What is wrong with DNS?

Markus Granström
Helsinki University of Technology
magranst@cc.hut.fi

Abstract

This paper presents a few problems related the Domain Name System (DNS) and reviews solutions to them. Since DNS is one of the cornerstone services of modern Internet it is imperative to guarantee its integrity and availability.

Some threats include an attacker trying to inject malicious DNS information into the DNS response message. Others stem from the fact that DNS is a huge interconnected system and contains a lot of interdependencies. The DNS may also be used for DoS-attacks to other services. Finally some protocol restrictions cause troubles nowadays.

Fortunately many of the vulnerabilities can be countered by running an up-to-date server software. However the Domain Name System Security Extensions (DNSSEC) would be useful to protect the DNS transactions in the long run. It requires the Extension Mechanisms for DNS (EDNS0) - a technique to add new options and functionality in a backward compatible way. Although these techniques exist and are considered effective they still lack world wide adoption.

KEYWORDS: domain name system, threat, vulnerability, dnssec, edns0, secure dns

1 Introduction

The function of the domain name system (DNS) is to translate host names into IP-addresses understood by the Internet infrastructure [8]. It is one of the oldest services still employed in the Internet. The current DNS dates back to 1987 [8, 9], but has been evolving ever since.

Several threats have been identified concerning DNS [5]. Two of the most well-known threats include cache poisoning attacks (sec. 3.1) and simple packet interception or man-in-the-middle attacks (sec. 3.2). Also larger scale problems caused by the complex interdependencies between names and name servers can cause undesireable effects [11].

Two major extensions to the DNS are reviewed in this paper, namely the Extension Mechanisms for DNS (EDNS0) (sec. 4.1) and the Domain Name System Security Extensions (DNSSEC) (sec. 4.3). The EDNS0 extends the DNS protocol to accomodate the growing needs of the system [14]. The DNSSEC on the other hand is an extension that provides end-to-end integrity between the actual DNS provider and the user [4].

This paper reviews the current threats to the domain name system and evaluates the solutions to those threats. We start with a brief overview on how the DNS works, in section 2. Section 3 presents a collection of threats related to DNS. Some of the threats presented here are not unique to DNS but rather general attacks that have a DNS specific implementation. In section 4 we review a few solutions proposed for the threats against DNS and try to evaluate how well they perform. Finally, in section 5 we summarize the subject and draw our conclusions.

2 A brief overview of DNS

The domain name system is a distributed database in nature and it is based on the client-server architecture. It consists of three main components: name servers, resolvers and the domain name space and resource records [8].

Name servers consist of a server process and a database containing the information about the domain tree structure [8]. Name servers are organized in a hierarchial structure so that each domain and subdomain have one or more authoritative servers. These servers publish the information about that domain and can also point to the correct server for information about any other part of the tree. At the top level of this hierarchy lies the root name servers, which provide information about the top level domains (TLDs), such as .com, .net, and country codes like .fi and .uk.

Resolvers are the client side of the client-server architecture in the domain name system [8]. The resolvers provide the clients the information requested by querying at least one name server. A resolver is usually implemented as a library directly accessible by client applications.

The domain name space and resource records specify a tree structured namespace and the data associated with the names [8]. Every node and leaf in the tree contains a set of information. DNS queries are used to extract information from these sets. Queries specify a domain name and the resource that is required. For example, a query requesting an address resource would return an Internet host address.

Regarding name servers, it is worth mentioning the two different operating modes: recursive and non-recursive, since in some cases the vulnerabilities might depend on the name server operating mode [5].

The recursive mode is the simplest from the client perspective. In this mode, if the name server does not have the information requested, it will follow the referrals leading to the source itself and returns the answer [8]. So from the client point of view the name server acts as a resolver and always provides either the answer or an error code.

The non-recursive mode on the other hand is the simplest from the server point of view. In this mode the name server replies with an answer, an error or a referral that leads closer to a correct part of the tree [8]. It is left to the client to follow
the referrals and perform the additional queries if needed. The non-recursive mode is considered as the default one and all name servers must implement at least this mode [8].

