworthiness and no overall benefit to the end user.

### 3.3 Registrar

As the interface between Registry and Registrant, without the support of the Registrar, DNSSEC has little hope of widespread deployment.
Many Registrars point to a lack of demand for DNSSEC as the primary reason for not supporting it. Registrants, conversely, point to the lack of Registrars supporting DNSSEC as a barrier to deployment. Without sufficient support and deployment by these entities, end users will not reap the benefits of DNSSEC or any subsequent innovation.

An isolated view of support and deployment costs for Registrars and Registrants also enters into this standoff.

Unfortunately this mindset can again lead to either forgoing the benefits of DNSSEC altogether due to lack of support or building untrustworthy end user/registry interfaces and operations to support it. In the early stages of DNSSEC deployment, an excessive number of failures and/or unprofessional handling of incidents would be a death knell for further DNSSEC deployment efforts. With a loss in perceived benefit, Registrants would cease to request DNSSEC and Registrars drop support for it. Without a widely deployed, secure and trustworthy DNSSEC infrastructure, the promise of innovation in the prior section becomes a pipe dream.

### 3.4 Registry

Registries responsible for the TLDs have many of the same concerns as Registrars regarding the lack of demand and cost.

However, the adoption rate here is not so bleak. The decades of DNSSEC development by the Internet community has brought along with it awareness for those operating TLDs. Most Registries accept that they will need to eventually deploy DNSSEC [16].

Due to cost constraints, some of the smaller TLDs may forgo deploying DNSSEC or end up with untrustworthy deployments. As will be described later, there are multiple approaches to overcoming cost issues.

Deployment need not be expensive if proper practices are put into place and expectations set appropriately. There are also multiple inexpensive or free secure [17] outsourcing options.

### 3.5 Root

DNSSEC was deployed on the root July 15 2010 [1] and supported by a key management process requiring the direct involvement of 21 trusted community representatives from around the world. This ensures trustworthy management of the root keys with global buy-in in a way that encourages a simple validating structure relying on one key.

Feedback and suggestions for improvements on this system is continually taken by ICANN from the Internet community.

### 3.6 ISP

The ISP’s direct relationship with the end user places it in a position of trust. In this position, the ISP typically operates a DNS caching resolver on behalf of end users to take advantage of aggregation and speed up DNS response time. To support DNSSEC and thus cryptographically validate responses before passing them to the end user, the
ISP need only enter a copy of the root key and switch on this capability. Unfortunately, few ISPs have done so, fearing additional support calls and additional, albeit minor, maintenance in installing DNSSEC root key material in their resolvers.

Making the leap to turning on DNSSEC validation should be considered carefully since misconfiguration of the ISP resolver would lead to DNS lookup failures and bad end user experiences. The danger of which may be a major setback for turning on DNSSEC validation again and hence keeping DNSSEC benefits from the masses.

### 3.7 End User/Relying Party

The end user or relying party is the most influential entity in the chain of trust. Although in an ideal setting, where DNSSEC has been fully deployed on the Internet, the end user would see no signs of DNSSEC, without end user awareness of the benefits of DNSSEC, it may never reach the critical mass needed for innovation to flourish.

The end user, unaware of DNSSEC’s benefits, might point to “this new DNSSEC service” as a problem during its rollout by, say an ISP, where a poorly operated external domain name may have failed validation by no fault of the ISP. As a result, the end user and ISP may simply turn off validation as an expedient approach to avoid any more support calls. This would further frustrate DNSSEC deployment and block the future benefits it could bring.

Conversely, an end user who is aware of DNSSEC’s benefits (either through Registrant, ISP, Registry or other education) can drive adoption and subsequent DNSSEC application development that will provide a better Internet experience for all.

### 4 How to avoid failure – Raising the bar

Although the previous section did not paint an encouraging picture for the newfound DNSSEC infrastructure ever becoming a source of trust on the Internet, let alone playing a role in critical applications, there is hope.

DNSSEC deployment can be used as an excuse to revamp or improve the processes and practices surrounding DNS operations to become a cornerstone for security on the Internet and to be relied upon for critical applications. By applying practices used by established sources of trust on the Internet and benefiting from their lessons learned, we can transfer and even improve upon many of the same qualities to DNSSEC.

Borrowing from the many decades of experience CAs have developed in selling trust and all the legal, financial, and reputational aspects that entails allows us to bootstrap that same trust into DNSSEC operations. This has been done at the root as well as a few top level domains.

Armed with this experience, below are suggestions for each of the entities in the chain of trust described in the previous section.

#### 4.1 For End Users/Relying Parties

Finding a path toward building trust into DNSSEC deployment is only half the solution toward a trusted DNSSEC infrastructure that benefits all. The other half is encourag-
ing the relevant entities to take the path and implement the processes and procedures borrowed from the CA community.

As the largest beneficiary of a trusted infrastructure, it is the 2B [18] end users of the Internet that need to demonstrate their interest in security by gravitating toward secure Web sites, ISPs, and other solutions on the Internet. By doing so, entities in the chain of trust will be incentivized to take the path that builds trust in their offerings.

Therefore, building end user awareness regarding the benefits of a trusted DNSSEC infrastructure is the key step in ensuring that trustworthy DNSSEC deployment becomes a race to offer the best product instead of a race to the lowest cost and quality service resulting in an untrustworthy deployment.

4.2 For the ISPs

For ISPs, the steps needed to support DNSSEC are deceptively simple. Since the majority of DNS resolver implementations already support DNSSEC, it is only a matter of switching on DNSSEC validation functionality. This will ensure that records for domains with DNSSEC deployed on them must be validated before passing to the end user.

There will likely be increased computational load on the DNS resolvers, that may require additional servers. However, given the gentle uptake of DNSSEC this can be an incremental process.

Due to the public facing nature of ISPs, which can greatly amplify even the slightest misunderstanding on the part of the end user, additional education for support staff regarding DNSSEC should accompany a decision to offer DNSSEC validating service on an ISP’s resolvers.

For example, as Registrants progress along the DNSSEC learning curve, there are bound to be situations that make the Registrant’s site temporarily fail to validate and thus become unreachable by no fault of the ISP. But the complaint reaches the ISP first.

Large ISPs have already begun to roll out support for DNSSEC in their resolvers [19] or are the process of considering it.

Finally, ISPs can use this relatively minor investment by promoting DNSSEC as a differentiator amongst products and competing providers.

4.3 For Registrants

Registrants have a number of options for trustworthy DNSSEC deployment on their domain name.

They can build their own DNSSEC signing system adopting the same practices used by Registries and the root. As will be described below, depending on individual requirements and risk profiles, this need not be costly but it must have transparency and thorough documentation among its key aspects.

Much of the software and equipment needed, as well as training, is readily available.

Rolling your own DNSSEC deployment might be appropriate to high security domain names where financial, legal, patient, or other critical information is regularly exchanged. However, for the vast majority of Registrants outsourcing the generation,
use, and rollover of DNSSEC keys and domain signing will be the only reasonable option.

Here various reputable options exist including some with estimates as low as 2USD/year [20] or as part of packages [21, 22]. DNSSEC signing services may also be provided by the DNS and Web hosting providers currently being used by the Registrant. In each case, the Registrant should check the suitability of any public documents describing the signing services for linking to its own site since its own reputation will depend on them. This would be particularly important if legal or reputational issues rely on the integrity of services provided via the Registrant’s Web site.

Similarly, the ability to seamlessly move DNSSEC operations from one operator to another is critical to protect against operator system or reputational failure. This is often an afterthought but it should not be [23].

Finally, Registrants can differentiate their Web site and other services from competition by promoting the security afforded by DNSSEC, whether home grown or riding on the reputation of the outsourced DNSSEC provider.

4.4 For Registrars

As the interface between Registrant and Registries, the Registrar plays a pivotal role in the chain of trust by:
- Ensuring the accuracy of Registrant contact and technical data.
- Protecting the integrity of Registrant data.
- Providing secure, authenticated paths for communication to Registrant and Registry.
- And supporting the propagation of DNSSEC parameters (e.g., DS records) from Registrant to Registry

A conscientious Registrar can set the level of trust in DNSSEC and associated applications while accelerating DNSSEC deployment.

The direct relationship between Registrars and Registrants also places them in a unique trusted position of giving Registrars the opportunity to offer a wide range of services from traditional DNS, Web, and email hosting to DNSSEC key management/signing, SSL CA services and enterprise SMIME PKI management.

As the basis for most of the practices used to build trust in DNSSEC deployment are the same as those for CAs, there are cost savings in the form of shared facilities, personnel and third party audit requirements.

Finally, by promoting DNSSEC and its associated applications, there is an opportunity for the Registrar to build loyalty programs and to offer differentiated services within its product line as well as with respect to its competitors.

4.5 For DNS Operators (Registries, Registrants and Registrars)

Whether DNSSEC operations are carried out by the Registry, Registrant, Registrar or by a third party DNS operator we believe the shortest path to deploying a trustworthy operation is to build on practices that have been honed over the past few decades by CAs. This has the advantage of not only capturing the best physical, access, logical, and crypto (engineering) practices but also allows bootstrapping many of the audit and legal practices used in the CA industry.
Based on our experience in developing practices and procedures for deploying DNSSEC at the root and DNSSEC deployment discussions with large Registries, the following key concepts make up a trustworthy deployment:

**Transparency** Often assumed but tedious to implement, transparent operations is the key to gaining the trust of your relying parties or public.

One of the first steps towards transparent operations is to set clear expectations and predictability by publishing a practices statement describing how your operations are implemented and contingency plans.

For DNSSEC we borrow directly from the established framework that the CA industry has had in place for creating Certificate Practice Statements (CPS). This framework was developed in cooperation with international accounting organizations which perform audits and provide certifications for CAs and related IT operations.

For DNSSEC this is called a DNSSEC Practices Statement (DPS) and specific frameworks for the same have been developed in the IETF [24].

The development of a DPS serves many purposes. It not only provides an opportunity to set reasonable expectations (e.g. response time, physical security, response to incidents and disaster recovery) but also helps limit liability. Most importantly it forces the DNS operator to decide the level of risks it is willing to accept which will be a primary factor in determining cost. For example: it might be reasonable to forgo the cost of dedicated armed guards protecting your private keys if you have a well documented approach (that is also outlined in the DPS) for detecting and recovering from the unlikely event of compromise.

For the relying party, the DPS allows them to evaluate their own environment and their associated threats and vulnerabilities to determine the level of trust they may assign to DNSSEC in the given domain and the level of risk they are willing to accept.

