The Domain Name System from a security point of view

Simon Boman Patrik Hellström
Email: {simbo105, pathe321}@student.liu.se
Supervisor: David Byers, {davby@ida.liu.se}
Project Report for Information Security Course
Linköpings universitet, Sweden

Abstract

The Domain Name System is used everyday by all the people surfing the Internet. Most of the people probably don't even have an idea that it exists and how important it is in order to make the Internet fully operational. Among those people who have heard of the system or even knows what it is, only a small group probably knows how easy it is to exploit it. In this paper we bring up the fundamentals of the domain name system but also what different attacks that exist towards it such as DNS cache poisoning, some incidents that have happened in the past and what is being done in order to make the system more secure, for example DNSSEC.

1. Introduction

Internet today is very widely spread and used by millions of people every day. In order to make access to various hosts easy, the Domain Name System is used to map a domain name to an IP address. Today there are over 100 million different domain names registered in the world [11] and the numbers are increasing every day. With the increasing numbers in mind and the fact that DNS was never developed with consideration of security makes it a very feasible target for various attacks.

In this paper we will talk about the Domain Name System from a security perspective and discuss what weaknesses are there, what are the consequences of a security breach and what are being done in order to make DNS more secure are the main questions this report will focus on.

In chapter two we begin with a brief history of DNS, then we move on to the technical specifications and how DNS works. In chapter three we will bring up some of the possible attacks and in chapter four a few successfully performed attacks will be mentioned. In chapter five we move on to the developments that have been made in order to make DNS more secure and what further evolvement, concerning security, DNS will face in a near future. The last chapter summarize the report and we discuss our findings of the security of the DNS.

2. The Domain Name System

In this chapter we will talk about the Domain Name System in general. We will bring up some history, the protocol and how it works.

2.1 History

When DNS was invented in 1983 the practice of using a name instead of a numerical address had actually been in use for decades. It was when ARPAnet was invented the method was firstly implemented but not in the form as a distributed system like the DNS. Back then one file were used. This file, named HOSTS.TXT, was retrieved around once a week by every computer from a computer at the SRI's Network Information Center. The file contained all of the currently active mappings between numerical address and a name. As the Internet grew a more scalable and dynamic system was needed and that was the beginning of the domain name system. [17]

2.2 The DNS protocol

In the Internet, a DNS message is sent either in UDP datagrams or TCP datagrams, both on server port 53. In this chapter we mention some details about what the message contains. All content is taken from RFC1034[1] and RFC 1035[2].

2.2.1 Message format

The DNS protocol has a standard message format to send DNS queries and responses. At the top of the message format it's a header containing a number of fixed fields, and four sections which carry query parameters and resource records.

2.2.1.1 Header

The header section is always present and it in-cludes fields that specify which of the remaining sections that are present. It also specifies whether the message is a query or a response. The first field of the header is a 16 bit identifier which is generated by the requesting resolver. A major field in the header is a four bit field named an OPCODE which divides different queries. Of 16 in total, one OPCODE is part of the official protocol (standard query), two are options (inverse query and status query), one is out of date, and the rest are unassigned.

2.2.1.2 Question

The question section is used to carry the query name and other query parameters, like QNAME, QTYPE and QCLASS. Where the QNAME field has the domain name represented as a sequence of labels. The QTYPE specifies the type of the query and QCLASS specifies the class of the query, for example IN for the Internet.

2.2.1.3 Answer

The answer section usually contains records that directly answer the question of the message, where several answers are possible. The Answer, Authority and Additional sections all have the same format which consists of a domain name, a type, a class, a TTL (time-to-live), the length of the RDATA and at least the RDATA field.

2.2.1.4 Authority

The authority section holds the names of the name servers which are being sent back to the client. The section can optionally carry the SOA resource records for the authoritative data in the answer section.

2.2.1.5 Additional

The additional section contains extra information that may be of value to the client, for example the IP address of a name server in the authority section.