There are two kinds of data contained in the name servers: authoritative and cached [9]. Authoritative data is the complete database for a particular subtree. This set is called a zone. The cached data is DNS information gathered by a local resolver. This data may be not up to date but improves the performance when non-authoritative data is requested. Cached data has a defined life time called time to live (TTL) [9]. When this time period expires, the data is no longer usable and has to be refreshed.

For further reading and more detailed specification of DNS please refer to RFC 1034 and 1035 [8, 9].

3 Threats

3.1 Cache poisoning

Cache poisoning attacks abuse the weaknesses in the DNS protocol and its implementations [12]. As the name suggests, the attack takes advantage of the fact that DNS servers tend to cache information to improve the performance. If an attacker manages to supply bogus information to DNS server that in turn caches it, the cache is considered poisoned. In other words, any information that ends up in the name server cache but did not originate from an authoritative name server is considered malicious [13].

DNS allows the server to include additional information in the reply messages [8]. This additional information does not need to be related to the actual response in any way. The fact that the resolver then possibly caches this extra information gives the server a possibility to inject malicious data into the resolvers cache [12].

Another way to carry out a cache poisoning attack is to exploit a bug in BIND [2], the Berkeley Internet Name Daemon that is a widely used name server software in the Internet. Earlier versions of BIND did not randomize the query ID. In addition to source and destination IP addresses and ports, the query ID is the only form of authentication in a DNS response. This allowed the attacker to easily predict the next query ID and send a spoofed response. [13]

Birthday attack is a mathematical insight which raises the probability of hitting the correct randomized query ID of a DNS response [3]. The basic idea of the attack is to send \( n \) simultaneous recursive DNS requests for the same domain name and then flooding the name server with spoofed responses in the hopes of hitting a correct query ID and injecting false information into the servers cache. A flaw in BIND caused the name server to send multiple recursive queries for the same domain name. So instead of one response the server was expecting \( n \) responses, all with a unique query ID. This significantly increases the probability of hitting a correct query ID with a spoofed response. In contrast if the server were to send only one query for the domain name, the probability would be much lower. The power of the birthday attack can be expressed by the fact that it nears 100% success at around 700 packets whereas with 700 packets the conventional attack would have a success probability of 1.07%. [3, 13]

3.2 Packet interception

The basic types of packet interception attacks are man-in-the-middle attacks and eavesdropping attacks combined with spoofed response messages [5]. Usually the attacker tries to convince the resolver to believe something that benefits the attacker. This type of attack is not DNS specific but the nature of DNS requests makes it particularly easy to execute against DNS. Since DNS uses unsigned and unencrypted UDP messages to transmit queries and responses all the attacker needs is the ability to intercept these messages and inject his own substitute message [5]. This usually means that the attacker needs to have control over the transit or shared network.

3.3 ID guessing and query prediction

ID guessing technique is based on the fact that the queries between the client and the server are matched only by the ID field in the DNS header and the client UDP port [5]. The server port is a well known fixed value, therefore matching the transport protocol parameters is fairly easy. The ID field in the DNS header is only 16 bits long and the client source port is also a 16-bit value [9]. This gives the attacker only 2^32 possible combinations of port and id, which is not that much. The fact that in practice the client port number and the ID can be predicted from previous traffic and that the client port number can also be a fixed value instead of a randomized one usually lowers the number of possible combinations to 2^16 or even lower [5].

To be able to inject false information into the clients cache the ID and the client UDP port number alone are not enough. The attacker must have information about the query type (QTYPE) and the query domain name (QNAME) of the active query also. In this case the resolver might have a hard time telling the difference between the false and the correct response. [5]

Since this attack is based on guessing the ID and the client port, the attacker does not need to be in a transit or shared network with the victim. This might simplify the execution of the attack. On the other hand, the attack only succeeds when the attacker guesses correctly, which again lowers the probability of success. [5]

3.4 Name chaining

Name chaining attacks can be considered as a subset of cache poisoning attacks [5]. Several different types of name chaining attacks exist but they all include a resource record (RR) whose resource data (RDATA) field contains a DNS name or something that directly maps to a DNS name. Through these records the attacker might be able to inject malicious data into the victims cache and therefore redirect the victims query to suite the attackers needs.

