Development of DNS security, attacks and countermeasures

Karl Andersson
Linköpings Tekniska Högskola
karan496-at-student.liu.se

David Montag
Linköpings Tekniska Högskola
davmo024-at-student.liu.se

Abstract

We use the Internet every day. The Internet relies in its very foundations on DNS, the Domain Name System. What most people don't know is how fragile DNS actually is. This paper discusses the shortcomings of DNS, and how it is being secured. We explain common attacks, such as cache poisoning and DNS forgery. Then we mention some incidents briefly, before moving on the improvements. The improvements range from simple fixes, such as randomized ID numbers and acceptance policies, to more elaborate solutions such as TSIG and DNSSEC.

1. Introduction

The Domain Name System is a critical part of the Internet. It is the directory service responsible for (among other things) translating domain names to IP addresses. Since it is such a critical component, security is a very important issue in DNS. We will study the domain name system and the security issues related to it in chapter 2. The development of attacks will be discussed in chapter 3. Then we will move on to security incidents involving DNS in chapter 4, and how they were resolved. Finally in chapter 5 we will study how the protocol and its implementations have evolved to meet the security challenges. In chapter 6 we express some general thoughts and sum up the report.

2. Domain Name System

The Domain Name System is a system for resolving names into text/numbers. Its need is motivated by people generally being very bad at remembering arbitrary numbers, but being much better at remembering letters and names hierarchically arranged. Initially in the original ARPANET, HOSTS.TXT files were used. These files would be distributed to computers with FTP, and they would map every computer name to an address. The growth of the number of hosts on the Internet required a better general-purpose name service. There were many ideas but eventually we got to what we today call DNS. The DNS is structured as a distributed tree with caching. This is needed in order for DNS to work because of the many many domain names existing. It's still in development and numerous RFCs concerning DNS is still being released.[10]

DNS uses text separated with dots to build up a so-called fully qualified domain name (FQDN), for instance *www.example.com.*. This domain name consist of four parts, the root (which is empty), the top-level domain (*com*), the subdomain, e.g. a company or some other name (*example*), and a service (*www*). When a user wants to resolve a record for this domain name, he/she will send a request to a DNS resolver that will check its local cache. If nothing is found there it will ask a root server (.). The root server will reply with an NS record stating the name of the *com.* nameservers, along with some records indicating at which IP address the *com.* servers are. This extra information is called glue. Now the resolver will continue with sending a request to a com nameserver, which will reply with information about *example.com.* and so on.

2.1. Packet format

A DNS packet consists of five parts: the header (holds information about the packet itself), the question, the answer, the authority (which describes who is the authority for a domain name) and an additional section (which holds additional information that could be useful for the resolver). For an exact description of the DNS packet see RFC1035[2].

2.1.1. Header

The header consist of many different parts, the most important part from a security perspective being the ID number. The ID number "can be used by the requester to match up replies to outstanding queries."[2] It's notable that it says *can* and not *must*. This highlights the fact that the RFC was not written with security in mind. The ID is most likely used by every DNS server in use today as it significantly helps prevent cache poisoning attacks.

The OPCODE field specifies the type of question in the packet. The most commonly used OPCODE is the stan-

dard query. Another used OPCODE is NOTIFY, which is used to force a slave DNS server to initiate a zone transfer from its master.[11] There are also three other OPCODES: STATUS, UPDATE and IQUERY. The IQUERY was intended to be used to invert an ordinary record, i.e. to resolve A 192.0.2.1 to example.com. The IQUERY was obsoleted with RFC3425[13]. The STATUS OPCODE is not defined in any RFC and therefore remains unimplemented. The UPDATE OPCODE is defined in RFC2136[12] and defines a way to dynamically update RRs. There are an amount of flags which define how the query/response should be interpreted.

