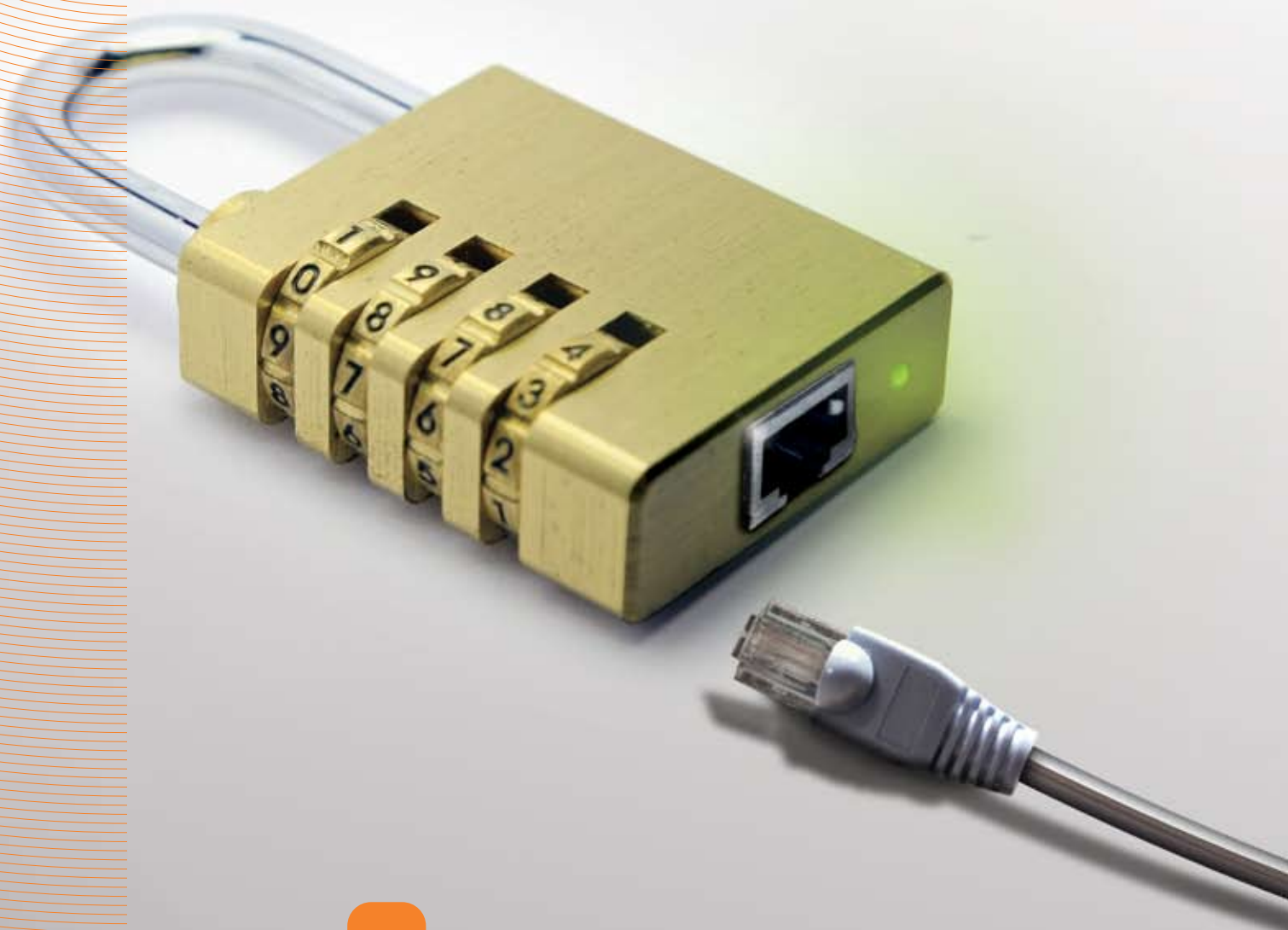


SURF

NET



HARDENING THE INTERNET

The impact and importance of DNSSEC

TABLE OF CONTENTS

MANAGEMENT SUMMARY	5
READING GUIDE	5
1 THE DOMAIN NAME SYSTEM: ROAD SIGNS ON THE INTERNET	7
1.1 Purpose of DNS	7
1.2 History	7
1.3 The solution: the Domain Name System	7
1.4 Resolving an IP-address	8
2 WHY THE DOMAIN NAME SYSTEM IS INSECURE	11
2.1 Introduction	11
2.2 Changing a road sign	11
2.3 Replacing the road sign company: the Kaminsky attack	12
2.4 Patching against Kaminsky	12
2.5 Example: breaking the PiggyBank	13
2.6 How DNSSEC solves this problem	13
3 WHAT IS DNSSEC?	15
3.1 Definition	15
3.2 Security by signing	15
3.3 Chains of trust	15
3.4 Trust anchors	16
3.5 Islands of trust	16
3.6 Alternatives	16
4 IMPLEMENTING DNSSEC	18
4.1 Introduction	18
4.2 Timeline	18
4.3 Strategy for rolling out DNSSEC in an organisation	19
5 CONCLUSIONS	21
5.1 DNSSEC provides trust in DNS responses	21
5.2 There is no alternative in the long term	21
5.3 Managing DNSSEC is different from managing DNS	21
5.4 You need to do it	21
6 ACKNOWLEDGEMENTS	22
7 ENDORSEMENTS	23
APPENDIX A TERMS AND ABBREVIATIONS	25
A.1 Terms	25
A.2 Abbreviations	26
APPENDIX B REFERENCES	27
B.1 Further reading	28
APPENDIX C TECHNICAL DISCUSSION OF DNSSEC	29
C.1 Introduction	29
C.2 DNSSEC for resolvers	29
C.3 DNSSEC for authoritative name servers	30
APPENDIX D OUTSOURCING	34
D.1 Introduction	34
D.2 Division of responsibilities	34
D.3 Protecting against vendor lock-in	34
APPENDIX E ALTERNATIVES	35
E.1 Introduction	35
E.2 The DNS arms race	35
E.3 TSIG and SIG(0)	35
E.4 DNScurve	36
E.5 IPsec	36
E.6 SSL or TLS	36



**MANAGEMENT SUMMARY
READING GUIDE**

MANAGEMENT SUMMARY

The Domain Name System (DNS) is one of the basic building blocks of the Internet. Vulnerabilities in the DNS system can affect the security of the entire Internet.

DNS converts logical names for resources on the Internet to IP addresses, making it possible for a user to type a logical name (such as `www.surfnet.nl`) rather than an IP address (such as `194.171.26.203`).

It has been known for a long time that DNS has a number of vulnerabilities in its basic design. However, a recent exploit (the Kaminsky attack, see [5]) has shown how easy it is to abuse these vulnerabilities, leading to a renewed sense of urgency within the Internet community.

An extension to DNS has been developed to address its vulnerabilities: DNSSEC¹. Although DNSSEC has been available for some time now, deployment has not yet taken off on a large scale. This, however, is changing rapidly, as the need to secure DNS has become more apparent by the Kaminsky attack. This specific attack may have been mitigated, but others are likely to follow.

While DNSSEC deployment can be complex, tools are now available which handle most of the complexity, resulting in a more straightforward implementation. It still requires an organisation to make changes to its systems and to its processes though.

In the long run, every organisation will have to implement DNSSEC. However, given the stage of developments, only organisations with relatively high levels of technical expertise, like universities, research centres, banks, ISPs and some top-level domain administrators are currently implementing it. At the time of writing, a number of top-level domains² have implemented DNSSEC, while many others are in the process of implementation³. Any organisation with any kind of IT infrastructure should at least be planning an implementation at some time in the near future.

READING GUIDE

This white paper is aimed at decision-makers, responsible for the IT infrastructure, in non-commercial institutes and commercial enterprises. It explains, at an abstract level, how the current DNS is vulnerable and what organisations should be doing about it.

The first five chapters give a high-level overview of the following subjects: what the Domain Name System is and why it was introduced; why it is vulnerable to attacks by design; how DNSSEC can address these vulnerabilities, what DNSSEC is, a strategy for deploying DNSSEC and finally the conclusions of this paper.

The remainder of the document consists of several appendices that contain more detailed technical information about DNSSEC; these appendices are aimed at DNS administrators to give them a general insight into the technical implications of implementing DNSSEC within their infrastructure.

¹ DNSSEC: Domain Name System Security Extensions

² Currently (December 2008) the country top-level domains of Sweden, Brazil, Bulgaria, the Czech Republic, Puerto Rico, as well as the .museum top-level domain and the ENUM domain.

³ For example, the US government top-level domain .gov will implement DNSSEC as of January 1, 2009.



1

**THE DOMAIN NAME SYSTEM:
ROAD SIGNS ON THE INTERNET**

1 THE DOMAIN NAME SYSTEM: ROAD SIGNS ON THE INTERNET

1.1 Purpose of DNS

To reach a computer (or any other resource) on the Internet, some form of addressing is needed. The address has to be unique, so the network can route your requests to the correct destination. For this purpose, every computer connected to the Internet has an IP address – a numerical identifier which routers and other networking equipment can understand.

IP addresses, however, are meant to be used by machines, not people. They are hard to remember, and have no logical structure people can relate to. For that reason, Internet resources usually have a logical name in addition to an IP address.

In order to translate logical names into IP addresses, a translation system is needed. This translation system is called the Domain Name System (DNS). DNS translates logical names, such as **www.surfnet.nl**, into IP addresses, such as **194.171.26.203**⁴.

1.2 History

To understand the workings of the Domain Name System, a brief explanation of its history is needed.

When the Internet was created, each node on the network needed to have an address. For this purpose, IP addresses were defined.

Early adopters of the Internet soon ran into the problem of missing a logical structure to link computer systems to IP addresses and came up with a simple solution: they created a file that mapped an IP address to a logical name. This file was called the 'hosts' file as it contained a list of hosts connected to the network. An example of such a hosts file is shown in Table 1.

194.171.26.203	www.surfnet.nl
194.171.26.204	tag.surf.nl
194.171.26.205	redactie.surfnet.nl
...	...

Table 1 - Example of a hosts file

The hosts file was maintained and shared between the system administrators of the computer systems that were connected to the early Internet. But when the Internet started to expand, it soon became clear that this solution would not scale to fit the need of the many new users connecting to the network.

⁴ Although all the examples in this paper use IPv4 addresses, the issues are exactly the same for IPv6.

1.3 The solution: the Domain Name System

In response to the fast expansion of the Internet and with it an increase in IP addresses, the Domain Name System – or DNS for short – was introduced in 1983.

Domain names, and the DNS system, are hierarchical: there is a root domain (represented by a single dot "."), a set of top-level domains, such as **.com** or **.nl**, and any number of levels under these top level domains. Figure 1 shows an example of the domain name hierarchy.

