DNS

Security Guide

- DNS fundamentals
- DNS Security
- Vulnerabilities and threats in DNS
- Fortifying DNS
- DNSSEC

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO

INTECO
Instituto Nacional de Tecnologías de la Comunicación

Autor

Antonio López Padilla

Coordinación

Daniel Fírvida Pereira

CONTENTS

1 PURPOSE OF THIS GUIDE. 5

2 BASICS OF DNS. 6

WHAT IS THE DNS? 6

THE COMPONENTS OF THE DNS. 6

Domain Name Space. Hierarchy and Syntax. 7

Name Servers. 9

Resolvers. 10

DNS RECORDS. FORMAT AND TYPES. 10

DNS COMMUNICATIONS AND TRANSACTIONS. 12

DNS Protocol. 12

DNS Messages. 12

DNS Transactions. 16

CRUCIAL CONCEPTS. 21

3 SECURITY IN THE DNS. 22

THREATS AND VULNERABILITIES IN THE DNS. 22

Attack Vectors and Threats in a DNS Scenario. 22

Vulnerabilities and Weak Points in the DNS Specification. 23

DNS CACHE POISONING AND DNS SPOOFING. 25

Description of Attacks. 25

Measures against Cache Poisoning Attacks. 26

4 DENIAL OF SERVICE ATTACKS. 29

DNS AMPLIFICATION ATTACKS. 29

Description of Attacks. 29

Protecting a Server against DNS Amplification Attacks. 30

DENIAL OF SERVICE (DOS). 31

ATTACKS AGAINST THE DOMAIN REGISTER. DNS HIJACKING. 32

Description. 32

Measures against DNS or Domain Hijacking. 32

5 FORTIFYING A DNS SERVICE. 34

SECURITY OF THE BASIC SYSTEM AND SOFTWARE ENVIRONMENT. 35

Operating System. 35

Software Configuration. 35

Network Topology. 39

Internal Monitoring. 42

Summary of Measures in the Base Environment of the DNS Service. 42
SECURITY MEASURES IN TRANSACTIONS. 43
Security in DNS Queries and Responses. 43
Security in Zone Transfer Transactions. 46
Security in Notifications. 47
Security in Dynamic Updates. 48
Summary of Measures for Protecting Transactions 49
SECURITY MEASURES FOR PROTECTING DATA. 49
Zone Files. Start of Authority Record Parameterization. 49
Restricting the Information Provided by Various Types of Record. 50
Summary of Measures for Protecting Data 51

6 DNSSEC. 52

WHAT IT IS AND HOW IT WORKS. 52
Components and Operation. 52
Difficulties in the Use of DNSSEC 57
Deployment of DNSSEC. 58

LISTS AND REFERENCES 59

REFERENCES 59
DOCUMENTATION 60
LIST OF ILLUSTRATIONS 60
LIST OF CONFIGURATIONS 61

APPENDICES 62

TRANSACTION SIGNATURE. TSIG. 62
PRACTICAL EXAMPLES OF THE USE OF DIG 64
USEFUL LINKS AND TOOLS 72

1 PURPOSE OF THIS GUIDE.

The aim of this guide is to offer an overview of the DNS service, to describe the principal attacks to which this protocol is subject through inappropriate use being made of it, and to provide guidelines for good practice for application in making it more secure.

The guide is intended for operators and administrators of systems and networks and has the purpose of aiding them in implementing and reinforcing the service.

Although the focus of this document is on the DNS in general, particular emphasis is laid on the open-code software BIND for the examples and implementations suggested, since this is by far the most widely used package

This document is made up of five principal sections:

I. Basics of DNS.

   This explains the concepts, objectives and functioning of a DNS system.

II. Security in the DNS.

   This section identifies possible attack vectors in a typical DNS scenario and the assets affected.

III. Vulnerabilities and Threats in the DNS.

   The weaknesses intrinsic to the design of the DNS protocol are explained, as are the principal attacks taking advantage of these.

IV. Fortifying DNS.

   This section investigates the security measures that should be implemented in the three main planes of attack on the DNS service: Infrastructure of the DNS Service, Communications and Transactions, and Data.

V. DNSSEC.

   Finally, an introduction to DNSSEC is given. This is a development in DNS security in which the introduction of encryption is intended to give the DNS service an effective mechanism for the historical vulnerabilities of the design.

2 BASICS OF DNS.

This section describes the elements that make up a DNS infrastructure, their names and hierarchical organization, and the protocol in itself. Details are given of the format of fundamental messages, operations and transactions, with the aim of giving a clear view of the concepts necessary for understanding the vulnerabilities affecting the protocol.

WHAT IS THE DNS?

The *Domain Name System* (DNS) is a system distributed across the world that is scalable and hierarchical. It offers a dynamic database associating IP addresses of computers, services or any other resource connected to the Internet or a private net with information of various types. It supports both IPv4 and IPv6, and information it is stored in the form of *Resource Records* (RR) of varying kinds, as they can store IP addresses or other sorts of information. This information is grouped into zones, corresponding to a name space or domain, and maintained by the authoritative DNS server for these.

Fundamentally, the DNS is responsible for translating IP addresses of network resources into names easily read and memorized by people, and vice versa. This action is known as "DNS resolution". In this way, a user-friendly mechanism is established for locating and identifying resources. It is common to use the analogy of a telephone directory in which it is possible to find the number associated with a given name or the name linked to a given number. In this comparison, the numbers would represent IP addresses and the names, records in the domain space.

THE COMPONENTS OF THE DNS.

The DNS has a structure involving three main components:

- **The Domain Name Space:** This consists of a hierarchical tree structure in which each node contains zero or more records (Resource Records, or RRs) with information about the domain. From the root node, situated on the highest level, branches run out, making up the zones mentioned above. These in their turn may contain one or more nodes or domains, which likewise can be divided into sub-domains at lower levels in the hierarchy. See *Illustration 1. Hierarchy of the Name Space.* Hierarchy of the Name Space
- **Name Servers:** These are servers responsible for maintaining and providing information about the name or domain spaces. On the one hand, there are servers that store full information for one or for several sets in the name space (domains) for which they are responsible. These are called authoritative servers for the zones or domains in question. On the other hand, there is a different type of server which stores sets of records for different zones or domains, which it obtains by consulting the authoritative servers that correspond to them (recursive searching). This information is stored locally for a short period (cached) and is renewed periodically. These are termed cache servers. The name servers and their inter-linkage achieve distribution and redundancy in the domain space.
- **Resolvers:** These are cache servers or client programs responsible for generating the necessary queries and obtaining the requested information so as to pass it on to the requesting user.

- **Hierarchical Structure.**

The DNS comprises a domain name space organized as a tree hierarchy in which nodes are linked, each of them representing a level in the domain space. The highest level in the entire hierarchy is the root domain, represented by "." (a full stop or period, read as "dot"). Just one level lower are the *Top Level Domains* or TLDs. These act as mother nodes for other lower levels known as second level TLDs. The hierarchy continues successively downwards until it reaches a final node that represents a resource. The name formed by the entire chain is called the *Fully Qualified Domain Name* (FQDN).

A zone is a portion of the domain name space for the administration of which is delegated to a DNS server that acts as "authority" for that portion or domain. This server is known as the *authoritative server* for the zone.

The hierarchy commences in the root zone ".", which is the highest level. Although it is normally not shown, all complete domain names end in a final full stop or dot ".", which indicates the end of the space in the root zone. For example, *"www.example.com"* is really *"www.example.com."*, where the final dot at the extreme right represents the root zone. This full domain designation is what is termed the Fully Qualified Domain Name (FQDN).

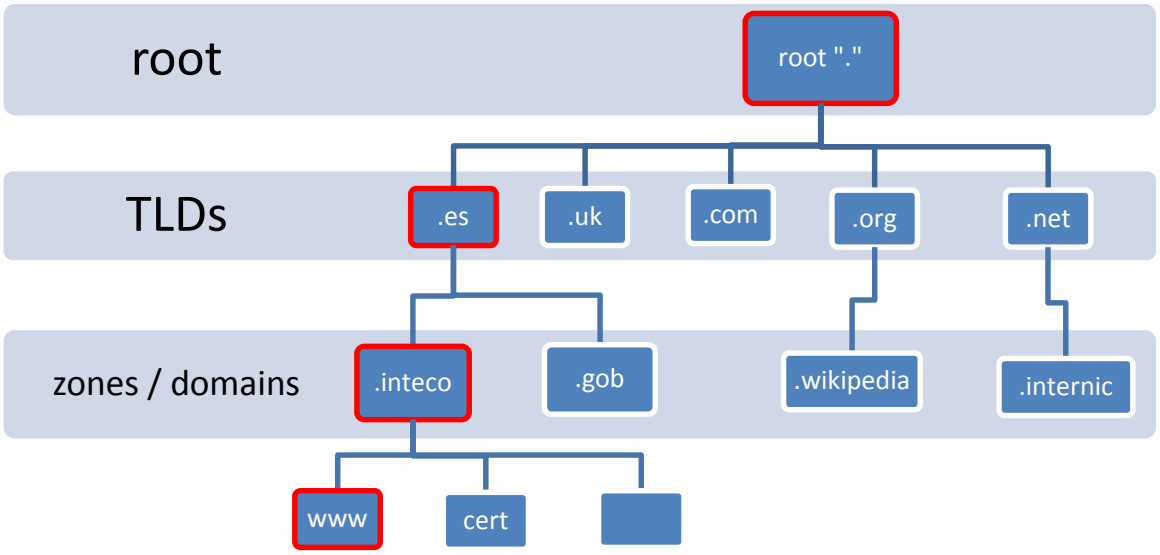


Illustration 1. Hierarchy of the Name Space.

- **DNS Naming.**

Depending on its position in the hierarchy, each name in the domain name space is made up of one or more labels separated by a dot ".", each of them with a maximum length of 63 characters. A final FQDN name may contain up to a maximum of 255 characters, including the dots ".".

These labels are built up from *right* to *left*, where the rightmost label represents the Top Level Domain (TLD) for the domain. For example, .**es** is the TLD for the zone *inteco.es*. Labels are separated by a dot ".".

Following the TLD, each label added to the left represents a sub-division or sub-domain. As indicated, each label may be up to 63 characters long and can in turn be sub-divided into other sub-domains, as long as the final FQDN does not exceed the maximum of 255 characters. This standard provides some flexibility when the hierarchy of sub-domains dependent upon a given domain is being designed. Finally, the leftmost part of the FQDN usually indicates the name of a machine or end resource, generically known as a host.

In domain names no distinction is made between upper and lower case letters. For instance, the domain names *www.mysite.com* and *www.MySite.com* will be considered identical

- **The Address Domain Space IN-ADDR.ARPA**

In the DNS the domain *in-addr.arpa* is used to define the IP address space. Thanks to this domain, inverse resolution of an IP address into its corresponding name can be guaranteed. This facilitates searching for them on the Internet.

The sub-domains of *in-addr.arpa* have a structure of up to four labels (IP version 4), each of which would represent one byte of an IP address. Thus, for instance, information about the IP address *213.4.108.69* would be located in domain *69.108.4.213.in-addr.arpa*. It may be seen how a hierarchical criterion is followed from *Illustration 2. Domain 69.108.4.213.in-addr.arpa.*

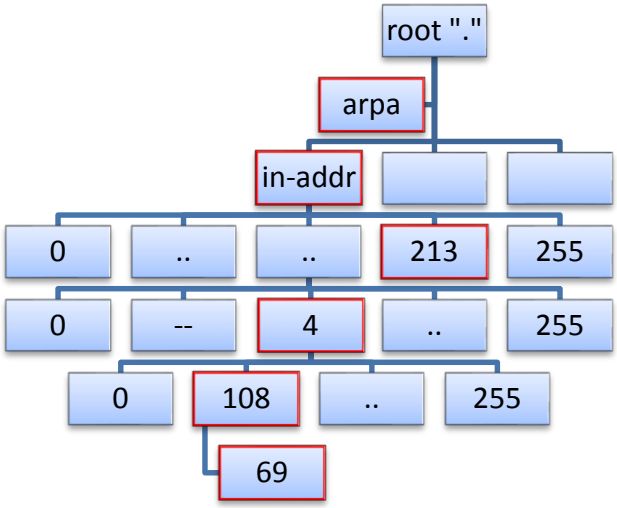


Illustration 2. Domain 69.108.4.213.in-addr.arpa.

The next illustration shows an example of inverse resolution using the Domain Information Groper (DIG) utility.

```
kuko@DNS ~ dig -x 213.4.108.69

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> -x 213.4.108.69
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63960
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;69.108.4.213.in-addr.arpa.     IN      PTR

;; ANSWER SECTION:
69.108.4.213.in-addr.arpa. 21599 IN     PTR     www.interdomain.org.

;; Query time: 144 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Mar  3 10:21:32 2014
;; MSG SIZE  rcvd: 76
```

Illustration 3. Inverse Resolution of IP 213.4.108.69.

NAME SERVERS.

A name server is a computer that is used for storing and providing information about the name space and address space. It thus provides the translation (or resolution) of a name into an IP address and vice versa. This information is termed the DNS Record and will be described in more detail below.

- **Authoritative Servers.**

An authoritative name server is one that maintains zones that are locally stored and provides responses to requests relating to them. It should be remembered that a zone is a set of domains that in turn contain information about records. Authoritative servers provide responses only for the domains for which they have been configured by the administrator. Authoritative servers may be *masters* or *slaves*. In *masters*, or primary servers, definitive versions of records are kept and administered, these being transferred to *slave* authoritative servers which hold a copy that is updated every time a change occurs. This updating is called a *zone transfer*.

When a domain name is registered through a registration service, a request is made for the allocation of a primary server and at least one secondary server, so as to provide redundancy in the case of failure of any of the servers and thus to keep the information about the domain accessible. Generally, *primary servers* are *master authoritative servers* and *secondary servers* are *slave authoritative servers*. When it is an authoritative server that provides a response to the client, this is marked with a flag that indicates that it is an authoritative answer or AA. When the client receives the response from some other intermediate cache server, the answer is received as not authoritative. The next illustration shows the difference between an answer obtained from an authoritative server and one got from a cache server:

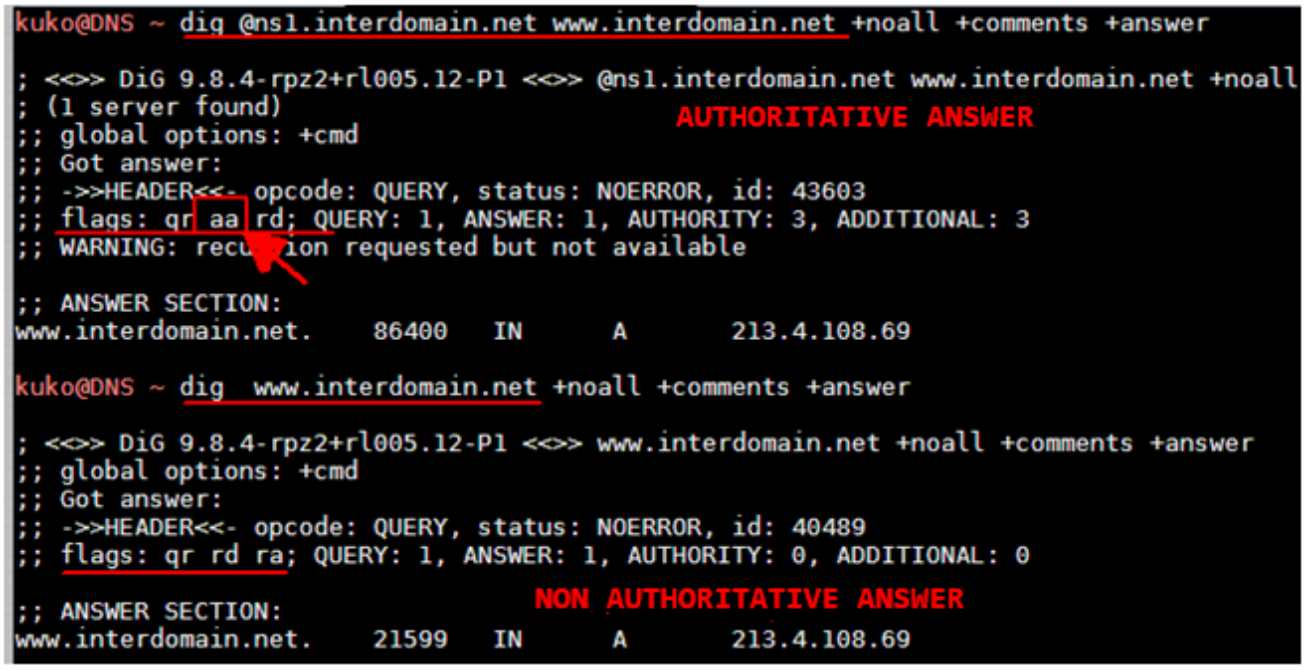


Illustration 4. Authoritative Answer and Cached Answer. Note the flag aa (authoritative answer).