2.1.2. Resource Record

Resource records are used to represent different kinds of data. The A record is a 32-bit IPv4 address, the AAAA record is an 128-bit IPv6 address. Other commonly used RRs are NS, MX, CNAME and PTR. NS is the authoritative name server record and consist of a domain name. MX specifies a domain name which can be used for mail exchange. The MX record consist of a 16-bit preference number and a domain name. CNAME is a canonical name and instructs the resolver to continue its resolving at the domain name pointed at by the CNAME. PTR is used to reverse-resolve IP addresses to domain names. In IPv4, this is done by taking the IP address (for instance 127.0.0.1), reverse it (1.0.0.127), append ".in-addr.arpa." to it (1.0.0.127.in-addr.arpa.) and resolve it. The PTR record is used to give names to IP addresses and is often the hostname of the host. Many ISPs however give names reflecting the city, location and/or interface speed.

2.1.3. Question section

The question section is used to carry the query sent to an DNS resolver. The resolver copies it to the question section in the response, unchanged. One can have multiple questions in one packet, but usually only one question is sent. In the question, the type and class of the question are specified.

2.1.4. Answer section

The answer section contains the actual answer to the question. As with questions, it's possible to have multiple answers. The format for Answer, Authority and Additional section is the same, and consists of a domain name, a type, a class, a 32-bit time-to-live (in seconds), a length of the RDATA and finally the RDATA which is the data associated with a record.

2.1.5. Authority section

The authority section defines which nameservers are authoritative for a domain name. This is done by passing NS RRs for servers that are authoritative for the domain. This section can also optionally include a SOA RR for the domain.

2.1.6. Additional section

The additional section holds data that may be useful for the resolver/user when using the other sections. For instance, when resolving an A/NS/.. record for a domain name, the replying server sends along A/AAAA records for all the nameservers because of the bootstrap problem. It's notable that previously the data passed in the additional section was accepted into the cache without any filtering, which made cache poisoning very easy.

2.2. Shortcomings

The obvious shortcoming when looking at DNS is that it wasn't designed with any security in mind. The RFCs for DNS published in 1987 contain very few references to security. One example is, as mentioned above, that the ID number "can be used by the requester to match up replies to outstanding queries.". [2] This is hopefully done by everyone today but still it shows the mentality at that time. Other examples are that there is no real definition for how the resolver should handle RRs from the additional section. Should it only cache things from the same domain, or should it accept anything?

Another shortcoming that is inevitable with a caching infrastructure is that information can become stale and possible intentionally incorrect in some parts. This incorrect information will spread, and if the DNS infrastructure is built with too many servers caching from too few and not querying the authoritative nameservers, compromising these few will spread the incorrect RRs widely. The root servers are of course the most endangered here, but since they are so big targets, they're also probably well protected. Therefore there is a greater danger for the DNS servers that only act as caching servers for other caching servers.

There is no real integrity or confidentiality in DNS today. There is DNSSEC, but this is still not widely used and DNSSEC leaves some things for improvement.

3. Attacks on DNS

Many different attacks against DNS exists. Almost all of these focus on changing a record somewhere, and it can be done in many different ways. There are many ways to change a record but we'll concentrate on attacks aimed at the DNS protocol, and not on compromising an actual DNS-server by some exploit. The distributed nature of DNS, and

Answer:	Empty
Authority:	evil.com. 3600 IN NS ns.example.com.
Additional:	ns.example.com. 3600 IN A 192.0.2.1

Table 1. Additional section injection

the amount of different servers under different administrations in use make DNS vulnerable to attacks. Also, the caching infrastructure doesn't exactly improve security. Below, different types of attacks will be discussed.

3.1. Cache poisoning

Cache poisoning can be achieved by changing or adding a record to a nameserver's cache. An attacker can use this technique to change an A record to an IP address under his control, thus redirecting traffic to himself. Cache poisoning is the general concept of changing something in or adding something to the cache of a nameserver. What makes this technique very effective is the heavy use of forwarders. Forwarders are nameservers that a resolver forwards its incoming requests to. Thus if a record is poisoned in a forwarder, all resolvers that forward to it will also be poisoned. The widespread use of forwarders is discussed further in a white paper by Dan Kaminsky[14].