2.2.2 Resource records

In the Domain Name Space each node has one or more resource records, which contain information about the domain name. A resource record consists of several types of records. The most common types of resource records are the A (address) record which maps a hostname to a 32-bit IPv4 address and the AAAA record which maps a hostname to a 128-bit IPv6 address. Further on there are NS, SOA, CNAME, PTR and MX. The NS (name server) record maps a domain name to a list of DNS servers authoritative for that domain. The SOA (start of authority) record specifies authoritative information to the DNS server, for example about an Internet domain. CNAME (canonical name) record is commonly used when running multiple services (such as a mail server and web server) from just one IP address, for instance m1.isp.com and www.isp.com. Instead of mapping hostname to an IPv4 address, the PTR (pointer) record maps an IPv4 address to the canonical name for that host. The PTR uses reverse DNS lookup for that address. MX (mail exchange) record is used for mail servers as it maps a domain name to a list of mail exchange servers for that domain.

2.3 How it works

This part describes how the domain name system is constructed and how it works in practice. We will also mention a common configuration and how a typical DNS-request could be performed.

2.3.1 DNS servers

The domain name system is a hierarchical system built up by many different DNS servers. Some of these servers are what is called authoritative DNS servers which provide the functionality of publishing information about its domain and all the domains and name servers beneath it. At the top of the hierarchy are the root name servers which are used when a query for a top-level domain name is performed.

2.3.2 DNS resolvers

A DNS resolver provides the functionality of looking up resource records information for different nodes. The resolver has the knowledge of how to communicate with name servers by sending requests and passing answers to the requesting instance. In most cases the resolver has a cache in which it saves the most recent answers. In this way a lot of time can be saved by the fact that a common query get its answer directly from the resolver’s cache and doesn’t have to be looked up every single time.

2.3.3 Common configuration

In the simplest scenario (which is probably also the most common one) the user interacts with the

DNS server in an indirect manner; the user uses a program, for instance a web browser, which sends a query to the local DNS resolver (implemented by the operating system) which in turn will check its cache for a match. If no match can be found the resolver will send a request to a DNS server (most likely your ISP´s name server if no other configuration has been made). The DNS server has a cache as well and if an answer to the current request can be found in it, the name server will send an answer to the requesting resolver which will forward the answer to the user program. If no matching cache entry can be found, the DNS server will act as a resolver which tries to find an answer. If the DNS server doesn't have any information in its cache it will do this by beginning to query the root server about the address to the authoritative server for the correct top-level domain (for example .com if a host under a .com address is to be looked up). It then continues in an iterative way by asking the "com-server" about the address to the requested host and so it continues. When the answer has been successfully obtained the DNS server pass this on to the local resolver which forwards it to the user program. [2]

3. DNS attacks

When the domain name system was developed, security was not the main concern and when looking at the earliest released RFC:s describing the system one won't find much information about the security of it. When the system began to grow at a rapid speed it obviously became a tempting target for attackers. Nowadays the system's security has evolved but is still not flawless and the numbers of unpatched and old version system out on the Internet is probably huge.

In this chapter we will mention some of the attacks that have been possible and some that still are possible to perform to the Domain Name System.

3.1 DNS forgery

One of the most common attacks on the domain name system is the one referred to as DNS forgery. In this attack the attacker might eavesdrop on a connection and if a DNS request is seen the attacker sends a forged reply to the client, beating the reply from the DNS server. Since DNS requests and answers are sent over UDP in an unsigned and unencrypted packet, this attack is quite simple to perform for an attacker who is able to intercept packets on a shared network. Hence in a wireless network where all data can be seen by every node this attack is very straightforward.[3]

3.2 DNS cache poisoning

Another form of DNS forgery is the case when the attacker is not on the same network as the victim. In this case the victim's DNS server has to be attacked instead. This is done by inserting false data in the DNS server's cache so that when the victim sends a request to the server the forged answer will be sent back to the client.

3.2.1 Old-school

In the early versions of the DNS software the additional section of an answer was trusted blindly and therefore all the data in it would be put in the DNS cache. This could be used by an attacker by setting up his own DNS server and in some way trick a resolver to connect to it. When receiving a request, an answer with a specially crafted additional section containing the A records that are to be poisoned, is sent to the requesting client. In more recent versions of name server software this has been fixed and all the data in the additional section will be validated.