- **Cache Servers.**

DNS cache servers store information concerning DNS queries for a given period of time termed *Time To Live* (TTL) for each DNS record. Cache servers optimize the use of the network by reducing DNS traffic on the Internet. This is because they store records that have been consulted, and can thus provide them directly without having to repeat the recursive search. They also reduce the load on authoritative servers, especially those for the root zone, or root servers.

RESOLVERS.

Resolvers are programs or services with which users interact through their machines to generate a DNS query. They are responsible for formatting requests in accordance with the specifications necessary for the DNS message and managing communication with the server for sending and receiving information about the records required.

DNS RECORDS. FORMAT AND TYPES.

A domain name is identified with a node in the DNS hierarchy. Each node contains a set of pieces of information called records (*Resource Registers,* RR) for which it is responsible (the authority).

DNS Records: Resource Records (RR).

There are various types of records, each identifying one type of information. This information is formatted as a record made up of six fields, used when the information mentioned is transmitted in DNS messages. The following table describes the six fields possibly present in a DNS message.

Field	Description	Length (bytes)
NAME	Name of the domain to which the record belongs.	Variable string
TYPE	Code for the record type.	Two bytes
CLASS	Code for the record class.	Two bytes
TTL	Time in seconds during which the record is kept cached.	Four bytes
RDLENGTH	Indicates the length in bytes of the field RDATA.	Four bytes
RDATA	Variable-length string describing the record in accordance with its type and class.	Variable string

Table 1. Record Format. Resource Record (RR).

The **TYPE** field contains a code that identifies what type of record is involved. There is a large range of types of records defined in various RFCs[1] to cover an equally large range of functions. Some of the commoner types are shown in the following table:

TYPE (Value of the TYPE Field)	Function
A = Address	Translates (resolves) names of resources into IP addresses, version 4.
AAAA = Address	Translates (resolves) names of resources into IP addresses, version 6.
CNAME = Canonical Name	Allows the creation of additional names (aliases) for the resource.
NS = Name Server	Indicates which server(s) store information for the domain subject of a query.
MX= Mail Exchange	Associates a domain name with a list of mail exchange servers for that domain. It has a balanced load and priority for the use of one or more mail services.
PTR = Pointer	The inverse of record A, translating IPs into domain names.
SOA = Start of Authority (for the zone)	Indicates the primary DNS server for the zone, responsible for holding information relating to it.
HINFO = Host INFOrmation	A description of the CPU and operating system holding information about a domain. Usually kept hidden.
TXT = TeXT	Permits domains to provide additional data.
LOC = LOCation	Allows indication of the geographical co-ordinates for a domain.
SRV = Services	Information on the services offered.
SPF = Sender Policy Framework	Aids in combatting Spam. In this field there is a specification of which host or hosts are authorized to send mail from the given domain. A server receiving mail consults the SPF to compare the IP from which it has arrived. It use is intended to be given up in favour of the TXT field[2].
ANY = Any	Requests all records available.

Table 2. Commonest Values for the TYPE Field.

[1] RFC: Requests for Comments (RFCs) are a set of notes about the Internet and systems connected to it which commenced publication in 1969.
[2] IETF.RFC 6686 Resolution of the Sender Policy Framework (SPF) and Sender ID Experiments https://tools.ietf.org/html/rfc6686

The **CLASS** field is usually set at the value IN (Internet) for DNS records related to *host names*, servers or, in inverse resolution, IP addresses. There are also the classes CH (Chaos) and HeSiod (HS) for other, less common, systems.

In the **TTL** field there is a numerical value that indicates the time in seconds for which the record will be cached. A value of 0 indicates validity only for the transaction under way and that the associated record will not be cached. SOA records always have TTL equal to 0.

The **RDATA** field describes the contents of the record in accordance with the type indicated in the TYPE field: SOA, A, NS, MX, and so forth. The length of this information is shown in the field **RDLENGTH**.

DNS COMMUNICATIONS AND TRANSACTIONS.

DNS PROTOCOL.

DNS uses port 53 for communications, both UDP datagrams and TCP packets. DNS activity generally uses UDP datagrams, as they require fewer processing and network resources. When the size of a response exceeds the maximum specified in the DNS standard for a UDP packet (512 bytes, not counting IP or UDP headers) and use is not made of ENDS0[3] (which allows extension of the DNS request up to 4Kb), TCP is used because of the need to keep control over the transport layer so as to ensure correct transmission. In this case, the server responds with the flag *truncated* (TC) and the client retries the response through TCP. Other operations, such as zone transfers, use TCP immediately.

The implementation of DNS with UDP as the main basis for its communications implies the presence of a multitude of threats related to the intrinsic lack of reliability of transmissions under this protocol. As there is no control over the data transmitted by UDP, it is taken for granted that the source is reliable and that the response is always received by the requester. This has a great impact on the security of communications and constitutes and easily exploitable attack vector. This will be discussed below in relation to the security of the protocol.

DNS MESSAGES.

- **Generic Format of DNS Messages.**

All communications in the DNS protocol follow a standard format called *message*. The message is divided into a HEADER and four sections: QUESTION, ANSWER, AUTHORITY and ADDITIONAL. Depending on the type of message one or more sections may be null. The HEADER is always present, as it contains important information about the message contents.

SECTION	Description

[3] See Section 2.4.2.4. Format of the DNS Extension Mechanism EDNS0.

HEADER	Contains information about the type of message. Includes fields giving information about the number of entries in other sections of the message.
QUESTION	Contains one or more requests for information (queries) sent to the DNS server.
ANSWER	Contains one or more records responding to the query or queries.
AUTHORITY	Contains one or more records indicating the authoritative server for the domain in question.
ADDITIONAL	Records with additional information not essential for responding to the query

Table 3. Generic Format for a DNS Message.

- **HEADER of a DNS Message.**

The **HEADER** section of a DNS message consists of 16 bytes, broken down into the following fields:

ID: (16 bits = 2 bytes). The first two bytes are given over to the identity of the message. This field is of particular importance, as it identifies the packet and will be the target for attack when any attempt is made to falsify a message.

QR: (1 bit). This is used to indicate whether it is a query (0) or a response (1).

Opcode: (4 bits). This indicates the type of query as a standard query, an inverse query, a notification, a dynamic updating or a server state.

Flags (4 flags each of 1 bit). **AA**: Authoritative Answer. **TC:** Truncation – this indicates that the message is truncated because it has gone over the maximum length permitted in the transmission. **RD:** Recursion Desired, specifying that a recursive query is being requested. **RA:** Recursion Available – this denotes a response offering the possibility of recursion.

Z: (3 bits). Reserved for future uses.

RCODE (4 bits): A fixed field in responses to indicate their status as No Error, Format Error, Server Error, or Rejected.

QDCOUNT, ANCOUNT, NSCOUNT, ARCOUNT: (16 bits). Fields intended to specify the number of entries or records in the sections QUERY, ANSWER, AUTHORITY and ADDITIONAL.

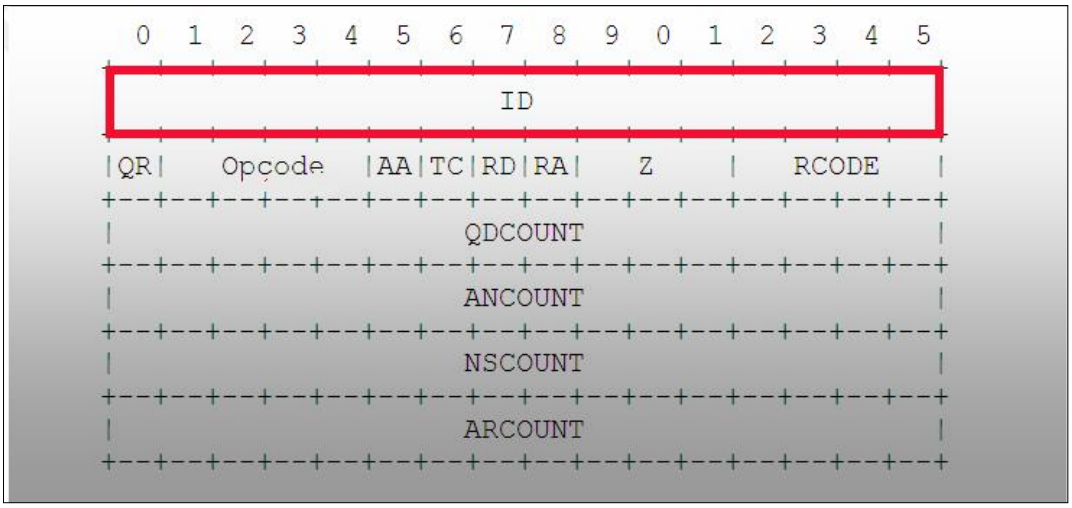


Illustration 5. Section Header in a DNS Message.
The ID Field, Identifier for the Message. Source: RFC1035[4].

- **The Format of DNS Query and Response Messages.**

Depending on whether the DNS message is a question or an answer, one or another of the fields may be absent.

DNS Query Messages (QUESTION).

In *DNS query messages* there is a **QUESTION** field which contains the query being directed to the DNS server, which the client (resolver) formats according to the three-field structure shown in the following table:

DNS Query Message. QUESTION Field.	
QNAME	**Indicates the domain about which the question is being asked.**
QTYPE	**Type of information (record) required in the query**.
QCLASS	**Class of record**

Table 4. The QUESTION Field in a DNS Query Message.

In the **QYTPE** field, which indicates the type of information, all the types defined in the **TYPE** list (Table 2. Commonest Values for the TYPE Field.) are valid, as are other additional values allowing the specification of operations, such, for instance, as **AXFR** in zone transfers. In the **QCLASS** field, all the values for **CLASS (IN, CH, HS)** defined in the record format are valid. This field generally takes the value **IN** (INternet). An example is given in Illustration 6. An Example of an A Type DNS Query

[4] RFC1035. http://www.ietf.org/rfc/rfc1035.txt

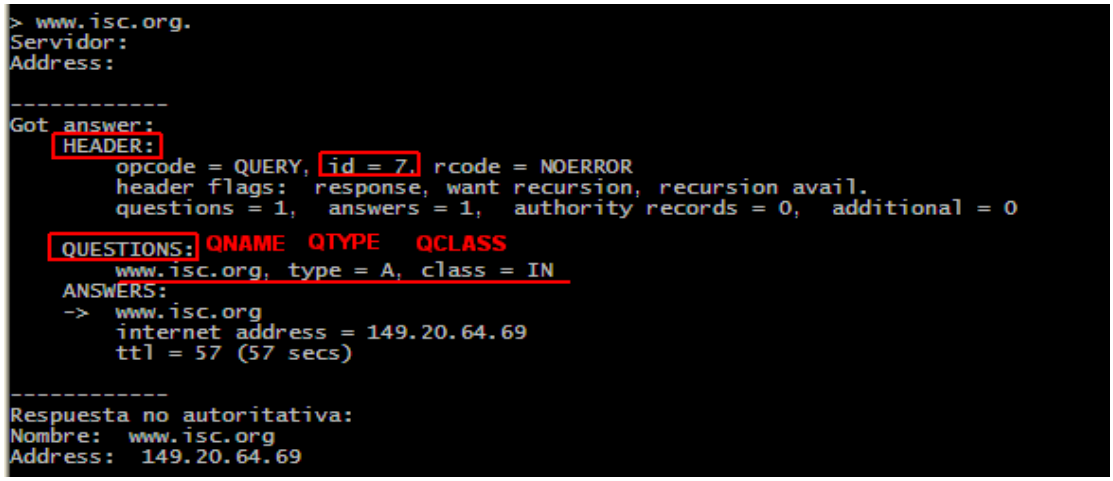


Illustration 6. An Example of an A Type DNS Query.

DNS Response Messages

In DNS response messages the sections **ANSWER, AUTHORITY** and **ADDITIONAL** appear, all following the DNS record format (RR) described in Table 1. Record Format. Resource Record (RR). Hence, they are made up of the fields **NAME, TYPE, CLASS, TTL, RDLENGTH**, and **RDATA**.

- **The EDNS0 Format Extension Mechanism for DNS.**

DNS messages may make use of the extension mechanism defined in RFC2671 and thus allow communication of messages of a greater length than the pre-fixed 512 bytes in UDP. This function permits the use of a larger buffer for UDP datagrams. It is generally employed in operations that need a length of more than 512 bytes or in zone transfer operations. In versions of Bind from 9 upwards, the EDNS0 format is used by default. A server highlights its capacity to use EDNS0 by specifying a pseudo-record called OPT in the **ADDITIONAL** section of the DNS message. This can be identified in a query with the DIG utility, where it is shown as "OPT PSEUDOSECTION", as may be observed in the following illustration:

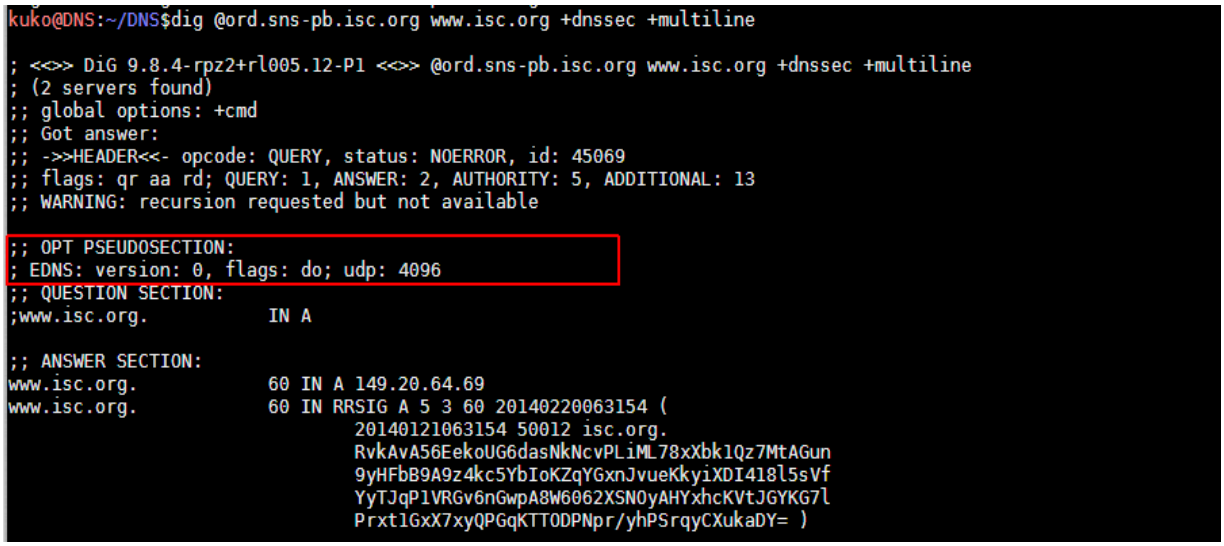


Illustration 7. EDNS0 Extended Format. 4096 Byte UDP.