3.2. Additional section injection

Additional section injection occurs when unrelated data is passed in the additional section of a reply, in the hope that the targeted nameserver will cache this. The attacker sends a query for a domain name he controls to the targeted nameserver. The nameserver will then ask the attacker's nameserver. In the old days the attacker's nameserver would probably just pass the A records that he wanted to poison back in the additional section. Nowadays some more complicated way would be used by passing an NS record in the authority section saying that ns.example.com is the authoritative nameserver for his domain and in the additional section state that ns.example.com is at his IP address. See table 1 for an example.

3.3. DNS forgery

DNS forgery is an attack in which the attacker forges a reply to a DNS query. This is done by beating the reply from the real server back to the client. This scenario is of particular importance when it comes to wireless networks.

Every DNS query and reply contains a 16-bit ID number. The number in the reply must match the number in the query. Without this keeping of state, an attacker could keep flooding a victim with reply packets for a domain the attacker knows the victim will look up, i.e. google.com. When a query was made, the victim would accept one of the flooded reply packets instead.

The ID numbers make this harder, as the attacker has to match the ID number of the reply with the query. In a wireless network, where all traffic is seen by all nodes, DNS forgery is a big issue. The attacker could then simply intercept all DNS queries on the network, and send back forged replies to the victims he or she wants to attack. This won't work on a wired network though. On a wired network, the attack will need to calculate or predict the ID numbers of queries.

3.3.1. ID/port number prediction

If the attacker has no direct connection to the victim, and cannot intercept traffic, then the victim's DNS server must be targeted instead. This is done by sending a query to the DNS server, immediately followed by a reply to the query. What happens is, when the DNS server receives the attacker's query, it will recursively find out the reply, and send it back to the attacker. If the attacker spoofs the reply packet before it arrives from the actual DNS server that knows the real reply, the attacked DNS server will accept this false reply and cache it. When someone else then queries the DNS server for the spoofed domain, the victim will be returned the attacker's cached records.

You probably see the problem here. The attacked DNS server's recursive DNS queries each have an ID. Every reply must have the corresponding ID for the query it matches. This means that the attacker must send 65536 packets with different IDs to be sure to match the ID of the recursive query. This is a lot of packets. Here we can utilize the birthday paradox. If the attacker sends, say, 100 queries to the DNS server (which in turn will send 100 recursive queries), and then 100 replies with different ID numbers, the probability of one recursive query ID sent matching a spoofed reply ID is very high.

There's also the additional problem of matching the UDP port numbers of the requests and spoofed replies, i.e. the spoofed reply must be sent to the UDP port the recursed request was sent from. Some BIND configurations send all requests with source port 53. This would make it very easy to execute an attack like this. Randomized source ports would mitigate this problem. A birthday attack that bruteforces the port numbers is possible, but not very effective. This attack vector is hard to protect against. Also, some nameservers might be behind firewalls and thus can only use a fixed port.[4]

3.3.2. Phase-space analysis

Imagine if the ID numbers for the queries were generated by a counter. Then an attacker could, by setting up an own DNS server and querying the victim DNS for a domain that

the attacker's DNS server served, log the ID numbers of the requests. By simply knowing one or a few ID numbers, the attacker could, in this scenario, predict the next ID number and thus eliminate the need for guessing. If the port number is known, the poisoning could be done with two packets.

The above example used a counter to generate ID numbers. What if ID numbers were generated by a pseudo-random number generator (PRNG)? Would we be safe? The answer is no. The attacker could set up the above scenario again, and log the ID numbers. By performing phase-space analysis on the collected ID numbers, the next one in sequence can be predicted.[5] This poses a great risk if the attacker can obtain a few ID numbers from the DNS queries.