3.2.2 New-school

Despite that the information in the additional section is validated in modern name servers there is still one way to perform cache poisoning. The method is similar to the one with DNS forgery but with some extra steps.

The attack is performed by sending a request to the target's DNS server directly followed by a forged response. If the server doesn't already have the answer in its cache a recursive lookup will be made. If the attacker's answer beats the answer from the recursive lookup it will be cached and the correct answer from the recursive lookup will be discarded. All other request to the DNS server querying for this domain name will now get the attacker's answer. There is however one problem with this scenario.

As seen in the previous chapter every DNS packet has a 16-bit id number which is used in order to match an answer to the correct request. When the targeted nameserver performs its recursive lookup the query packet will get a unique id number and in order for the attacker to make the targeted name server accept his answer, he must make sure to get the very same 16-bit id number or the packet will not be taken as a correct answer. The simplest way of getting around this issue is by flooding the name server with faked answers which all have different id numbers and hope that one of them is correct (1 out of 65535). However, this is not a very effective solution but with the help of a so called birthday attack, explained in the next section, the odds of succeeding will increase.

3.2.3 ID/port number prediction

One way of increasing the chance of guessing the correct id number is by using a so called birthday attack which has its roots in the birthday paradox. This says that "*in a gathering of 23 persons, it's likely that 2 of these persons will have the same birthday date*" [12]. This paradox can be used in the way that if the attacker sends *n* requests to the victim's DNS server and directly after sends *n* spoofed answers his chances of success increase dramatically. In fact if he sends 300 requests and 300 spoofed answers; his chance of success is 50% [4].

Another way to increase the odds in succeeding is by making "qualitative" guesses. If the id number is created by a counter, the attacker could easily find out the number the counter is at for the moment and start his guessing in the surrounding of that number. On the other hand, if a pseudo random number generator is used to create the id number, phase space analysis can be used.

This leaves the attacker with one last problem; the port number. Most of the name servers always use port 53 to send its requests. If this is the case this part of the attack becomes very simple. If the port number on the other hand is chosen by random by the name server the attack becomes much more difficult. Also in this case the attacker could resort to performe a phase space analysis if he knows how the pseudo random number generator, used by the name server to create the port number, works. The birthday attack could also be used.

3.2.4 Phase space analysis

Phase space analysis is a mathematical method which can be used when trying to predict id and port numbers. Suppose that the id number is created by a pseudo random number generator. With the help of phase space analysis an attacker could find out just how random these numbers are and he might even find a pattern in how they are created. This increases the chance to guess the correct id number drastically.

3.3 DNS cache snooping

The process of DNS cache snooping is when you determine whether a given resource record is (or is not) present on a given DNS cache. [5] Today there are two ways to figure this out, and that is what we are going to present here.

3.3.1 The Ecological Way

Using iterative queries is the most effective way to snoop a DNS cache. The first query sets the RD (recursion desired) bit in the query to zero and asks the cache for a given resource record of any type. If the answer is cached the answer will be valid, other-wise the cache will reply with information from another server which better can answer our query better, or more usually, send back the content of the root.hints file.

3.3.2 The Polluting Way

If you're not allowed to use non-recursive queries, you still have some chance to succeed with DNS cache snooping by using recursive queries. But there is one major disadvantage with using it; it will pollute the cache, so if a given record isn't present in the cache when you do the attack, it will be there after the first query has been sent. One way to see if the attack has been succeeded, is to check the TTL field of the cached response, if it's much lower then the initial set TTL the cache hasn't been polluted. Another way to check if the cache has been polluted is to observe the time that the query takes to process. If the query time is approximately equal to the round trip time (RTT) of a packet to the server, then the answer probably should be in the cache. Otherwise the cache has been polluted.