Finally, as has been demonstrated in business [25] and more recently in DNSSEC deployments citeUK.FR, regular communication with the public via established channels (e.g., Web site), even if to describe a problem, are critical to building trust. The publication of incident reports describing a problem and the corresponding response [26, 27]] have not only garnered praise from other operators but has raised the bar by sharing lessons learned and increasing transparency expectations.

Now that we have said what we will do with the DPS, we need to prove that we did what we said.

Continuing along the same CA track we can provide this proof combining multiple elements. One is audit. This may be performed by a certifying third party (e.g. SysTrust, WebTrust certifications) or even internally depending on your administrative structure. Such audits can be expensive but can provide comfort to a wider range of industries that may not have a sufficient understanding of DNS operations.

Another element borrowed from the CA world is the key ceremony. A key ceremony is a recorded (and sometimes broadcast) event where those responsible for key generation and use follow a script with witnesses to ensure documented procedures are being followed. Such an event also provides the opportunity to involve those outside of normal DNS operations and therefore help broaden trust.
Security

Securing any operation can become an impossibly expensive task if limits are not set based on acceptable levels of risk. Therefore, before embarking on deployment, a DNS operator should do a risk assessment based on the level of service they will be providing.

CAs typically break security into physical, logical, and cryptographic elements. Each one of these seek to protect content and key material from theft, loss, modification, and compromise.

Physical security is usually described in terms of concentric tiers with access to lower tiers required before gaining access to higher tiers. Progressively more restrictive physical access controls to each tier are applied. This could be, for example, tier 1 – a data center requiring authorized personnel to sign-in at a guard desk; tier 2 – smart card access on to the data center floor through a man-trap; tier 3 – a cage or rack area only accessible by DNS operator personnel; tier 4 – a safe whose combination is only known by limited DNS operator personnel authorized to access key storage devices. To deter collusion, access to the different tiers may be split across different personnel.

Although recovery procedures may be in place to deal with loss or compromise, an undetected compromise could lead to much greater damage due to its duration. A common approach to address this is to make use of motion detectors and video facilities (that may be part of the data center) and to rely on notifications and logs kept on the output of these systems. These logs along with entry and exit logs become part of any audit material.

Finally, the use of inexpensive tamper evident packaging to protect key material and associated devices goes a long way to proving that critical components have not been unknowingly compromised.

Logical security in the form of passwords and PIN codes are used to protect access to sensitive components. This includes off-net as well as off-line systems that must rely on configuration access to firewalls. Logs for such access would also fall under the category of audit material.

Logical security may also be used to further limit possible collusion. In the tier example above, activating the key storage device in tier 4 may require a PIN controlled by yet another person. This "separation of roles" is a key concept borrowed from CA operations and visible in most CA certificate practice statements.

Cryptographic security is central to CAs and DNSSEC operation. It includes ensuring that suitable algorithms and parameters are chosen and are implemented correctly; random number sources are trusted and have sufficient entropy; and keys are securely backed up and maintained.

Although this can be implemented in off the shelf software, the assurances certification brings has CAs generally using certified specialized hardware in the form of Hardware Security Modules (HSM) to implement key generation, use and maintenance.

Certification usually comes in the form of the US/Canadian based FIPS 140 standards [28] although other national standards exist. These devices vary widely in features, level of protection (including erasure on tamper attempt), and cost. In some cases instead of the 20000USD high speed networked HSM, a 5USD smartcard validated to FIPS 140-2 level 3 might serve as an HSM for a domain name with very few entries.
Hence, modeling DNSSEC security on CA security methods need not cost much more than existing DNS operations in a typical data center. However, increased documentation, separation of roles requirements and greater involvement of staff to conduct key ceremonies will indirectly add to costs.

For Registrars, added requirements for secure exchange of information between Registries such as NS and DS records (as defined by the Registries) and authenticated secure auditable mechanisms between Registrants are necessary to fortify the trust in these links.

**Availability** The final key concept is service availability. Although ensuring the availability of DNS services is nothing new (e.g. with backup sites), with DNSSEC and its dependence on signatures and absolute time, failures have shown that operations need to improve.

Providing this increased level of service is further made difficult with DNSSEC’s added key management complexity.

Monitoring DNSSEC operations to provide early warning notifications of eminent signature expirations has helped avoid some failures.

A balanced application of automation to key management reduces the possibility of signature expiry and administrative burden. Some Registries have done this by pre-generating signature and key material so as to allow automated processes to take over for extended periods of time.

Experience has shown that the complexities associated with key management that cannot be handled with automation can be tamed with documented procedures, checklists and sufficient coverage by personnel.

**Overview** Although these principles are not foreign concepts to most IT centric organizations, the implementation of the various controls necessary to adhere to them often are. This is understandable given the difference in cultures between CAs (and some of the industries that rely on them) and Internet engineering departments.

Controls consist of documented and published processes and procedures, extensive logging, physical intrusion detection (motion and video), key ceremonies, separation of duties through multi-person access control – physical as well as logical, specialized cryptographic hardware to protect keying material, and audit.

Applying these basic but sometimes new practices to DNSSEC deployment brings the trusted framework established for the CA environment to DNSSEC and along with it the hope that it will fulfill the goal of a global PKI/trusted platform for innovative security solutions for the Internet.

### 5 Summary

With awareness building, setting reasonable expectations, and building on decades of CA practices we believe DNSSEC can be cost effectively supported by all entities in the DNS ecosystem thus creating an infrastructure that critical applications can rely on with opportunities that drive a race to the top benefiting all.
References

1. ICANN http://dns.icann.org/ksk/ds19036/
6. IETF, Domain-Based Email Authentication Using Public Keys Advertised in the DNS (DomainKeys) http://www.ietf.org/rfc/rfc4870.txt
7. EFF SSL Observatory http://www.eff.org/observatory
8. EEF https://www.eff.org/files/DefconSSLiverse.pdf
11. COMODO, Compromise demonstrates need for DNSSEC migration http://www.digitalsociety.org/2011/03/comodo-compromise-demonstrates-need-for-dnssec-migration/
15. IETF, Using Secure DNS to Associate Certificates with Domain Names For S/MIME http://tools.ietf.org/id/draft-hoffman-dane-smime
19. DNSSEC Rollout by large ISP (3M Internet customers) http://www.dnssec.comcast.net/
25. Tylenol Crisis, 1982  http://iml.jou.ufl.edu/projects/fall02/susi/tylenol.htm
29. WebTrust, Certification required by some OS vendors to include a CA root key  http://www.webtrust.org/principles-and-criteria/item27818.pdf
30. IANA DNSSEC Root SysTrust Certification  https://www.iana.org/dnssec/systrust
31. .SE DPS  http://www.iis.se/dl/DPS-PA9-ENG.pdf
32. Root DPS  https://www.iana.org/dnssec/icann-dps.txt

6 Acknowledgements

The author would like to thank Fredrik Ljunggren, Jakob Schlyter, Tomofumi Okubo, and Roy Arends as experts in their respective fields for their time and patience in enlightening the author.

7 Biography

Dr. Richard Lamb Rick has over 25 years of engineering, business, and policy experience in the Internet arena. Currently responsible for DNSSEC efforts at ICANN, Rick was the technical and policy architect for ICANN’s root DNSSEC deployment. Prior to this he was director of global IT policy at US Department of State where he worked to bridge technology and policy. He has founded a number of computer networking startups the last acquired by Microsoft. Rick received his doctorate from MIT in 1987.
Methodologies and Applications for a Secure and Robust DNS
Detecting Hidden Anomalies in DNS Communication

Ondřej Mikle, Karel Slaný, Ján Veselý, Tomáš Janoušek and Ondřej Surý

CZ.NIC
Americka 23, 120 00 Prague, Czech Republic
ondrej.mikle@nic.cz, karel.slany@nic.cz, ondrej.sury@nic.cz

Abstract. The aim of this paper is to implement a profile-based anomaly detection methodology for extracting hidden anomalies in network traffic published by G. Dewaele et al. and to examine its usability for detecting DNS communication anomalies. The used methodology does not require any previous knowledge about the analysed traffic. Even though the method does not require any additional reputation information it is able to detect anomalies with very low intensities. To make it more efficient several modifications to the method with respect to specific characteristics of DNS communication were introduced. As a result a tool capable of detecting hidden anomalies in DNS traffic has been made publicly available.

1 Introduction

The Domain Name System (DNS) [6,?] supplies an important service in the Internet. Most of the communication in the Internet begins with several DNS lookups. Command and Control (C&C) communication used by botnets [4,?] makes no exception to this behaviour. Static blacklists containing per domain name and per IP address reputation data have been used to identify DNS communication originating at malware-compromised sources. In combination with static reputation lists dynamic monitoring systems are being developed.

DNS reputation systems like Notos [1] and Exposure [3] rely on local monitoring of the traffic on local recursive DNS (RDNS) servers. The system Kopis [2] is designed to monitor the traffic at the upper DNS hierarchy. The Kopis system analyses streams of DNS communication at authoritative name servers (AuthNS) or top-level domain (TLD) servers by the means of a statistical classification model. The system extracts features like requester diversity, profile of the querying RDNS servers and the reputation of the the IP address space of the resolved domain name in order to train classifiers. Given a set of known legitimate and malware-related domains as training data, the system is able to learn classifiers that can predict whether the domain is malware-related or not. Kopis claims to have very accurate detection performance even when no IP reputation information is available.

This paper describes a less complex DNS traffic anomaly detection system. It is based on random projection techniques and multi-resolution non-Gaussian marginal distribution modelling [5]. The method was designed to blindly analyse large-scale packet trace databases. For the purpose of DNS traffic monitoring the method has been slightly modified in order to get a better fit on the specifics of TLD DNS communication.
2 Method Description

The used method is based upon the paper *Extracting Hidden Anomalies using Sketch and Non Gaussian Multiresolution Statistical Detection Procedures* by G. Dewaele et al. Primarily the method was designed to analyse large-scale packet trace databases without any previous knowledge. The following section gives a brief description of the methodology. For more detail see [5].

2.1 Method

A continuous stream of packets is being analysed within a sliding time-window of duration $T$. Each time-window is then put through the following procedure.