See also the example in Appendix B, Practical Examples of the Use of Domain Information Groper (DIG).

DNS TRANSACTIONS.

The commonest transactions in the DNS are:

Questions and Answers (DNS Queries).

Zone Transfers: A mechanism for replicating zone files between servers.

Dynamic Updates: A mechanism used to bring zone files up to date in a DNS server.

Notifications: Transactions used by an authoritative server to notify changes in its zones database.

- **DNS Queries.**

These are the commonest type of DNS transaction. Queries may involve a question or an answer. A DNS query of this sort has its origin in a resolver and is directed to a DNS authoritative server or cache.

DNS queries carried out by a resolver may be *iterative* or *recursive*:

a) An ***iterative query*** is one in which the resolver (client) requests the DNS server to return the best response based on its zone or cache files. If the resource asked for is not to be found in the server itself, it will in its response return a ***referral***, that is, a pointer to the authoritative server at the lowest level of the domain requested, to which it must immediately turn to continue the iteration. For example, if server A is questioned about the domain *www.mydomain.org*, and this server A has not got this information, it will respond with a referral to the authoritative server of the root domain "." so that it can be asked for the name. The resolver will then continue the query iteratively, asking the root server for the domain, which will return a referral to the authoritative server for the domain .org. The resolver repeats (iterates) the process until, by running through the referrals, it reaches the authoritative server for the desired domain, from which it obtains the answer or an error (if there is no such record). Normally the final resolver requests a ***recursive query*** from the DNS server that acts as intermediate resolver (recursive cache) avoiding a need for the client to carry out the iteration.

b) A ***recursive query*** is one in which the resolver asks the DNS server for a final response or an error (if the resource does not exist). In this case, the DNS server acts as an intermediary, performing the necessary iterative queries to get the answer or the error.

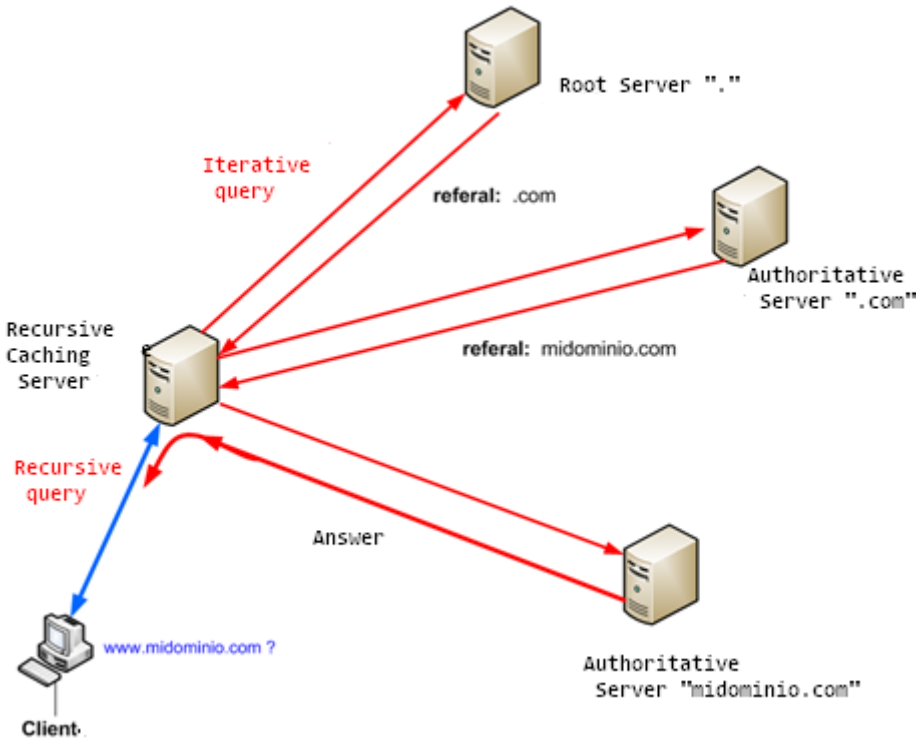


Illustration 8. Iterative and Recursive Queries.

- **Resolution Mechanism in a DNS Query.**

The procedure followed in a DNS resolution is as follows. The client (resolver) passes on the query to the DNS server:

a) If the DNS server is configured as authoritative and receives a DNS query about a domain for which it is authoritative, it will return a response by consulting the records stored in its configuration and will mark this answer as an *Authoritative Answer* in the "ANSWER" section of the response message. If it lacks this information, it will respond with the message NXDOMAIN (*Non-Existent-Domain*).
b) If the DNS server is authoritative and not configured as recursive, and it receives a query about a domain for which it is not authoritative, it will respond with a message containing records in the "AUTHORITY" and "ADDITIONAL" sections informing the resolver that it does not provide recursion and where it should direct its query so as to obtain authoritative information on the domain requested. This is known as a *Referral Response*.
c) If the DNS server is not authoritative, but is configured to be recursive, and it receives a query, it will initiate iterative queries (recursion) to find the authoritative server for the domain. Once it has the answer, it returns the record to the client (resolver), indicating that is a non-authoritative answer. The information is cached, so that if it is asked about the same resource once again and the time that the record is marked with for expiry (TTL, or Time To Live) has not elapsed, it will reply by consulting this cache.

An example of the flow in a recursive query about the domain www.example.com would be:

1. The resolver launches a query, asking the DNS server for a resolution of the name *www.ejemplo.com*.
2. The server, lacking the answer, initiates iterative query to find the record. For this purpose, it questions the root servers for the domain *.com*.

3. The query reaches the root DNS servers, which answer with AUTHORITY and ADDITIONAL records referring to the authoritative servers for the domain *.com*.
4. The servers referred to by the root server are questioned about the domain *www.ejemplo.com*.
5. The authoritative DNS server for the zone .com also returns a referral response with a pointer to the authoritative DNS server for the domain *ejemplo.com*.
6. When the recursive server has learnt the addresses of the DNS authoritative servers for the domain ejemplo.com. through referral responses it redirects the query about *www.ejemplo.com*
7. The authoritative DNS server for *ejemplo.com* seeks the record requested amongst the pieces of information it stores and returns the answer.
8. Finally, the recursive server caches the response with the TTL as configured and provides a resolution for the resolver's request.

Illustrations 9 and *10* give a graphic view of the process described above:

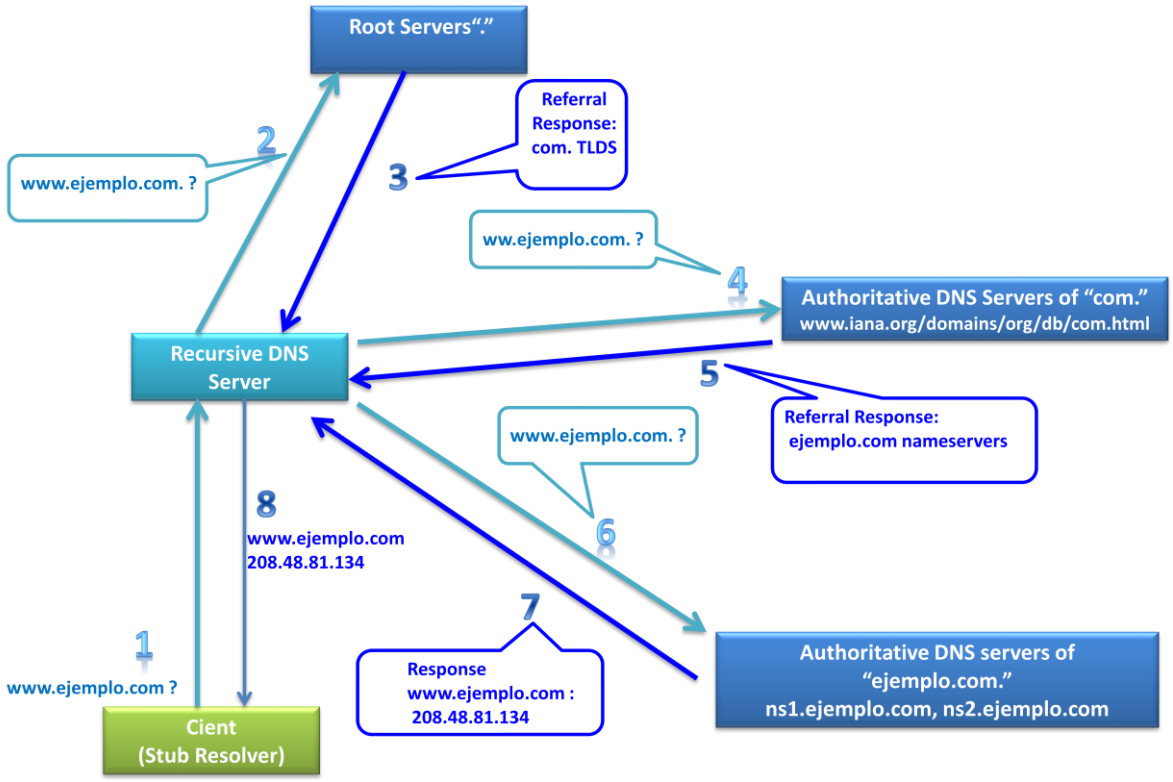


Illustration 9 Succession of Queries in a Recursive Resolution.

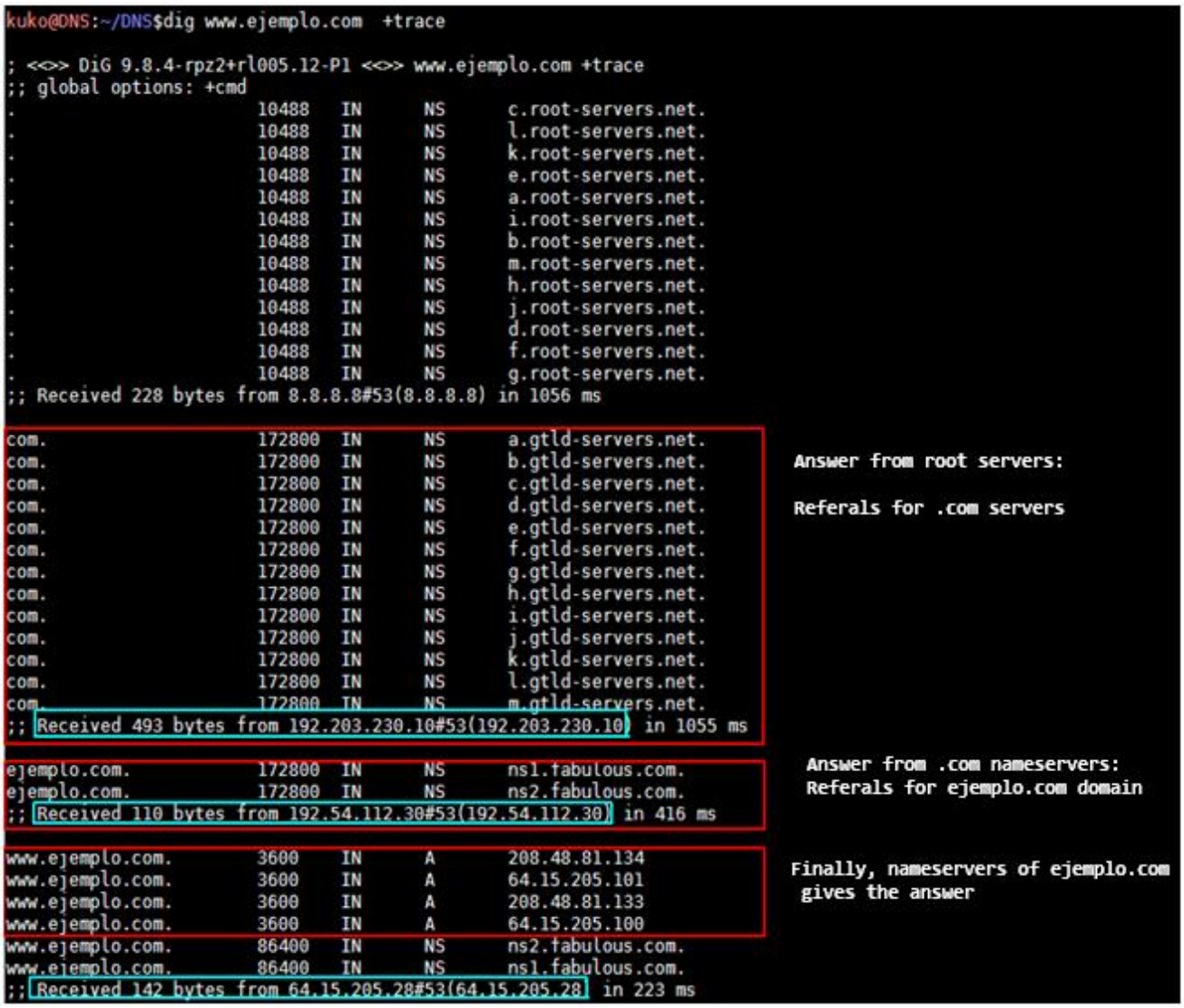


Illustration 10. Succession of Iterative Queries in a DNS Resolution.

- **Zone Transfers.**

A zone transfer is a transaction in which a *secondary* (slave) DNS server updates the zone contents from a *primary* (master) server and thus keeps a copy that is synchronized with the master databases. The transaction starts with a *zone transfer query*, in which a request is made for all the records (resource records or RRs) for a domain. A zone transfer query is generated automatically in the secondary server in two possible circumstances:

1 A notification message << NOTIFY >>is received by the primary server to make it known that there have been changes or modifications in the contents of the zone.
2 The time specified by the value <<REFRESH >> in the RDATA field of the SOA record for the zone has elapsed. (*Illustration 11).*

```
kuko@DNS:~/DNS$dig isc.org SOA  +multiline

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> isc.org SOA +multiline
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39002
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;isc.org.                       IN SOA

;; ANSWER SECTION:
isc.org.                7200 IN SOA ns-int.isc.org. hostmaster.isc.org. (
                                2014012101 ; serial
                                7200       ; refresh (2 hours)
                                3600       ; retry (1 hour)
                                24796800   ; expire (41 weeks)
                                3600       ; minimum (1 hour)
                                )

;; Query time: 88 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Wed Jan 22 15:28:55 2014
;; MSG SIZE  rcvd: 79
```

Illustration 11. Record of the Type SOA (Start of Authority). Note the Refresh Value.

- **Dynamic Updates.**

In certain environments the number of records and zones grows or varies frequently, making manual management unviable. For example, this would apply to records of names of type "A" in a Dynamic Host Configuration Protocol service and their converse pointers to names (PTR) within any large service-providing business. This need gave rise to the concept of dynamic updating. The mechanism for such updates provides for two operations: *adding* or *erasing* records in a zone file. The case of an update is covered by an erasure followed by later reconstitution of the record. A detailed specification for this can be consulted in RFC 2136 Dynamic Updates in the Domain Name System (DNS UPDATE)[5]

Keeping in mind the two possible operations, adding and erasing, defined for the process of dynamic updating, the possible actions would be:

- ✓ Adding or erasing individual records.
- ✓ Erasing sets of records meeting a specific criterion within a given domain.
- ✓ Erasing an existing domain (for instance, all records in the domain mydomain.com).
- ✓ Adding a new domain with one or more records.