The attacker can after the exploit determine which domains that have recently been resolved via the attacked name server, and also which hosts that have been recently visited. So why would an attacker want to have that information? One example is that an advertising agency could see which web-surfing patterns some people has, and then use this to make pointed publicity to those people. A more serious purpose is if an attacker was interested in whether your company utilizes the online services of a specific financial institution, they would be able to use this attack to build a statistical model regarding company usage of earlier mentioned financial institution. You can also use this information to find business-to-business partners, external mail servers and so on.

3.4 Betrayal by trusted server

Another way to do a packet interception attack is the trusted server who appears to not be so trustworthy, whether on purpose or by an accident. A lot of client machines only have a basic configuration, and use trusted servers to perform all of their DNS queries. Most commonly the trusted server is supplied by the user’s ISP and advertised to the client thorugh DHCP or PPP options.

This problem is specific for frequent travellers who bring their own computer and expect it to work wherever they are. These travellers need trustworthy DNS services without the need to care about who operates the network which they are connected to, or what equipment the operators are using.

The simplest solution of this problem is to choose a more trustworthy server for the client, but in practise that is not an option for the client. Almost in every network environments a client machine has only a limited set of recursive name servers to choose from, but there is no guarantee that some of them will be trustworthy. Even if the user of the client machine is willing to port filtering or perform some other forms of packet interception, it may prevent the client host from make a DNS request. The source of this problem is not the DNS protocol, this kind of exposure is a threat to DNS clients, and to just change to another recursive name server is not a good option of defence.

The only thing that differs between this kind of betrayal and a packet interception attack is that the client has freely sent its request to the attacker. To protect against this is the same method used as with a packet interception attack. The resolver must either use TSIG (or some other signature check) or check the DNSSEC signature by itself, to secure that the server is a trusted one. But it is important to be careful, just because you are using TSIG doesn’t guarantee that a name server is trustworthy. The only thing TSIG can do is to protect the communication with the name server who the resolver already decided to trust in an earlier stage.

3.5 Denial of service attacks

As with any kind of network service, DNS is also vulnerable against denial of service attacks. Some people may think that DNSSEC will defend against it but that is wrong, it may make the problem even worse for resolvers that check signatures. When you check signatures it both increases the amount of messages that needs to answer a query and the processing cost per DNS message. Another thing that DNSSEC does not defend against is that DNS servers also are at risk of being used as a denial of service amplifiers. Since DNS response packets usually are bigger than DNS query packets, this can be used to do a denial of service attack.

Another way to do a denial of service attack is to make an amplification attack. This is based on the fact that small queries can generate larger UDP packets in response, when you do a recursive DNS attack. In the initial DNS specification, UDP packets were limited to 512 bytes. With an amplification factor of 8.5 and 512 byte response, you could at most use a 60 (512/8.5 ~ 60) byte query to generate this response. With this in mind it is quite easy to make a denial of service attack if the attacker has a botnet consisting of a couple of hundred machines.

4. Successfully performed attacks

4.1.1 DNS cache poisoning

On March 4 2005 the Internet Storm Center reported that they were getting a lot of reports from several sites indicating that users were being re-directed to various malware sites. Sites as google.com, ebay.com and weather.com all directed the user to a “bad site”. It later was discovered that the affected users were using a system which used a Symantec firewalls with DNS cache. The firewall contained a vulnerability which had been exploited in order to poison the firewall’s cache and in that way re-direct the users when surfing the web for common sites. [16]

4.1.2 DDoS attacks

On October 21 2002 a large distributed denial of service attack targeting the thirteen DNS root nodes was performed. The attack began at approximately

20:45UTC and lasted for about 2 hours until 22:00UTC. Attacks targeting the root nodes of the domain name system are not unusual but what made this attack so serious was that all of the thirteen nodes were attacked at once and nine of them were disabled. A analyze of the attack however showed that no end-users noticed the attack in no more way than that a lookup might have taken some extra time compared to the normal case. [13]

5. Countermeasures against DNS attacks

Over the years several things concerning security have been made to the domain name system. The problem with the additional section was fixed by always validate the data it held, the problem with id number prediction was fixed by implementing better pseudo random number generators and the port number over which the DNS messages are sent are now chosen by random. The next major event regarding security is to implement DNSSEC. This will add functionality such as origin authentication of DNS data and data integrity.