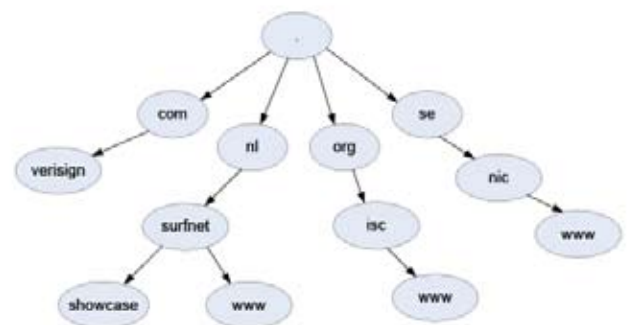


Figure 1 - Example of the DNS hierarchy

The DNS translates human-friendly domain names (such as **www.surfnet.nl**) into IP addresses used by computers to look up its destination.

Information for a certain domain is stored on a computer system, a so-called *authoritative name server*. This name server manages what is called a "zone". A zone contains records, mapping names to resources – for instance: the zone for **surfnet.nl** contains a record that maps the name **www** to the IP address **194.171.26.203**. Each entry in the zone is a domain name.

A name server may redirect entities that are requesting information for a resource within a zone to another name server. This is called delegation. For instance: the name server for the **.nl** zone can delegate management of information for the **'surfnet.nl'** zone to SURFnet's name server.

The zone containing the top of the hierarchy (the "." domain) is called the root zone. The entries in this zone are the top-level domains. At the moment there are two types of top-level domains: generic top-level domains (such as **.com** for companies, **.edu** for educational institutions, etc.) and country code top-level domains (such as **.nl** for The Netherlands, **.cn** for China, etc.).

1.4 Resolving an IP-address

What has not been discussed yet is how a user can query the DNS to find the address for a given name. To do this, a so-called *resolver* is used. If a resolver is asked to find the address for **www.surfnet.nl**, it queries the DNS top-down. The diagram below shows how this is done:

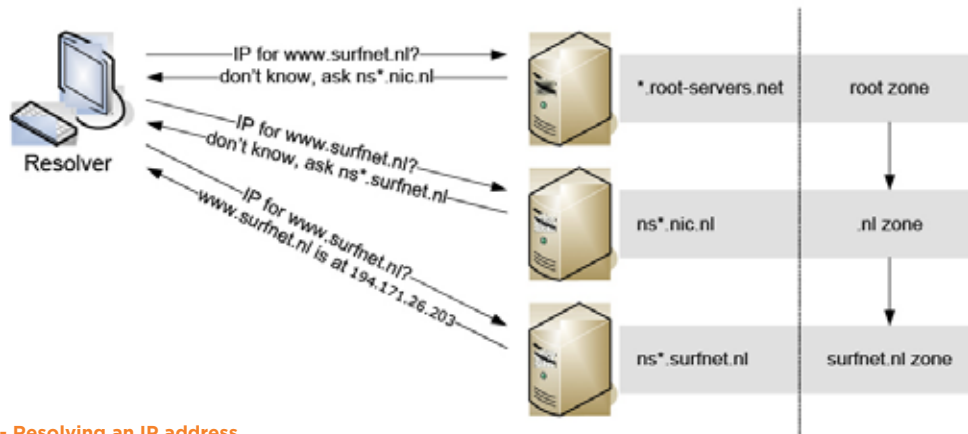


Figure 2 - Resolving an IP address

The diagram shows how a resolver goes about finding the IP address:

1. The resolver asks one of the root name servers if it knows the IP address for **www.surfnet.nl**. The server tells the resolver that it doesn't know, but that the resolver can ask one of the name servers for the **.nl** domain. It also provides the names and addresses of these **.nl** name servers.
2. The resolver asks one of the name servers for the **.nl** domain if it knows the IP address for **www.surfnet.nl**. The server tells the resolver that it doesn't know, but that the resolver can ask one of the name servers for the **surfnet.nl** domain. It also provides the names and addresses of these name servers.
3. The resolver asks one of the name servers for the **surfnet.nl** domain if it knows the IP address for **www.surfnet.nl**. It responds by telling the resolver that **www.surfnet.nl** is at **194.171.26.203**.

This only leaves the question: "how does the resolver know where to find the root name servers?". The answer to this is simple: the 'boot strap' for a DNS resolver is the so-called *hints file*. This file contains a list with the names of the root name servers together with their addresses. Regular updates of this file are made available by InterNIC on their website⁵.

⁵ <http://www.internic.net/zones/named.root>

Instead of having a fully functional resolver on each client, it is common practice to expedite the resolving process to a *recursive name server*. This recursive name server allows clients to request any name to be resolved. It then does all the 'hard work' for the client, resolving the name. An additional bonus is that the recursive name server can maintain a cache of answers that it has received (thus becoming a *recursive caching name server*). The answers can be re-used when another client asks for the same name to be resolved. All answers provided by name servers have a *time-to-live*; when this period expires the answer is removed from the cache.

In this case, the client has only a very simple resolver (called a *stub resolver*) that cannot perform the recursive lookups itself. The stub resolver is configured to talk to a recursive (caching) name server. Figure 3 below shows a recursive caching name server in action.

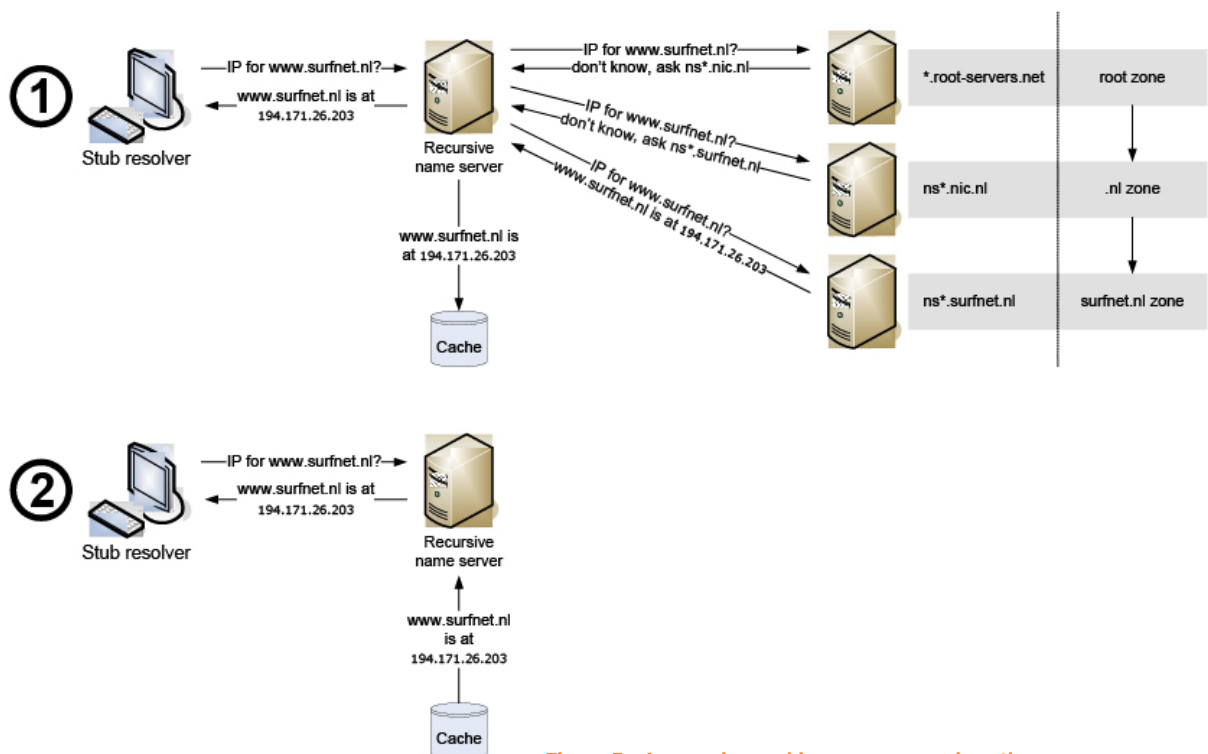


Figure 3 - A recursive caching name server in action

The first time a client requests the address for **www.surfnet.nl** (1) the server goes through the entire resolving process. The second time (2), however, it can re-use the address that has already been stored in its cache.

A recursive caching name server actually stores a lot more information in its cache than just the address of **www.surfnet.nl**. During the process of resolving **www.surfnet.nl**, it also encountered the names and addresses of the name servers for the **.nl** zone and for the **surfnet.nl** zone. As a result, it now knows where to go for other names within the **surfnet.nl** zone, so that a subsequent query for **showcase.surfnet.nl** can be directed straight to the **surfnet.nl** name server.



2

**WHY THE DOMAIN NAME
SYSTEM IS INSECURE**

2 WHY THE DOMAIN NAME SYSTEM IS INSECURE

2.1 Introduction

The Domain Name System was designed during the early years of the Internet. During this era all users were academia, military organisations and computer enthusiasts, who – in general – could be trusted not to abuse the network.

All the above implies that security was not one of the main design goals of the Domain Name System. As a consequence, there are vulnerabilities in the system (some of which are even by design). The most significant vulnerability results from the fact that name servers query each other, without a method to verify that the results are genuine, or even originate from the proper name server. This allows for a type of attack called *cache poisoning*.

To illustrate the problem of cache poisoning, an analogy is used. An apt analogy for DNS is the use of road signs. Similar to road signs which point you in the direction of a geographical location if you are looking for an address, DNS points you in the right direction if you are looking for a specific Internet address.

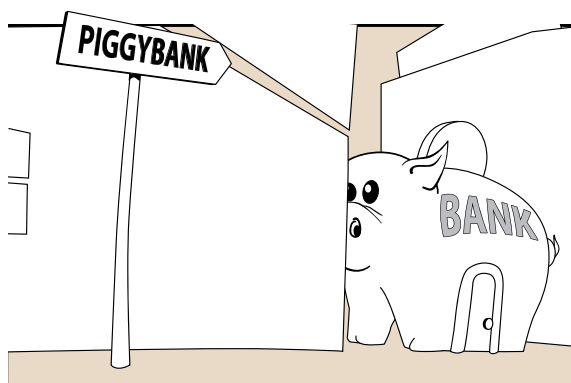


Figure 4 - A road sign showing where PiggyBank can be found

2.2 Changing a road sign

As was already described in the previous chapter, it is common practice for clients to make use of a recursive caching name server that does all the hard work of resolving, and keeps the answers in a cache so they can be re-used for other clients. Most LANs have one or more recursive caching name servers. Such a server is contacted by clients on the LAN, and thus its cache is an ideal tool for reducing the strain put on the Internet's DNS infrastructure. This is a win-win situation: clients get their results more quickly, and downstream DNS servers have less work to do.

But imagine that it would be possible to fool the recursive caching name server into accepting a wrong answer for a query it has sent downstream. This incorrect answer would then end up in its cache and be served out to all clients requesting the same address until the time-to-live of the wrong answer runs out. As a result, users could be sent to the wrong bank, or

to a website with malware on it; their e-mails could be sent to the wrong address and even their telephone calls could be redirected.

This would be the equivalent of replacing a road sign by a new one pointing in another direction. And if it's done correctly, the worst thing is that users can't tell the difference between the real and the fake road sign.

This scenario is exactly what is possible in the Domain Name System at the moment. When a resolver sends out a request, it is possible for an attacker to send wrong answers to the resolver. If the attacker serves up an acceptable answer quickly enough, then the resolver will accept that answer⁶. The real answer will be discarded, since the resolver has already received a response to its request.