3.4. Amplification attacks

The DNS amplification attack gets its name from how it works. It can turn a few kilobytes of traffic into megabytes. This makes it very effective as a tool for executing distributed denial of service (DDoS) attacks. Here's not it works:

The attacker generally begins by compromising a nameserver and inserting a large TXT RR into a zone for which the nameserver is authoritative. The attacker then queries a list of public nameservers for the hacked TXT RR, putting it in the caches of the public nameservers. Public nameservers are nameservers that recursively resolve queries from the public Internet. The attacker finally executes the attack by sending spoofed queries to the list of public nameservers for the TXT RR (and possibly other RR). The spoofed queries have as their source address the victim's address. For each query sent by the attacker, the victim will get one packet. That's just 1:1, so what? [26]

The key point here is that the replies are much larger than the queries. This is made possible by RFC2671[27], Extension Mechanisms for DNS, that allows for increased UDP buffer sizes. Naturally, the nameservers will have to support RFC2671 in order to participate effectively in the attack. This way, a request with a size less than 100B can generate a reply larger than 4KB.

Now consider the scenario where the attacker has a botnet consisting of thousands of machines. Each of these machines could simply use its ISP's nameservers to execute the attack. With the amplification 100B→4KB and 1000 bots, that would yield a theoretical payload of 4MB on the victim, with just 100B per bot. Make every bot send 10KB worth of requests, and you'll have an effective DDoS.

4. Incidents

This section will cover some DNS-related security incidents that have taken place over the years.

4.1. The InterNIC incident

In 1997, an affiliate of the registrar AlterNIC, Eugene Kashpureff, poisoned the caches of major nameservers. People trying to visit www.internic.net, the homepage of the InterNIC registrar, would instead be presented with AlterNIC's page.[6] AlterNIC was an set of alternative root-servers with alternative TLDs. The cache poisoning was done by attacking a vulnerability that existed in BIND prior to version 4.9.6 and 8.1.1. The vulnerability allowed an attacker to redirect queries to the attackers nameserver and then pass along bogus glue records in the additional section. [24] So why was this very offensive attack done? In July Networks Solutions who at the time administrated InterNIC claimed ownership of the top domains .net, .com and .org. Their agreement with National Science Foundation was about to expire and they therefore claimed ownership of the domain name at the Securities and Exchange Commission. Kashpureff felt this wrong and said that domain names should be public, he therefore initiated this DNS attack which he later was convicted for. [21][22]

Still today alternative root servers exist but in a smaller extent then at the time of this incident. The Internet Architecture Board spoke harshly against alternative roots in RFC2826 since they divide the global Internet, the RFC finish with "There is no getting away from the unique root of the public DNS.". [23]

4.2. DDoS attacks

There have been two significant DDoS attacks against DNS over the years. The biggest one took place on October 22, 2002. During this attack, which lasted for about an hour, nine of the thirteen root nameservers were disabled. On February 6, 2007, another large-scale DDoS attack was launched against Internet nameservers. After this attack, US officials announced that they will be prepared to fight back with cyber counterattacks and, in the worst case, even actual bombings of attack sources.[7][8]

It was not because of a vulnerability that these attacks could be carried out, so why are we mentioning them? We're mentioning them because we want to emphasize how important proper DNS operation is for the Internet, and nowadays, for the world.

5. Security development of DNS

This section will cover how the security aspects of DNS have improved. It will discuss different technologies, along with their strengths and weaknesses.

5.1. Random ID numbers

As mentioned earlier, it is important that the ID numbers of queries are non-predictable. Therefore recent versions of popular nameservers use random IDs for their queries. Proper randomization can prevent prediction attacks. Phase-space analysis may still be effective though. Port numbers of requests are also randomized (by the OS though). These two factors combined make cache poisoning hard.

5.2. The additional section

Recent versions of BIND handle data from the additional section in a more sane way than they used to. Different policies can be employed, both on the server and client sides. The default policy is to only accept information that is related to the query, or that the responding DNS server is authoritative for. For instance, a caching resolver looking up www.yahoo.com may only cache glue records from yahoo.com, com and the root servers.[9]

There are different ways to evade the problem described in chapter 3.2. Depending on the scenario, different policies can be employed. We have looked at how MaraDNS[28], a DNS server package, deals with the problems related to the additional section.