5.1 DNSSEC

DNSSEC stands for DNS Security Extensions and is a collection of a few new resource records and protocol modifications that will add more security to the DNS. There are three major functionalities that it provides:

- Origin authentication of DNS data
- Data integrity
- Authenticated denial of existence

The former two deals with the concept of how to make sure that a resolver is indeed communicating with the correct authoritative DNS server and that no information in a packet is changed during transit. The latter one is used in order to “prove” that a certain record does not exist. The way DNSSEC provide these functionalities is by using public key cryptography to sign and authenticate DNS resource records sets (RRsets). The DNS zone administrator begins with signing all of the RRsets in the zone with a private key and then he publishes these signatures for each RRset in the zone file along with the public key. The next step is to get this public key signed by his parent zone administrator. In this way a “chain of trust” is added to the domain name system.

In order to achieve this four new resource records are added to the DNS protocol. These are Resource Record Signature (RRSIG), DNS Public Key (DNSKEY), Delegation Signer (DS), and Next Secure (NSEC).

5.1.1 Resource records

The four added resource records added by DNSSEC are presented in this section.

5.1.1.1 RRSIG

The Resource Record Signature, RRSIG, is used to store digital signatures and is used in the DNS authentication process. In order for a validator to identify the DNSKEY RR, which will be used to verify the signature, the RRSIG contains, apart from the signature, the signer’s name, the algorithm used and the key tag. Another thing that is specified in the RRSIG is a validity time which tells how long the signature is valid.

5.1.1.2 DNSKEY

In order for a resolver to validate a signature it will need the signer’s public key. This is provided by the DNSKEY RR. The way this works is that a zone signs its authoritative RRsets by using a private key and the resolver then uses the public key in the DNSKEY to validate the signature. In addition to the public key, three other fields are provided in the DNSKEY RR. The first one is a 7 bit field containing various flags, which for example indicates if a DNS zone key or some other type of DNS public key is held by the record. The next field is the protocol field which have the value 3. The third field is the algorithm field which tells what cryptographic algorithm that has been used.

5.1.1.3 DS

The DS, Delegation Signer, RR is used to indicate that a delegated zone is digitally signed and which key is used to sign it. The record refers to a DNSKEY RR by storing the key tag, algorithm number and a digest of the DNSKEY RR. By authenticating the DS, a resolver is able to authenticate also the DNSKEY RR to which the DS RR points. The place at which the DS record is stored is at the upper, or parental, side of a delegation. [14]

5.1.1.4 NSEC

To provide authenticated denial of service the NSEC (Next Secure) RR is used. The record consists of two fields where the first one indicates the Next Domain Name in the zone that has authoritative data. In this way it proves that there are no names in between the NSEC's owner name and the name in the Next Domain Name field. The second field of the record is a bitmap which identifies the different RR types that exist at the NSEC RR's owner domain name.

The NSEC RR comes with one drawback, an attacker who repeatedly asks queries for NSEC records can eventually retrieve all of the names in the zone. This concept is known as "walking the zone" or zone enumeration. [15] In March 2008 NSEC3 was released in order to address this problem. See [18] for more details.

5.1.2 How it works

When the server has got a request from the client, DNSSEC adds additional data to the responses that provide information to allow the client to authenticate the RRset data response. With the point of view of transferring the protocol between a query agent and an authoritative name server, it's an addition of a RRSIG part to the data response where a response can be generated. If there is no authoritative data to respond in the query the use of NSEC RR response and its companion RRSIG is added. It's also added an RRSIG response covering records in the authority section and one or more RRSIG responses to records in response at the additional section.

The client can check the hash of the RRset data matches the decrypted RRSIG hash. To generate the hash of the data, the client takes the RRset response and uses the algorithm referenced in the RRSIG record. To decrypt the hash in the RRSIG record, you encrypt the RRSIG value with the DNSKEY public key. To accomplish this, the client must also have the DNSKEY record for the zone.