<table>
<thead>
<tr>
<th>used symbol</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>number of hash functions</td>
</tr>
<tr>
<td>$n \in {1, \ldots, N}$</td>
<td>hash function number used to create a sketch</td>
</tr>
<tr>
<td>$M$</td>
<td>hash table size</td>
</tr>
<tr>
<td>$m \in {1, \ldots, M}$</td>
<td>sketch output number computed by hash function $h_n$</td>
</tr>
<tr>
<td>$J$</td>
<td>size of the set of aggregation levels</td>
</tr>
<tr>
<td>$\Delta_j$</td>
<td>aggregation level, $j \in {1, \ldots, J}$</td>
</tr>
<tr>
<td>$X_{n,m}(t)$</td>
<td>aggregated hashed time series for scale $\Delta_j$</td>
</tr>
<tr>
<td>$\alpha_{\Delta_j}$, $\beta_{\Delta_j}$</td>
<td>estimated $\alpha, \beta$ of Gamma law fit for $X_{n,m}(t)$</td>
</tr>
<tr>
<td>$D_{\alpha_{\Delta_j}, \beta_{\Delta_j}}$</td>
<td>estimated Mahalanobis distance for $\alpha, \beta$</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>distance threshold value</td>
</tr>
</tbody>
</table>

Table 1: Symbols used in algorithm description.

**Random Projection - Sketches** Let $P = \{p_i\}, i \in \{1, \ldots, I\}$ denote the set of $I$ packets captured within the sliding time-window. A packet $p_i$ is represented by a tuple $(t_i, A_i)$, where $t_i$ is the arrival time-stamp of the given packet $p_i$ and the packet identifier $A_i$. The packet identifier $A_i$ is formed from the packet attributes, such as source IP address, destination IP address, source port and destination port.

Let $h_n$ be one of the $N$ independent $k$-universal hash functions which are generated with different random seeds (construction see [9]). Let $M$ denote the fixed size of the hash tables. The packet identifier $A_i$ is used as hash key.

Each hash function $h_n$ splits the original trace $P$ into $M$ sub-traces (sketches) $X_{n,m} = \{(t_i, m_{n,i})\}_{n,m}$, where $m_{n,i} = h_n(A_i), m_{n,i} \in \{1, \ldots, M\}$. The sketch $X_{n,m}$ contains captured packet time-stamps and packet identifiers.

**Multi-Resolution and Aggregation** The sub-traces $X_{n,m}$ are aggregated jointly over a collection of aggregation levels $\Delta_j$ to form the $X_{n,m}(t)$ time series.
The aggregation function gathers all the packets which arrived during a small period (given by the used aggregation level) and returns the number of packets arriving. The process creates a series of time aggregated packet counts. The time aggregated data become significant during the next step of statistical modelling. Here the parameters of a probability density function (PDF), which fits the aggregated data, are estimated.

**Non-Gaussian Modelling** Aggregated traffic in every sketch is modelled using Gamma distribution laws $\Gamma_{\alpha,\beta}$. The $\Gamma_{\alpha,\beta}$ distributions are stable under addition of independent random variables. The difference between corresponding Gamma parameters $\alpha$ (shape) and $\beta$ (scale) relates to the short-time statistical dependencies along time.

From $X_{\Delta_j}^{n,m}(t)$ the corresponding collection of parameters $\{(\alpha_{\Delta_j}^{n,m}, \beta_{\Delta_j}^{n,m})\}$ for all aggregation levels $\Delta_j$ are estimated (by means of standard sample moment procedure).

**Reference Values** Average behaviour and typical variances of corresponding values for each $h_n$ are estimated: $\gamma_{\Delta_j}^{m,R} = \langle \gamma_{\Delta_j}^{n,m} \rangle_m$ and $\sigma^2_{m,\gamma,\Delta_j} = \langle \langle \gamma_{\Delta_j}^{n,m} \rangle_m \rangle_m$, where $\langle \cdot \rangle_m$ denotes the standard sample mean estimator and $\langle\langle \cdot \rangle\rangle_m$ variance estimator, computed from $m = 1, \ldots, M$. Either the $\alpha$ or $\beta$ parameter of the Gamma distribution can be used for $\gamma$.

**Computing the Statistical Distances** Anomalous behaviour of the given sketches described by $\{(\alpha_{\Delta_j}^{n,m}, \beta_{\Delta_j}^{n,m})\}, j \in \{1, \ldots, J\}$ is measured by computation of the statistical distance from the reference behaviour $\gamma_{\Delta_j}^{m,R}$. Here the Mahalanobis distance $D_{\gamma,\Delta_j}$ is being used.

$$ (D_{\gamma,\Delta_j})^2 = \frac{1}{J} \sum_{j=1}^{J} \frac{(\gamma_{\Delta_j}^{n,m} - \gamma_{\Delta_j}^{m,R})^2}{\sigma^2_{m,\gamma,\Delta_j}} \quad (1) $$

When $D_{\gamma,\Delta_j} \leq \lambda$, where $\lambda$ is the threshold value, then $X_{\Delta_j}^{n,m}(t)$ consists of normal traffic. When $D_{\gamma,\Delta_j} > \lambda$ then $X_{\Delta_j}^{n,m}(t)$ is said to contain at least one anomaly. The detection threshold $\lambda$ is to be chosen.

**Anomaly Identification by Sketch Combination** Reversing the hashing procedure allows the identification of the hashing keys associated to the identified anomalies. When an anomaly in the $m$-th output of the hash function $h_n$ is found then the corresponding attributes $A_i$ are registered within a detection list $A_n^p$. Combining all of the $N$ hash functions $h_n$ and taking the intersection of the $A_n^p$ results a list of attributes $A_o^p$ which correspond to the detected anomalies within the scanned time-window.

In a nutshell – the method analyses the packets within a sliding time-window by dividing the traffic randomly into sketches (groups). Each sketch is evaluated by statistical means resulting in several characteristics which can be compared. The characteristics are compared by computing a statistical distance from the average behaviour. When the computed distance reaches a given threshold all packets in the sketch are declared
anomalous. This process is repeated using a different random projection resulting different anomalous sketches. Intersections between anomalous sketches are computed in order to isolate the anomalous packets. The whole process is repeated until the number of anomalous packets no longer changes by consequent iterations or when maximal iteration count is reached.

3 Application in DNS Traffic

During the implementation of the algorithm a decision to extend the analysis past the standard TCP/IP connection identifiers (i.e. source address, destination address, protocol, source port, destination port) was made. A system of modules called policies has been implemented in order to test various packet identifiers and their viability for the tested detection method.

Two policies have been designed and implemented. Identifiers selected by policies are used as hashing keys for dividing packets into sketches.

**IP Address Policy** It is based on the original idea which uses part of the TCP/IP connection identifier as hash key. Here we use the source IP only. Destination address, destination port and transport protocol show little to none variability in TLD DNS traffic. The inclusion of source port was left for future testing.

**Query Name Policy** This policy is based on application layer data. The first domain name in a DNS query is extracted and used as hash key.

3.1 Tool Functionality

The behaviour of the algorithm can be adjusted by setting several parameters. This section presents a list of them with a short description how they can impact the results.

`-w, -window-size=<seconds>`

The application analyses captured DNS communication within a fixed size time-window. The length of the window is set in seconds. Setting the window to 5–10 minutes yields good results. The lower bound on the usable window size depends on the traffic density as higher traffic will generate more reliable reference behaviour in shorter time windows.

`-i, -detection-interval=<seconds>`

The interval between the starts of the analysis is passed via the detection interval parameter. Using shorter detection intervals than the length of the detection window allows the application to work with overlapping time-windows.

`-a, -aggregation-count=<num>`

The application uses an exponential aggregation scheme. Given the parameter of aggregation level count, the application will utilise aggregation levels of \{1, 2, \ldots, 2^{num-1}\} seconds. Low aggregation count (2) will turn focus in the direction of short-time anomalies (containing slightly elevated false positive rate in the short-term range). Higher count (4 up to 8) will lean towards long-term anomalies.
-p, --analysed-gamma-parameter="shape"|"scale"|"both"
Selects whether to analyse the shape, scale or both of the Gamma distribution parameters. Setting shape or scale yields similar results, setting both leads in several cases more precise results – in our case less false positives were emitted.

-t, --detection-threshold=<num>
The detection (distance) threshold parameter is left to user's choice. It determines the boundary past which the sketches are marked as anomalous. The threshold setting serves as tradeoff between sensitivity and false positive rate. Threshold of 0.8 seems to be a good choice when analysing scale or shape. When analysing both the value should be raised by factor from 1.4 to 2 to get reciprocal behaviour.

-P, --policy="srcIP"|"dstIP"|"qname"
The choice of the policy strongly affects the type of detected anomalies as will be pointed out in following text. Choices are srcIP, dstIP and qname.

-c, --hash-count=<num>
The user is free to select the count of the used hash functions. The ideal count of hash functions (algorithm iterations) to be used is the least number such that the set of resulting anomalies remains unaltered by adding another hash function (performing consecutive iteration). The purpose of increasing the number of used hash functions is to minimize the probability of a packet identifier $A_k$ to be mapped repeatedly together with an anomalous identifier $A_l$ into same sketches – thus minimizing the probability of marking a non-anomalous identifier as anomalous.
The application currently does not determine the ideal count. Ideal value depends on the volume of analysed data and is loosely related to sketch count. (In general, increasing sketch count allows the decrease of the count of hash functions.) Too high values slow down the application with marginal detection improvement.

-s, --sketch-count=<num>
The size of the hash tables can be set via the sketch count parameter. Low values generate improper results, values between 16 to 32 seem to be a good choice.

A graphical illustration of the anomaly detection method can be found in figures 1 and 2.

### 3.2 Performance Measurements

The speed of the application is demonstrated using a 1 hour packet dump file. The file holds .cz ccTLD traffic from Tuesday, April 12, 2011 captured between 16:00 and 17:00. The file holds 10365678 packets totaling 2.3GB of data.

Given the arguments which are shown in the table 2, and which are also used in latter experiments, the tool processes the dump file very quickly. The analysis time varies from 10 to 45 seconds resulting speeds from $2.3 \cdot 10^5$ up to over $10^6$ packets/second. In our case the performance measurements have presented that the application is capable of analysing real-time traffic. The main factor affecting the analysis speed is the selected policy. The query name policy consumes about 4 times more CPU time than the source IP policy.
4 Anomaly Detection Experiments

In all test cases the DNS traffic analyser was run with parameters shown in table 3. Detected anomalies were consequently inspected (and tracked back if necessary) in order to identify the type and origin of the anomaly. The higher number of hash functions was chosen in order to avoid identifiers being reported by appearing in the same sketches as anomalies.

The traffic being used at testing origins from DITL\(^1\) 2011 data collected in April 2011 from authoritative name servers for the .cz TLD. Traffic was captured in the course of 50 hours and amounts to 300 GB of packet dumps.