Now this attack is only possible if the attacker either intercepts the original request or generates the request himself, and if he succeeds in giving the resolver a spoofed answer before the authoritative server does.

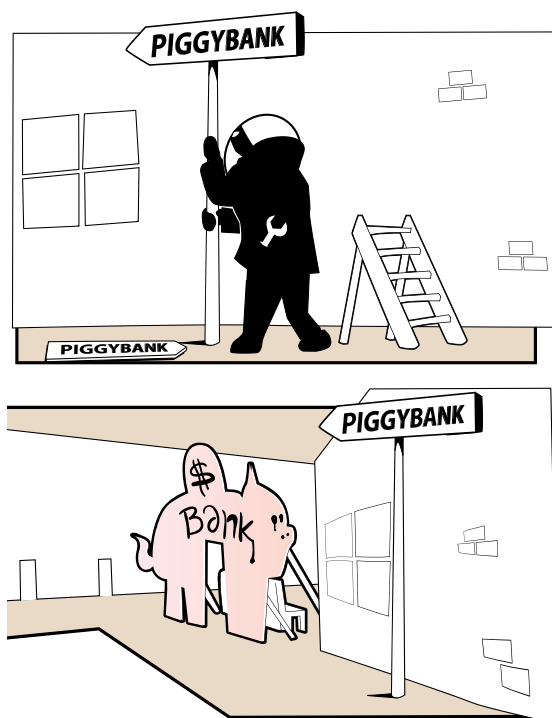


Figure 5 - Changing a road sign

⁶ Technically, this involves getting some numbers right, an excellent – albeit very technical – explanation is given in [15]

2.3 Replacing the road sign company: the Kaminsky attack

The trouble doesn't end there. Going back to the road sign analogy: what if it would be possible to trick the local council into hiring a different company for putting up road signs?



Figure 6 - Replacing the road sign company

Unfortunately, this is possible in the Domain Name System. This is known as the Kaminsky attack. To explain this attack, a little bit more detail is needed.

In the previous chapter the process of resolving was explained. In this process, name servers can tell a client that they don't know the answer but redirect the client to look elsewhere. The information that is supplied by the name server is so-called 'authority' information. The name server says "I'm not authoritative for the domain you're looking for but this other server is". It then very conveniently supplies both the name and the address for this authoritative name server as part of the answer to the query. This information is commonly known as "glue". In effect, the answer to every query consists of 3 parts:

- The answer to the query (can be empty if the query cannot be answered)
- Authority information (who is authoritative for the domain being queried)
- Additional information (the address information for the authoritative servers)

It is easy to see how this can be abused: the attacker can try to answer before the name server to which the request was sent. If the attacker succeeds, he can supply falsified "glue" referring all further requests for the domain that is being subverted to his own name servers (i.e. he replaces the "additional information" section of the reply with information of his own).

Three things are particularly troublesome about this attack:

- The attacker can carry out this attack at any time (instead of having to wait for cached replies to time out). He simply queries the recursive caching name server he wants to subvert for (non-existent) host names he knows are likely not to be present in the cache. He knows that the server

will forward this request because the name is not in the cache, giving the attacker the opportunity to insert his own incorrect answers which will then automatically be entered in the cache.

- This attack subverts a whole domain instead of just one host name. And an attacker will typically set a long time-to-live on the falsified answer, to make sure it stays in the cache for a long time. Because the whole domain has been subverted all traffic to it can be redirected, including e-mail.
- This attack cannot be mitigated by protecting your web site using SSL (https). As the example in section 2.5 will show, it is trivial for an attacker to redirect users to an SSL secured site that may seem completely valid from a user's point of view.

2.4 Patching against Kaminsky

The Kaminsky attack was made public only after a patch was available for all of the common name server software, and a large number of caching name servers had already been patched. This is, however, not a solution that works for the long term since it is part of an arms race between system administrators and attackers (see also section E.2). The basic flaw in the Domain Name System – that there is no way to ensure that answers to queries are genuine – remains.

Why patching is not the solution is best illustrated by some numbers:

- Unpatched servers can be poisoned within as short a time as 3 seconds
- Research performed by CZ.NIC – the top-level domain registrar for the Czech Republic's .cz domain – has shown that it is possible to poison a fully patched server within one to eleven hours

So even though patching helps delay an attack – giving administrators a warning window in which to detect the attack, for instance using traffic analysis – it shows that an attack is still viable. And since an attacker has "all the time in the world" there are many ways in which an attacker can mask the attack, for instance by performing it in small bursts from many different hosts.

2.5 Example: breaking the PiggyBank

The example described below shows how insidious the Kaminsky attack actually is. It also demonstrates that even an SSL secured web site is not safe.

Assume that there is a bank called PiggyBank. This bank provides Internet banking services to its customers. The bank provides a link to its online banking portal through its web page on

<http://www.piggybank.nu>. Normally, when a user clicks on this link, he is redirected to the online banking portal **<https://my.piggybank.nu>**.

Now suppose an attacker intends to hack PiggyBank's on-line banking system and wants to trick customers into allowing him access to their bank accounts. Using the Kaminsky attack, this is really straightforward:

- The attacker would begin by copying PiggyBank's web site and on-line banking portal to make sure that they look familiar to the customer;
- Then the attacker would set up his own domain, **[secure-piggybank.nu](https://my.secure-piggybank.nu)**;
- Using the Kaminsky attack, he can now poison the cache of a major ISP directing queries for the **[piggybank.nu](https://my.piggybank.nu)** domain to his own server;
- He can now lead users to his own modified version of the PiggyBank website. This website is identical to the original, except that the link to **<https://my.piggybank.nu>** now refers to **<https://my.secure-piggybank.nu>**;
- Anyone can request SSL certificates from most suppliers as long as they can prove they own a domain. As our attacker owns the **[secure-piggybank.nu](https://my.secure-piggybank.nu)** domain, he can request a certificate for **my.secure-piggybank.nu**;
- Users will end up on an SSL secured site, padlock present, and all that looks completely valid from their point of view whereas in fact they are on a phishing site. Only a user who notices that the address bar shows **<https://my.secure-piggybank.nu>** rather than **<https://my.piggybank.nu>** might realise that there is something wrong.

Had both PiggyBank and the ISP used DNSSEC, then the resolver of the ISP would have been able to check the authenticity of the reply it got in response to its queries for information in the **[piggybank.nu](https://my.piggybank.nu)** domain; DNSSEC can effectively prevent this attack from taking place.

Obviously, the scenario above applies to any service using DNS: instant messaging, e-mail, VoIP, et cetera. All of these services can be subverted through the Kaminsky attack. Deploying DNSSEC can prevent this line of attack.

2.6 How DNSSEC solves this problem

The main goal of DNSSEC is to introduce authentication of answers to DNS queries. This is achieved using digital signatures. To put it simply: each DNS record is signed using a cryptographic algorithm and resolvers have the means to check these signatures thus proving the authenticity of the information supplied. The cryptographic algorithm is strong enough to prevent casual subversion by an attacker. The way DNSSEC works is explained in more detail in chapter 3.

To put this solution in the perspective of the analogy of road signs: by using DNSSEC a special code is printed on each road sign that can be checked for authenticity thus proving that the road sign is genuine and can be trusted.

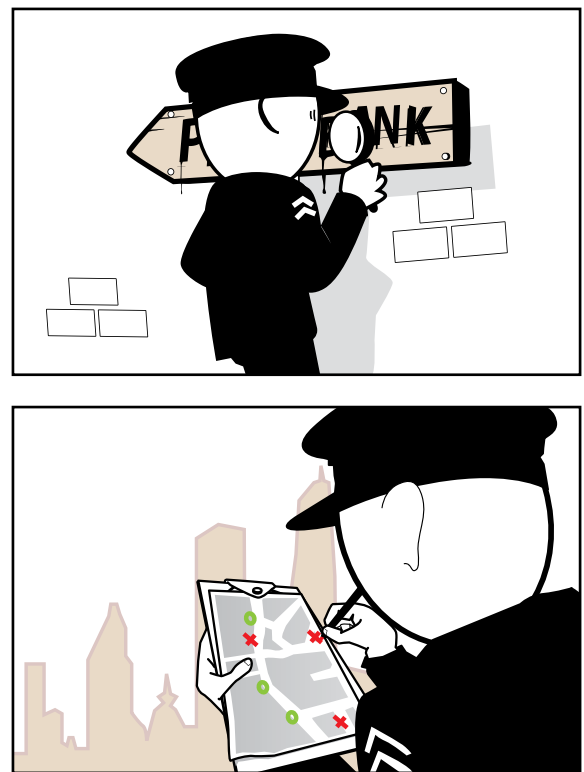


Figure 7 - Checking the authenticity of a road sign



3

WHAT IS DNSSEC?

3 WHAT IS DNSSEC?

3.1 Definition

DNSSEC stands for “Domain Name System Security Extensions”. DNSSEC is an extension to the DNS protocol⁷. It is defined in several specifications by the Internet Engineering Task Force (IETF)⁸. This chapter contains a high-level overview of DNSSEC and the mechanisms involved.

3.2 Security by signing

The security that DNSSEC provides is based on signing information cryptographically using public key cryptography (this means that a key-pair is used: signatures are created using a private key, and can be validated using the associated public key).

DNSSEC is implemented on a zone level: the DNS information for an entire zone is signed.

An important feature of DNSSEC is that signing takes place off-line. It would be infeasible, because of the amount of computation this would require, to create on-the-spot signatures (this would put a strain not only on the server but also on the resolver and caching name servers). Therefore a DNSSEC Signer signs zones up-front. The signed zones are then stored on and served from a DNS server that supports DNSSEC.

By providing signed zones, DNSSEC provides authenticated responses to DNS queries. A recursive caching name server or even a resolver on the client can validate the DNS response it receives by checking the signature on the response against the appropriate public key.

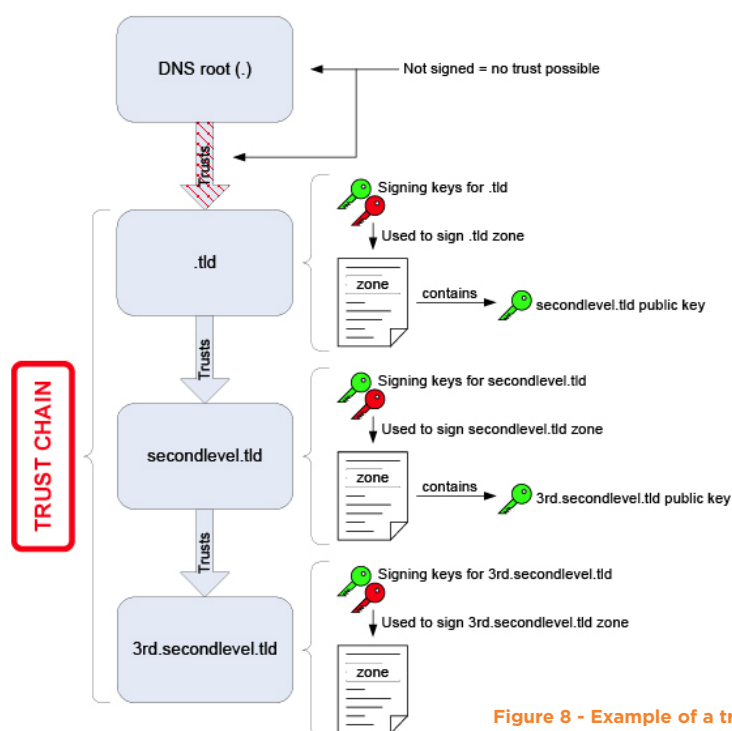
It is important to note what DNSSEC does not provide: confidentiality. DNSSEC only proves that a response is genuine, but it does not keep the response hidden.