Especially the CNAME, NS, and DNAME classes of RRs are especially bad in this case since they can cause the query to be redirected to a place of the attackers choice. The MX and SRV RRs can also be used to trigger further queries but they are not considered as harmful as the previously mentioned RR classes. The address records (A and AAAA) do not contain DNS names in their RDATA field but the reverse
entries (IN-ADDR.ARPA and IP6.ARPA) on the other hand do include a DNS name which means these records can be used in name chaining attacks.

The basic name chaining attack proceeds as follows:

1. The victim issues a query. This might be provoked by the attacker himself or some other party. The query might be directed to an unrelated name in which case the attacker is just using the query in order to inject false data about some other name.

2. The attacker feeds the false response by means of query guessing, packet interception or possibly even by being a legitimate name server.

3. The response then contains RRs with false information in the RDATA field. The object may be to either poison the victim's cache or redirect the next stage of the query to another place.

It is fairly easy for the attacker to instigate a DNS query to a name of his choosing. For example by embedding a small picture in an HTML email the attacker might be able to trigger the email client to request that picture and therefore make a DNS query for that particular name.

### 3.5 Betrayal by trusted server

The fact that many clients are only supplied with a stub resolver which needs a recursive DNS server to handle all the queries on it is behalf can raise a question of trust [5]. The server information could be provided by the users ISP through automatic configuration (e.g. DHCP). The server might provide false information due to a server bug, break in or it might even purposely provide unexpected information to the client in an effort of trying to genuinely be helpful or trying to further some other goal [5].

This scenario is particularly difficult in a mobile environment where the user might not always be able to choose a trusted DNS server of their choosing. The network might have port filtering in place and it could also only provide a few name servers whose trustworthiness can not be verified. [5]

### 3.6 DNS amplification attack

Denial of Service (DoS) attacks are a threat to almost every service in the Internet and DNS is no exception. The fact that makes DNS interesting regarding DoS attacks is that it can be used as an amplifier when performing a DoS attack [5]. This is due to the fact that DNS response messages tend to be much longer than the query messages. This means that an attacker could create DNS queries that seem to originate from the victim’s IP address. These queries would then make the DNS server send the responses to the victim. In this case the attacker would only need to create the queries, which add up to only a fraction of the traffic that ends up blocking the victims connection.

### 3.7 Authenticated Denial of Domain Names

An attacker might be able to remove certain RRs from a response message. This might pose a threat with RRs whose absence cause an action instead of an immediate failure. These kinds of RRs, namely MX and SRV, fall back to A records. It is still a bit unclear how big of a threat this really is but since it is an identified weakness it is reasonable to assume that someone somewhere will eventually find a way to exploit it. [5]

### 3.8 Protocol limitations

The DNS specification limits the UDP message size used in DNS messages to 512 bytes, excluding the UDP and IP headers [9]. This limit seems to be too small for features that might be desired now or in the future [14]. Although TCP can also be used to make DNS queries, it is recommended that UDP is used instead since it provides better performance and lower overhead [9].

The second full 16 bits in the DNS header defines which kind of query is in question (OPCODE), the response code (RCODE) and a bunch of flags [9]. This 16-bit space is already mostly utilized and does not allow any more flags or codes to be defined even though it might be required [14].

The first two bits of the format in which the domain labels are transmitted denote the type of the label [9]. Two of the four possibilities are currently allocated to denote a normal domain label and a pointer that is used in the compression scheme and two are left for reserve. Far more label types have been proposed than what the current specification allows [14].