Anomalies detected by the tool can be divided into the following categories:

- generic (generated by recursive resolvers)
- blind or dictionary-based domain enumeration
- domain enumeration with the knowledge of its content

---

\(^1\) Day in the Life of Internet data collection at [https://www.dns-oarc.net/ditl/2011](https://www.dns-oarc.net/ditl/2011)
Fig. 2: Progress of computing Mahalanobis distances for three aggregation levels. When a line exceeds the boundary given by the $\lambda$ parameter (represented by thick dashed horizontal lines at -0.8 and 0.8) then the corresponding sketch is marked as anomalous. Note that the plotted distances do not correspond to the values of $D_{\gamma_{n,m}}$ as defined by equation 1 – the plot shows the values of the expression $(\gamma_{n,\Delta_j} - \gamma_{m,R})/\sigma_{m,\gamma,\Delta_j}$ for 16 sketches.

<table>
<thead>
<tr>
<th>srcIP</th>
<th>qname</th>
<th>srcIP</th>
<th>qname</th>
<th>srcIP</th>
<th>qname</th>
<th>srcIP</th>
<th>qname</th>
</tr>
</thead>
<tbody>
<tr>
<td>-w 600</td>
<td>-w 1200</td>
<td>-w 1800</td>
<td>-w 3600</td>
<td>-i 600</td>
<td>-i 1200</td>
<td>-i 1800</td>
<td>-i 3600</td>
</tr>
<tr>
<td>scale</td>
<td>11.9</td>
<td>44</td>
<td>12</td>
<td>41</td>
<td>12.4</td>
<td>37.9</td>
<td>10</td>
</tr>
<tr>
<td>both</td>
<td>12</td>
<td>44.5</td>
<td>12.2</td>
<td>40.4</td>
<td>12</td>
<td>39.6</td>
<td>10.2</td>
</tr>
</tbody>
</table>

Table 2: Speed experiments. Testing an 1 hour packet dump on a 64bit Intel E5400 CPU @2.70GHz. The parameters $-c$ 25 $-s$ 32 $-a$ 8 $-t$ 0.8 are common for all experiments. The resulting numbers are the average of 10 consequent runs in seconds.

- Traffic generated by broken resolvers or scripts
- Repeated queries due to short TTL in a resource record

The tool pinpoints flows which are statistically different from the other flows in the detection window but does not understand semantically the content of the communication. Thus the interpretation of the pinpointed traffic has to be done by other means – in our case manually.

### 4.1 Domain Scanning Anomalies

Query bursts with lots of NXDOMAIN results are usually a sign of blind domain enumeration or spam attempts (which depends on the type of resource records queried – NS, A, MX). A case of domain enumeration was apparent when inspecting the anomalous queries. The queried names were either dictionary-style in alphabetical order or concatenation of a dictionary stem with a permutation of prefixes and postfixes (possibly for purposes such as search for brand infringement).
Table 3: General experimental set-up. The values have been experimentally selected. They appear to present good detection performance while maintaining the computational load at a reasonable level.

<table>
<thead>
<tr>
<th>parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>window size</td>
<td>10 minutes</td>
</tr>
<tr>
<td>detection interval</td>
<td>10 minutes</td>
</tr>
<tr>
<td>hash function count (iterations)</td>
<td>25</td>
</tr>
<tr>
<td>hash table size (sketch output count)</td>
<td>32</td>
</tr>
<tr>
<td>number of aggregation levels</td>
<td>8</td>
</tr>
<tr>
<td>threshold level</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Dictionaries chosen by the enumerating clients seem to consist of names harvested from generic top-level domains. Sample flow with detected domain enumeration anomaly is shown in figure 3a. Aside from hosts guessing the domain names from an arbitrary dictionary, occasionally the domain enumeration anomaly discovers a host evidently very familiar with the zone’s content since the NXDOMAIN ratio is almost negligible.

A specialized case exhibiting high traffic is a resolver tied with a web crawler. Such detected traffic can be seen in figure 3c. The distinction from a generic resolver of a big ISP would not be apparent from the traffic contents alone, extra information about the network is necessary.

With increasing length of the analysed time-windows the detection favours picking up caching resolvers of large ISPs. The traffic pattern is unique compared to other resolvers in a longer time frame. Due to anycast addressing of the .cz TLD server cluster, data originating from some of the TLD DNS servers manifest ISPs’ resolvers as anomalous more often than others.

4.2 Low Traffic Anomalies

The tool is able to detect anomalies which do not generate high volume traffic. A broken resolver is characterized by numerous repeated queries for the same resource record (sometimes with no sense) over a very short time period. In this regard such a resolver cannot be distinguished from a script, which may be malfunctioning or intentionally embodying non-standard behaviour. Example of such traffic is displayed in figure 3b. Repeated queries for A records of popular domains with short TTLs by various resolvers yield a similar pattern, e.g. an domain containing anti-virus software updates shown in figure 3d.

A distinct pattern appeared concerning a foreign embassy’s A record has also been detected. A sample of 10 minute traffic is recorded in figure 3g. The spikes correspond to an almost unique set of few hundred resolver IP addresses concentrated into few seconds. The pattern repeats itself throughout the whole 50-hour packet dump and is appearing exclusively as the characteristic spikes. During the 50 hours the A record was queried by a set counting about 12000 unique IP addresses evenly spread across autonomous systems and countries.
Fig. 3: Examples of detected anomalies. The anomalies shown in figures (d), (f), and (g) were reported by query name policy, the rest was detected by source IP policy. Note that the anomalies do not need to create traffic peaks compared with total traffic flow in order to be detected.
4.3 Malware-Related Anomalies

The test search did not show any anomalies for the domains listed in blacklist databases abuse.ch\(^2\) and Google Safebrowsing\(^3\), only a single C&C server in the .cz domain was listed in the databases and the server has already been blocked. The following manual search exposed only a single DNS request for the given C&C domain in the complete packet dump.

Domains recorded in Google Safebrowsing appeared to run compromised legitimate services where a piece of javascript or other code has been implanted in order to redirect users to drive-by downloads (i.e. no botnet C&C server present). As such the DNS traffic related to these sites is expected to be mostly legitimate. Traffic related to the most requested domain listed in the Safebrowsing database consisted of less than one request per minute.

Although we discovered no anomalies matching known malware-related domains, some detected patterns suggest the presence of compromised hosts. One of such patterns may be a burst of MX queries originating in a single host – as presented by figure 3e – which may present circumstantial evidence of a spamming bot.

Similar evidence graphed in figure 3f testifies requests generated by approximately 160 hosts querying for an identical MX record which may suggest a botnet carrying out a spam delivery command. The figure is plotted with time-window length increased to 30 minutes to demonstrate the duration of the anomaly. Nevertheless, the anomaly was initially detected with the original 10 minute time-window. The former case concerning a single host querying MX records was encountered when the size of the sliding time-window was decreased to the interval of 1-3 minutes.

5 Conclusions

Built upon multi-scale modelling, the implementation of the anomaly detector is able to pinpoint low-volume anomalies in DNS traffic as well as high-volume anomalies like scanning. Minor adjustments to the original method\(^5\) have been proposed in order to comply with DNS traffic specifics. Changes involve the extraction of the query name to be used as a hash key in the process of sketch creation.

Packet attributes which are selected to serve as hash keys affect the classes of anomalies which are identified by the tool. Domain enumeration attempts and invalid resolver communication can easily be detected by the source IP policy. Whereas the query name policy divulges domain-related incidents such as the presence of domains with short TTL – an attribute which fast-flux domains are characterised with.

An open source implementation of a DNS traffic anomaly detector utilising the above mentioned approach is available for download\(^4\). The application was created with the aim to supply a simple tool able to detect suspicious behaviour at a TLD beyond the range of the content of reputation lists containing known malware compromised

\(^2\) abuse.ch botnet tracker. Zeus, SpyEye and Palevo botnet IP & domain name blocklist at https://www.abuse.ch/

\(^3\) Google Safebrowsing database at https://code.google.com/apis/safebrowsing/

\(^4\) Project repository can be found at git://git.nic.cz/dns-anomaly/.
sources. Unlike such databases the tool is capable of pinpointing suspicious, abnormal or just interesting behaviour in real-time traffic. The analysis of such captured traffic is left to manual inspection or automatized heuristics. Currently we do not deploy any type of automated classification of the pinpointed traffic.

References


Author Biography

Karel Slany received its Ph.D. focusing on evolutionary algorithms at Faculty of Information Technology, Brno University of Technology. Since 2010 he is a programmer for research and development at CZ.NIC

Ondrej Mikle is security researcher for CZ.NIC Labs, mostly focusing on SSL/TLS and PKIX.
What an IP-over-DNS Tunnel
-A Case Study on Large Operational Network

Ziqian Liu
China Telecommunications Corporation
liuzq@chinatelecom.com.cn

Abstract. China’s main Internet providers’ WiFi access points are exposed to a leak of authentication, and have being publicly compromised by a profit-seeking DNS tunneling software called LoopcVPN. The paper reveals the technical principles and details of such kinds of IP-over-DNS software. The prevalence of the software usage is assessed across an large operational IP network. To inhibit the usage of such IP-over-DNS software, the paper proposes and validates a set of countermeasures, in which SM/RFC algorithm is to suppress the recursive queries for anomalous volume of distinct names within common domains. The method shows promising performance by being deployed on an operational recursive nameserver.

1 Introduction

Nowadays, many large Internet Service Providers (ISPs) provides wireless Internet access services to the public. ChinaTelecom (CT), as the leading ISP in China, runs its own WiFi-IEEE 802.11 infrastructures called “ChinaNet”\(^1\). In most cases, ChinaNet will charge users for Internet access fee in terms of duration or traffic volume, and customer authentication for WiFi access should be mandatory and inevitable as designed. Recently, a profit-seeking software called LoopcVPN [1] claims to provide free Internet access through ChinaNet and CMCC (the WiFi deployment of ChinaMobile) without need of user authentication. By saying “free”, it means that a user does not need to pay to the ISPs, but does need to purchase, though at a much lower price, a valid password from a C-to-C Taobao\(^2\) account for monthly usage of the software. Such behaviors really hit a nerve of the ISPs.