When a reply as the one in table 1 is received, the nameserver shouldn't cache both records received, as all it really needs to know is the IP address of the nameserver for evil.com. Therefore it only caches:

```
evil.com. 3600 IN NS 192.0.2.1
```

This way, only the relevant information is cached, and the bogus information is automatically discarded.[29]

Naturally, records in the answer and authority sections must be for the domain being queried for. Other records are ignored.[29]

5.3. TSIG

Transaction signatures (TSIG) are a mechanism to secure communication between DNS servers, or more specifically, the dynamic updates sent between DNS servers. TSIG uses shared symmetric keys combined with cryptographic hashing to authenticate the sender of updates. The update and the secret key are hashed into a MAC. A time stamp is also added. The receiver of the update can then check the authenticity of the update by performing the exact same hashing procedure. The time stamp prevents the request from being replayed later.

5.4. DNSSEC

DNSSEC is a set of extensions to DNS that are meant to provide extra security. DNSSEC's goals aren't very well specified but the general idea is that it should provide data integrity and data origin authentication. This ensures that the original zone data not been altered in transit, and that it in fact is the original zone data that has been acquired. DNSSEC does not provide data confidentiality or any form of client authentication (for instance for access control). For DNSSEC to provide end-to-end data integrity, from the nameserver to the end user, the end user's recursive resolver (or better yet the end user's stub resolver), should be DNSSEC aware and have the trusted keys installed. This is a potential problem when it comes to both deployment and maintaining strong security.[15]

DNSSEC works by introducing a concept of signed zones. Signed zones contain one or many public DNS key RRs (DNSKEY), Resource Record Signature RRs (RRSIG), Next Secure RRs (NSEC) and optionally a Delegation Signer RR (DS). A resolver starts with an authenticated key which can be acquired in three ways:

1. It can be configured in the resolver and is then called a trusted anchor. Trusted anchors can either secure the whole hierarchy, or create a secure island.

2. The most common way is by verifying that a DS RR and its DNSKEY is signed by an authenticated key. This is part of the authentication chain that a resolver builds from a trusted anchor to a RRSIG used to sign a RR.

3. Find a corresponding key that has been signed by an old verified public key.

There are many reasons why DNSSEC isn't widely deployed. Some of these are important technical issues that need to be resolved, and others are political ones, such as who should hold the keys. The US government, the ICANN, the UN? An important technical issue that needs to be resolved is how trusted anchor key rollover should be resolved, i.e. how a potentially compromised key can be changed, when it is present in millions of resolvers. There are some privacy issues with the NSEC RR that allows for zone enumeration. NSEC is used to deny the existence of a domain name in an authenticated way. Also, since DNSSEC is relative new, more testing is require to strengthen the DNSSEC standard and the DNSSEC implementations.[17]

5.4.1. Resource Records

The description of the different RRs useds in DNSEC are taken from RFC4034 [25].

- DNSKEY - DNS Public Key
  The DNSKEY RR is used to store the public key whoes corresponding private key is used to sign resource record sets (RRsets, an RRset is a collection of RR).

example.com. 86400 IN DNSKEY 256 3 5 (AQPSKmyW ... av4w==)

Table 2. DNSKEY example

The RR consists of four parts, a flag field, a protocol field that MUST be 3, a algorithm field that defines what algoritm is used and last the public key. The algorithm that is mandatory to implement according to RFC4034 is RSA/SHA-1. See table 2 for an example of a DNSKEY RR.

- RRSIG - Resource Record Signature
  The RRSIG RR is used to store the digital signatures that signs an RRSet. The RRSIG signes a specific RRSet with a particulary name, class and type. The RRSIG consists of a specification on what type is covered, the algorithm used, the signer's name and a keytag to specify what DNSKEY should be used to verify the records, there is also an signature start and expiration time.