### 3.9 Dependencies between names and name servers

Since the resolution of a domain name to an IP address might depend on more than one name server, there are complex dependencies between name servers. A DNS query might first need to contact one name server that in turn redirects the query to another name server whose IP address needs to be resolved. Therefore a query might directly or indirectly depend on several name servers and a compromise in even one of the servers might affect the integrity of DNS. [11]

A survey done by Ramasubramanian and Sirer [11] reveals that the resolution of a domain name depends on average on 46 servers. From this 46 servers only 2.2 are directly designated by the owner of the name. The rest of the servers involved in the resolution are totally out of the control of the name owner. They also show that 30% of domain names they surveyed could be hijacked by gaining access to only two name servers per domain using only well known exploits. A notable result of the survey also points out that 125 name servers control a huge portion of about 10% of the whole namespace. A fifth of these servers belong to educational institutes that may not provide enough security to guarantee integrity.
4 Solutions

4.1 EDNS0

The EDNS0 extension to the DNS protocol provides a backward compatible way to extend the existing protocol [14]. The current protocol contains many fixed fields which are no longer adequate.

As discussed in section 3.8, the amount of label types available is hardly enough. EDNS0 provides a solution for this problem by introducing new label type. The new label type uses one of the two available label types reserved in the DNS protocol, namely "0 1". This label type indicates an extended label type. The specifics of the extended label type are then encoded in the lower six bits of the first octet of a label. [14]

EDNS0 also introduces a pseudo-RR called OPT. This can be included in the additional data section of a message. It is called a pseudo-RR since it does not directly relate to any DNS data but instead is a part of a certain transport level message. This also means that the OPT RRs shall never be cached or forwarded. The OPT RR contains a fixed part for some popular protocol elements and a variable part for options encoded as \{attribute, value\} pairs. [14]

The sender includes its maximum UDP payload size (Maximum Transmission Unit, MTU) in the fixed part of the OPT RR. This enables the responder to expand the limit of 512 bytes to the desired value. Both the sender and the responder should however take into account the smallest MTU size on the path since it can be smaller than that of either end point. Also the extended RCODE values and flags are included in the fixed part of the OPT RR. [14]

4.2 Secret key transaction authentication for DNS (TSIG)

TSIG is a way to ensure the integrity of the communication between a client and a server or between two servers. In other words TSIG provides point-to-point integrity and authentication for transactions. This is achieved using shared secrets and one way hashing functions. [15]

TSIG adds a new meta-RR called TKEY, which is not a conventional RR. This RR is always created per transaction and must never be cached. Hence the name meta-RR. The TKEY RR is used to authenticate between peers that have established a shared secret. TSIG uses HMAC-MD5 algorithm to calculate the signature for the message and this signature is stored in the TKEY RR. [15]

TSIG requires that the both parties in the transaction share a secret key but it does not specify a way to distribute this secret [15]. A way to share the secret keys safely is defined in RFC 2930 [6]. This RFC defines a new meta-RR called TKEY which is used to agree on and distribute the secret keys. The TKEY mechanism supports variety of ways to agree on the shared secret, for example basic Diffie-Hellman key exchange protocol can be used. The TKEY RR is similar to the TSIG RR in the way that it is not supposed to be cached or forwarded at any time.

4.3 DNSSEC

4.3.1 Overview

The main purpose of DNSSEC is to introduce data integrity protection and origin authentication into DNS. The extensions also provide means to distribute the public keys. It is good to note that confidentiality is not provided by DNSSEC which means that DNSSEC is only concerned with the security of the DNS data, not the security of the channel. The extensions are implemented by new types of RRs and protocol modifications. [4]

DNSSEC introduces four new resource record types: Resource Record Signature (RRSIG), DNS Public Key (DNSKEY), Delegation Signer (DS), and Next Secure (NSEC) and two new header bits: Checking Disabled (CD) and Authenticated Data (AD). Since the new payload increases the DNS message size the EDNS0 extension is also required by DNSSEC. [4]

The RRSIG record contains a digital signature which is the signature for the zone data [4]. The signature is associated with the whole zone. With this signature a DNSSEC enabled client can authenticate the zone’s data if it has obtained the public key for the zone.