In this paper, we study the packets captured on the LoopcVPN clients and reveal that the software takes advantage of the fact that most of the operational Broadband Remote Access Servers (BRAS) turn on green light for UDP port 53 traffic generated by customers even before customers get authenticated. This makes it possible for a remote proxy to pretend as a DNS server and to set up a tunnel encapsulated in DNS protocol with users behind BRAS. The paper studies two scenarios this software currently works in to bypass the authentication process, and points out the potential evolution of the

---

\(^1\) ChinaNet is originally known as the name of ChinaTelecom IP backbone network (AS4134). ChinaTelecom adopts the same name for its WiFi access point. In this paper, ChinaNet means the WiFi access point if without specification.

\(^2\) Taobao is China’s leading e-business website. [http://www.taobao.com](http://www.taobao.com).
software which may further abuse the provider’s recursive nameservers. Countermeasures are proposed from mechanism perspectives, and the experiments on operational networks show the measures are practical and effective.

There exist several implementations for DNS tunneling [2] [3] [4], but to our best knowledge, none of them are like LoopcVPN to actually set up infrastructures to provide public services for profit-seeking purpose. This work is for the first time to investigate and assess the prevalence of such IP-over-DNS applications across a large-scale operational network. The fact is that DNS systems are facing unprecedented challenges from being manipulated as intermediate agents for many other Internet applications [2] [3] [4] [5] far from what the DNS born for, let alone the variety of DDoS attacks which inflict abnormally high volume of packets on upper DNS hierarchies [6]. These issues are all relevant to manipulating the DNS queries. We hence propose a heuristic algorithm called SM/RFC algorithm to identify the anomalous domains that get involved in such undesirable abuses and to mitigate the impact of the evil recursive queries to the whole DNS infrastructures.

2 Authentication Bypass Methodology

2.1 LoopcVPN’s Tricks

We first review the normal authentication process for ChinaNet, as illustrated in Fig. 1. When a user wants to access to ChinaNet, he may arbitrarily input a URL in web browser, the generated HTTP traffic (Step 3) will destine to the IP address resolved by DNS beforehand (Step 1 and Step 2). When the BRAS sees the traffic, it will redirect the user to a captive portal (Step 4). After the user inputs the account information (Step 5), the captive portal server forwards these information to the AAA server (Step 6). Only when the user information is valid for passing the authentication (Step 7) can the BRAS let the user’s traffic pass (Step 8).
Fig. 2: Interactions between the LoopcVPN client and the server in Scenario 1.

The BRASes usually permit DNS traffic to go through in pre-authentication domain. In addition, a few of CT’s BRASes do not regulate the destination address of the users’ DNS queries in the pre-authentication domain, letting the UDP port 53 traffic to go anywhere; while the rest majority of BRASes explicitly set this destination to be the CT’s recursive nameserver’s IP addresses. LoopcVPN takes advantage of these facts and works in either circumstance.

**Scenario 1**

Given the pre-authentication domain on BRAS does not require for specific destination IP address for DNS traffic, the client can send/receive UDP port 53 traffic to/from LoopcVPN servers, see the two dashed arrows in Fig. 2. LoopcVPN now have 12 proxy servers across China. When launching the LoopcVPN client software, a user can choose which server to connect to or let the software to choose the optimal based on the measured round-trip time. Fig. 3 shows a set of traces captured from the LoopcVPN client side (we actually buy one for investigation), where the client IP address is 1.202.0.38 and the VPN server is hn.loopc.com. The traces from entry No.31 to No.34 are normal DNS resolution to get the IP address for hn.loopc.com, which is 218.75.152.141. Things become mysterious since entry No.35, where the client sends a “malformed” (interpreted by Wireshark) DNS query to the VPN server, and the VPN server replies with a “malformed” DNS response in No.36, which contains a RSA key as highlighted in Fig. 3. This is very likely to be the key for users to encode the following HTTP request and decode the answers from the VPN server. We then browse google websites and keep the packet-capture running. Indeed, the communication between the client and the server thereafter are all “malformed” UDP port 53 packets, without any HTTP traffic as usual. During the whole surfing process, 92 queries and 101 responses are captured. The amounts of queries and responses are close while their packet size distributions differ a lot. 80% of the queries are less than 119 bytes, but 70% of the responses are larger than 180 bytes and 47% of the responses have unified size of 1356 bytes. Such an
obvious difference is consistent with the fact that the volume of the HTTP downstream traffic should be much larger than that of the upstream.

**Scenario 2**

As mentioned above, the majority of the BRASes do specify the provider’s recursive nameserver IP address in the pre-authentication domain, which blocks the way of directly tunneling to a remote server by UDP port 53. However, LoopcVPN still works around this by treating the nameservers as intermediate proxies. The challenge here is that, since all UDP port 53 traffic generated by a client has to go to the provider’s nameserver, how to make the nameserver follow the client’s words to forward whatever the client wants to deliver.

LoopcVPN sets up 2 of the 12 servers which only work in Scenario 2. The two servers have different names (one of them is named just-try-it.loopc.com) but both resolved to a same CNAME cry555.com with A record 114.80.209.166. Fig. 4 shows the interactions between the client (180.109.247.129) and the provider’s nameserver (218.2.135.1) when the client wants to visit www.baidu.com. The client generates a normal A query for the Baidu name, followed by a NULL query with a long encoded prefix and a fixed suffix cry555.com, as highlighted in Fig. 4. Such a request hits nothing in the cache of the provider’s nameserver, and triggers iterative queries to the authoritative nameservers for cry555.com. The authoritative nameservers for cry555.com are seemingly to be two boxes, ns1.loopc.com and ns2.loopc.com, but are essentially one server with IP address 114.80.209.166. Since this server is not a real authoritative nameserver and will not correctly response, the provider’s nameserver replies to the client a SERVFAIL (see entry No.1682).

In order to make sure that the remote VPN proxy is available, the client runs a keep-alive mechanism by periodically sending a NULL type record query for a same name in cry555.com. The provider’s nameserver will, after querying the upper DNS, answer the
query with the two authoritative nameservers in the response, which has a 0-sec TTL meaning no caching at all.

The webpage contents are fetched by the LoopcVPN server, and are delivered directly to the client without the nameserver’s cooperation. Indeed, under Scenario 2, the direct communication between the client and the LoopcVPN server is unidirectional, from the LoopcVPN server to the client only. These packets are all encapsulated in DNS response format, with encoded payload including a characteristic “RSV”-start bit string. The downstream path these packets follow is routable since a user gets a public IP address by DHCP from BRAS even before authentication. We argue that such an IP address assignment policy in pre-authentication domain is ill-considered, because it brings extra address source consumption and security risks.

Fig. 5 illustrates how the mystery goes in Scenario 2. The client encodes the user’s HTTP requests by the RSA key (Step 1), encapsulates them into DNS queries for names in cry555.com and sends them to the provider’s nameserver (Step 2). After receiving these queries and looking through the cache, the provider’s nameserver finds no hit and starts iterative queries (Step 3). These queries are then received by the authoritative server for the zone cry555.com, which is actually the LoopcVPN proxy server. The LoopcVPN server will decode the queries and fetches for the client the desired contents (Step 4). After that, the LoopcVPN server encodes those contents and encapsulates them into pseudo DNS responses, which are then transmitted to the client through the BRAS (Step 5). Finally, the client will decode and reassemble these packets and get what it wants (Step 6).

2.2 One More Choice - Iodine

In Scenario 2, we see a clear triangle path as indicated by the dashed arrow-lines in Fig. 5, i.e., the upstream path is from the client through the provider’s nameserver to
Fig. 5: Interactions between the LoopcVPN client and the server in Scenario 2. the LoopcVPN server, and the downstream path is from the LoopcVPN server directly to the client. We then consider a more aggressive way this software might work, which keeps the upstream unchanged and make the downstream pass through the provider’s nameserver too. That is, comparing to Scenario 2, the LoopcVPN server could response to the provider nameserver’s queries, and the provider nameserver forwards these responses to the client. We refer to this as Scenario 3. The LoopcVPN software already shows the talent to work in Scenario 3, since it uses NULL type queries and sets RR TTL to be 0 seconds. NULL type significantly expands the payload format and size that the response can obtain, and 0-sec TTL makes the provider’s nameserver not cache these RRs at all [7].

Our tests show that currently LoopcVPN cannot work in Scenario 3. While we do find a software call Iodine [2] that claims to work exactly in this mode, and we set up an experimental environment to see how well it can work. Since the test nameserver is very light-loaded and close to the client, the performance is fair. We manually change the DNS query types following the order from NULL, TXT, SRV, MX, CNAME to A, and our experiences get worse. Moreover, we highly suspect that the LoopcVPN software is actually developed based on Iodine, since the packet traces captured from the Iodine experiments imply a fairly close behavior to that of LoopcVPN.

3 Prevalence Assessment

We assess the prevalence of the LoopcVPN software throughout ChinaNet from two aspects: the frequency the LoopcVPN-relevant domain names are requested and the traffic volume destined to/originated from the IP addresses relevant to LoopcVPN. We implement a 48-hour long measurement from 23:00 Jun 13 to 23:00 Jun 15 2011 Beijing Time (UTC+8) to get statistics.

At the time, LoopcVPN has 12 proxy servers distributed over 6 provinces in China, and there are altogether 25 relevant domain names used by these servers. We record the
Fig. 6: Traffic behavior and classifications of LoopeVPN servers.
number of requests to these domain names on CT’s main DNS nodes. The fact is that only a fraction of our DNS nodes encounter the relevant requests, and we count totally 23918 such requests in two days, namely 8.3 requests per minute or 0.14 QPS, which is negligible given 60 million ChinaTelecom broadband Internet populations.

We use Arbor Peakflow SP [8] to analyze the Netflow dataset, which are collected from all the core routers in CT IP backbone (AS4134) following a unified 1:5000 packet sampling rate. Fig. 6 demonstrates the LoopcVPN-relevant traffic behavior and classification, where the x-axis shows the time period and the y-axis indicates the traffic volume. The application traffic are identified by the classical 5-tuple method, e.g. HTTP is of TCP port 80 and 8080, and DNS is of UDP port 53. We observed a clearly diurnal behavior of the LoopcVPN traffic, since 75% of the traffic are HTTP and DNS traffic which normally involves human interaction. We classify the rest quarter into “other”, most of which does not match any well-known 5-tuple character. The positive y-axis (+out) indicates the traffic volume originated from the LoopcVPN servers and negative y-axis (-in) indicates the traffic destined to those servers. With respect to the DNS traffic (the dark blue portion), the LoopcVPN servers send out more and receive less. This is because in both Scenario 1 and 2, the downstream traffic sent from the LoopcVPN servers to the clients contains encoded Internet content while the received DNS traffic by the VPN servers are just client requests for the desired content. In contrast, since the LoopcVPN servers act as proxies to fetch the Internet contents for clients, the HTTP traffic amount destined to these servers (the green portion) is much higher than the servers originate (hardly seen from the figure).