- **Notifications and Updates Arising from Zone Transfers.**

Every time that there is a change in the zone files in the primary authoritative server, the secondary server must be informed of the modification and thus be enabled to update its copy of the zones, requesting a zone transfer from the primary server. Thanks to this mechanism, after any change in the zone records the master server sends a message (NOTIFY) to the secondary servers to alert them to the modification. In addition, although less precise than the primary server NOTIFY message, there is a zone transfer process triggered in the secondary server when the time specified

[5] RFC2136. http://www.ietf.org/rfc/rfc2136.txt.

in the stored SOA record (the *refresh* value) has elapsed. By default, BIND uses the notification method (*NOTIFY*).

When the master server needs to issue a notification, it selects the NS (name servers) records specified in the zone file, to which it sends the NOTIFY message. When the slave server receives the notification, it resets the "*Refresh*" value to zero and checks whether the *serial* (the number identifying the version of the zone) has been incremented, in which case it requests the transfer. This process of notification by default may be able to notify other servers that do not appear as name servers in the zone files. In secondary servers, the directive *allow-notify* identifies those servers from which it is permissible to receive notifications.

CRUCIAL CONCEPTS.

- **Resolver:** A DNS client responsible for composing and sending DNS messages to servers to obtain information required for a given domain. It can be a server itself or only a client (stub resolver).
- **Open Resolver:** A server that offers a recursive DNS service accessible publicly to any client (resolver) requesting it.
- **Recursion:** The actions that a DNS server takes so as to hand over requested information to a resolver by questioning other servers.
- **Authoritative Server:** The DNS server that maintains, distributes and responds to DNS queries by consulting the information stored in its records, Resource Records (RRs). It may be primary or secondary.
- **Master (Primary) Authoritative Server:** This is the authoritative DNS server which holds the definitive versions of the records it administers.
- **Stealth (Hidden) Authoritative Server:** A primary authoritative server for certain zones but not appearing in the NS records for these. The aim is to keep it hidden from queries of the NS type. This may be useful, for instance, for internal servers.
- **Slave (Secondary) Authoritative Server:** This is the authoritative DNS server that stores a copy of the records administered by the master server. When some change takes place in the records of the master or primary server, it is notified to its slaves, which request and initiate a zone transfer.
- **DNS Cache Server (Recursive Resolver):** This is an intermediary DNS server that obtains answers to DNS queries by consulting authoritative servers, and stores them in cache so as to have them available and pass them on to clients (resolvers). Its function is to improve the performance for responses and to contribute to reducing the DNS traffic load on the Internet.
- **Zone:** A database that an authoritative server holds, relating to a set of domains.
- **Zone Transfer:** A communication (transaction) between DNS servers so as to replicate contents of a zone among them. It is a client-server TCP communication existing in two forms: complete (AXFR) or incremental (IXFR, bringing changes up to date).
- **FQDN:** Fully Qualified Domain Name. This is the absolute and complete name that identifies a resource in the distributed database of the DNS space.
- **DNS Record or RR:** Resource Record. This contains the information from a DNS record that is sent in DNS messages. See *Table 1. Record Format. Resource Record (RR).* It is made up of six fields: NAME, TYPE, CLASS, TTL, RDLENGTH and RDATA
- **DNS Message:** A structure designed for IP communication among parts of the DNS space and to transmit information. It is made up of five fields: HEADER, QUESTION, ANSWER, AUTHORITY and ADDITIONAL. Table **3**. Generic Format for a DNS Message

3 SECURITY IN THE DNS.

THREATS AND VULNERABILITIES IN THE DNS.

In a DNS environment several points can be identified from which potential attacks might develop. These points or "attack vectors" are to be found both locally in the DNS server and local network themselves, and in communications between servers and clients.

ATTACK VECTORS AND THREATS IN A DNS SCENARIO.

In a typical DNS scenario, shown in *Illustration 12*, five main areas can be picked out as presenting a plane susceptible to threats. These threats and possible countermeasures to them can be summarized as follows:

1 *Local Threats:* In preventing local threats, the simplest solution is to implement security measures and policies in the internal network. *Anti-spoofing Mechanisms*, *Intrusion Detection Systems* (IDSs) or *Intrusion Protection Systems* (IPSs)[6], together with protection of the access channels to servers and their files, will be the main thrust of protection in this area.

2 *Server-to-Server Threats*: Dynamic Updating. These risks are present when the size of the organization or the number of servers to be administered makes it necessary to centralize administration of data through DDNS (*Dynamic DNS*). One valid route for ensuring communication would be to have a dedicated restricted communication channel or to implement transaction signatures (TSIG) or both.

3 *Master Server to Slave Server Threats*: Zone Transfers. When an organization has slave servers, it needs to carry out master-to-slave zone transfers. In such cases one solution worth consideration is the implementation of TSIG (*Transaction SIGnature*), so that zone transfer operations are signed with a key known to both parties. In addition, security of communications can be reinforced by using Secure Sockets Layer (SSL) or Transport Layer Security (TLS). Other options would be a private network channel for the transaction, or in an extreme case the disabling of automatic operations, having them carried out manually, which is not really a functional alternative.

4 *Master Server to Client Cache Server or Resolver Threats.* As will be seen in the section on *Randomization of the Transaction ID and Port* of Origin, improvements implemented in recent versions of Bind, involving the introduction of randomization in the ports from which queries originate, as also message identifiers, make the possibility of cache poisoning in DNS servers less likely, but even so attacks continue to be possible. The only solution considered effective is to adopt DNSSEC.

[6] IDSs/IPSs: Intrusion Detection Systems and Intrusion Prevention Systems. These systems either prevent or detect threats, or both. http://es.wikipedia.org/wiki/System_de_prevenci%C3%B3n_de_intrusos

5 *Server to Client Threats*: In the flow of information between a client or resolver and master or cache server, there is a possibility of local attacks to intercept data and of spoofing with the aim of supplanting the DNS server. Once again, DNSSEC is the countermeasure that is effective against this threat.

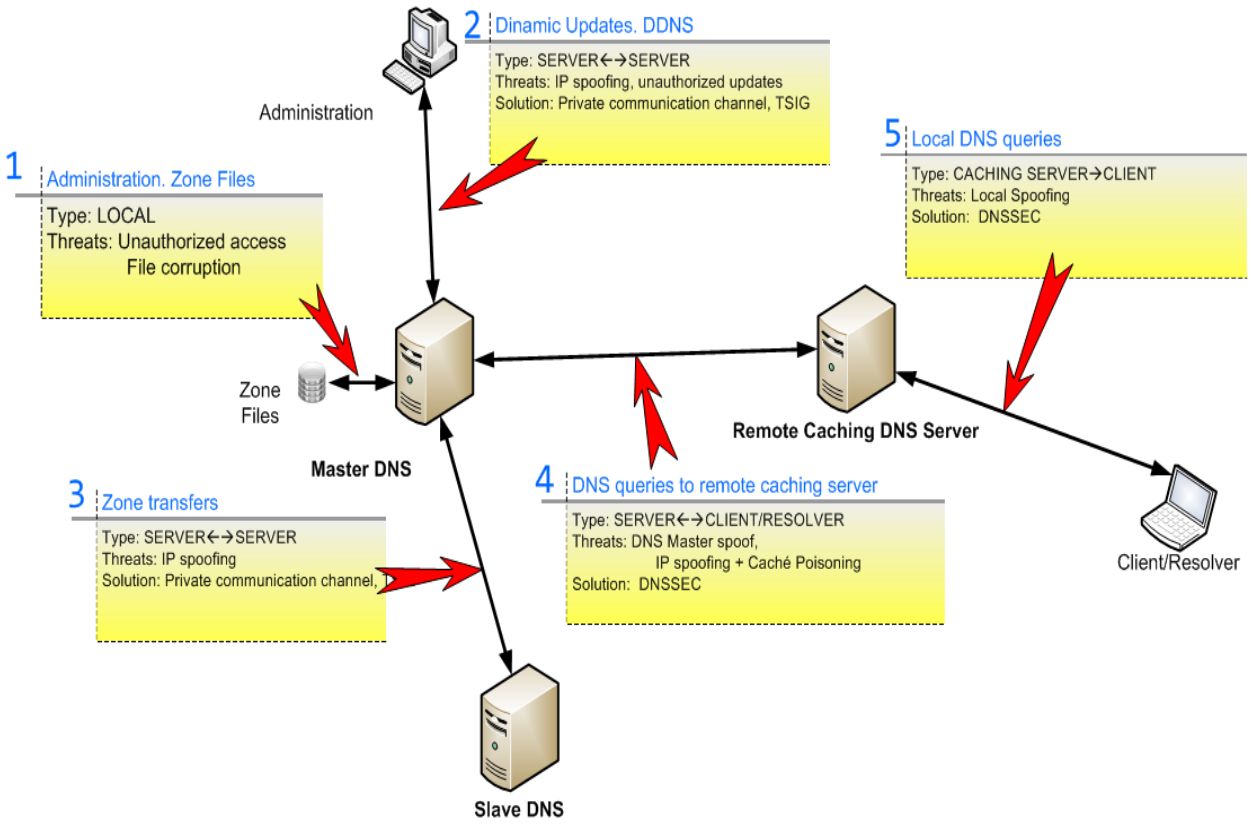


Illustration 12. DNS Scenario. Attack Routes and Classification of Threats.

VULNERABILITIES AND WEAK POINTS IN THE DNS SPECIFICATION.

- **UDP Transport Layer and IP Spoofing.**

The main weakness suffered by DNS has its origin directly in the use primarily of the UDP protocol to transmit messages. UDP is a network transport protocol in which speed of transmission is given pride of place and which sends and receives information without prior establishment of a connection or confirmation of, or check on, delivery or reception of any message. This makes it feasible to falsify IP addresses (IP spoofing) and the substitution of question and answer messages. Although the DNS envisages the use of TCP for transmitting messages, in specifications for implementation it recommends employing UDP for queries, for reasons of efficiency. It suggests limiting use of TCP to zone transfer transactions and those queries that exceed the maximum size of 512 bytes for messages in UDP. In view of the absence of any check or confirmation in UDP transmissions, final responsibility for validating a message falls directly upon the DNS protocol.

- **Weaknesses in Identification and Validation of DNS messages.**

In parallel to the problem of the use of the UDP protocol for DNS message transport, there are further design weaknesses in respect of the identification and validation of packets that favour falsification of them.

As described in *Generic Generic Format of DNS* Messages *Message,* in the HEADER section of a DNS transmission the **ID** field is intended to identify the message. This identifier, represented by a number of just 16 bits, plays an important part in the mechanism for validating answer messages. As explained below, its limited length, combined with a weak validation procedure in UDP makes attacks supplanting IPs possible with relative ease.

Response Validation.

Nonetheless, the ID field is not the only element checked when validating a response. As can be inferred from "RFC1034[7]", the minimum requisites for accepting an answer are:

- The destination port in the response datagram must be the same as the port of origin of the question.
- The ID of the answer message must be the same as the ID of the question message.
- The ANSWER field must refer to the same resource as the QUESTION field.
- The AUTHORITATIVE section contains the authoritative servers for the ANSWER section.

All these conditions, except the transaction identifier ID, are easily picked out and it is simple to construct a fake response if the resource requested is known. In this way, an attacker who succeeds in finding the ID with which the query was issued will have the information necessary for providing a false answer. This, together with transmission using UDP, which lacks any checking or validation of the communication, has the result that such a false response will be accepted by the server as valid for the query previously made.

Message Identifier (ID)

Owing to the limited length assigned to the ID field of the message (16 bits) and to weaknesses in the way the sequence of IDs is generated, it is computationally relatively simple to produce a sufficient number of IDs in a short time and thus hit upon the original ID. However, many aspects of the protection of the ID and other values in DNS messages have improved since 2008, when Dan Kaminsky[8], a researcher at Security IT, presented his work on "*DNS Cache Poisoning*", in which he demonstrated how easy it was to manage to falsify the response to a DNS query and thus get the requesting server to store it in its cache.

These weaknesses in message transmission and validation turn the DNS protocol into an easily exploitable target for the two main types of attack based on *DNS IP spoofing*: *DNS Cache Poisoning* and *Service Denial through DNS Amplification*.

[7] RCF1034. Domains Names Concepts and facilities. http://tools.ietf.org/html/rfc1034#section-5.3.3
[8]Dan Kaminsky. a security researcher known for having discovered the error allowing poisoning of DNS caches in 2008 affecting the Rootkit of Sony. http://es.wikipedia.org/wiki/Dan_Kaminsky

DNS CACHE POISONING AND DNS SPOOFING.

As has been seen, in a DNS query the **ID** field of the **HEADER** section of the message is used to identify the transaction and its corresponding response. With UDP and if no other checking is used, an attacker can send a mass of responses (*flooding*), each with a different ID until hitting upon the ID generated in the query. If this is so, and it is possible to get the false response to arrive before the legitimate one does (a race condition), the server that initiated the query will accept it and store it in its cache. In this way, it is possible to "poison" the cache of a recursive DNS server with a manipulated record. From that moment on, for the length of time the record remains stored in the cache (TTL), the victim server will redirect to an illegitimate IP all the requests from a resolver that consults it about the resource that has been manipulated.

DESCRIPTION OF ATTACKS.

The sequence that is produced by a cache poisoning is shown in *Illustration 13* and is the following:

Attackers, having a DNS server under their control, request a name which belongs to the domain that they wish to supplant (1). They ensure that this name is not cached. The victim server, not finding in its cache the resource that has been requested, initiates the sequence of repetitive requests beginning with the root servers (2) and following the TLDs indicated in referrals (3) until it finds out which server is authoritative for the resource in order to query it (4). At this moment, the malicious server starts a bombardment of responses (6) with different IDs with the aim of getting one that matches the ID corresponding to the original query from the victim server. In these responses it is indicated that the authoritative server (authority) for the domain that it is intended to supplant is to be found at the IP of the malicious server. If it is feasible to get the fake response (6) to arrive before the original does (5), the victim server will store the false record in its cache with the IP for the malicious server as the authoritative server, supplanting the legitimate authoritative server. The real response, arriving later, is discarded.

At this point, the *cache poisoning* of the victim server has been successfully completed, and all requests from resolvers arriving from sub-domains belonging to the supplanted domain will be directed to the malicious server. This will ensure that IPs under its control will be passed on as suits it.

If no other defence is in place, an attacker just needs to be quick enough in generating a number of responses sufficient to hit upon the original ID.

The process may be seen in graphic form in the illustration below.