At the part of the additional section of a DNSSEC response, you would normally get the DNSKEY. If the DNSKEY is not validated within some locally defined period, the client also needs to validate the DNSKEY value. This results that the RRSIG record on the DNSKEY value needs to be verified. Due to the domain zone key validation, a trust chain back to a trust anchor point is created. If the domain key is not already a trust anchor, then the client needs to query the parent zone for the DS record of the child zone, and this will returns both DNSKEY RR and an RRSIG value, and a public key. The returned public key needs to be validated using the DNSKEY of the parent zone, and that parent zone public key must be validated and so on… This will construct a trust chain that probably, leads back to a trust anchor. When it has, the DNS response has been validated.

5.1.3 DNSSEC issues

To make a non-secure system like DNS secure, when it is of such importance in the internet and so widespread, is not an easy task to accomplish. This has lead to that DNSSEC has some problems. Here we will bring up the most critical issues:

- It's very complex to implement DNSSEC and it includes some unpleasant cases that require careful coding. Trivial zone configuration errors or expired keys can cause serious problems for a resolver who uses DNSSEC.

- As we have mentioned earlier, DNSSEC increases the size of DNS response packets. This will make DNS servers who uses DNSSEC, be even more effective as denial of service amplifiers.

- In most cases key rollover is really challenging, specifically in the case with keys for the DNS root zone. That is: how can a key, which is present in millions of resolvers, be changed?

- The zone enumeration issue is a vulnerability which has always been present in DNSSEC and has been a difficult issue to resolve. But in March 2008, NSEC3 was developed to solve this kind of issue.

6. Conclusions

One of the conclusions we have made during the writing of this report is that even though the domain name system is very widespread, used by millions of people every day and not that secure, it still performs very well. But with the rapid growth of the Internet and the increased knowledge about its technologies, more and more people are sure to find out what "bad things" can be done with the domain name system. This makes the work concerning security in DNS even more important and to continue the work on DNSSEC (and other security technologies as well) is of great importance and should never end.

References

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

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

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

[4] Joe Stewart, DNS Cache Poisoning – The next Generation, *http://www.ida.liu.se/~TDDD17/literature/dnscache.pdf*, January 2003

[5] Luis Grangeia, DNS Cache Snooping or Snooping the Cache for Fun and Profit, *http://www.rootsecure.net/content/downloads/pdf/dns_cache_snooping.pdf*, February 2004

[6] AR. Arends et al, DNS Security Introduction and Requirements, RFC4033, August 2004

[7] AR. Arends et al, Resource Records for the DNS Security Extensions, RFC4034, March 2005

[8] AR. Arends et al, Protocol Modifications for the Security Extensions, RFC4035, March 2005

[9] Geoff Huston, DNSSEC – The Theory, *http://ispcolumn.isoc.org/2006-08/dnssec.html*, August 2006

[10] Geoff Huston, DNSSEC – The Opinion, *http://ispcolumn.isoc.org/2006-10/dnssec3.html*, October 2006

[11] Domain Counts & Internet Statistics, *http://www.domaintools.com/internet-statistics/*

[12] DNS spoofing techniques *http://www.securesphere.net/download/papers/dnsspoof.htm*

[13] Paul Vixie et al, Events of 21-Oct-2002, *http://d.root-servers.org/october21.txt*

[14] O. Gudmundsson, Delegation Signer (DS) Resource Record (RR), RFC 3658, December 2003

[15] S. Weiler et al, Minimally Covering NSEC Records and DNSSEC On-line Signing, RFC4470, April 2006

[16] Kyle Haugsness, Global DNS cache poisoning attack?; Update..., *http://isc.sans.org/diary.html?storyid=469*

[17] Domain Name System, *http://en.wikipedia.org/wiki/Domain_Name_System*

[18] B. Laurie et al, DNS Security (DNSSEC) Hashed Authenticated Denial of Existence, RFC5155, March 2008