Although the WiFi authentication leak widely exists and the LoopcVPN client software is publicly available on the Internet, the prevalence of this IP-over-DNS software is very limited in terms of both the name requests and traffic volume. Such results surprise us a little bit at first as we had expected a relatively higher popularity. We then consider two main reasons for the current status. First, the geographic distribution of the ChinaNet APs is still very limited to a few of hotspots in large cities so far but not the ordinary residential areas. Moreover, mostly when users roam into these hot areas they could surf the Internet free of charge for like 10 hours or so which CT already offers in the set menu. Second, the user experience of this LoopcVPN highly depends on factors like the real tunneled Internet applications, the distance between client and VPN servers, the load of VPN servers and the cooperation of provider nameservers, which all cause the DNS tunnels quite unstable. Hence, the customers’ motivation to actually turn to a pre-paid IP-over-DNS tunnel remains low.

4 Counter Measures

The existence of the LoopcVPN-like IP-over-DNS software poses threats to ISPs interests in two aspects. First, it escapes the accounting while consuming the network resources. Second, it brings undesirable burden to provider nameservers (in Scenario 2

---

3 Generally DNS traffic may include both TCP and UDP port 53 traffic, however in this case, we use UDP port 53 only since no TCP-based DNS traffic is discovered.

4 Though a very tiny fraction of the traffic is identified as SSL (TCP port 443/993/995), this may be due to the remote control requirements to these servers.
and 3) because the nameservers are forced to do lots of iterative queries which can be resource-consuming and dangerous [9].

4.1 BRAS Configurations

We insist that we should inhibit such kinds of software from the mechanism perspective rather than just blackhole some IP addresses. Our strategy is straightforward. We modify the BRAS pre-authentication domain configuration by two steps. Step 1, we mandatorily specify the provider’s nameserver IP address to be the only valid destination for DNS traffic to go to from clients. By this means we eliminate the possibility of Scenario 1 and degrade the transmission efficiency to a large extent. Step 2, we supplement an inbound white list on the basis of the concurrent outbound white list, and the inbound list specifies that the valid source IP address of the DNS traffic should only be the provider nameserver’s address, and thus blocks the downstream path from the LoopcVPN server to the client (Step 5 in Fig. 5). Such a set of bidirectional ACLs on BRAS turns out be instantly effective in eliminating both Scenario 1 and 2 at the pilot points in our operational networks.

4.2 SM/RFC Algorithm

If LoopcVPN evolves towards Scenario 3 or Iodine is in use, the previous measures should be in vain as the DNS traffic exchange between the client and the provider’s nameserver is still permitted and remains unexamined. We thus propose a heuristic algorithm called Suffix Matching/Requested Frequency Counting, or SM/RFC for short, to tackle the problem. Here, the suffix means the common domain of multiple domain names; the prefix means the rest apart from the suffix in a domain name. The intuition comes from the fact that the IP-over-DNS software has to rely on some certain domains, e.g., cry555.com in this case, as the suffix following the encoded content, and this suffix will remain unchanged during a relative long period. In contrast, the prefix, namely the encoded desired content, should frequently changes and remains distinct. In addition, the length of such requested names tends to be much longer than normal ones. Take LoopcVPN as an example, the sizes of the encoded requested names except those for keep-alive purpose are all longer than 80 characters. Thus, the main idea of the SM/RFC algorithm is focusing on the queries for very long names sent out from the recursive nameserver, merging the requested names on the basis of their common suffix, and counting the requested frequency of each distinct suffix. Since the provider’s recursive nameserver will be driven to send out lots of long name queries with distinct prefixes but same suffix, the algorithm periodically outputs the top frequently request suffixes, and hopefully the domains that IP-over-DNS software adopts should be unveiled.

We emphasize that since what we concern most is the unwanted queries from the recursive nameserver to the upper DNS hierarchies, the method focuses on this part of queries rather than queries from clients to the recursive nameserver. SM/RFC basically fulfills three missions, i.e., to identify the anomalous recursive queries for certain domains, to suppress the corresponding domain queries and to release the suppressed
domains. The way of how to suppress and release domains inherits some ideas from BGP route flap damping [10].

For every one minute, the algorithm parses through the queries saw in the minute, merging the query names into domains according to their common suffix, counting the number of requests for each suffix. A request frequency threshold ($RFT$) is empirically set beforehand. Any count that exceeds $RFT$ will be recorded and the corresponding suffix will be marked as the suspect suffix. If a suffix count exceeds $RFT$, the suffix will be assigned a fixed penalty value. If a suffix’s accumulative penalty exceeds the suppressing threshold ($ST$), the suffix will be suppressed, meaning that any recursive query for names with that suffix will not be forwarded by the recursive nameserver. In order to keep the accumulative penalty value from being too big, we ceil it with $max\_penalty$. The algorithm sets a parameter called $decay\_half\_life$, and its value means time duration in minutes after which the accumulated penalty value of a suffix will be reduced by half. Again, in order to prevent a suffix from being suppressed for too long, we introduce $max\_suppress\_time$, which correlates with $max\_penalty$ as:

$$max\_suppress\_time = decay\_half\_life \times \log_2\left(\frac{max\_penalty}{ST}\right).$$

The penalty decay process follows an exponential function as

$$penalty_t = \left(\frac{1}{2}\right)^{\frac{\Delta t}{decay\_half\_life}} \times penalty_{t-\Delta t},$$

where $penalty_t$ is the cumulative penalty value at the end of the current minute, and $decay\_half\_life$ is duration time in minutes.

In order to prevent accidentally suppressing of “good” domains, the algorithm maintains a white list which includes well known domains. If the request frequency of any domain on the white list exceeds the $RFT$, suppress will not be executed automatically, but a dedicated syslog will be triggered and the relevant messages will be shown on our DNS monitoring system.

For every minute at time $t$, the algorithm functions as the below:

```
#Pseudo code for suppression process
for each penalty_{t-1,i}
    do Equation (2) to get penalty_{t,i}
end
for each domain name
    if domain name length > name_length
        do SM to get [suffix_i, count_{t,i}, penalty_{t,i}]
        if count_{t,i} > RFT
            penalty_{t,i} = penalty_{t,i} + penalty
        if penalty_{t,i} > max\_penalty
            penalty_{t,i} = max\_penalty
        if penalty_{t,i} > ST
            if suffix_i ∈ white list
                do extra human judgments
            else do suppress suffix_i
```

else continue
else continue
else continue
end

#Pseudo code for release process
for each suppressed suffix
    if penalty_{t,i} < ST
        do release suffix_i
    else continue
end

Table 1: Statistics of Suppressed Domains in 24 hours by SM/RFC algorithm.

<table>
<thead>
<tr>
<th>Suppressed Domains</th>
<th>Total Requests</th>
<th>Max QPM</th>
<th>Avg. QPM</th>
<th>Total Supp. Duration/ above RFT</th>
<th>Longest Supp. Duration(min)/ Supp. Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>www-geme456.net</td>
<td>6937090</td>
<td>98093</td>
<td>82675</td>
<td>173/173/1</td>
<td></td>
</tr>
<tr>
<td>vwwvv-garne456.com</td>
<td>3402737</td>
<td>141704</td>
<td>67173</td>
<td>200/200/1</td>
<td></td>
</tr>
<tr>
<td>emga456.com</td>
<td>2198819</td>
<td>305873</td>
<td>217092</td>
<td>77/77/1</td>
<td></td>
</tr>
<tr>
<td>vwwvv-garne456.com</td>
<td>1735165</td>
<td>258406</td>
<td>157519</td>
<td>97/57/2</td>
<td></td>
</tr>
<tr>
<td>214sp.com</td>
<td>844037</td>
<td>171130</td>
<td>113814</td>
<td>62/62/1</td>
<td></td>
</tr>
<tr>
<td>qamer456.com</td>
<td>564760</td>
<td>250660</td>
<td>13825</td>
<td>40/40/1</td>
<td></td>
</tr>
<tr>
<td>gamenc456.com</td>
<td>411593</td>
<td>203801</td>
<td>123961</td>
<td>30/30/1</td>
<td></td>
</tr>
<tr>
<td>18wsm.com</td>
<td>333293</td>
<td>21499</td>
<td>13608</td>
<td>131/120/2</td>
<td></td>
</tr>
<tr>
<td>qamea456.com</td>
<td>158022</td>
<td>135165</td>
<td>68252</td>
<td>17/17/1</td>
<td></td>
</tr>
<tr>
<td>000390.com</td>
<td>155954</td>
<td>22346</td>
<td>14584</td>
<td>82/82/1</td>
<td></td>
</tr>
<tr>
<td>at5.com.cn</td>
<td>115219</td>
<td>15710</td>
<td>12168</td>
<td>67/67/1</td>
<td></td>
</tr>
<tr>
<td>337070.com</td>
<td>51996</td>
<td>26739</td>
<td>19496</td>
<td>6/6/1</td>
<td></td>
</tr>
</tbody>
</table>

4.3 Results

We implement the SM/RFC algorithm by a dedicated server which parses the mirrored inbound/outbound traffic from one of CT’s main recursive nameservers. The server equips with an 8-core Xeon CPU and 8G memory, and runs Ubuntu Linux 10.04. The experiment lasts for complete 24 hours from 21:30 Jul 18 to 21:30 Jul 19 UTC+8. During this period, the nameserver received at an average rate of 25.6K QPS and a peak rate of 38.8K QPS from customer side, and it sent queries to the upper DNS hierarchies at an average rate of 2.7k QPS and a peak rate of 7.3K QPS. As introduced before, the algorithm parses the queries from the recursive server to the upper DNS hierarchies only, and for every one minute, it outputs the suppressed/released domains. The processing time for one-minute data calculations lasts only 2.3 sec at average.
Fig. 7: Suppression process for domain vwvvw-garne456.com.