3.3 Chains of trust

As was already discussed in section 1.3, DNS uses delegation to assign the responsibility for domains to different parties. For instance: the holder of the **.nl** top-level domain delegates responsibility for the **surfnet.nl** second-level domain to SURFnet. This is done by adding an NS (Name Server) record in the **.nl** zone that refers to the authoritative name servers for **surfnet.nl**.

In a manner similar to how domain delegation is performed, it is also possible to delegate responsibility for the signing of domains. The holder of a domain can delegate signing of a sub-domain by expressing trust in the key that is used to sign the sub-domain.

This practice creates chains of trust. The advantage of these chains of trust is that a validator does not have to trust each sub-domain public key explicitly. All it needs to do is to trust the key at the top of the trust chain. Whenever it needs to validate a DNS response, all it has to do is walk the chain of trust from the top until it arrives at the appropriate sub-domain. Figure 8 shows an example of a trust chain for the fictional **3rd.secondlevel.tld** domain.



⁷ The DNS protocol is described in RFC 1034 (see [6]) and RFC 1035 (see [7])

⁸ See [9], [10], [11] and [14]

Figure 8 - Example of a trust chain

A resolver needs to have a starting point for the trust chain when it wants to validate a DNS response. Ideally, the chain of trust would start at the root of the domain name system. Unfortunately, the root zone is not signed at this moment⁹. This means that a complete trust chain from the root down is not possible. Another issue is that there can be gaps in a trust chain; DNSSEC was explicitly designed to be deployed in such a manner that trust can start and end at any point in a domain path.

For example: if the top-level domain **.nl** is not signed but **surfnet.nl** and all the domains below are signed, a trust anchor for a resolver would be the public key used to sign the **surfnet.nl** zone.

At the time of writing of this white paper, the majority of top-level domains as well as the root zone remain unsigned. Only a handful of top-level domains support DNSSEC. A larger number of top-level domain administrators have announced that they are going to support DNSSEC in the future.

Because the root zone is not yet signed, any domain currently deploying DNSSEC will form an island of trust. The big disadvantage of this situation is that any resolving party has to decide whom to trust (i.e. which islands) and will have to negotiate some way of establishing trust in key material supplied by the parties it wants to trust. Figure 9 below shows an example.



3.6 Alternatives

DNSSEC solves the basic security flaw within DNS: the fact that a name server cannot know whether a response it receives from another server is genuine, or even that it comes from the right server. There are several other initiatives to secure DNS, but none of these solve this basic flaw in a scalable manner. Appendix E discusses some of these alternatives.



16



4

IMPLEMENTING DNSSEC

4 IMPLEMENTING DNSSEC

4.1 Introduction

This chapter gives a high-level overview of how to implement DNSSEC. It introduces a possible timeline for DNSSEC roll-out. A high-level strategy for deploying DNSSEC within an organisation will also be presented.

A detailed technical discussion on how to roll-out DNSSEC can be found in Appendix C.

4.2 Timeline

Before a strategy for rolling out DNSSEC can be discussed, it is important to reflect on the current state of affairs. At the time of writing of this white paper, large scale adoption of DNSSEC has not yet taken off. As was already mentioned in section 3.4, however, there are indications that the adoption of DNSSEC is gathering momentum. The diagram below shows a possible timeline for DNSSEC deployment across the Internet, based on the current situation:

The figure shows four distinct phases:

- Phase one is the research phase¹¹, which has already been completed. This is the period during which researchers realised that DNS needed to be secured, and developed the DNSSEC standard. During this phase, some zones were signed experimentally.

- Phase two is the early-adopters phase. This is the phase we are currently in. During this phase, early-adopters like universities, research institutions, banks, some companies and a few top-level domains start signing their zones and deploying DNSSEC. These early-adopters play an important role in gathering momentum to convince more top-level domains to start supporting DNSSEC. There is some support for DNSSEC in major operating systems. During this phase, preparations are underway to sign the root and the remaining top-level domains.
- Phase three is the commodity phase. Having gained momentum, DNSSEC takes off as a multitude of top-level domains start offering support for DNSSEC. Major operating systems start supporting DNSSEC out-of-the-box. The number of DNSSEC enabled zones grows rapidly. The root zone is expected to be signed somewhere during this phase.
- Phase four is the latecomers phase. The growth in usage of DNSSEC diminishes as the majority of zones are already signed. Growth is accounted for by latecomers adopting DNSSEC and by new domains being registered.

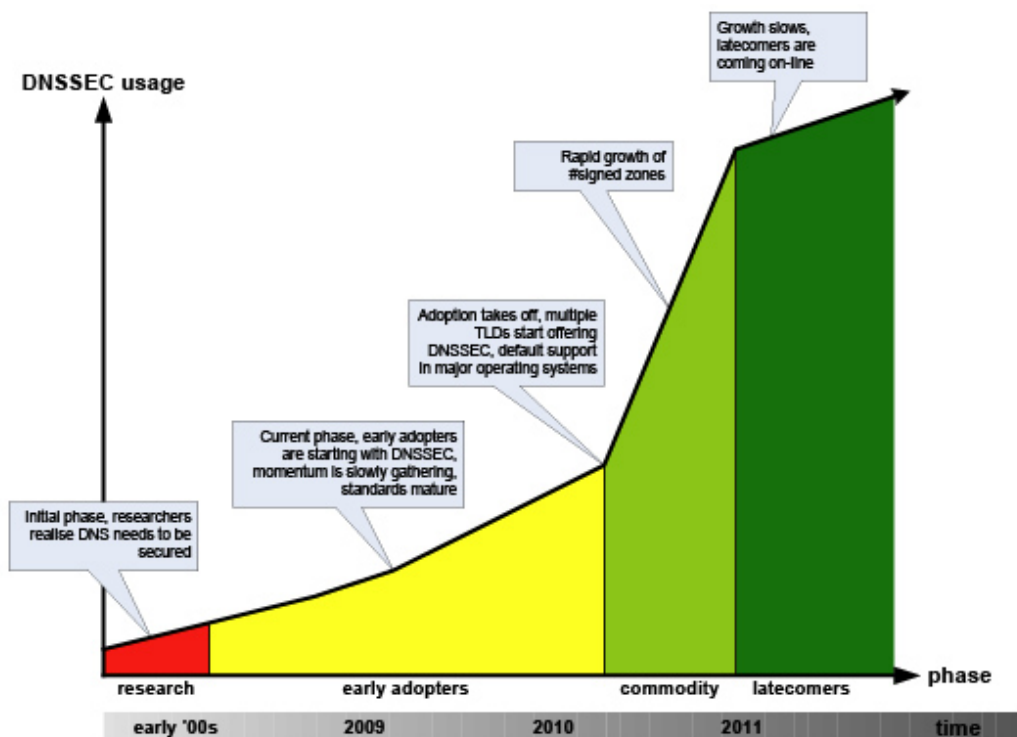


Figure 11 - Possible timeline for DNSSEC roll-out across the Internet

¹¹ This phase started outside of the diagram during the latter part of the 1990's

4.3 Strategy for rolling out DNSSEC in an organisation

4.3.1 When to start rolling out

Sooner or later, any organisation with any kind of IT infrastructure will have to implement DNSSEC. Even organisations without a domain name of their own will still need to enable DNSSEC, in order to validate the DNS data from other domains.

The most important question that needs answering is when to start rolling out DNSSEC. This depends on what role an organisation has in society, and on the level of expertise within an organisation. Based on the phases from the previous section, some recommendations are given below:

- In phase two (early adopters), organisations with a relatively high-level of technical expertise can start rolling out DNSSEC. Such organisations will include universities, research institutions, ISPs, banks and some top-level registrars. The role of these early adopters is significant, as they are the ones that create the momentum for a large-scale deployment of DNSSEC across the Internet. During this phase, tools are still maturing.
- In phase three (commodity) almost any organisation should be able to deploy DNSSEC. Tools have matured and are commonly available.
- It is not recommended to start as late as phase four (latecomers) as this will make an organisation a likely target for attacks.

4.3.2 What to start with

The implementation of DNSSEC within an organisation actually consists of two separate activities: enabling DNSSEC on all recursive caching name servers, and signing the zones on the authoritative name servers. These activities can be executed sequentially.

4.3.3 Resolving

An organisation that wants to start deploying DNSSEC should start by updating or replacing their recursive caching name servers (resolvers). The reason for resolving securely is twofold:

- It allows an organisation to start validating DNSSEC information from zones that are already DNSSEC enabled;
- Without it, it is impossible to validate one's own zones.

Many (commercially) available resolvers already support DNSSEC, and will only need an update or a parameter change, as well as a configuration change to set up the *trust anchors*. Other resolvers may need a larger upgrade or a replacement. Appendix C provides an overview of these steps, under heading C.2 “DNSSEC for resolvers”.

4.3.4 Signing zones

Once the resolver infrastructure is in place, an organisation can start signing their zones and publishing the DNSSEC enabled zones on their primary name servers. This requires a more significant effort, as tools for signing zones are – at the moment – less readily available and have a higher learning curve. Managing the keys for zone signing is too complex to be done manually, so a set of tools is necessary to automate parts of the process. Especially for smaller organisations and/or those with less technical expertise it is recommended to outsource signing or to invest in a DNSSEC signer appliance – that is: a dedicated machine in the network that can perform all the important signing functionality out-of-the-box¹².

As part of the DNSSEC deployment, the organisation will have to establish a set of procedures for key management and roll-over, and ensure that there are individuals within the organisation responsible for these procedures. If IT functions such as DNS management have been outsourced, some of these responsibilities may well be included in the outsourced service. However, the organisation should take care that this does not result in a vendor lock-in. Appendix D provides some hints for organisations wishing to outsource these activities.

¹² Such appliances are starting to become available on the market and there are also open source initiatives for making such software available such as the OpenDNSSEC project (<http://www.opendnssec.org>)



5

CONCLUSIONS

5 CONCLUSIONS

5.1 DNSSEC provides trust in DNS responses

As was shown in chapter 2, DNS has major security issues that are there by design. DNSSEC solves these issues by giving confidence in the authenticity of domain information. Spoofing is much harder, and cache poisoning is no longer a threat when DNSSEC is deployed.

Not everybody has to participate at once to make DNSSEC work – there is no need for a big-bang deployment; the design of DNSSEC allows starting anywhere within the DNS hierarchy. Organisations can start now, without waiting for others. The more organisations adopt DNSSEC, the stronger it becomes.

5.2 There is no alternative in the long term

There is no credible alternative to DNSSEC, at least not in the long term. The DNS patching arms race that is currently ongoing is a battle that will most likely be lost in the long run, simply because security was not one of the design criteria when DNS was developed. Although there are some alternatives, none of these address the actual problem of verifying the authenticity of DNS responses.