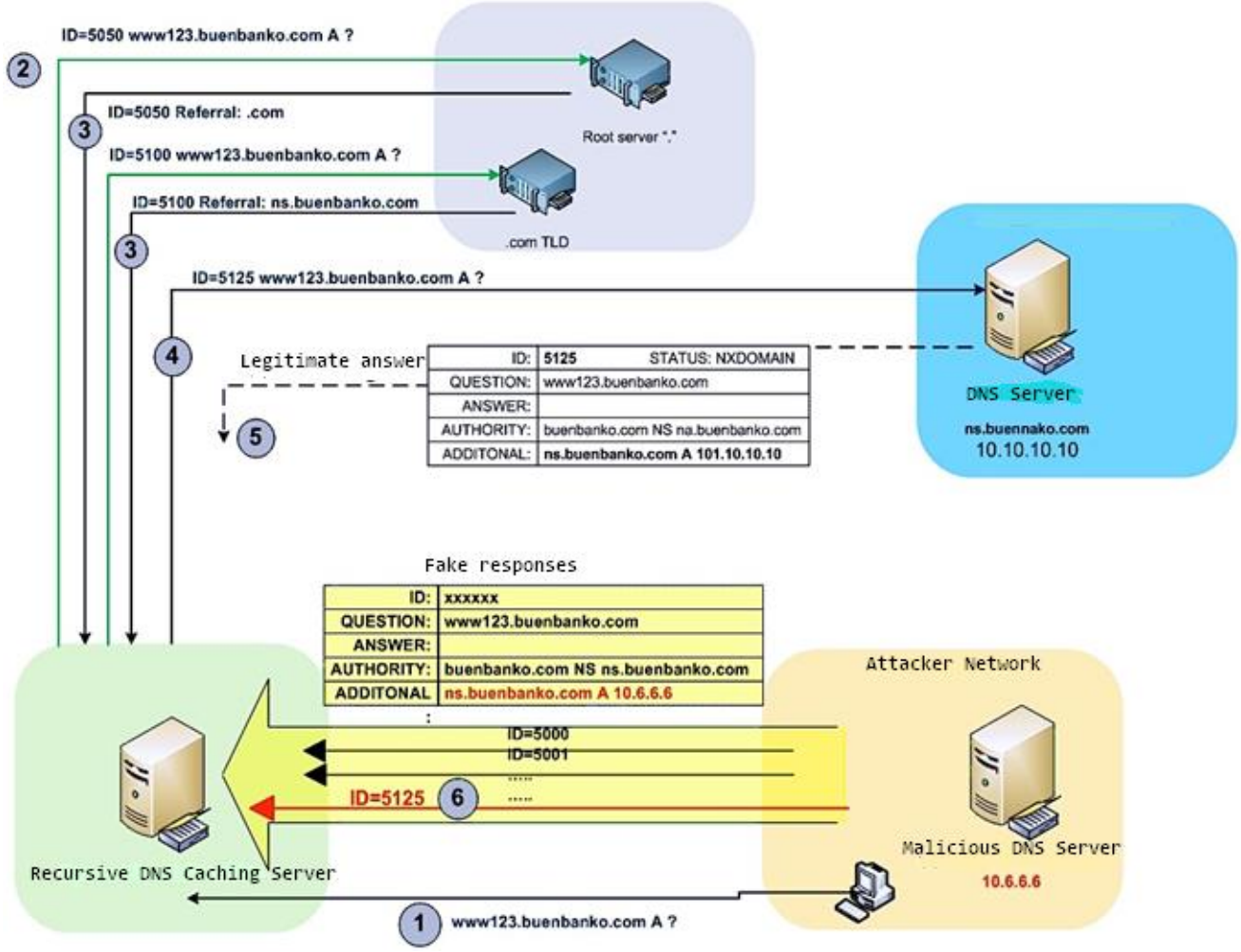


Illustration 13. Cache Poisoning Attack. Once the original ID has been discovered, a false response is sent.

MEASURES AGAINST CACHE POISONING ATTACKS.

The document from the Internet Engineering Task Force *RFC5452 Measures for Making DNS more Resilient against Forged Answers*[9] describes the problem of DNS spoofing and sets out some guidelines for implementations of DNS software with the aim of detecting and avoiding this threat, stating that:

"DNS data is to be accepted by a resolver if and only if:

a) *The question section of the reply packet is equivalent to that of a question packet currently waiting for a response.*
b) *The ID field of the reply packet matches that of the question packet.*
c) *The response comes from the same network address to which the question was sent.*
d) *The response comes in on the same network address, including port number, from which the question was sent.*
e) *In general, the first response matching these four conditions is accepted."*

On the basis of these indications, DNS software should implement the measures described below.

[9] RFC5452 . http://tools.ietf.org/html/rfc5452

- **Randomization of the Transaction ID and Port of Origin.**

Since the ID field is the key to identification of DNS messages DNS, and ever since the cache poisoning attack shown by Dan Kaminsky, attempts have been made to make it difficult to predict the ID of the transaction by randomizing ID generation in queries and thus making them less foreseeable. This measure, however, has proved insufficient, owing to the limitations of ID field, assigned just 16 bits, which means only 2^{16} - 1 = 65,535 values (other than zero) are possible. At a later stage randomization was introduced for the port of origin of queries, which previously had been fixed as port 53. The ports available for random assignment, once the privileged ports running from 1 to 1023 are taken out of account, allows for 2^{11} = 2,048 ports. The overall outcome of these two measures yields $2^{11} \times (2^{16} - 1)$ = 134,215,680 possible non-zero values. This larger number of values that may arise makes obtaining the ID of the transaction in the limited time available before the arrival of the legitimate response much more difficult (assuming there is no denial of service such as to delay it).

- **0 × 20 Bit Encoding.**

As a complement to randomization of the transaction ID and port of origin, there are other additional factors that some manufacturers like Nominum[10] implement. The technique of *0 × 20 bit encoding*[11] consists of making DNS queries have randomly alternating capital and lower-case letters. Since the DNS protocol does not distinguish case, the resolution will be exactly the same for *WwW.Example.Com* as for *www.example.com*. However, the implementation of the software will validate a response only if its capitalization of letters coincides with that in the query. In this way, it is more difficult for a fake response to be accepted, as it is unlikely to coincide with the upper- and lower-case lettering of the query.

- **Validation of Responses and Detection of Spoofing. Retransmission with TCP**

The introduction of randomization mechanisms when selecting ID and port of origin in the generation of queries does succeed in making spoofing attacks difficult. However, they are theoretically still feasible. Hence, additional checks on the response in itself are needed.

A good resolver should detect attempts at spoofing applying criteria like those noted in RFC5452. So, if application of these criteria leads to the discarding of many reply packets responding to a given query, the request is abandoned in UDP form and is tried again using TCP.

- **Limiting Recursion.**

An improvement complementary to those above is that of limiting recursion as far as possible, and thus reducing the areas exposed to attacks. In fact, the large number of recursive servers offering a public service (known as *open resolvers*) constitutes the main route used for bringing into play powerful attacks, such as service denial by DNS amplification.

---

[10] Nominum Vantio Cache Server. http://nominum.com/infrastructure/engines/caching-dns/

[11] IETF. Use of Bit 0 × 20 in DNS Labels. http://tools.ietf.org/html/draft-vixie-dnsext-dns0x20-00

- **A Solution to Poisoning: DNSSEC.**

Finally, it is felt that the most effective solution for eliminating this threat would lie in the implementation of DNSSEC[12]. Some ideas about this improvement to the DNS are put forward in *Section* 6 *DNSSEC.*

[12] DNSSEC, Domain Name System Security Extensions, is a set of specifications aimed at authenticating the origin of data in DNS messages.

4 DENIAL OF SERVICE ATTACKS.

Because of its intrinsic vulnerability to IP spoofing, the DNS protocol has become a powerful ally when denial of service attacks are being brought into play. This, combined with its wide distribution and access throughout the world, makes this type of attack one of the most efficacious and widely used.

DNS AMPLIFICATION ATTACKS.

DESCRIPTION OF ATTACKS.

Once again, the use of UDP for carrying DNS messages, together with the huge number of recursive servers accessible through the Internet (open resolvers) makes it possible to use the service to bring distributed denial of service attacks to bear on other servers. One of the main forms, based on DNS, is the *DNS Amplification Attack.*

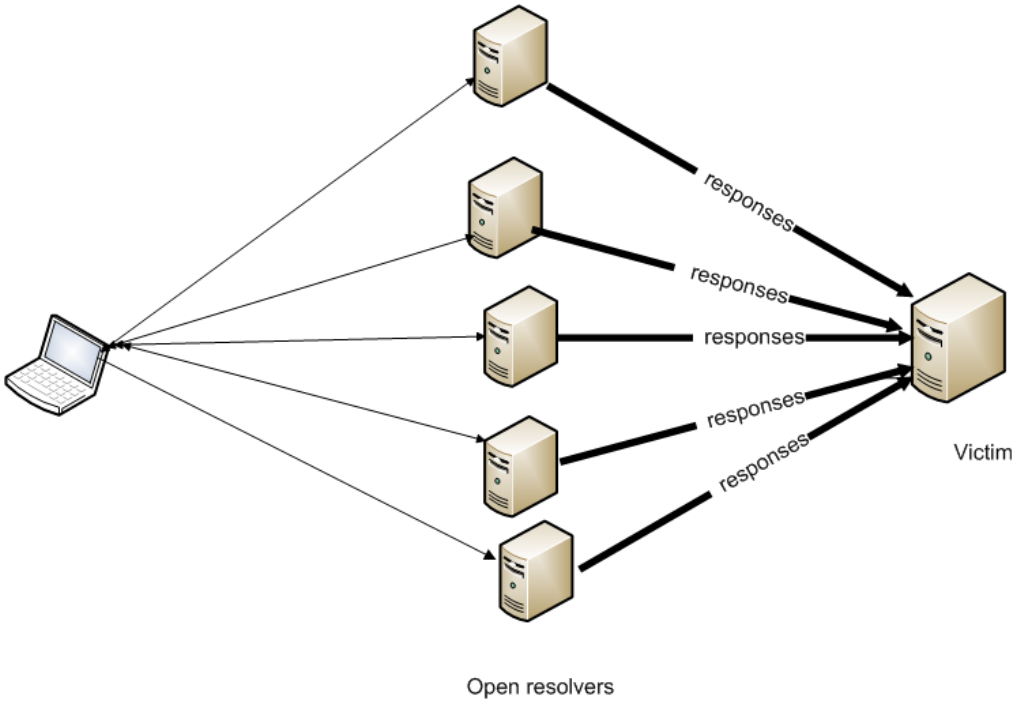


Illustration 14. A DNS Amplification Attack.

In a DNS amplification attack (*Illustration 14*) an attempt is made to overwhelm the capacity of a server to respond by making a large quantity of DNS data arrive at it. The procedure consists of launching DNS queries at an *open resolver* forging the IP of origin to be the IP of the server or host to be attacked. This technique is known as *IP spoofing* and is widely used with protocols based on UDP, where the lack of control over the connection makes it possible to forge IP addresses. In the manipulated queries, the origin IP address is changed to be the IP of the object of the attack, and queries are sent on a massive scale to as many servers (*open resolvers)* as possible. These resolvers, upon receiving the queries, respond by sending the answer to the IP address indicated. With a sufficient volume of queries, by requesting resources with a response that is much bigger than the query issued, for example, TXT or ANY records, it is feasible to generate a huge volume of

traffic sent to the target, but based on a limited amount of data. This will trigger congestion of resources in the victim and lead to a loss or degradation of service. When the attack is launched simultaneously from different origins, the volume of traffic is even greater. In this case the attack is called a Distributed Denial of Service or DDoS.

The following illustration (*Illustration 15*) shows the amplification factor clearly, since 2,066 bytes are obtained. A query is usually around 66 bytes

```
kuko@DNS ~ dig dhs.gov ANY  +bufsize=4096

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> dhs.gov ANY +bufsize=4096
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56170
;; flags: qr rd ra; QUERY: 1, ANSWER: 21, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;dhs.gov.                       IN      ANY

;; ANSWER SECTION:
dhs.gov.                475     IN      A       173.252.133.166
dhs.gov.                475     IN      RRSIG   A 8 2 900 20140313183320 20140303180959
cnIF2s859F zpa72ufdM3wATFIxTVtDWoYYVwgfGcZw9BBe+vaSZc475HA637bjfMlD NYM=
dhs.gov.                21175   IN      NS      asia2.akam.net.
dhs.gov.                21175   IN      NS      use3.akam.net.
dhs.gov.                21175   IN      NS      use1.akam.net.
dhs.gov.                21175   IN      NS      usc2.akam.net.
dhs.gov.                21175   IN      NS      usw3.akam.net.
dhs.gov.                21175   IN      NS      eur2.akam.net.
dhs.gov.                21175   IN      NS      asia3.akam.net.
dhs.gov.                21175   IN      NS      usw4.akam.net.
dhs.gov.                21175   IN      RRSIG   NS 8 2 28800 20140313230021 201403032249
MH1stBao89Ctp KFkoBF7mu18S5Frrxf2uLQwEyt6HzV2GXtuu4aW/MmOOsNsEL0uh2paL r0g=
dhs.gov.                21175   IN      SOA     ns5.dhs.gov. dnsseclnet.cbp.dhs.gov. 206
dhs.gov.                21175   IN      RRSIG   SOA 8 2 28800 20140314084458 20140304074
EAQ9LDprmeRDxc LQ4+6ae9lyCDFGYtJmKOXKZqTLYHMuhzsUoEPlDlAkrUdD68YCKUPwb5 G0o=
dhs.gov.                21175   IN      MX      10 mail.us.messaging.microsoft.com.
dhs.gov.                21175   IN      RRSIG   MX 8 2 28800 20140313082229 201403030816
xN0eiqQZOWdUl GYoJaw0NdXuPtlXLdokkfISJyUeSSagI53TSF+Bk7q2yg4iyjcEApkRx iMA=
dhs.gov.                21175   IN      DNSKEY  256 3 8 AwEAAc9nhbytBN/boahvvLyf2Nk04wnl
2RD86gR/jLWleOz7VevO yDAiqKYH
dhs.gov.                21175   IN      DNSKEY  257 3 8 AwEAAdCPBJACeS7+jhZV2p76YkyjtnL/
kC70snNdWmWLkIbo74P4 SbxCXIzBYU+D/hfhC3pyFl63U8JFlWTBOi1hNmh0dXycUlRTkkxFv4V2 3okFI1+kpF
dhs.gov.                21175   IN      RRSIG   DNSKEY 8 2 432000 20140313052936 201403
2wxtpthR/EI13NmSo 2cpXEc9WnceHp0v0suZVeMNMTvhcjleZcwrAiQA8vsrAMDr3biAZLuwT pPz4LOCO3UqA2
j2NdrBgSEQv5Po9VzaysMq8I9ZHIOtLillDW37y 0IK5bQ==
dhs.gov.                21175   IN      RRSIG   DNSKEY 8 2 432000 20140313052936 201403
z8m4PNHOPKrD2mvE4R bVb6T/fA81WImFaCEKZCEW0FOUn3CTPNXzijVq10/R+Emx6XIcs7f9sr tp0=
dhs.gov.                475     IN      NSEC3PARAM 1 0 10 CE73FD6B7A5EF6FC74F72799
dhs.gov.                475     IN      RRSIG   NSEC3PARAM 8 2 900 20140313010739 201403
LrrW6+Q6l3DKhRAlGhJ +DUMEyng2db2lMfAJDYWotUOZqv73vetBbncTx0Cjq6GZ4+nTK/FBet5 mCA=

;; Query time: 82 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Tue Mar  4 11:31:20 2014
;; MSG SIZE  rcvd: 2066
```

Illustration 15. The Amplification Factor. A query of type ANY, with a length of 66 bytes, gets a response of 2,066 bytes.

PROTECTING A SERVER AGAINST DNS AMPLIFICATION ATTACKS.

Overall, the problem of denial of service based on DNS amplification has its roots in the very large number of DNS servers distributed around the world and configured as open resolvers, that is, offering their service without any type of restriction to any requester using the Internet. There are

projects like the *Open Resolver Project* aimed at encouraging control of open resolvers directed towards the owners of DNS servers with an eye to getting them to control or limit recursive queries proceeding from locations outside their network[13].