It is a pity that before our measurements, CT has taken actions to block LoopcVPN from many aspects, which cause our measured node seeing no LoopcVPN-relevant queries coming in. In practice, we thus omit the name_length limit and raise the RFT to a higher level. The settings are as follows, \( RFT = 6000 \) QPM (namely 100 QPS), \( \text{penalty} = 100 \), \( \text{max\_penalty} = 51200 \), \( ST = 200 \), \( \text{decay\_half\_life} = 30 \) min. During the 24 hours, totally 29.34M distinct domain names are recorded, and 13 domains have ever been suppressed. Table 1 lists the details. The suppressed domains are arranged top-down following their number of requests in 24 hours, as shown in the 2nd column. For domain emga456.com, its maximum QPM is 305873, namely 5098 QPS high, and the average of all request frequencies above \( RFT \) is 217092 QPM or 3618 QPS. The 6 out of 13 domains are in similar forms of “game456.com”, and nearly 100% of their requests are for nxdomain. The extremely high volumes of requests for such domains are recently proved to be caused by botnet-related Trojan [11]. The rightmost column in Table 1 lists three numbers for each domain: the total suppression time, the longest suppression duration, and the suppression times in the 24 hours. Two domains have ever been released once (Supp. Times = 2), implying that the corresponding request frequencies decrease and remain normal levels for an enough period of time for the penalties to decay below \( ST \).

Domain vwvvw-garne456.com gets suppressed for 200 minutes in 24 hours, which is the longest among the others. Fig. 7 demonstrates the whole suppression process for this particular domain.

5 Conclusion and Future Work

LoopcVPN is a DNS tunneling software that can bypass the provider’s WiFi authentication by encapsulating data in DNS format. Although the authentication leak widely exists, the software is far from popular as what we had thought.

We propose a set of counter measures to inhibit LoopcVPN from mechanism perspective, and we foresee the potential threat of such kind of IP-over-DNS software to the provider recursive nameservers since the amount of queries for names with frequently changing prefixes will pose an undesirable burden on DNS systems. SM/RFC algorithm is proposed to automatically identify and suppress/release such abused domain names,
and shows good performance on the operational DNS nodes. To investigate such kinds of DNS misuse based on the deployment and improvement of the SM/RFC algorithm will be our major work in the near future. We last emphasize that it is vital for Internet providers to keep an eye on the queries from their recursive nameservers to the upper DNS hierarchies, which has been overlooked for quite a long time. This is where the wild DNS traffic lies in.

**Acknowledgements.** W. li, H. Xu, J. He, C. Shen, G. Yu, H. Pan, L. Chen of Chinatelecom helped with the packet trace capturing and analysis. L. Long from DNSII helped with the deployment of the SM/RFC algorithm.

**References**

2. Iodine. [http://code.kryo.se/Iodine/](http://code.kryo.se/Iodine/)

**Author Biography**

**Ziqian Liu** joined Chinatelecom Corp. in 2007. He has been the principal network engineer of ChinaNet (AS4134) since 2008. He is also the chief architect and the principal operator of the DNS system for Chinatelecom Next Carrier Network (CN2, AS4809). Now he serves as the Director of the Network Security Operation Center (SOC) of Chinatelecom Headquarter.

Ziqian recived Ph.D. from Beijing Jiaotong University. He has been a visiting scholar at CAIDA in 2006, playing with the first two sets of DNS-OARC data of root servers. His interests includes routing, topology, DNS and network security-related stuff. He had dreamed to be a scientist, but now he is on his way trying to become an enginner and manager with scientific insights.
Design of robust DNS adaptable to dynamic Ad hoc networks

Younchan Jung\textsuperscript{1} and J. William Atwood\textsuperscript{2}

\textsuperscript{1} School of Information, Communications and Electronics Engineering
Catholic University
Puchon, Kyonggi-do, Korea

\textsuperscript{2} Department of Computer Science and Software Engineering
Concordia University
Montreal, Quebec, Canada

Abstract. The Domain Name System (DNS) is indispensable in today’s Internet and will be equally important to future military networks based on mobile nature of Ad hoc networks. However, the Ad hoc nature of future military networks makes it much more challenging to plan for, deploy and update DNS. In this paper, our approach to providing dynamic DNS focuses on the use of smart phones having two address (e.g., IP address and mobile phone number (PN)) so that the parent name server (NS) may reach its slave NSs, no matter where they roam, no matter what IP address they are currently using. We also propose a protocol for ensuring that a slave NS is kept up-to-date, and evaluate its performance.

1 Introduction

The Domain Name System (DNS) is a fully distributed system that provides network applications to translate from logical user friendly names into an IP address [1–4]. The essential role is to send queries to a distributed database consisting of hierarchical Domain Name Servers [5, 6]. DNS is indispensable in today’s Internet and will be equally important to future military networks where the dynamic, bandwidth constrained and mobile nature of future military Ad hoc networks makes it much more challenging to plan for, deploy and update DNS [7]. Morera and McAuly [7] nicely explain the dynamic characteristics of the future military networks as follows: a) any node, including servers, can be mobile; b) links have variable bandwidth; c) server and network connectivity can be intermittent due to link instability; d) networks may be split into independent islands and whole parts of the network may be permanently lost. As shown in Fig. 1, the most important part of the future military tactical and combat network can be the mobile Ad hoc network (MANET), which is an autonomous system of mobile nodes with all routing within the domain occurring in a highly dynamic manner. A MANET can have a connection to a remote MANET across the dynamic, bandwidth constrained Red networks, together with the secure overlay networks, that is, Black networks (especially those created by IPsec tunnels) [8, 9]. (See [8] for the definition of Red and Black networks.) This Ad hoc nature of future military networks makes it much more challenging to plan for, deploy and update DNS.
Even in existing commercial networks, the limitations in the DNS have lead to proposals for more robust, secure, flexible and dynamic alternatives [10–13]. In 2006, a “smart phone” combined the functions of a PDA and a mobile phone [14] and had both Wi-Fi and mobile network access. Today, a “smart phone” is considerably more powerful, and has additional network interfaces. This makes it highly likely that smart phones will be used as components of the MANET. This adoption of smart phones in MANETs makes it possible to design a more inclusive solution for a robust, secure and dynamic DNS. The adoption of smart phones in MANETs enables a DNS design in which each Name Server (NS) is identified by a phone number, rather than by an IP address. The NS can be reached, no matter where it roams, no matter what IP address it is currently using. Mobility control is essential in today’s mobile networks. The well-proven system for handling mobility is based on queries to a distributed database consisting of Home Location Register (HLR) and Visiting Location Register (VLR) [15]. So in this paper, we use NSs with combined functions of Wi-Fi and mobile network access.

Section II describes DNS for future mobile Ad hoc networks. Section III gives our recommended design for DNS update in future military networks, followed by conclusions in section IV.
Fig. 2: Name space recommended for military networks
2 DNS for future mobile Ad hoc networks

2.1 Name space

DNS is integral to the existing Internet, converting user-friendly names to IP addresses that routers can use to reach the destination. DNS is implemented using a distributed database of Resource Records (RRs) stored in Name Servers (NSs). Applications access DNS through a local client resolver process that sends a DNS request packet to the configured NS. Typically, DNS records are indexed using unique DNS names. To provide a scalable and manageable solution in military networks, Fig. 2 shows how the name space is constructed as a tree. A domain is defined as a complete sub-tree that cuts the domain name tree in one place. Domains are named based on the root of their sub-tree. Zones are any contiguous part of the domain name tree that can cut the tree in one or more places and are under a particular authority [7]. Every node is uniquely identified by a specific domain name, which contains the names of each node in the tree from the named entity to the root. To provide scalability each NS knows a limited number of RR and an even smaller number of other NSs.

2.2 Configuration of zones

We distinguish two Server Types: Master (Primary) server and Slave (Secondary) server. A Master Server (MS) is an authoritative server for the domain for which it is responsible. A Slave Server (SS) is similar to a caching server, with additional properties, which will be described below. A Master Server manages any changes within the zone for which it is authoritative; it has read and write permissions. A Slave receives RR from either the master or another slave of the zone; it has read only permission.

A military domain can be divided into its zones. Fig. 3 gives an example of dividing a br1 military domain into three level zones. The advantage of multi-level zones, which constitute a layered architecture, is that data can be replicated in a more hierarchical manner. Each part of the domain becomes independent of the other parts if they are not in relationship of vertically adjacent layers, which can reduce complexity on data transfer over the Ad hoc nature of future military networks.

In a normal Internet environment, a single processor hosts the authoritative Name Server for a domain or a zone. DNS NSs can be distributed in a zone in different ways. For example, a Master or Slave NS could be in the zone placed at the boundary between the Black network and Red networks (e.g., br1 zone). A couple of Slave NSs could be placed in the intermediate zones (e.g., batt3 zone). Given the requirement that a military MANET must withstand network partitioning, the Master Server will be the authoritative server for a domain, while the Slave Servers in the MANETs will need to have a full copy of the data held by the authoritative server, which implies that each mobile Ad hoc station has the role of a slave NS in the lowest level zones (e.g., co3 zone).

2.3 The effect of smart phone adoption for DNS name servers

We assume that the possibility of smart phone adoption by the personnel using future mobile Ad hoc networks is very high. Demand for advanced mobile phones in future
military operations will not stop. Future military Ad hoc stations would operate over both wireless networks (e.g., 3G or 4G networks) and IP-based networks. In this paper, we assume that combined functions with Wi-Fi and mobile network access are needed to meet the requirements on stations in future mobile Ad hoc networks. This means that DNS NSs in this paper will have at least two address: an IP address and a mobile phone number (PN). Then, NSs can send and receive SMS messages to devices with mobile networks such as 3G or satellite networks. SMS communication has the advantage that it will dynamically detect the location of the child NSs because in Ad hoc networks, where DNS servers can be mobile, robust operation requires that the linkages from the parent NS to its child (slave) NSs be automatically and securely established as network topology changes.

The fact that the NSs are implemented on a smart phone permits the Child NSs to be identified by a phone number, rather than by an IP address. The parent NS can reach its Child NSs, no matter where they roam, no matter what IP address they are currently using.

The parent NS has a DNS application that allows its child applications to request and receive RR data using HTTPS (HyperText Transfer Protocol Secure). DNS name servers can cooperate with SMS (Short Message Service) server. That is, NSs in the Ad hoc networks operate with interfaces to two different networks: Wi-Fi networks and mobile networks. Because roaming is basic in mobile networks, we argue that use of SMS would provide more robust operation in maintaining linkages between parent NS and its child NSs even in the dynamically changing Ad hoc network environments.
Then, each NS can be used with an SMS recipient for receiving the SMS message as well as a DNS application for sending SMS and updating RRs.