The zone public keys are stored in the DNSKEY record. The private keys are not accessible through DNS and should be kept offline if possible. The clients can obtain a zone public key by the means of normal DNS resolution. To reliably obtain a public key the target key must be signed by another key already authenticated or by a preconfigured authentication key. The way the system works is that the resolver creates an authentication chain through the already authenticated keys all the way up to the new key. The keys authenticated earlier are used to authenticate the new key. This means that there has to be a way to authenticate the first key. This is done by a preconfigured trust anchor in the resolver. [4]

The DS record resides on a zone’s delegation point. The record indicates the public key(s) corresponding to the private key(s) that are used to self-sign the child zone’s DNSKEY RRset. The child zone’s data is then signed using the private key(s) corresponding to the public key(s) in this DNSKEY RRset. [4]

The NSEC record is used to provide authentication for negative response on a name or type non-existence. The NSEC records produce a chain that describes the empty spaces between the domain names. It also contains information about the existing RR types. NSEC records are signed and authenticated with the technique described above. [4]

Even though channel security mechanisms such as TSIG [15] and IPsec [7] could be used to sign the DNS messages it would not be feasible because of the high processing costs per DNS query. Also these techniques do not provide any end-to-end integrity between the actual DNS provider and the user. [5] The integrity is only provided on a point-to-point basis. The end-to-end integrity provided by DNSSEC requires that DNSSEC is properly implemented on all nodes.

DNSSEC provides protection against many of the threats described here. As mentioned earlier DNSSEC must be implemented properly through out the path of the query, including the client. This implies that the client must check
all the DNSSEC signatures itself and not rely on anyone else. This way it ensures the integrity of the data it receives. Since DNSSEC provides end-to-end security it is a powerful protection against packet interception attacks and forged responses. It is possible for the resolver to check that the data associated with the name is actually from the proper authority. On the other hand, there is very little that DNSSEC can do against attacks like denial of service. In fact, DNSSEC might even make the situation worse since the response messages carry more payload and could therefore be used in amplifying the attack. [5]

It is important to notice that resolvers that do not perform DNSSEC signature checking themselves must use some other means of verifying the integrity of the communication. TSIG can be used to guarantee the integrity of the communication channel to a recursive DNS server which does check the DNSSEC signatures [5].

4.3.2 Problems

Osterweil et al. [10] conducted a study on the state of the global DNSSEC adoption. They gathered data from October 2005 through January 2008 using a monitoring project called SecSpider [1]. They aimed to quantify the status of the DNSSEC deployment through the attributes of availability, verifiability and validity.

In the study [10] they made a few interesting discoveries. First of all, they discovered that unforeseen availability problems can occur because of the larger message sizes caused by the added payload. As mentioned in section 4.1 the EDNS0 enables the participants to advertise their MTU size. The problem arises when the smallest MTU on the path from the sender to the responder is smaller than the response message size and the larger packets may be fragmented or dropped. When this happens the server might become unreachable.

Second, the idea of the authentication chains is that a resolver would only have to configure one trust anchor which would enable it to build the authentication chains to most of the secured zones. The study shows that the DNSSEC hierarchy today is very fragmented. In fact, 662 of the 871 secure zones included in the study had no authentication chain leading to them. This means that instead of the ideal one trust anchor the resolver would have to configure 662 anchors in order to verify all the DNSSEC data. [10]

Lastly they discovered that stale DS records in parent zones might make it impossible to verify the DNSKEYs in the delegate zones. This would also break the authentication chain. Their numbers show that 9% of the authentication chains were broken in this fashion. [10]

5 Summary and conclusion

The current domain name system has many issues, weaknesses and actual vulnerabilities. Some of them can be fixed by running an up-to-date server software since they only appear in outdated pieces of software. Others are not so much of implementation shortcomings but actual flaws of the protocol. These protocol issues can be addressed with extensions like EDNS0 and DNSSEC. General denial of service attacks are even more difficult to address and DNSSEC might even make the situation worse during an attack.

DNSSEC seems to be the answer to threats to the security of DNS transactions. The low level of adoption however makes it more or less worthless at the moment. Global deployment would be desirable but there seems to still be some obstacles that need to be resolved before large scale deployment can be achieved. DNSSEC is generally considered complicated and it requires time and effort to maintain.

References