- DS - Delegation Signer
  The DS RR is used to delegate the zone signing authority. The presence of a DS indicates that a delegated zone is digitally signed, it also indicates what key is used to sign it. For *example.com.* a DS is present in the *com.* zone and NOT in *example.com.*, this simplifies the delgation but requires extra response processing. The DS RR consist of the DNSKEYs keytag, algorithm number and a digest of the DNSKEY RR, this way when verifying a DS one has verified the DNSKEY also.

- NSEC - Next Secure
  NSEC is used to deny the existence of a specific domain. NSEC consist of Next Domain Name and a type bitmap that identifies what RRs are present at that NSEC owner domain name. NSEC is used in a cirular way where each record points to the next one in canonical ordering (see [25], all the NSEC are signed with RRSIG. This enables the denying of existance of a domain name without needing access to the publickey, the domain nameserver simply sends back an empty record and the NSEC before and after the requested one.

- NSEC3 - Next Secure 3
  NSEC3 resource record is currently only in draft form and is meant to deal with the issue of enumeration of zones. Instead of including the next domain name a cryptographic hash of the next domain name is included. See [19] for more information.

5.4.2. Records verification

DNSSEC protects against an attacker trying to modifiy an existing record, inject extra records or dropping records via any of the attack vectors discussed earlier. It can do this by building a chain of DNSKEY and DS from a trusted anchor all the way to the zone. A RRset can from the security-aware resolver point of view be in one of the following states.

- Secure
  RRset and RSIG can be verified with a chain of DNSKEY and DS to a trusted anchor.

- Insecure
  RRset which is known to have no chain of DNSKEY and DS RR from any trusted anchor. This can occur when the zone has no DS record and this is verified with NSEC records, also it can happen when the RRset is a descendent of an unsigned zone.

- Bogus
  RRset and RSIG should be able to be verified with a chain of DNSKEY and DS RR to a trusted anchor, but the resolver is for some reason unable to do so because of missing/incorrect signatures. This could indicate an attack or missconfiguration.

- Indeterminate
  RRset that the resolver is unable to obtain the necessary DNSSEC RRs for an i thus unable to verify if the RRset should be signed or not. This can occure if the resolver is unable to contact a parent. [16]

Any record that is Secure or Insecure could be accepted since they are either secure of verified insecure they can therefor be accepted. Bogus records should be denied since the resolver is expecting a signature but is unable to accquire one. The tricker part is Indeterminate records, according to RFC4033 [20] its up to local policy whether this records should be denied or accepted. For maximum security they should probably be denied.

5.4.3. Zone enumeration

Zone enumeration is probably one of the biggest problems with DNSSEC, and is seen by some as a security vulnerability. Usually one doesn't want all the domain names in a zone disclosed to the public, because of obscurity reasons. Whether this is the correct way or not can be discussed. Still, the truth is that not many nameservers allow public zone transfers today, and to give away your zone as a consequence of enabling DNSSEC is unwanted. In the case of NIC-SE, they clearly state that all domain names are viewed as public data and therefore doesn't need to be protected.[18] The current solution in development is to use

hashed denial of existence records instead of the real record. This uses the NSEC3 RR, and an Internet Draft is available at IETF.[19]

To enumerate a zone one simply sends an NSEC type query to the nameserver. One will then receive the next name in the zone and can use this to enumerate the whole zone. See the appendix for a small shell script showing this.

5.4.4. Trusted anchor key rollout/rollover

Another key issue is that there is no adequate way to rollout or rollover the trusted anchor key at the root. [15] Rolling out a key is how all resolvers should acquire the key from an authority in a secure way. Rollover is when this key needs to be changed, changing the root key can be necessary if it has been compromised or is suspected to have been compromised. Work on this issue has been done, but no good way has been presented. Right now NIC-SE uses out-of-band communication with a mailing list and PGP signed keys.

6. Conclusions

Our conclusion is that DNS was not designed to be secure and was designed at a time when security was not the top priority. DNS is actually rather insecure, and there is much left to be wished. However, it works considerable well. The attacks available today are considarably more advanced than those used a few years ago. The authors beleive that DNS security and attacks on DNS will continue to play an important role, even in the future. DNSSEC is a step against a more secure DNS, but it leaves some things to be wanted, and is not quite ready for full deployment. Some of the barriers that DNSSEC has to overcome are non-trivial, such as who should have control over the root keys, and a number of technical problems.