2.4 DNS linkages

The Domain Name System (DNS), which has provided DNS service on the Internet for the past 20 years, has DNS linkages mostly manually constructed based on IP addresses of static DNS NSs. However, in ad hoc networks, where DNS servers can be mobile (that is, servers also change their IP address), robust operation requires that these linkages be automatically and securely configured and reconfigured as network topology changes. However, new protocols are needed to dynamically configure such linkages. We propose that DNS be configured as follows. We need one platform NS per one zone. Each platform NS functions as 1) a main slave of the level above it and 2) a main parent NS of the level below it in the domain hierarchy. Because Ad hoc networks use wireless links with intermittent connectivity, it is necessary to have a couple of additional slave NS per zone. In the zone of the lowest level in the domain hierarchy, wireless stations communicate directly without an access point. This means stations function as communication nodes that operate in the pure Ad hoc mode. We propose to maximize the replication of information for the slave NSs in the zone of the lowest level in order that they may receive faster response to the DNS queries. Thus, having ‘Child Synchronization’ with each wireless station as a slave of a platform NS of its same level zone would provide good robustness because each wireless station already has a copy of the RR.

3 Recommended approach

3.1 RR replication

Under the assumption of relatively few updates, it is advantageous to maximize the replication of information [7]. Thus, using ‘Child Synchronization’ with each NS as a slave of the level above it in the domain hierarchy would provide good robustness. Using a multi-level replication database would provide even greater robustness and allow more updating of RRs. Then, the slave finally would have a copy of the RR of the master through the Parent-Child link to the slave of the level above it in the domain hierarchy. This section describes our recommended approach for RR update in the Child from the Parent NS (that is, the platform NS of the level above it).

As shown in Fig. 5, the parent NS’s state alternates between Active mode and Quiet mode [8]. The parent NS goes to Quiet mode, in which only Hello messages are periodically sent, to manage DNS linkages from a certain platform NS to all slave NSs as Child NSs of the level below the platform NS in the DNS hierarchy. The Quiet mode results in a considerable savings in energy consumption, which is important in mobile devices.

A quiet mode state is defined as the period when no RRs are changed. Figs. 6 and 7 show how to manage DNS linkages in mobile Ad hoc environments. DNS linkage management occurs during Quiet mode in which Hello SMS message is sent every T
Fig. 4: Child Synchronization with platform NSs in the domain hierarchy
seconds from the Parent NS to its Child NSs. That is, Hello Timer interval is $T$ seconds. The Hello message contains Parent NS’s IP address. When Child NS receives the Hello SMS message, it uses the HTTPS HEAD method instead of the more common GET and POST methods to send an acknowledgement to its Parent NS [16]. HEAD works the same as GET, except the web-server will only return the HTTP headers and not the entire web-page. The Child NS only needs to see the header, so using the HEAD method significantly reduces the amount of data that needs to be transferred. So the Hello mechanism in Quiet mode aims to monitor and manage configuration of Parent-Child NSs (that is, DNS linkages from Parent NS to every Child NS are periodically refreshed based on their IP addresses). This Quiet mode operation is important because it enables the Parent NS not only to inform its Child NSs of its IP address but also to identify each IP address of the Child NSs. So the Parent NS refreshes DNS linkage information every $T$ seconds. Considering that the platform NS in each level in the DNS hierarchy is responsible for the Parent NS, hierarchical linkage managements will be made.

Figs. 8 and 9 show how to update RR data in a secure way. When the parent NS goes to Active mode, it begins with Hello plus RRUpdate messages sent to its Child NSs. The Active mode in a certain parent NS starts when any RRs of the NS are updated or it has been reestablished. During Active mode, Hello plus RRUpdate SMS messages are sent just once to every Child NS associated with the parent NS. While the Hello message contains Parent NS’s IP address, RRUpdate contains ‘nonce’.
Fig. 6: Refreshment of DNS linkage information every T seconds

Fig. 7: DNS linkage management
Fig. 8: RR update procedure

Fig. 9: Downloading RR data from Parent NS to Child NS
Fig. 8 shows the recommended protocol to update RRs in the child NS. Replication communication consists of message pairs of RRUpdate and UpdateConfirm. The detailed description of the protocol is presented in Fig. 9. The parent NS begins to send the SMS message, which contains its IP address and temporally generated nonce value, to the child NS with phone number (PN) = nnn. The nonce value will be used as directory for RR database to be replicated. When SMS recipient in the child NS detects a new RRUpdate request, it automatically reports this request by informing its DNS application with [IP address + nonce]. Then, the DNS application in the child NS sends the HTTPS request message to the parent NS. ‘GET’ command of the request line can be ‘GET /somedir/nonce’. That is, the nonce value will be used to identify the location of the RR to be replicated in the parent NS whose address is ‘IP address’. Also the nonce value in the HTTPS request line assures that this is not a replay of an old message because the nonce value will never be reused between the parent NS and child NS. Then, the DNS application in the child NS (as HTTPS client) receives the response message containing a copy of the RR in a secure way owing to the HTTPS protocol. When this protocol is used, the parent NS sends its certificate, which includes its RSA public key. Then, the child NS validates the certificate, authenticates the parent NS, and obtains the server’s public key. The public key enables the generation of session keys used to encrypt the RR data. So they will exchange encrypted RR data for replicating RR.

3.2 Required bandwidth for the Hello mechanism

The configuration load for DNS linkage refreshment occurs when the Hello mechanism operates every $T_{HELLO}$ (Hello timer interval in Quiet mode) seconds. The required bandwidth for the Hello mechanism can be limited to a negligible degree if we choose $T_{HELLO}$ value of 30 seconds [8]. It is because one HTTP session (‘HTTPS request: HEAD method’ and ‘HTTPS response: HTTPS headers’) is needed every 30 seconds for a pair of the Parent NS and Child NS.

3.3 Load for the RR data replication mechanism

In a strict sense, it is very difficult to obtain quantitative results that can answer the questions: ‘How much bandwidth is needed to receive a copy of RR data from the Parent NS?’ and ‘What are the average and worst case latency for the Child NS to update a limited portion of RRs?’ Also the answers will depend on the operational scenario of mobile networks, that is, the average number of records the NS needs to hold and the average rate of IP address change of servers are difficult parameters to obtain.

In this paper, we focus on an example of RR update in the Child database. To limit the use of wireless bandwidth when there is a significant number of updates, it is desirable to reduce the average size of replication data for a update session (HTTPS response: ‘A copy of RR data’) shown in Fig. 9. A good compromise for a dynamic military network, with using wireless networks with intermittent connectivity, is to make the database for the Child and the Parent NS synchronized at any time. In our RR data replication mechanism, they operate in a master-slave manner. The Parent NS determines RR data to be replicated as a master. This means that only the parent NS has...
complete authority over its Child NSs. Then, the RR database for the slave and the master are always synchronized even though the Parent NS sends a limited portion of RRs that has changed. So our partial RR data replication mechanism has advantages in regards to latency and overhead. Overhead is low since the average bandwidth to perform a update session can be reduced. Also, the average latency to complete a update session is reduced. Though the Child NS uses incremental updates only on receiving a small part of RR database, there is no possibility that the database for the Child and the Parent were not synchronized since the recommended RR data replication mechanism operates based on a master-slave manner.

4 Conclusion

Under the assumption that smart phones will be in widespread use in military MANETs, we have proposed a methodology and protocol exchanges for DNS updates that takes advantage of the special characteristics of smart phones. The methodology makes use of the dual interfaces of smart phones to ensure secure, correct updates in spite of the mobility of the name servers, and accounts for the intermittent connectivity of the smart phones.

We have given the details of the protocol exchanges and demonstrated that the communications overhead is expected to be low. Thus, the methodology represents a convenient way to inexpensively keep tracks of mobility and resource record changes in military MANETs.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0015200) and by the Natural Sciences and Engineering Research Council of Canada, through its Discovery Grants program.

References

Robust DNS adaptable to dynamic Ad hoc nets


Author Biographies

Younchan Jung received the M.S. and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST) in 1991 and 1994, respectively. From 1980 to 1989, he worked as a senior research engineer in the Agency for Defense Development (ADD), Korea. He is currently a Professor at the School of Information and Communications Engineering in the Catholic University of Korea. In 2002, he was Visiting Scientist and Lecturer, Computer Science, Concordia University, Canada. Also he was Visiting Scientist at the School of Engineering, Blekinge Institute of Technology, Sweden in 2008. His research interests include Wireless VoIP QoS, Mobile network Security and Military Ad hoc Network Design.

J. William Atwood graduated from McGill University in 1963 with a Bachelor of Engineering, from the University of Toronto in 1965 with a Master of Applied Science and from the University of Illinois at Urbana-Champaign in 1970 with a Doctor of Philosophy, all in Electrical Engineering. He is a Distinguished Professor Emeritus in the Department of Computer Science and Software Engineering at Concordia University, Montreal. His research interests include the design, analysis, validation and deployment of protocols for secure multicast, secure routing and secure DNS.
## Author Index

<table>
<thead>
<tr>
<th>Author Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atwood, J. William</td>
<td>116</td>
</tr>
<tr>
<td>Deccio, Casey</td>
<td>54</td>
</tr>
<tr>
<td>Fukuda, Kensuke</td>
<td>27</td>
</tr>
<tr>
<td>Gijsen, Bart</td>
<td>7</td>
</tr>
<tr>
<td>Hardaker, Wes</td>
<td>65</td>
</tr>
<tr>
<td>Jamakovic, Almerima</td>
<td>7</td>
</tr>
<tr>
<td>Janoušek, Tomáš</td>
<td>93</td>
</tr>
<tr>
<td>Jung, Younchan</td>
<td>116</td>
</tr>
<tr>
<td>Koç, Yakup</td>
<td>7</td>
</tr>
<tr>
<td>Lamb, Richard</td>
<td>77</td>
</tr>
<tr>
<td>Liu, Ziqian</td>
<td>103</td>
</tr>
<tr>
<td>Mikle, Ondrej</td>
<td>93</td>
</tr>
<tr>
<td>Mitamura, Takeshi</td>
<td>27</td>
</tr>
<tr>
<td>Morrison, Wayne</td>
<td>65</td>
</tr>
<tr>
<td>Mundy, Russ</td>
<td>65</td>
</tr>
<tr>
<td>Sato, Shinta</td>
<td>27</td>
</tr>
<tr>
<td>Slaný, Karel</td>
<td>93</td>
</tr>
<tr>
<td>Story, Robert</td>
<td>65</td>
</tr>
<tr>
<td>Surý, Ondřej</td>
<td>93</td>
</tr>
<tr>
<td>Suresh, Krishnaswamy</td>
<td>65</td>
</tr>
<tr>
<td>Veselý, Ján</td>
<td>93</td>
</tr>
</tbody>
</table>