This means that although DNSSEC is more complex than DNS, there is no other option in the long run.

5.3 Managing DNSSEC is different from managing DNS

As has been explained in this white paper, different skills are required to manage a DNSSEC enabled zone. This may require some changes in an organisation and may require administrators to acquire some new skills. It is important to note, however, that there is a lot of effort ongoing to make managing a DNSSEC enabled zone easier.

DNSSEC is becoming easier every day. Appliances are already starting to appear on the market (including open source initiatives such as OpenDNSSEC). Tooling is also maturing quickly, rapidly making it easier to automate many of the harder tasks in DNSSEC. And recently third parties have started offering hosted DNSSEC.

5.4 You need to do it

To make the Internet safer, DNSSEC should be adopted by everyone. Thus, organisations must start deploying it. Implementing DNSSEC will enable an organisation to secure its own domain, and to validate DNS data from others.

Starting by addressing the changes required for recursive caching name servers, the deployment can take off, and once set in motion, signing zones is only one step away. It is of vital importance that top-level domains get signed as quickly as possible, and this will only happen if the stakeholders in these domains (the organisations that have a second level domain under these top-level domains) adopt DNSSEC themselves, thus paving the way. Early adopters should be those organisations that have technical skills available, such as universities, research centres, banks and – of course – ISPs.



6 ACKNOWLEDGEMENTS

SURFnet would like to thank the following people for their contribution to this white paper:

For co-authoring the white paper:

- Paul Brand of Stratix
- Rick van Rein of OpenFortress
- David Yoshikawa of Stratix

For reviewing the white paper:

- Jaap Akkerhuis of NLnetLabs
- Anand Buddhev of RIPE NCC
- Aart Jochem of GOVCERT.NL
- Piotr Kijewski of NASK/CERT Polska
- Olaf Kolkman of NLnetLabs
- Esther Makaay of SIDN
- Krzysztof Olesik of NASK
- Jeroen van Os of GOVCERT.NL
- Carol Overes of GOVCERT.NL
- Antoin Verschuren of SIDN
- Wouter Wijngaards of NLnetLabs

7 ENDORSEMENTS

The organisations listed below have collaborated with SURFnet to make this white paper possible. By endorsing this white paper, these organisations underline the importance of DNSSEC for the security of the Internet:

GOVCERT.NL

“GOVCERT.NL is the Computer Emergency Response Team of the Dutch government. We work on preventing and dealing with IT security incidents 24/7. We support organizations that carry out public tasks such as government agencies, and work together with vital sectors, such as water boards and energy companies. We inform the general public about measures and current risks regarding computer and Internet use.

Securing the basics of the Internet must be a priority of every organisation relying on trustworthy services or providing public services on the Internet. The opinion of GOVCERT.NL is that DNSSEC is an important improvement of the integrity of DNS. Spreading knowledge and advice about implementing the system is needed and GOVCERT.NL is happy to endorse this white paper.”

NLnet Labs

NLnet Labs is a research and development foundation that focuses on those developments in Internet technology where bridges between theory and practical deployment need to be built; areas where engineering and standardisation takes place.

Since its foundation in 1999, NLnet Labs has been active in DNSSEC standardisation, deployment engineering, training, and software development. NLnet Labs developed NSD, an authoritative DNSSEC aware, high performance name server in used by various root-zone and top-level domain operators. We have assumed responsibility for the development of Unbound, a DNSSEC aware DNS recursive name server and stub resolver and maintain the Idns and Net::DNS software library suites.

OpenFortress^{*} digital signatures

OpenFortress Digital Signatures is a specialist in cryptography, aiming to develop the general availability of this useful technology in the form of practical applications that are easy to use. OpenFortress has keenly awaited a wide adoption of DNSSEC since 2004.



Founded in 1992, the RIPE NCC is an independent, not-for-profit membership organisation that supports the infrastructure of the Internet. The most prominent activity of the RIPE NCC is to act as a Regional Internet Registry (RIR) providing global Internet resources and related services to a current membership base of over 6,000 members in 75 countries.

More information about the RIPE NCC is available at: <http://www.ripe.net>

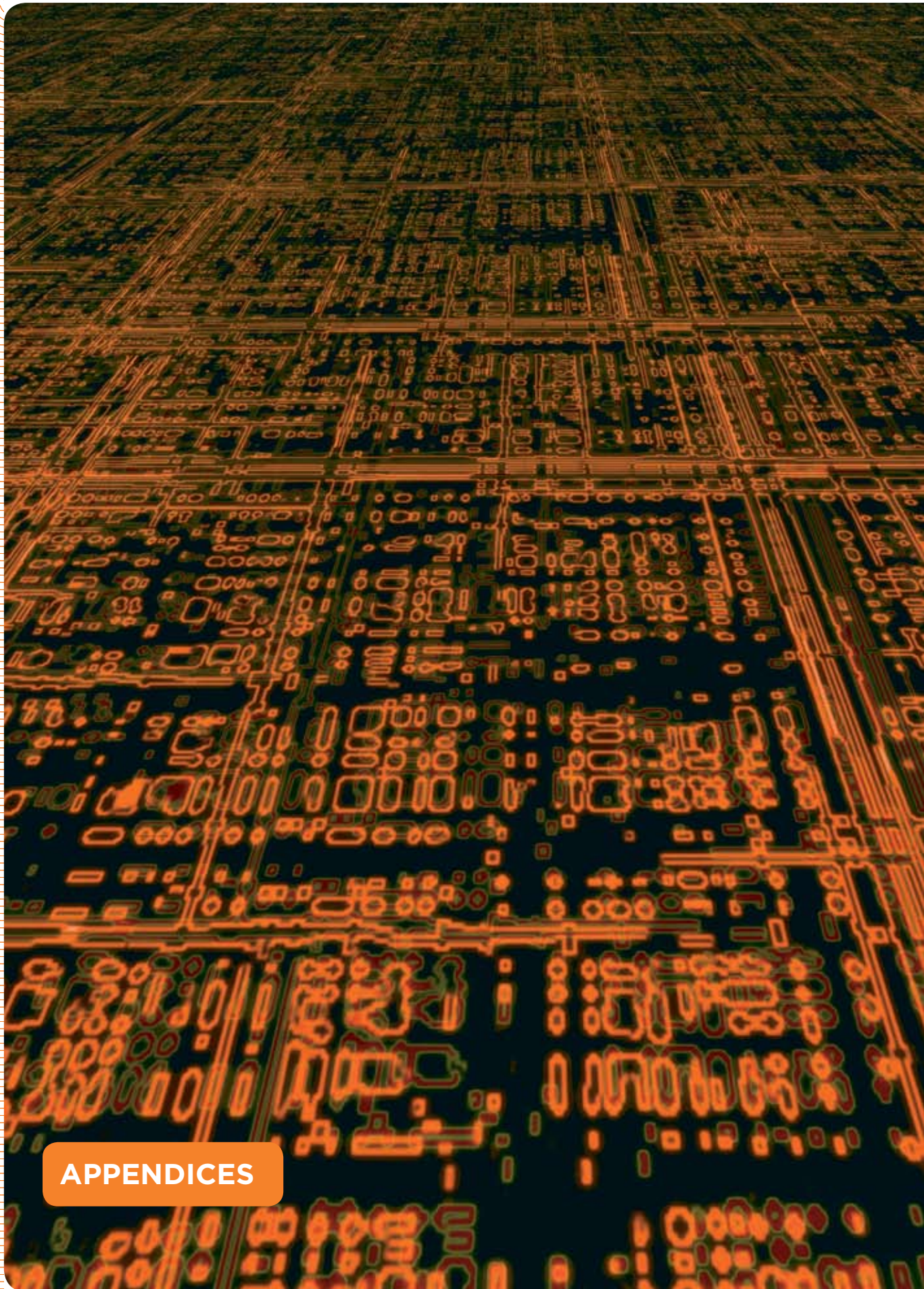


SIDN is responsible for the functional stability and development of the .nl Internet domain. As well as registering and allocating .nl domain names, the organisation enables Internet users all over the world to make use of these labels at any given moment. SIDN's rapidly growing domain name register now contains more than three million .nl domain names, which are the subject subject of almost one million successful searches a day.

SIDN also plays an active role in the technical, regulatory and political development of the Internet, at the national and international levels



SURFnet develops and operates innovative services for higher education and the research community. We focus on network infrastructure, authentication and authorisation services and multimedia platforms for online collaboration. About one million end-users access our services on a daily basis. SURFnet is part of SURF, an organisation in which universities, polytechnics and research institutions collaborate on groundbreaking ICT innovations. SURFnet has been a leader in ICT innovation in The Netherlands for over 20 years.



APPENDICES

APPENDIX A TERMS AND ABBREVIATIONS

A.1 Terms

Authentication

Verifying the authenticity of one's identity

Authoritative name server

A name server that is the base source of information about a certain domain; the fact that a name server is authoritative for a certain domain is indicated in the so-called "start-of-authority" (SOA) resource record.

Digital signature

A cryptographic operation based on public key cryptography used to uniquely prove that a given piece of information is trusted by the owner of a given private key; the authenticity of a digital signature can be verified using the corresponding public key.

Domain

A namespace in the Domain Name System; for instance: surfnet.nl

Hints file

A file containing the names and addresses of the root name servers; this file is distributed by InterNIC on their website.

Host name

The meaningful name given to a computer system.

Kaminsky attack

The cache poisoning attack on the Domain Name System published in 2008 by Dan Kaminsky.

See also: reference [5]

Phishing

Identity theft by luring users to a fraudulent website – users are enticed to enter their private data such as their usernames, passwords and credit card numbers on a spoofed website by an e-mail seemingly from a party they trust.

See also: <http://en.wikipedia.org/wiki/Phishing>

Public key cryptography

An asymmetric cryptographic scheme with two complementary keys: the public key and the private key. A signature computed using the private key can be validated using the public key, conversely a piece of data can be encrypted using the public key and decrypted using the private key.

See also: http://en.wikipedia.org/wiki/Public-key_cryptography

Recursive (caching) name server

A name server that can be used by clients to perform DNS resolving tasks; a caching server will store all the answers it receives to queries and re-use these when appropriate.

Resolver

A piece of software that uses DNS servers to map a DNS name to an IP-address.

Resource record

Usually abbreviated to RR; a resource record is an entry in the Domain Name System. There are several types of resource record. The most commonly used ones are A records for mapping a name to an (IP-) address.

See also: http://en.wikipedia.org/wiki/List_of_DNS_record_types

Time-to-live

The time period during which an answer to a DNS query remains valid and can thus be cached.

Trust anchor

The top of a trust chain that can be used by validators as a starting point to validate the authenticity of a signed DNS record at any point in the trust chain.

Validating resolver

A resolver that validates the DNSSEC signatures on the answers it receives to DNS queries.

Validator

A piece of software that verifies the authenticity of a digital signature using the public key that was used to create the signature.

Zone

A collection of related resource records that is served as a unit by a name server.

See also: http://en.wikipedia.org/wiki/DNS_zone

Zone walking

The ability to retrieve the complete content of a zone using the sequence of NSEC records in a zone.