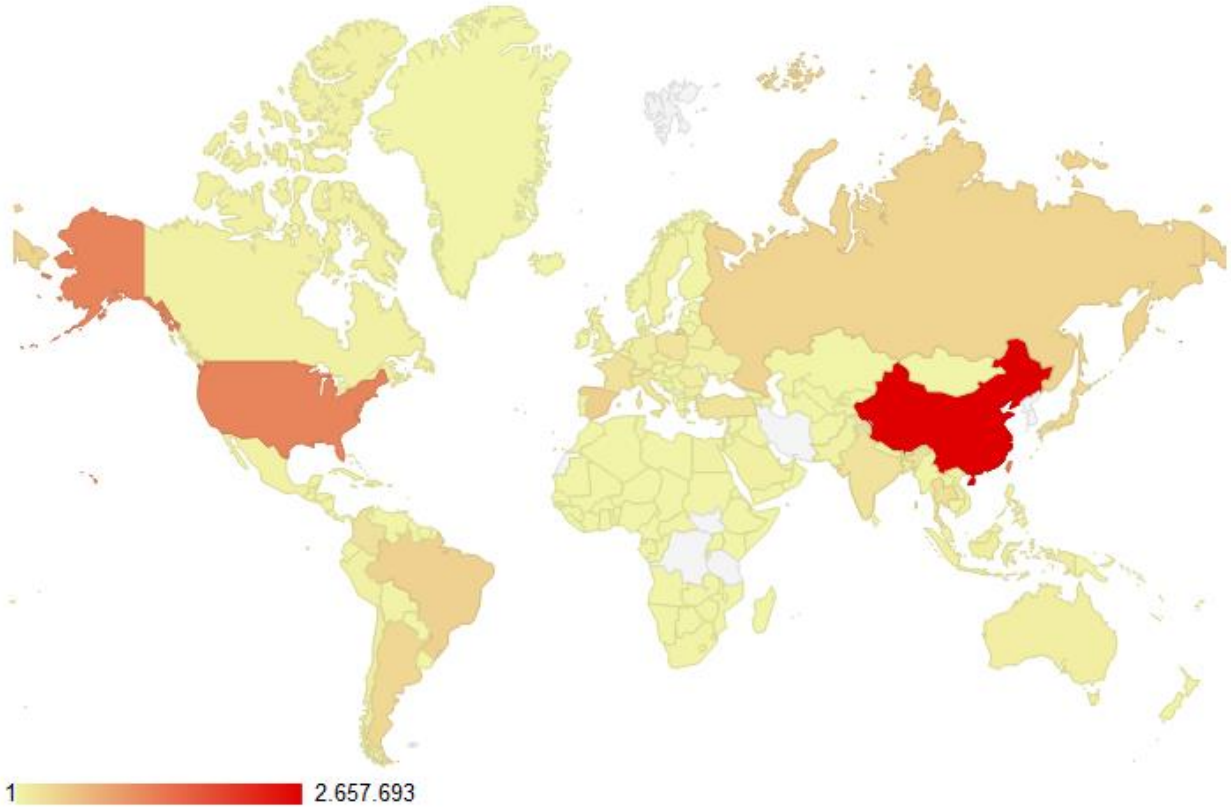


Illustration 16. World Distribution of Open Resolvers. Source: DNS Amplification Attacks Observer.

Specifically, administrators of DNS resolvers may carry out a series of tasks to prevent their systems from being used to launch denial of service attacks. Among the tasks that should be considered are anti-spoofing measures, traffic filtering and techniques like *Rate Response Limit* and a correct configuration for recursiveness such as are described in the section on *Security in DNS Queries and Responses.*

DENIAL OF SERVICE (DOS).

Denial of service attacks are extremely difficult to avoid, and any resource that is publicly accessible can become a potential target for such an attack. In the case of DNS, its characteristics and the intrinsic weakness of the carriage by UDP on which it is based, make the service itself a victim, not merely a collaborator, as happens in the case of amplification attacks. Given the difficulty of locating and blocking an attack over UDP with fake IP addresses, it is vital to have reaction mechanisms to defend against a denial of service attack when the final victim of such an aggression.

[13] Open Resolver Project: http://openresolverproject.org

In general terms and with reference to network architecture, some of the commoner errors to be avoided are:

- Putting all the servers in one single sub-net.
- Putting them behind the same router.
- Using just one line or route for Internet provision
- Having a single autonomous system (AS)[14]

ATTACKS AGAINST THE DOMAIN REGISTER. DNS HIJACKING.

DESCRIPTION.

Many domain register services, operating with a large number of businesses of considerable commercial clout, have automated procedures to provide a speedy way to check records. Many attacks on such registrars start from an acquaintance with, and analysis of, these procedures. Thus, for instance, awareness that e-mail may well be the preferred method for notifying changes in configurations, contacts, record renewals, and so forth may lead an attacker to use this information in attempting to succeed in hijacking a domain, redirecting to an IP controlled by the attacker, or changing configurations through phishing, social engineering, or both.

A poor security policy relating to control or access to the administration account for domain records, whether on the part of the registrar or of the client, also often leads to compromising of a domain.

MEASURES AGAINST DNS OR DOMAIN HIJACKING.

In the case of domain registration services, the measures that should be kept in mind to guarantee protection of their clients' assets should focus on preventing unauthorized access and verifying the authenticity of the identities of records or changes requested. Clients, in turn, should agree with the registrar a commitment to maintain a security policy with reference to access control, checking contacts and verification of identities to prevent supplantation arising from phishing or social engineering attacks.

- **Record Verification**.
  Domain record accounts should have a verification method to ensure the authenticity of requesters and their contact details, with the aim of reducing identity theft and abuse of domains. Studies of phishing[15], experiences with botnets, and fast-flux attacks[16] make plain how important it is to check domain record accounts for cyber-crime activities.
  Hence, it is vital for the registrar to implement verification mechanism for contact e-mails and confirmation of records by a hyperlink sent to the account stated.

- ***Strengthening the Authentication System***.
  Policies and mechanisms of proven reliability should be implemented for managing, maintaining and transmitting passwords used for access to the record account. Multi-factor authentication and secure connections (SSL, VPN) can be considered highly advisable additional measures.

[14] AS: Autonomous System. A group of networks with their own routing policy.
http://es.wikipedia.org/wiki/System_aut%C3%B3nomo
[15] Global Phishing Survey: Trends and Domain Name Use in 1H 2012
http://docs.apwg.org/reports/APWG_GlobalPhishingSurvey_1H2012.pdf
[16] Fast-Flux. http://en.wikipedia.org/wiki/Fast_flux

- ***Multiplicity of Contact Points***.
  Permitting the specification of various contact points provides greater security and granularity with regard to the confirmation of actions to be performed.

- ***Renewals Policy***. Similarly to verification of contacts and identities in requests for records, it is important to keep open a communication route in respect of renewals and changes relating to the domain registered. At times a failure to communicate and non-renewal of an expired record leads to theft by a third party who takes over the record with the aim of benefiting from it. This technique is known as cyber-squatting[17].

[17] Cyber-squatting. http://en.wikipedia.org/wiki/Cybersquatting

5 FORTIFYING A DNS SERVICE.

This section describes measures recommended for reinforcing and protecting a DNS service in a generic way and with specific reference to the DNS Bind software, the most widely used around the world, which has currently reached its ninth version. For this purpose, the elements making up the service as a whole are grouped into three layers to render more granular the identification of attack vectors and the measures applicable to them. This grouping is as follows:

- **Base Environment**. Basic elements of the service at the level of systems and communications.
  - Operating System.
  - Bind Software.
  - Network Topology.
- **Data**. Measures relating to data security.
  - Parameterization.
  - Information on zone records.
- **Transactions**. Protecting messages in DNS operations.
  - Queries. Questions and Answers.
  - Zone Transfers.
  - Notifications.
  - Dynamic Updates.

Illustration 17. Reinforcing DNS. Layers.

SECURITY OF THE BASIC SYSTEM AND SOFTWARE ENVIRONMENT.

OPERATING SYSTEM.

The operating system of the server must be updated and patched. There are a large number of systems supported by BIND 9 software. Thus, whatever the option chosen as a function of the necessities of the service, it is important to follow a policy of maintaining patches up to date and following up any possible vulnerabilities that might compromise the system.

Unnecessary services should be de-activated. The server should be exclusively dedicated to the DNS service, with all other unneeded applications apart from DNS and system administration software disabled. Those firewall rules strictly necessary to permit the functioning of the DNS should be applied.

SOFTWARE CONFIGURATION.

- **Checking and Follow-up of Software.**

A review policy should be established for software, so as to ensure that it is correctly updated and takes account of any possible vulnerabilities or security patches. The status of the latest versions of BIND software can be consulted on the producer's website.

- **Hiding the Version.**

Any directives that might show information about the version of the software being executed should be disabled. This information can be requested with a query of the TXT type and CHAOS class, as shown in the following example:

```
@<server_dns> TXT CHAOS

or, with nslookup:

# nslookup -q=txt -class=CHAOS version.bind<server_dns>
```

Example 1. Checking the Version of BIND.

In the BIND DNS software package, the command that gives information about the version is specified in the configuration file named.conf. This information can be modified, as shown in the following configuration:

```
// File: named.conf

options {version "version not available"; }
```

Configuration 1. Hiding Information about the BIND Software.

- **Executing the DNS Software as a Non-Privileged User.**

The DNS service should never be executed as a root or privileged user of the system. This measure, combined with "caging" of the service in a chroot environment, will avoid putting attackers in a situation where they can control the system if protections are compromised.

A specific user should be created. Generally, the username is set as *named*, with the account being blocked so that it is not possible to log in as the user *named*:

```
# groupadd named

# useradd -g named -d /chroot/named -s /bin/false named

# passwd -l named
```

Configuration 2. Non-Privileged User for Running BIND.

- **Creating a Restricted Environment: Chroot.**

A directory structure should be set up within which the service can be confined, for example */chroot/named*:

```
/chroot
+-- named
 +-- dev
 +-- etc
 |+-- namedb
 | +-- slave<
| +-- master< Directories where zone files will be stored
       | +--<
+-- var
| +-- run
|
 +-- log
```

```
mkdir -p /chroot/named
cd /chroot/named
mkdir -p dev etc/namedb/slave var/run
```

Configuration 3. Chroot. Structure of Chroot Directories.

Thereafter the files needed for executing the BIND software can be copied across into this area.

On the assumption that the starting point is a prior installation of BIND in the route */var/named* and */etc/named.conf*, these will be copied in the chroot environment and the necessary permissions will be assigned for the routes where the user named needs to be able to write

```
cp -a /var/named/* /chroot/named/etc/namedb/

## The general configuration file is usually linked to the file
##/etc/named.conf outside the cage or jail with the aim of giving it visibility
## within the system and facilitating administration

ls –s /chroot/named/etc/named.conf /etc/named.conf

Time-zone file

cp /etc/localtime /chroot/named/etc
```

```
# The user named will need to write in zone files where it is a slave (zone
transfer) or the PID for the process on starting up the service:

chown -R named:named /chroot/named/etc/namedb/slave
chown named:named /chroot/named/var/run

# # Creating the necessary device files, confirming major/minor
## numbers with "ls -lL /dev/random /dev/null"

mknod /chroot/named/dev/null c 1 3
mknod /chroot/named/dev/random c 1 8
chmod 666 /chroot/named/dev/{null,random}
```

Configuration 4. Creating a Jail (or Cage) Environment for Bind.

- **Assigning Permissions.**

Similarly, the correct assignment of permissions for file systems and their contents should be verified. This avoids unauthorized access to configurations or data files.

```
cd /chroot/named
chown -R named:named .# Establishing the owner named

find . -type f -print | xargs chmod u=rw,og=r# estableciendo permiso files
find . -type d -print | xargs chmod u=rwx,og=rx# permisos directorios

chmod o= etc/*.conf# restricting access to configuration files


## The directory etc/namedb is where zone files will be stored.
## The user named must have permissions to update or to create new files


findetc/namedb/ -type f -print | xargs chown named:named
findetc/namedb/ -type f -print | xargs chmod ug=r,o=

chownnamed:named etc/namedb/
chmod ug=rwx,o=etc/namedb/
chmod ug=rwx,o=rx var/run/

##logfiles
chownnamed:namedlogs/
chmod ug=rwx,o=rx logs/

## Protecting the jail environment:
chown root /chroot
chmod 700 /chroot
chown named:named /chroot/named
chmod 700 /chroot/named
cd /chroot/named
chattr +i etc etc/localtime var
```

Configuration 5. Protection and File Permissions for Bind.

- **Log File Configuration.**

Recording of log details should be configured through the logging directives in the configuration file named.*conf*. In addition, sending to remote servers should be active in the configuration of system logs (for example, *rsyslog.conf*)

An appropriate configuration might be as follows. In it two channels are defined: one for general purposes and another specifically for picking out security messages and separating them into an independent file.

NOTE: As the software is limited to a chroot environment, *rsyslog* must be configured to be able to communicate with the jail environment. For this purpose, it is necessary to add a socket so that rsyslogd will receive messages from, for example, */chroot/named/dev/log*. Specific details for each individual system can be found in the manual for the syslog demon for that system.

```
// File: named.conf

logging {

channel default_syslog {
// Send the majority of messages to syslog ( /var/log/messages )
syslog local2;
severity debug;
};


channel audit_log {
// Send security messages to an independent file.
file "/var/named/log/named.log";
severity debug;
print-time yes;
};

category default { default_syslog; };
category general { default_syslog; };
category security { audit_log; default_syslog; };
category config { default_syslog; };
category resolver { audit_log; };
category xfer-in { audit_log; };
category xfer-out { audit_log; };
category notify { audit_log; };
category client { audit_log; };
category network { audit_log; };
category update { audit_log; };
category queries { audit_log; };
category lame-servers { audit_log; };

};
```

Configuration 6. Logging Configuration.

- **Service Start-Up in a Restricted Environment.**

Once the chroot environment for BIND has been configured in */chroot/named*, the service should be started up by taking this route as its root.

```
named -t /chroot/named -u named -c /etc/named.conf
```

Configuration 7. Start-up of Bind in a Chroot Environment.

NETWORK TOPOLOGY.

A good implementation of DNS should always separate servers in accordance with their role. Authoritative servers and recursive caches will be two clearly differentiated functional components that require to be treated independently when designing the network architecture.

The design of the network architecture is always a critical point when implementing a publicly accessible service. In the case of the DNS, it is, if possible, even more important, because of being an element common to both the internal and the external structure of an organization.

Organizations generally need a DNS infrastructure that will fulfil two objectives. These are: (1) to allow their internal network to access the Internet, and (2) to offer resolution to external networks for their public resources.

To give the infrastructure greater security, it is necessary to segment servers according to their roles and importance, and thus to establish different layers of exposal. The advantages of this arrangement are security and resilience to attacks. The drawbacks are higher cost and greater complexity of administration.

- **Segregation of DNS Server Roles. Authoritative and Cache.**

In a DNS infrastructure two main server roles can be distinguished *(Illustration 18).* Authoritative servers are those responsible for maintaining and distributing name domains. Recursive cache servers are those which request and temporarily store resolutions for domains obtained from authoritative servers. In their turn, authoritative servers are divided into two sorts.

Primary or *Master.* Maintains and administers its own local database with the domains and records of which it is owner.

Secondary or *Slave.* Has no local database, but rather obtains a replicate from a master server through a zone transfer.