References

[1] P. Mockapetris, *DOMAIN NAMES - CONCEPTS AND FACILITIES*, RFC1034, November 1987

[2] P. Mockapetris, *DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION*, RFC1035, November 1987

[3] M. Lottor, *DOMAIN ADMINISTRATORS OPERATIONS GUIDE*, RFC1033, November 1987

[4] C. Liu, P. Albitz, *DNS and BIND, Fifth Edition*, p. 271, May 2006

[5] Strange Attractors and TCP/IP Sequence Number Analysis, BindView RAZOR White Paper, *http://www.bindview.com/Services/Razor/Papers/2001/tcpseq.cfm*

[6] C. Liu, P. Albitz, *DNS and BIND, Fifth Edition*, p. 282, May 2006

[7] David McGuire and Brian Krebs, *Attack On Internet Called Largest Ever http://www.washingtonpost.com/ac2/wp-dyn/A828-2002Oct22, October 2002*

[8] E. Messmer, Network World, February 8, 2007, *http://www.networkworld.com/news/2007/020807-rsa-cyber-attacks.html*

[9] D. J. Bernstein, *http://cr.yp.to/djbdns/notes.html*

[10] András Salamon, *http://www.dns.net/dnsrd/rfc/*, DNS related RFCs, 2004

[11] P. Vixie, *A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)*, RFC1996, August 1996

[12] P. Vixie, Editor, *Dynamic Updates in the Domain Name System (DNS UPDATE)*, RFC2136, April 1997

[13] D. Lawrence, *Obsoleting IQUERY*, RFC3425, November 2002

[14] Dan Kaminsky, *Explorations In Namespace: White-Hat Hacking Across The Domain Name System*, *http://www.doxpara.com/cacm_kaminsky.pdf*

[15] Atkins & Austein, *DNS Threat Analysis*, RFC3833, August 2004

[16] AR. Arends et al, *Protocol Modifications for the DNS Security Extensions*, RFC4035, August 2004

[17] Thierry Moreau, *DNSSEC Deployment at the Root http://www.circleid.com/posts/dnssec_deployment_at_root/*, May 2006

[18] *NIC-SE statement regarding NSEC zone walking http://www.ops.ietf.org/lists/namedroppers/namedroppers.2004/msg00663.html*

[19] B. Laurie et al, *DNSSEC Hashed Authenticated Denial of Existence*, January 2007 *http://www.ietf.org/internet-drafts/draft-ietf-dnsext-nsec3-10.txt*

[20] R. Arends etal, *DNS Security Introduction and Requirements*, March 2005

[21] Janet Kornblum, *Temporary order issued against AlterNIC http://news.com.com/2100-1033-201733.html*, July 23, 1997

[22] Courtney Macavinta, *AlterNIC takes over InterNIC traffic http://news.com.com/2100-1033_3-201382.html*, July 14, 1997

[23] Internet Architecture Board, *IAB Technical Comment on the Unique DNS Root*, RFC2826, May 2000

[24] *http://osaka.law.miami.edu/ froomkin/articles/icann-antitrust.pdf*

[25] R. Arends et al, *Resource Records for the DNS Security Extensions*, RFC4035, March 2005

[26] R. Vaughn, G. Evron, 2006, *http://www.isotf.org/news/DNS-Amplification-Attacks.pdf*

[27] P. Vixie, *Extension Mechanisms for DNS (EDNS0)*, RFC2671, August 1999

[28] MaraDNS, *http://www.maradns.org/*

[29] MaraDNS, *http://www.maradns.org/cache_poison_protection.html*

Appendices

```
#!/bin/sh

NAME=se.

while true
do
  echo $NAME
  NAME=`dig +short NSEC $NAME @a.ns.se \
    | cut -d ' ' -f 1`
  sleep 0.1
done
```