A.2 Abbreviations

BIND

Berkeley Internet Name Domain (a commonly used Domain Name Server package)

DLV

DNSSEC Look-aside Validation

DNS

Domain Name System

DNSSEC

Domain Name System Security Extensions

IETF

Internet Engineering Task-Force
(see <http://www.ietf.org>)

IP

Internet Protocol

ISP

Internet Service Provider

KSK

Key Signing Key

LAN

Local Area Network

NSD

Name Server Daemon
(a commonly used Domain Name Server package)

NSEC3

Next Secure version 3,
part of the DNSSECprotocol

RFC

Request For Comments (an IETF specification)

SSL

Secure Socket Layer

TCP

Transmission Control Protocol

TLS

Transport Layer Security

UDP

User Datagram Protocol

WAN

Wide Area Network

ZSK

Zone Signing Key

APPENDIX B REFERENCES

- [1] WikiPedia: The Domain Name System
http://en.wikipedia.org/wiki/Domain_name_system
- [2] DNS and BIND
Paul Albitz and Cricket Liu, O'Reilly Media, Inc., Fifth Edition, May 26, 2006
- [3] WikiPedia: DNSSEC
<http://en.wikipedia.org/wiki/DNSSEC>
- [4] WikiPedia: Public-Key Cryptography
http://en.wikipedia.org/wiki/Public-key_cryptography
- [5] Black Ops 2008: It's The End Of The Cache As We Know It
(or 64K should be enough for anyone)
http://www.doxpara.com/DMK_BO2K8.ppt
- [6] RFC 1034: Domain Names – Concepts and Facilities
<http://tools.ietf.org/html/rfc1034>
- [7] RFC 1035: Domain Names – Implementation and Specification
<http://tools.ietf.org/html/rfc1035>
- [8] RFC 2845: Secret Key Transaction Authentication
<http://tools.ietf.org/html/rfc2845>
- [9] RFC 4033: DNS Security Introduction And Requirements
<http://tools.ietf.org/html/rfc4033>
- [10] RFC 4034: Resource Records for the DNS Security Extensions
<http://tools.ietf.org/html/rfc4034>
- [11] RFC 4035: Protocol Modifications for the DNS Security Extensions
<http://tools.ietf.org/html/rfc4035>
- [12] RFC 4431: The DNSSEC Lookaside Validation (DLV) DNS Resource Record
<http://tools.ietf.org/html/rfc4431>
- [13] RFC 4635: HMAC SHA TSIG Algorithm Identifiers
<http://tools.ietf.org/html/rfc4635>
- [14] RFC 4641: DNSSEC Operational Practices
<http://tools.ietf.org/html/rfc4641>
- [15] An illustrated guide to the Kaminsky DNS vulnerability
<http://www.unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html>
- [16] ISC's DNSSEC look-aside validation registry
<https://www.isc.org/solutions/dlv>
- [17] Bootstrapping the adoption of Internet security protocols
<http://weis2006.econinfosec.org/docs/46.pdf>

B.1 Further reading

For more information on the various aspects of DNSSEC the reader is referred to the following sources:

- **The DNSSEC consortium web-site**
<http://www.dnssec.net>

This site is a source of up-to-date information about DNSSEC and the deployment of DNSSEC on the Internet.

- **The DNSSEC HOW-TO by NLnetLabs**
http://www.nlnetlabs.nl/downloads/publications/dnssec/dnssec_howto.pdf

A technical resource with hands-on information about configuring a signed zone; examples are based on BIND.

- **NIC.CZ information page**
<http://www.nic.cz/page/513/about-dnssec/>

High-level information about DNSSEC; contains a testing widget that shows (using either a red or a green key) whether or not you arrived on the page from a DNSSEC secured domain.

APPENDIX C TECHNICAL DISCUSSION OF DNSSEC

C.1 Introduction

This appendix addresses the technical and organisational implications of implementing DNSSEC within a DNS infrastructure. It addresses both the issues for resolvers as well as for authoritative name servers.

C.2 DNSSEC for resolvers

C.2.1 Introduction

DNSSEC requires resolvers to implement validation procedures for DNSSEC. Although not all software is equipped with these extra capabilities, the most important resolvers are. With BIND9 it is a matter of setting a few flags in the configuration files, and Windows Server 7 will also support DNSSEC in its role as a resolver.

C.2.2 Computing power

Validating resolvers are configured just like any cache, but may need a bit more computing power to perform their task, as a result of the public key cryptography involved in the validation process. The added performance requirement is a result of centralising the cryptographic validation procedures (rather than have each desktop complete the validation on its own) but it does save on total computing effort, as validated results can be shared among independent desktops. It is important to realise that delays in DNS are experienced as delays “in the Internet”. Experiments by the DNS experts at NLnetLabs indicate that delays are not expected to be severe enough that they warrant dedicated cryptographic accelerator boards in validating resolvers. In the estimated 10% to 30% of queries that cannot be answered from the cache, validating the trust chain may cause a somewhat slower response, but bearable on a human scale. Note that scaling up only means that more hits are delivered from a cache.

C.2.3 Setting up trust anchors

Without going into details for specific applications, it is good to be aware of the need to configure trust anchors. As with X.509 certificates, DNSSEC needs a starting point for its relationships (see sections 3.3 and 3.4). DNSSEC has been designed to support multiple starting points for chains of trust. For each of these entry points into the DNSSEC hierarchy, there is a so-called trust anchor, which comprises of DNSKEY records or their secure hashes that have usually been validated out-of-band before they are trusted. The name server software should be configured to rely on these trust anchors.

IANA has been commissioned to setup an Interim Trust Anchor Repository that will host validating information for the keys used to sign top-level domains until the root zone has been signed. This is a temporary solution that will allow time for the resolution of the complex issues related to the central position of the root zone.

C.2.4 Look-aside validation

The introduction of DNSSEC for a parent zone such as a TLD or a ccTLD is more difficult than simply signing one's end-user zones. This is a result of gathering a lot of information, handling lots of key rollovers and a more general responsibility. As a result, child zones are likely to be signed before their parent zones. As a pragmatic solution to this generation gap, a temporary construct of DNSSEC look-aside validation (or DLV for short, see [12], [16]) has been defined. Instead of registering a domain's DNSSEC keys with a parent, this works by registering them in another domain that happens to collect such keys, and makes them available for look-aside validation

To setup look-aside validation, the look-aside domain collects DLV records (which are similar to the DS records generated for the parent, except for the resource record type, see [12] for more details). Such records are created under another domain, so that the DLV record for **surfnet.nl** could be stored for look-aside validation under **dlv.isc.org** by creating the DLV record under **surfnet.nl.dlv.isc.org**. A resolver configured with a trust anchor for **dlv.isc.org** would lookup anything without a DS in the parent as a DLV record under the **dlv.isc.org** domain, and if it finds one it will treat it in a way similar to a DS record that should have been found in the parent zone.

Since the DLV domain is also a domain like any other, and since it is usually validated through DNSSEC, it will be necessary to follow-up on key rollover at the DLV domain. This can be done manually, about once a year, or it can be automated if your server software implements RFC 5011, and if the published domain also uses that standard to revoke keys as they have served their useful life.

C.2.5 Security issues

It cannot be avoided that the same resolver handles both secure and insecure domains; in fact, DNSSEC is sometimes used to establish a domain's insecurity. Validating resolvers must be aware of this mixed world-view because it would be a dramatic failure if an attack in the style of Dan Kaminsky's could be mounted against an insecure domain and let that alter a secure domain's validated records. If you select resolver software, pick one that is mature, so you can rely on a strict separation between secure and insecure data.

C.2.6 Resolver selection

The following points sum up the issues that help to select a validating resolver:

- Does the resolver support DNSSEC?
- Can the resolver be configured with the KSK for selected domains?
- Can the resolver be configured for look-aside validation; accepting the DLV KSK and sending DLV requests to a look-aside domain?
- Does the resolver have facilities to update trust anchors automatically? (optional)

C.3 DNSSEC for authoritative name servers

C.3.1 Managing a signed zone

In chapter 3 the concept of zone signing was introduced. More specifically: the actual information being signed are sets of related Resource Records, also called RRsets. An RRset comprises all resource records of one type, in one class, pertaining to a single resource (e.g. all address (A) records in the Internet (IN) class for one specific host name). An RRset can consist of one or several records.

Signed zones should not be maintained by hand, as the data in the zones is too complicated for that. Instead, automated tools should be used to manage the signing of data in the zones.

C.3.2 Key material and key management

C.3.2.1 Algorithms

DNSSEC is based on public key cryptography. RFC 4034 (see [10]) specifies that the following cryptographic algorithms may be used:

- DSA/SHA-1
- RSA/SHA-1

For two reasons it is recommended only to use the RSA/SHA-1 algorithm:

- Security: DSA keys are constrained to a maximum key length of 1024 bits; this may impact the security of DSA keys
- Performance: Signature validation of DSA signatures is an order of magnitude slower than signature validation of RSA signatures. This mainly has an impact on validating resolvers.

In the future elliptic curve cryptography is also going to be supported for DNSSEC. Currently, however, its use has not been standardised yet.

C.3.2.2 Key types

The current operational practice for DNSSEC is to use a two-tiered key model:

- A Zone Signing Key (ZSK) is used to sign RRsets within a zone
- A Key Signing Key (KSK) is used to sign Zone Signing Keys

Each of these keys has its own specific properties:

The ZSK is relatively short-lived; the recommended use period for a single ZSK according to RFC 4641 (see [14]) is one month. It is also recommended to use a moderate key-size for the ZSK (in the order of magnitude of 1024 bits). This is necessary in order to keep the time it takes to sign a zone within manageable limits.

The KSK is longer-lived; the recommended use period for a single KSK according to RFC 4641 is 1 year. It is recommended to use a minimum key size of 2048 bits.

If zone signing has been delegated by a parent zone by means of a DS record, then this DS record should reference the KSK. This means that the parent only has to be informed if the KSK is updated.

C.3.2.3 Key rollover

The KSK and ZSK both have a limited period of validity. This means that it is necessary to perform key rollovers at regular intervals. To allow time for information to propagate through the DNS and to allow time in case of unscheduled problems it is good practice to give subsequent keys slightly overlapping validity periods. For instance: the validity period of a KSK could be 13 months and the validity of a ZSK could be 1 month plus enough time to let the longest TTL-values expire, for instance 1 week.

It is also good practice to make the new keys available before their validity period commences. A new KSK should ideally be announced 1 month prior to key rollover and a new ZSK should ideally be announced 1 week prior to key rollover¹³. Alternatively, it is possible to have non-overlapping key validity, but to have temporarily overlapping signatures on DNS records. The former method is called pre-publication of keys, the latter is the double-signature method.

¹³ Note: the time intervals given in this section are based on the current best practice as described in RFC 4641 [14]; these time intervals are not absolute, sensible variations are possible and likely.