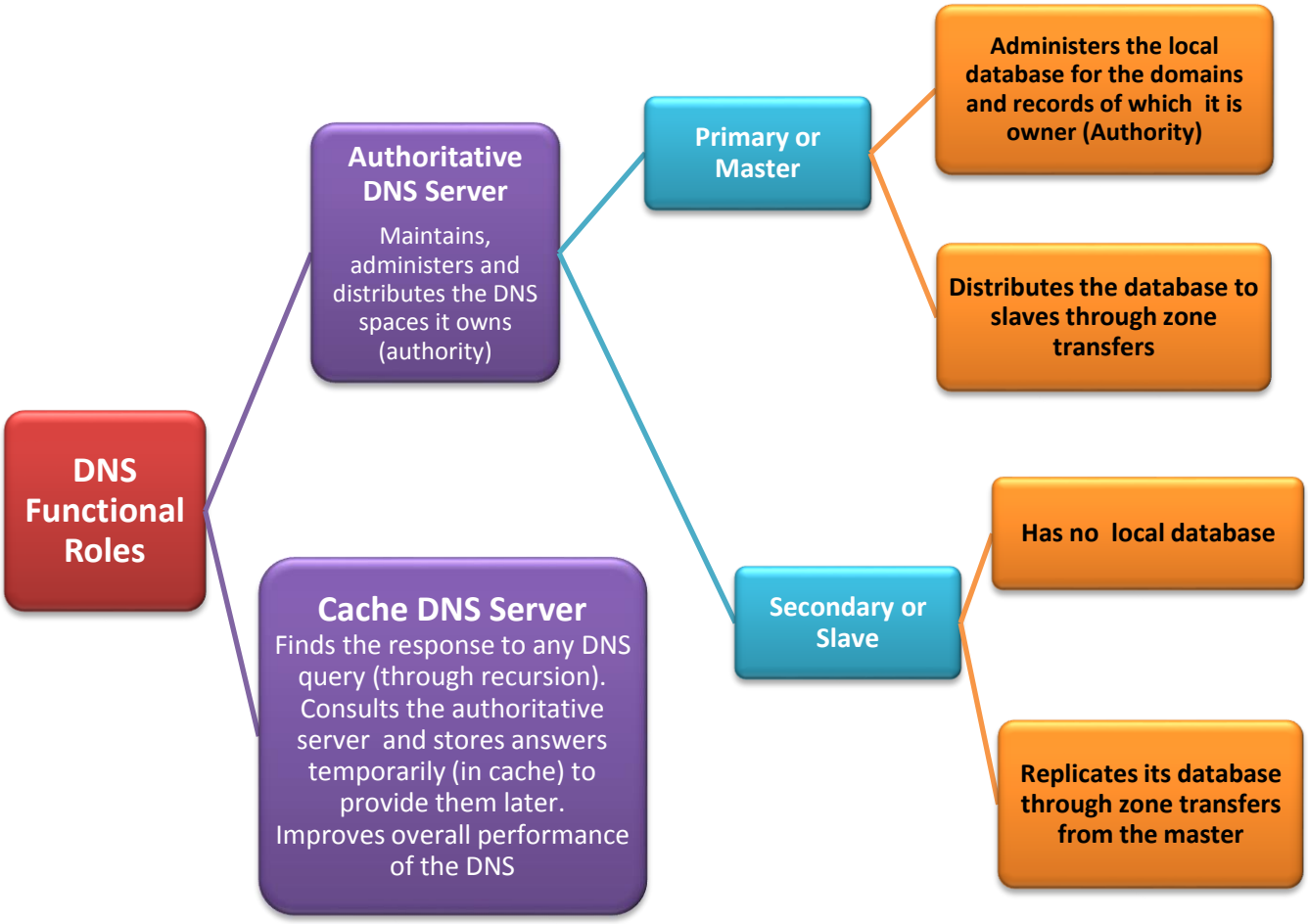


Illustration 18. Functional Roles of DNS Servers. Cache and Authoritative Servers.

- **Network Architecture Design.**

Implementation of Authoritative DNS Servers.

When a network architecture is being designed for a DNS service, a clear separation of functions depending on the information provided by the authoritative server should be observed, with the aim of separating public information from private. Furthermore, redundancy should be built in, so as to cope with possible cuts in communication, either deliberate or caused by technical incidents. These requisites may be matched by separating internal and external servers, and giving both geographical redundancy. An authoritative server should never permit recursion, this being reserved for cache servers.

On the basis of this strategy, authoritative servers should thus be separated into two distinct groups to serve public and private internal resources.

Public Authoritative Servers. These will respond to DNS queries from any external network, offering the public records from the organization's domain. They can be sited on an isolated public network for greater security, but generally they are put on a "De-Militarized Zone" (DMZ) network, that is, the segment of the network that is the boundary between the internal network and the Internet.

Private Authoritative Servers. These will respond exclusively to requests coming from the internal network itself. They will be sited on the internal network.

In addition, consideration might be given to installing an "invisible" authoritative master server, known as a *Hidden Master* or *Stealth Master.* A hidden DNS server is a primary server that does not appear in the NS records of a zone, even though it is the master for it. The hidden server is sited on the internal network behind a firewall and cannot be reached from external Internet networks. It is not used to provide information, either. Its sole function is the maintenance of zones and transfers of zones to secondary servers.

This recommended separation may be achieved in two ways. One is physical, dedicating exclusive hosts to the public and private parts of the set-up. The alternative is to use a single host known as Split-Horizon or Split DNS, which implements the separation internally by using the "views" functionality of DNS software. The use of Split DNS has the advantage of needing fewer physical resources (hosts) to achieve zone separation, but has the downside of leaving information about internal networks exposed in the case of any compromising of the server.

Implementation of Recursive Cache Servers.

The DMZ is where the dedicated recursive servers will be sited. They are the only servers that will be able to carry out recursion for requests proceeding from resolvers on the internal network and thus ensure the resolution of any Internet resource for this. They should NEVER permit recursive queries coming from the outside. These servers should not be used directly as resolvers, but rather should be accessed through a forwarder.

Internal recursive servers are generally configured as forwarders, to pass on queries that they cannot answer to the recursive servers in the DMZ (as a proxy DNS). For example, when an internal recursive server is not aware of the resolution for an external Internet domain, it will direct the query to the recursive server located in the DMZ, which will see to performing recursions and returning the response to the internal server.

Redundancy and High Availability.

To reduce the possibility of a loss of service, authoritative servers will be provided with geographical and network redundancy. This implies that they will be in independent sub-nets behind different routers and in physically distinct locations. In addition, a hidden authoritative DNS master server (stealth master) may be used, so that only secondary masters will be visible as name servers. These will take all their zones from the hidden master server through zone transfers or dynamic updates.

The network architecture strategy proposed may be seen in *Illustration 19. Network Architecture. Implementation of a DNS Infrastructure*.

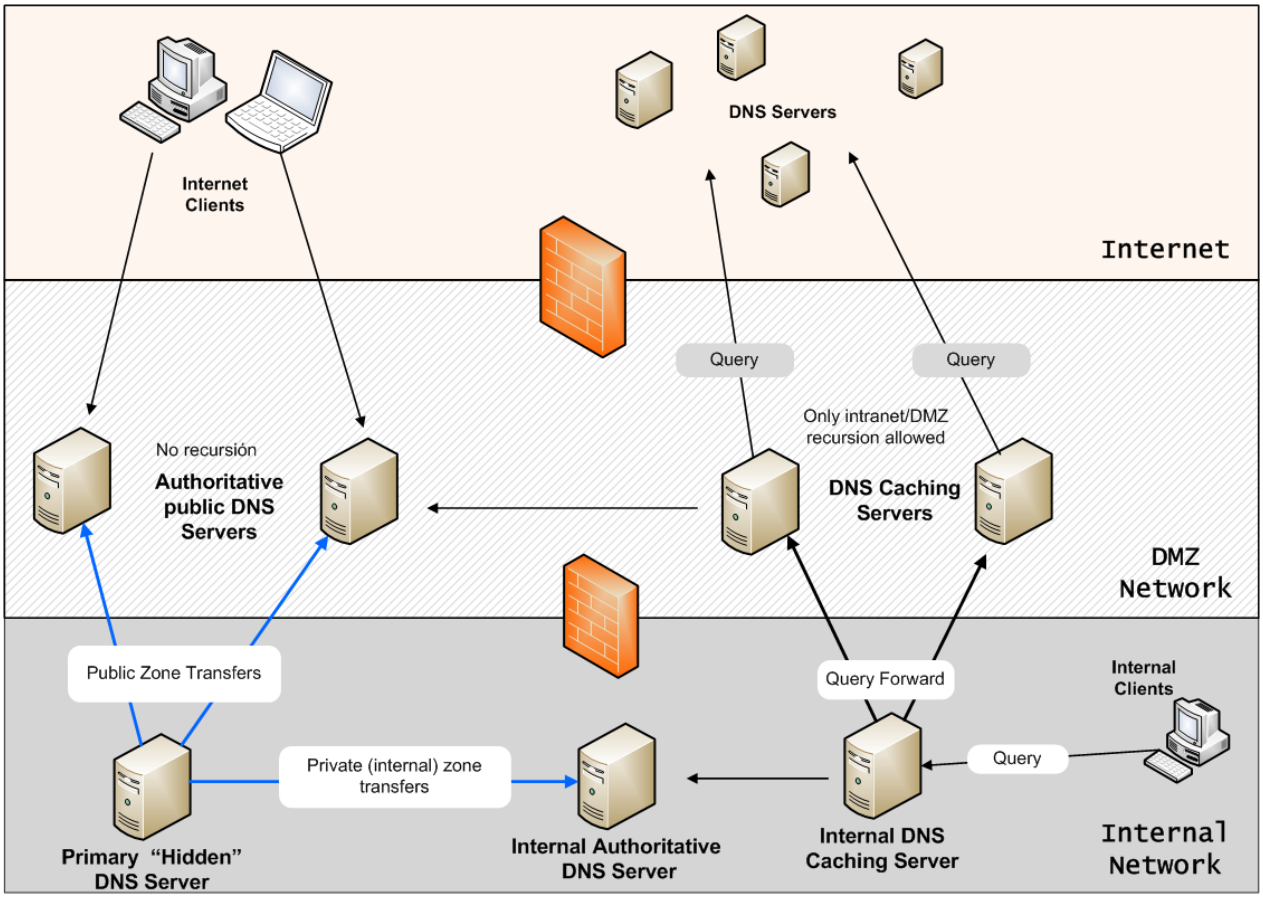


Illustration 19. Network Architecture. Implementation of a DNS Infrastructure.

INTERNAL MONITORING.

In a system of some importance or critical status, it is highly advisable to have a monitoring system to detect possible attacks or events occurring on the internal network. With this aim, it is recommended that intrusion detection mechanisms (IPS/IDS) be implemented, along with monitoring of accesses, log and event concentrators or SIEM (*Security Information and Event Management*), for instance.

SUMMARY OF MEASURES IN THE BASE ENVIRONMENT OF THE DNS SERVICE.

A summary of this nature is shown below in *Table 5 Checklist of Security Measures in the DNS Server Environment*.

Security in the Base Environment and DNS Software	
Operating System	
1	Operating system patched
2	Unnecessary services disabled
DNS Software	
3	Software updated and patched
4	Information on version hidden
5	Execution as non-privileged user
6	Isolation of the software environment (chroot)

7	Configuration logging
Network Topology	
8	Separating roles between different authoritative and cache servers
9	Geographical redundancy and a network of authoritative servers
10	When split DNS is used, configuration for a minimum view, external and internal
11	If a hidden (internal) master server is used, restriction of transfers to internal servers
Monitoring	
12	Consideration given to the implementation of systems for intrusion prevention or detection

Table 5. Checklist of Security Measures in the DNS Server Environment.

SECURITY MEASURES IN TRANSACTIONS.

SECURITY IN DNS QUERIES AND RESPONSES.

The principal threats affecting DNS queries are those related to spoofing, which can materialize as *DNS cache poisoning attacks*, as *service denials* or as *DNS amplification attacks*. Besides these, taking advantage of the weaknesses inherent in the DNS protocol, it is possible to manipulate responses captured in a local environment through network traffic *sniffing*.

- **Limiting Recursion. Segmenting Networks and Views.**

It is equally important to limit recursion to clients or networks that form part of the organization. For example, large companies like Internet service providers (ISPs) should offer recursion only to those clients to whom they provide access. This aim can be achieved in differing ways, depending on the topology chosen (see *Network Topology.*).

In the case of BIND, the use of views gives a method for segmenting and separating different origins to which the appropriate query capacities can be offered. Thus, for example, it is possible in a *split DNS server* (servicing both internal zones and external clients) to permit recursion for queries coming from internal zones and deny it to external clients**.** *In an authoritative DNS server recursion should always be disabled.*

An example of the use of views to separate internal zones from external might be the following:

```
// Example of a split configuration with views
// The internal networks to which recursion is offered (10.2.0.0/24)
// are situated in the acl “trusted”
// This will prevent external hosts using this server as a resolver for other
// domains

acl "trusted" {
// Our internal network
10.0.2.0/24;
localhost;
};
view "internal-in"{
// Internal view. Internal networks are permitted recursive queries and access
// to the cache.
```

```
// IMPORTANT: the views configured are applied in their order of appearance
// in the configuration

match-clients { trusted; };
recursion yes;
additional-from-auth yes;
additional-from-cache yes;
zone "." {
// Link in the root server hint file.
type hint;
file "db.cache";
};
zone "0.0.127.in-addr.arpa" {
type master;
file "master/db.127.0.0";
allow-query { any; };
allow-transfer { none; };
};
zone "localhost" {
 type master;
 file "db.localhost";

};
zone "internal.ejemplo.com" {
// Example of internal zone.
type master;
 file “etc/interna.ejemplo.com”;

};
};
// View for external DNS clients (not belonging to the acl “trusted”)

view"external-in" {
// External view, allowed for any client.
// No recursion or caching is allowed, avoiding the status of open resolver
// IMPORTANT: the views configured are applied in their order of appearance
// in the configuration
match-clients { any; };
recursion no;
additional-from-auth no;
additional-from-cache no;
// Link in our zones

zone "ejemplo.com" {
type master;
file “etc/zone_master.ejemplo.com”;
allow-query { any; };
};
};
};
```

Configuration 8. Restriction of Recursion and Zones. Use of views.

- **Defence against IP Spoofing. Traffic Filtering.**

Since even correctly configured servers can be exploited by attackers using a forged IP origin to submit DNS queries, it is advisable to apply the guidelines that are to be found in the publications

Best Current Practices BCP38 and *BCP84*. These were issued by the Internet Engineering Task Force (IETF) to identify and filter out traffic suspected of faking IP addresses.

These highly useful guides are, in full:

> **BCP38:** *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing* http://tools.ietf.org/html/bcp38
> **BCP84:** *Ingress Filtering for Multihomed Networks* http://tools.ietf.org/html/bcp84

- **Improvements in Authoritative Servers. Response Rate Limiting.**

Authoritative servers should be accessible so as to offer necessary information about the records for which they are responsible. It is crucial to check that **authoritative servers always reject recursive queries** and provide resolution only for records in their domain. Moreover, to combat amplification attacks it is advisable to implement a Response Rate Limiting (RRL) solution on authoritative servers, especially if no additional mechanisms (BCP38) are adopted to detect possible forgery of IP addresses. Thus, in view of the impossibility of determining whether a DNS query carried by UDP has a fake IP address, question and answer patterns should be taken into account so that in this way an attempt can be made to work out when an attack is under way. This information can be used to discard responses generated by suspect queries. Questions, unlike answers, are not affected by the RRL mechanism.

Sample Scenario:

Let it be supposed that the authoritative server for the domain *example.com* is receiving a large number of queries from an IP 1.2.3.4 with the flag for DNSSEC activated, so that the size of the answers will be extensive. Such behaviour is typical of flooding with requests in a DNS amplification attack and gives rise to suspicions of an attack directed against IP 1.2.3.4:

```
13-13-2013    12:27:34.102    queries:    info:    client    1.2.3.4#58540
(host1.example.com): query: testhost.example.com IN A +ED (1.2.3.4)
13-13-2013    12:27:41.606    queries:    info:    client    1.2.3.4#55979
(host1.example.com): query: testhost.example.com IN A +ED (1.2.3.4)
13-13-2013    12:27:59.196    queries:    info:    client    1.2.3.4#47516
(host1.example.com): query: testhost.example.com IN A +ED (1.2.3.4)
```

Illustration 20. BIND Logs. Flooding with Queries in a DNS Amplification Attack.

From version 9.9.4 of BIND onwards the RRL function has been incorporated and to implement it, it is enough to specify with a ***rate-limit*** clause any parameters desired in the section OPTIONS or VIEW of *named.conf*:

```
//named.conf. Rate limiting clause to combat DNS amplification attacks
// Queries detected as suspicious will be limited to five per second
options {
directory "/var/named";
rate-limit {
responses-per-second 5;
 // TEST: To check functioning uncomment the following line
 // log-only yes;
};
};
```

Configuration 9. BIND, Response Rate Limiting Configuration against Flooding Attacks.