The diagram below shows how these overlapping periods could work in practice:

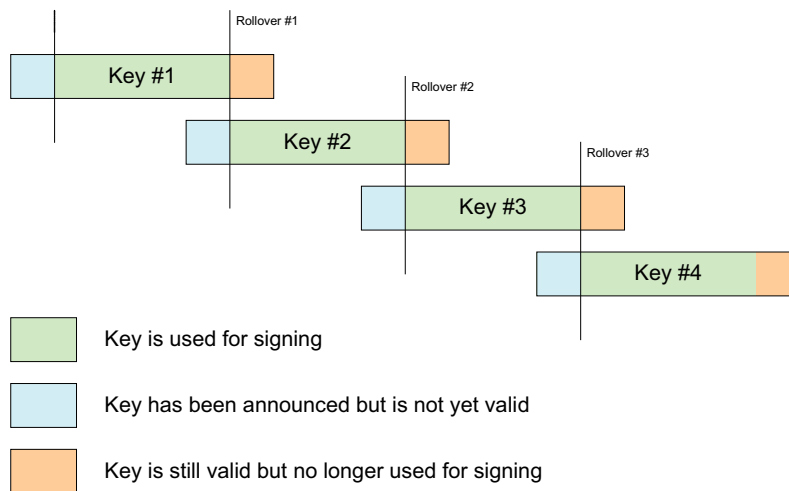


Figure 12 - Key rollover period overlaps

Whenever a key rollover of the KSK is to take place, the parent zone should be informed and should be supplied with a new reference record (called delegation signer, or DS) for this key. The new KSK should not be used for signing of ZSKs until the parent zone has been updated.

C.3.3 Authenticated denial of existence (NSEC/NSEC3)

C.3.3.1 Why authenticated denial of existence is necessary

A practical aspect of DNS is that it provides an explicit answer if a requested resource record does not exist. These explicit negative acknowledgements avoid retries and waiting for timeouts. From a security perspective, however, this introduces a threat of denial-of-service attacks.

To avoid such attacks, the absence of a resource record must be signed. But DNSSEC works with offline zone signing, making it impossible to predict any query against any name and sign for its absence.

C.3.3.2 NSEC and zone walking

The solution is not to sign for the name being absent, but to sort all names in a zone in a canonical order and to sign for statements like “after A comes C” so resolvers can infer that B does not exist. To facilitate this, the NSEC resource record was introduced (see [10]). This record lists the following information for a given host name:

- The next host name to be listed in the zone according to canonical ordering
- The types of records existing for the host name

As you can see this clearly defines that a given record “A” is followed by a given record “C” inferring that no intermediate record “B” can exist. It also goes

on to prove that for the given record “A” only signed RRsets exist of the specified types.

So if a name server gets a request for “B” it simply responds with the NSEC record for “A” thus proving in a secure way that “B” does not exist.

A long-time show-stopper for DNSSEC has been the lack of privacy of this construction; if one got hold of the name “A”, it would be trivial to get a link from A to C based on the NSEC record, from C one could then get a link to K, from K to Q etc. until the entire zone has effectively been enumerated. This iterative process that lists all names in a zone is commonly called “zone walking”. Although DNS data is usually public¹⁴, many felt this to be an unacceptable assault on their privacy and/or their ability to conceal experimental or private sub-domains from public viewing (which was possible because it is currently common practice to deny zone transfers to any but a few trusted parties).

C.3.3.3 How to solve zone walking: NSEC3

A recent improvement to DNSSEC addresses just this problem in the so-called NSEC3 resource record. This record type does not link the names in a domain, but the hashes of such names. A response that explains that B does not exist under a domain starts by calculating $\text{hash}(B)=4323\dots$ and finds its position in an ordered list of all names that occur in a domain. Perhaps $\text{hash}(Q)=381a\dots$ precedes the value of $\text{hash}(B)$ and $\text{hash}(C)=7bbc\dots$ might be the next. So an offline-signed link “after 381a... comes 7bbc...” is used to prove that $\text{hash}(B)=4323\dots$ does not occur.

¹⁴ There are exceptions to this rule, for instance: private DNS infrastructures behind a firewall

Since the hashes used are cryptographic/secure hashes, it is not possible to derive the original names Q and C from their hashes, so the privacy (or non-iterability) of the DNS zone is maintained while at the same time supporting the required proof that a name does not exist in the zone.

Thus the NSEC3 record is made up of 3 things:

- The hash of the host name it applies to
- The types of records existing for the host name
- The hash of the next host name in the zone in hash order (the zone is sorted from 0 to MAX(hash) and at the end it wraps back to the beginning)

So, in the example above, if a name server gets a request for “B” it simply sends back the NSEC3 record for hash(Q) allowing the resolver to verify that “B” does not exist in the given domain. Not every publisher will prefer NSEC3 over NSEC, so the two will probably continue to co-exist.

C.3.4 Dynamic DNS updates

Not all about DNSSEC is glorious. If DNS records update frequently, as in some dynamic uses of DNS, two problems arise:

- New data is not authenticated until a signature is made;
- Old data may float around as authentic until its signature expires

This means that DNSSEC and dynamicity in DNS are not an ideal combination. Dynamicity for DNSSEC could be implemented with short-lived signatures, but that leads to a lot of additional stress on DNS caches and resolvers, especially if they actively validate the signatures on DNS records (which is the most likely initial roll-out of DNSSEC). This is at least harmful for the scalability of DNSSEC.

Below, a few problems that are foreseen in existing networks are discussed, as well as some proposed workarounds.

C.3.4.1 Linking DHCP to DNS entries

The most common example of dynamic data in DNS is the mapping of a fixed host name to a dynamic IP. When acquiring an IP number through DHCP, a host may provide a host name, or its host name may be known thanks to a registered MAC address. Many DHCP servers will not only supply an IP lease to such a host, but will register the mapping in DNS at the same time.

The dynamicity of these IP addresses is usually not an issue. Other parts of the network, notably firewalls and routers, already require fixed IP addresses for exceptional systems such as servers. Only the client systems, being those that do not publish services, are treated as part of a uniform set without further discrimination based on their IP addresses.

This means that we expect the dynamic part of mappings from host name to IP address to cover clients, not servers. Since servers may be contacted from anywhere, their mapping is often vital to protect with DNSSEC, and since these mappings are static that ought to be no problem. Typical client systems with their dynamic name-to-IP mappings can be exempted from DNSSEC protection without much harm, since nobody will want to contact them using their DNS name.

DNSSEC offers a way out for such security exemptions. An insecure sub-domain of a DNSSEC domain can be constructed by referring to a sub-domain's name servers, but not accompanied by a key reference; for example the secured domain **harderwijk.edu** could have a sub-domain **dyn.harderwijk.edu** that does not support DNSSEC but uses plain DNS to map names to IP addresses. To construct this, the **harderwijk.edu** zone contains NS records for the **dyn.harderwijk.edu** domain, but contains no DS record(s) for **dyn.harderwijk.edu**, so exemption is explicitly verifiable by way of the signatures on the NS records in **harderwijk.edu**.

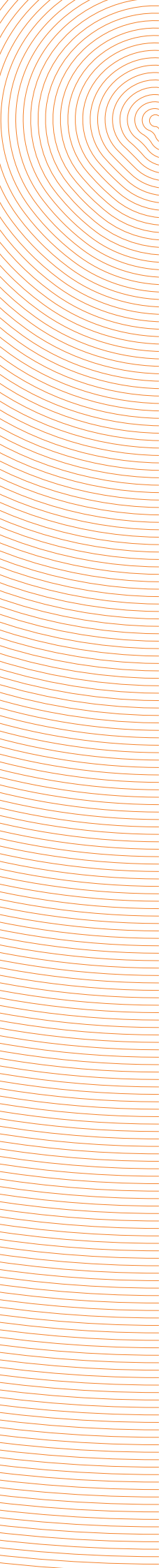
C.3.4.2 DHCP for Internet Service Providers

A specific form of the dynamicity in DNS due to DHCP concerns ISPs. The clients of an ISP may well want to run services on an IP address that is assigned to them through DHCP, which makes the IP address dynamic, at least in theory.

These DHCP assignments to cable and DSL customers are usually constant over a long period, and may therefore almost be treated as static assignments -- with the side note that a procedure must exist to alter them manually. This is actually the sort of situation that DNSSEC supports quite nicely, by way of regular resigning of a zone. Setting up a new IP address under DNSSEC is just some extra work that adds to the procedure of editing the usual DNS records. We do not expect this to cause major problems in practice. DHCP leases are usually supplied for periods of a week or so, and these periods may be synchronised with the regular DNSSEC signing procedure to even avoid the occurrence of signed faulty data in DNS.

In the ideal situation, the ISP signs their dynamically assigned records with DNSSEC, and given the long lease term from most ISPs that would not lead to scaling problems due to overloaded secure DNS caches. In any case, if a customer of an ISP defines their own domain and points it to the ISP-supplied IP address, it is possible to sign that; if the ISP is using DNSSEC it could just be a CNAME alias, but if the ISP does not define secure records it could be an A record (which is formally wrong but also is common practice).

If an ISP decides not to sign the dynamically bound mapping of names to IP addresses, it can explicitly opt-out for such addresses. This is done with a DNSSEC-signed statement that a sub-domain is



unsigned. Validating resolvers can use this statement to assure that nobody is suppressing a signature but that it is secure to assume that no signature is available. Note that ISPs failing to sign the dynamic mappings will cause additional A records in customer's own domains, so perhaps it is better to implement DNSSEC on ISP's DHCP leases. It is also worth noting that the reverse translation (from IP to host name) could be signed at the same time as the forward translation (from host name to IP).

C.3.4.3 Dynamically changing IPv6 addresses

In the interest of privacy, Windows has a default feature under IPv6 to assign a random bottom half in IPv6 addresses, and to change them regularly. This protects against visibility of one's MAC address (including the manufacturer code) in the bottom half of the address. Furthermore, since with IPv6 there is no need for NAT, all addresses are public and making long-term addresses known on the Internet does not provide the by-default security of being behind the client-only filter of NAT. Such dynamic IP addresses could be a problem for DNSSEC if it had to update its signatures. Fortunately however, this does not seem to be necessary in the situations that we currently anticipate.

Dynamic IPv6 addresses are intended for client side systems, and contacting them normally isn't a requirement. Server machines will use a manually set fixed IPv6 address over which their services will be acquired. Such fixed addresses are suitable for publication in DNS, including signatures.

Normal setups do not require the lookup of client systems in DNS, so they need not support DNSSEC; but even client systems (can) automatically create a fixed IPv6 address based on a MAC address, making them suitable for publication in signed DNS.

Note that operating systems support fixed and dynamic IPv6 addresses at the same time; it is common practice to have multiple IPv6 addresses co-existing on one interface. Only the static addresses would end up in DNS, with DNSSEC protection. The dynamic addresses, if they occur in DNS at all, can be in an unsigned sub-domain, and DNSSEC can explicitly opt-out that sub-domain.

C.3.4.4 Dynamic signing in name servers

The problems of signing dynamic content in DNS stem from the current practice of off-line signing. Future versions of BIND – the most commonly used software for DNS servers – are most likely going to support on-line signing. Microsoft is also working on DNSSEC support for Windows Server 2008 R2 and for Windows Server 7; it is likely that these will also support on-line signing since this would be required for Active Directory.

APPENDIX D OUTSOURCING

D.1 Introduction

For most organisations, managing the IT infrastructure is not part of their core business. In order to maintain a professional IT service, they tend to outsource at least part of their IT management to specialised vendors. In many cases the authoritative DNS servers for the organisation's domain will be hosted on an ISP's servers, and resolvers may be installed in-house but managed by an outside provider. This situation has implications for the deployment of DNSSEC.

D.2 Division of responsibilities

As in any outsourcing arrangement, the division of responsibilities between the organisation and outside vendor (or vendors) will have to be defined carefully. Some areas that an organisation may want to outsource, in relation to DNSSEC, are:

- Installing and configuring DNSSEC enabled resolvers, authoritative servers and signers;
- Managing DNS related equipment on-site or off-site, or hosting DNS servers on the vendor's platforms;
- Creating, storing, and deploying Key Signing Keys and Zone Signing Keys;
- Signing zones and deploying signed zones on the authoritative servers.

For each of these activities, there should be someone within the organisation with the overall responsibility, who can monitor the vendor's activities and address any issues that arise. The outsourcing contract should make clear what the vendor's responsibilities are in case of a problem, and who will assume liability for any resulting damages.

D.3 Protecting against vendor lock-in

Any outsourcing agreement will have provisions in case the organisation wants to migrate to a different vendor, or bring activities back in-house. However, DNSSEC introduces a few points that require extra care to avoid future difficulties in switching vendors:

- Clear agreement on the legal ownership of configuration data and cryptographic keys;
- Agreed procedures to hand over configuration data and cryptographic keys to the organisation or to a future vendor at the end of the contract (or earlier, if necessary);
- Agreed procedures to ensure keys remain available to the organisation in the case of disputes, take-over or bankruptcy of the vendor.

APPENDIX E ALTERNATIVES

E.1 Introduction

In this appendix the possible alternatives to DNSSEC are considered.

Although DNSSEC is a pragmatic solution rather than an ideal one, it has clear advantages over the alternatives available. Many of the alternatives below fail on account of not protecting the origin of DNS data. Although it is possible to protect DNS data by protecting every transaction between every client and server, this is unreliable for a number of reasons:

1. A chain of protected links can break at its weakest link, meaning that it is not possible to enforce minimum validation standards merely by configuring one's local resolver. Specifically, it is not possible to know if all links are individually secured or not. Being dependent on independently managed parts of the DNS infrastructure erodes the reliability of the system as a whole.
2. Between every two secured links sits a name server, either an authoritative name server or a recursive resolver. Even if the links are secure, then there is still the risk that the name server itself is poisoned with false data, which it will happily sign upon forwarding.
3. At some point in DNS, there is a need to connect from a local domain to a remote domain. This happens most often when a recursive/caching name server starts at the root name servers and proceeds downward in the DNS tree in order to resolve a query. There are quite a few practical problems related to protecting each link separately, especially because of the multitude of servers to be contacted.

E.2 The DNS arms race

The easiest "alternative" to DNSSEC is to do nothing. That is, not roll out anything to secure DNS and continue patching software as soon as a security problem arises.

DNS has not been designed for security, and name server software can only compensate to some degree. For example, in defence of Dan Kaminsky's attack there have been patches that use a random port for sending/receiving the DNS information. This effectively extends the information to be guessed by the attacker from 16 bits to 32 bits. This may defer the cache poisoning problems from all but the most determined attackers in the short term, but it is solely dependent on the possibility to squeeze these extra bits out of the existing systems. And the resulting 32 bits can by no means be classified as a securely large search space to defer attacks. It merely makes it simpler to detect attacks with an intrusion protection system, and it improves the chances of shutting down an attacker by way of an intrusion prevention system.

Rolling out intrusion detection and protection systems to protect a light-weight system like DNS can be considered overkill. The intrusion detection systems must be very powerful since DNS, thanks to its light-weight nature, can handle quite a lot of load on a single server. For instance, it is not uncommon for ISPs to service a whole country with only a few recursive DNS servers (DNS caches). Also, a determined attacker may simply fire at random in the full 32-bit search space over a long period of time, calling for intrusion detection systems that recognise patterns over a long period, which is infeasible as the system would have to consider so many attack patterns at the same time.

The sort of attacks and defences that are currently applied to DNS are an arms race, battling to manipulate or protect the technical data contained in the IP and UDP headers and the DNS payload. The attacks, if they are published at all, usually demand instant patches of one's systems, so time is of the essence.

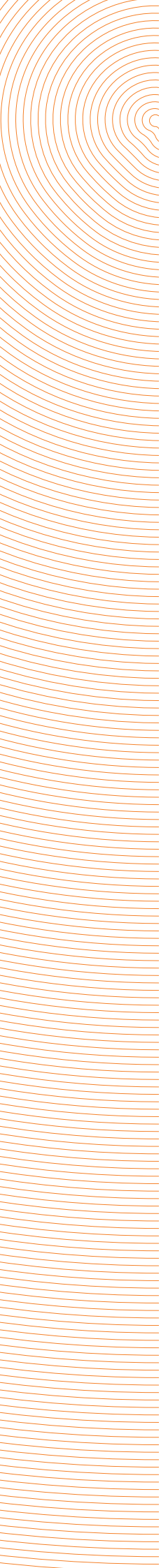
The major advantage of DNSSEC is that it introduces cryptography by way of digital signatures. The benefit of cryptography is that it creates an incredible gap between the abilities of the domain owner and an attacker: The simple fact that a private key is in the possession of a domain owner but unknown to an attacker places the latter in a greatly disadvantaged position. Potential attackers know this, and will generally avoid attacking the cryptographic aspects. If the remaining software is well-written, no attacks can realistically be mounted.

It could be argued that cryptography is an arms race of its own. This, however, is not an arms race that involves every single DNS administrator; it involves academic and government institutions that work on general cryptographic mechanisms such as RSA and SHA1. These mechanisms are widely used and widely tested by highly skilled people, and the general tendency is to be open about any possible problems that could compromise security. A few decennia worth of experience with cryptographic algorithms suggests that practical attacks hardly ever break an algorithm completely, but merely let their protection erode to such a level that the introduction of alternatives is required. This all happens at a much slower pace than the arms race of DNS as it stands today.

E.3 TSIG and SIG(0)

Early attempts to standardise DNSSEC have involved different kinds of resource records: TSIG and SIG(0). As will be explained, these are not without their use but they are unsuitable for a broad roll-out of DNSSEC.

TSIG is a facility for a shared secret between a pair of hosts. These hosts can exchange normal DNS information, and end with a signature based on that



secret. If no party but these two hosts holds that information, it can be inferred that no third party could have created the signature. The mechanism is light-weight and even has facilities for key rollover¹⁵. It is useful between paired hosts that have a long-term relationship (such as primary and secondary name servers for a domain) but it cannot scale up to a general solution for DNSSEC. For example, if the .com top-level domain were to be signed with TSIG, a shared secret would have to be negotiated between the .com authoritative name servers and every resolving name server on the Internet. Even if this would be technically feasible, there would still be the problem of co-ordinating the initiation of the shared secrets. Clearly, a public-key mechanism is more suited to the situation of general servers that are open to clients anywhere. In situations where the pairing can be fixed and secrets can be exchanged however, TSIG remains a valuable mechanism due to its small footprint.

SIG(O) is a public-key based mechanism that signs for selected queries and responses. As with TSIG, the signatures are made for a transaction between a client and a server, but it does not make the origin of data verifiable. Because SIG(O) signatures must be constructed by the resolver, it would overload that device if used to support validation of every query; it should be used sparingly.

A useful application of TSIG, TKEY and SIG(O) in a DNSSEC-rollout is to establish a secure link between a validating resolver and a relatively dumb local resolver, such as a modem/router or a single host on the network. These would request a TKEY record signed with SIG(O) using a trusted key for the connection to the validating cache. The TKEY record would relay a shared secret to the client, which can henceforth be used for light-weight TSIG security. These mechanisms have their uses, but not in a global roll-out of DNSSEC.

E.4 DNSCurve

DNSCurve solves another issue than DNSSEC; DNSSEC takes the viewpoint that information in DNS is public, and does not need encryption. It could however be argued that the actual DNS traffic does count as a security threat, even if the knowledge that is exchanged is public. On a broadcast network (cable Internet, WiFi) one shares a medium with potentially unreliable users who may be quite interested in learning that you are requesting domain information for a sensitive domain (such as a bank).

Just like TSIG and SIG(O), DNSCurve protects the query/response exchange between a client and a server. It does not provide authentication of the origin of data. Novel about DNSCurve is its use of elliptic curve cryptography, being a modern set of public key algorithms that has hitherto not been standardised for use in DNSSEC.

¹⁵ By means of the TKEY extension

E.5 IPsec

Assuming that IPsec would be omnipresent, it could spark the idea of being an alternative to DNSSEC. This is not true however -- it may help to authenticate the remote party being contacted, but not the origin of data that is received. If the remote party is in any way compromised, it can be loaded with invalid data. DNSSEC is about validating data to have come from the desired origin, and not just the proxy through which the information was obtained.

In addition to that, rolling out IPsec requires even more critical mass than DNSSEC, making it unlikely to ever hit the global Internet. Making this unlikely is the fact that IPsec is widely regarded to be a “board standard”, full of compromises to keep too many parties happy. The resulting standards are so full of options and alternatives that one cannot assume interoperability of solutions based on the mere premise that they support IPsec.

IPsec remains useful for generic traffic encryption and/or authentication in a locally controlled environment, acting as a standardised VPN, but it is not likely to gain sufficient traction for a globe-spanning secure network.

E.6 SSL or TLS

First and foremost, no proposals have been made to secure DNS with TLS or its predecessor SSL. The only relation between DNS and these protocols is that facilities have been proposed to store X.509 certificates in DNS resource records. This approach treats DNS as a database, but it has no security implications for DNS itself.

Using TLS (or SSL) in combination with certificates could have worked, if the whole infrastructure behind it wouldn't have been so riddled with questions and problems, ranging from who controls the list of trusted root certificates to what it means to sign a certificate. DNSSEC on the other hand gives a clear definition of these technical issues.

The standards for TLS and SSL are easily misinterpreted and there is ample room for disagreement on their interpretation. Keeping that in mind it is a small miracle that both standards have been so widely adopted. The main cause for their popularity is that they are embedded in browsers. Given a choice, cryptographers generally prefer other, more technically-inclined ways of application-packaging their cryptographic structures.

Finally, DNS has its own requirements, including densely packed data (which rules out X.509 certificates completely) for optimal use of cache and bandwidth. Furthermore, it is desirable to keep the validation process as simple as possible, in order to retain the light-weight and almost-instant nature of DNS as much as possible; interpreting complex structures such as certificate chains introduces counter-productive overhead that is unbearable in a DNS environment.

COLOPHON

Authors

Paul Brand
Rick van Rein
Roland van Rijswijk
David Yoshikawa

Editor

Roland van Rijswijk

Layout and graphic design

Paul Eversdijk

Illustrations

Tycho van der Klip

Copyright © SURFnet B.V. 2008-2009

This paper is distributed under the terms of the
Creative Commons License version 3.0
"Attribution-Non Commercial-Share Alike" Netherlands

A copy of this license can be obtained online:
<http://creativecommons.org/licenses/by-nc-sa/3.0/nl/deed.en>



SURFnet
Postbus 19035
3501 DA Utrecht
The Netherlands

T +31 302 305 305
F +31 302 305 329
E admin@surfnet.nl
I www.surfnet.nl

January 2009