Once attacks have been detected, the queries affected will be reflected in logs, whether active or in log-only mode.

```
13-Dec-2013 12:41:42.336 queries: info: client 1.2.3.4#53459
(host1.example.com): query: host1.example.com IN A +ED (1.2.3.4)
13-Dec-2013 12:41:42.336 query-errors: info: client 1.2.3.4#53459
(host1.example.com): would rate limit slip response to 1.2.3.0/24 for
testhost.example.com IN A(3ee9836b)
```

Illustration 21. BIND Logs. Detection of Flooding. Rate-Limit in Log-Only Mode.

```
13-Dec-2013 12:44:44.868 queries: info: client 1.2.3.4#57114
(host1.example.com): query: host1.example.com IN A +ED (1.2.3.4)
13-Dec-2013 12:44:44.869 query-errors: info: client 1.2.3.4#57114
(host1.example.com): rate limit drop response to 1.2.3.0/24 for
host1.example.com IN A(3ee9836b)
```

Illustration 22. BIND Logs. Detection of Flooding. Rate-Limit Activated and with Response Discards.

Because RRL is a flexible counter to various attack scenarios, it is highly advisable to undertake in-depth query of the BIND 9.9.4 Administrator Reference Manual[18]

Finally, it should be noted that the implementation of RRL can to some extent favour DNS cache poisoning attacks. This is because it may reject or delay legitimate responses, which could give an attacker more opportunities to manage to infiltrate a response. As was commented in the section *Attack Attack Vectors and Threats in a DNS*, the only reliable solution against spoofing attacks is the implementation of DNSSEC.

SECURITY IN ZONE TRANSFER TRANSACTIONS.

Zone transfer transactions are carried out with the TCP protocol, so in the light of its connection characteristics they are more difficult to manipulate. However, there still is some possibility of altering the transaction through *ARP spoofing* and *Man in the Middle* attacks, particularly in attacks on local networks.,

- **Use of Access Control Lists (ACLs) and IP Filtering.**

BIND makes it possible to restrict the IPs authorized to request a zone transfer by means of the command *allow-transfer*, but this method is not effective in a well-worked-out spoofing attack. It can be seen as a strengthening measure, but not as a solution.

```
//named.conf
options {
// Permits zone transfer only to 10.0.2.15
 allow-transfer { 10.0.2.15; };
}
```

Configuration 10. IP Filtering for Zone Transfer.

[18] BIND9 Administrator Reference Manual (ARM) https://kb.isc.org/category/116/0/10/Software-Products/BIND9/Documentation/

- **TSIG (Transaction SIGnature).**

TSIG is the method that is recommended for protecting zone transfer transactions. With this approach, communication between servers is authenticated by using a key shared among them. Originally devised for the protection of dynamic updates in RFC2845, it is also used to avoid the manipulation of zone transfers. A description of a TSIG configuration can be consulted in: *Appendix 7.1 TRANSACTION SIGNATURE. TSIG.*

```
zone "ejemplo.com"{
 Type master;
file "etc/zone_master.ejemplo.com"
allow-transfer { key "ejemplo.com";};//zone transfer only permitted
}// with the key ejemplo.com
```

Configuration 11. TSIG for Zone Transfers.

SECURITY IN NOTIFICATIONS.

The threat of spoofing in notification transactions (which are sent by the authoritative server to slaves when a change occurs in zones) is the same as in zone transfers. Nonetheless, its impact is smaller in so far as the transaction in itself carries no sensitive information. The effect that could be caused on a slave server by spurious notifications would be to force a zone transfer request, or in the extreme case, make it victim of a denial of service if the volume of notifications is very large.

- **ACLs and IP filtering.**

Unlike zone transfers, in the case of notifications, given that the impact of these transactions is smaller, IP filtering with the relevant *allow-notify* commands may be an acceptable approach. In a slave server notifications will be permitted from the master server specified in the masters' category in the zone statement. Notifications from other servers may additionally be permitted with an *allow-notify* clause, specifying allowed IPs for notifications.zone "ejemplo.com" in {

```
zone "ejemplo.com" in {
type slave;
file "etc/zona.ejemplo.com"
masters {10.0.2.2};// master server, notifications allowed
allow-notify {10.0.2.3}; // notifications also accepted from 10.0.2.3
};
```

Configuration 12. IP Filtering for Receiving Notifications.

- **Use of TSIG in Notifications.**

As in zone transfers, this is the effective solution for protecting the transaction, since IP forgery (spoofing) is an easily exploitable vulnerability, as has been stated several times. One possible configuration would be to use ACLs or to force the use of TSIG with a server statement.

```
server 10.0.2.5 {
 keys { ejemplo.com; };
};
```

Configuration 13. Use of TSIG in Server-to-Server Transactions.

In this way, transactions, queries, notifications, zone transfers and dynamic updates sent to server 10.0.2.5 will be signed with the key shared by both machines. Logically, a similar clause will be set up on 10.0.2.5 with the IP of the reciprocal server.

SECURITY IN DYNAMIC UPDATES.

These transactions are generally used in environments which require updating of zones quite frequently and with considerable volume, which makes manual administration of zone files difficult. As this is an administrative task with a direct impact on zone contents, it is a transaction which must be protected from malicious manipulation. In fact, RFC2845[19] on TSIGs, already mentioned above, was initially specified with an eye to providing a mechanism for the authentication of dynamic DNS update transactions.

- **TSIG for Dynamic Updates**

As has been seen, just as in notifications and zone transfers it is possible to force the use of TSIG in dynamic update transactions. For TSIG, use is made of a key shared among servers to authenticate the communication by adding a signature generated with it. The key is lodged in a file in the servers involved in the transaction and must be protected against unauthorized access. This is crucial, because the signature authenticates the transaction but not the authenticity of the origin of the data, so that a compromised host implies a threat for the servers it administers. See the appendix *TRANSACTION SIGNATURE. TSIG.*

The configuration needed for this option is shown in the following example.

```
zone "ejemplo.com" in {
 Type master;
file "etc/zone_master.ejemplo.com"
allow-update { key "ejemplo.com";};//updates with TSIG key
};
```

Configuration 14. Dynamic Updates with TSIG.

- **Security of the Communication Channel.**

Encryption of communication through a dedicated channel or an isolated network or Virtual Local Area Network (VLAN), or even an IPsec tunnel, might be measures complementary or alternative to TSIG. In smaller environments, if manual management can be undertaken, the suggestion would be to change to this option and thus reduce this attack vector.

[19] RFC2845. *Secret Key Transaction Authentication for DNS (TSIG)*. https://www.ietf.org/rfc/rfc2845.txt

SUMMARY OF MEASURES FOR PROTECTING TRANSACTIONS

Security of DNS Transactions	
Security in DNS Queries and Responses	
1	Limiting recursion. Segmentation of network and views
2	Defence against IP spoofing. Traffic filtering
3	Improvements in authoritative servers. Response rate limiting
Security in Zone Transfer Transactions	
4	Use of ACLs and IP filtering
5	TSIG (Transaction SIGnature)
Security in Notifications	
6	ACLs and IP filtering
7	Use of TSIG in Notifications
Security in Dynamic Updates	
8	TSIG for dynamic updates
9	Security of the communication Channel. Administration networks, VLANs

Table 6. Checklist of Security Measures in DNS transactions.

SECURITY MEASURES FOR PROTECTING DATA.

With regard to the data layer, objectives focus on protection and availability of information for zones. For this purpose, parameterization and record contents must be kept in mind. It is necessary to configure zones and information that may be provided in a suitable way to avoid undesirable leakage of information about elements such, for example, as data about the internal network.

ZONE FILES. START OF AUTHORITY RECORD PARAMETERIZATION.

The SOA record for a zone establishes the overall parameters for it. So as to optimize the latency and distribution of zone records and their updating, parameters affecting the SOA record should be selected carefully to regulate communication between primary and secondary servers. Besides the TTL parameter assigned to the record (the recommended value would be between two and seven days), a further five parameters are set. These parameters, included in RFC1912 together with recommended values for them, are the following.

- **`Serial`** (number): This value in RDATA field of the SOA record is used to indicate zone changes. It should be incremented whenever any modification is made to the zone data.
- **`Refresh`** (seconds): This communicates to the secondary servers how many seconds' wait there should be between zone transfers. If they are zones that are frequently updated, the recommendation is for a period of twenty minutes to two hours. If the updates are infrequent, the advice is for the interval to be set at between two and twelve hours. If DNSSEC is in use, the value should always be less than the period of validity of the signed zone record RRSIG[20]. If the primary server sends a NOTIFY message, the transfer to secondary servers will be made immediately, without waiting for the Refresh Value to elapse.

[20]RRSIG. A type of DNS de record used in DNSSEC that contains the signature for a set of records.

- **Retry** (seconds): This is the amount of time that a secondary server should wait before retrying a zone transfer that has failed previously. It must be a fraction of the Refresh Value specified, with a good estimate for this being values between five minutes and one hour.
- **Expire**: This is the time that a secondary server should take zone records as valid if it has not been possible to refresh them. It should be set as a multiple of the Refresh Value, preferably between two and four weeks.
- **TTLMinimum**: This is the value for the number of seconds that a secondary server should keep in cache a negative result (NXDOMAIN) for a record. Depending on the frequency of updating for a zone, it is recommended that it should be fixed at between thirty minutes (for dynamic zones) and five days (for static or infrequently updated zones). A value of five minutes can be taken as the minimum threshold value.

  An example of an SOA record is shown below.

```
$TTL 3d ; 3 days TTL
@ IN SOA example.com. root.ejemplo.com. (
199609203 ; Serial
28800 ; Refresh 8 hours
3600; Retry 1 hour
604800; Expiry 1 week
86400); Minimum TTL 1 day
```

Configuration 15. SOA Record Configuration.

RESTRICTING THE INFORMATION PROVIDED BY VARIOUS TYPES OF RECORD.

Among the sorts of DNS records that exist, those most often used to show information are TXT records, which store a text intended to give information to people and applications about networks, hosts, services, or other types of generic information. HINFO records contain information about the host where the DNS service is located and LOC records indicate the geographical location of the server. The system administrator should make sure that these records do not offer sensitive information that might be made use of by an attacker to recognize characteristics of the environment or any other piece of data that might reveal possible attack vectors.

The following configuration for BIND shows how to hide sensitive information, in this case the software version.

```
// named.conf
options {
version “Not disclosed”; // Hide information on versions
}
```

Configuration 16. Hiding the Version of BIND.

As an outcome of the application of the configuration shown above, it can be shown that the version of BIND is no longer indicated.

```
kuko@DNS:~$dig @ucdns.sis.ucm.es -c CHAOS -t TXT version.bind

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> @ucdns.sis.ucm.es -c CHAOS -t TXT version.bind
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59251
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;version.bind.                  CH      TXT

;; ANSWER SECTION:
version.bind.           0       CH      TXT     "Not disclosed"

;; AUTHORITY SECTION:
version.bind.           0       CH      NS      version.bind.

;; Query time: 51 msec
;; SERVER: 147.96.2.4#53(147.96.2.4)
;; WHEN: Wed Dec 11 17:10:02 2013
;; MSG SIZE  rcvd: 70
```

Illustration 23. Version of BIND Hidden.

SUMMARY OF MEASURES FOR PROTECTING DATA

Security in Protecting DNS Data	
Zone Files. Parameterization of SOA Records	
1	Values recommended: TTL : 2 to 7 days. Serial: Update with each change in zone files. Refresh: 2 to 12 hours or 20 minutes to 2 hours (frequent updates) Retry: 5 minutes to 1 hour Expire: 2 to 4 weeks TTL min: 30 minutes to 5 hours (threshold value 5 minutes)
Hiding Information	
4	Hide TXT information and the version of BIND

Table 7. Strengthening Checklist. Data Security Measures.

6 DNSSEC.

WHAT IT IS AND HOW IT WORKS.

DSNSEC, from the abbreviation of *Domain Name System Security Extensions,* is considered an effective mechanism against spoofing and message manipulation in the DNS protocol. By extension, it also provides a route to protection against cache poisoning and similar attacks. DNSSEC is based on an infrastructure of public key encryption, PKI[21], and the use of digital signatures to establish the authenticity of sources and validity of messages. When applied to DNS queries, it ensures the integrity of messages and the authenticity of the source sending them

DNSSEC, using PKI (Public Key Infrastructure) and a special set of resource records (RRs), specific signature records (RRSIGs) and key records (DNSKEY), allows resolvers with DNSSEC capacities to check on the following:

- **Authenticity of Origin**: It authenticates that the data received can come only from the zone requested.
- **Integrity:** It verifies the integrity of data, that is, it checks that data have not been modified during the course of the transaction.
- **Non-Existence:** In the case of a response of non-existent domain (NXDOMAIN), it confirms that the record truly does not exist in the zone requested and has not been deliberately eliminated through interception of the transaction.

In DNSSEC, the public key of a source is validated with a chain of verifications that begins in a trusted server (for instance a root server) and goes down through the hierarchy of the DNS name space, successively confirming the public key signature of a daughter node with its mother node. The public key of trusted servers is termed the *trust anchor*.

Once this verification of the public key of the source has been performed, the next step in DNSSEC is to authenticate the response. In this case, responses include not just the records requested, but also the digital signature of a set of encapsulated in a specific type of record, called RRSIG. The resolver can then use the public key previously verified to check the validity of the signature and ensure the response is authentic. If there is a negative response indicating the non-existence of a record, a specific record called NSEC is attached, with its corresponding signature. Verification of this ensures the validity of the response and confirms that the record has not been eliminated by some intermediate operation.

COMPONENTS AND OPERATION.

In DNSSEC there are two main processes: signing and delivering, and verifying signatures. These processes are performed through mechanisms based on public key cryptography. Although operationally it is not necessary to have more than a single pair of keys, one public and one private,

[21] PKI. Public Key Infrastructure. This is a set of components for establishing encrypted communications based on asymmetric public key cryptography. It is widely employed for authentication, encryption, digital signing and other uses in which relationships of trust are established through digital certificates.

it is very common to utilize at least two pairs so as to facilitate the tasks of renewing keys and re-signing zones. Moreover, key separation is good cryptographic practice, with a view to limiting the extent of any possible compromise. Thus, DNSSEC has two pairs of public and private keys. One pair is known as the *Key Signing Key* (KSK) used for signing DNSKEY key records and another, called the *Zone Signing Key* (ZSK), for signing records (RR sets) .

- **DNSSEC Records.**

DNSSEC makes use of the following specific records for its operation:

RRSIG (*Resource Record Signature***).** This records a signature. It contains information about a set of DNS records of the same type and the signature for this set (created with a private key).

DNSKEY. A public signature. It is used to verify the signatures adjoined to RRSIG records.

NSEC (*Next Secure***)**. When a non-existent record is requested, a record of this type is returned with its corresponding signature (RRSIG) to demonstrate to the resolver the inexistence of this record. It contains a list in canonical order of the next authoritative domain or delegated point (NS) and the set of types of records present in these. It is accompanied by the RRSIG with a signature for them. Together with its signature record (RRSIG), this constitutes the verification method. It has been replaced by the version NSEC3, since NSEC makes it possible through its behaviour to obtain information about zones (enumeration) by requesting non-existent records.

DS (*Delegation Signer***).** This contains a hash value for the public key of a daughter node. To make sure that the public key (DNSKEY) and signature records in a zone verified have not been both manipulated, a hash value for the public key is generated and handed on to the node immediately above. This node generates the DS record storing this hash value and signs it with its private key, yielding the corresponding RRSIG record. This chain of hand-ons continues upwards in the hierarchy until it reaches the final trusted node in chain, typically the root node.

The following illustration (*Illustration 14. A DNS*) shows an example of an RRSIG record and identifies the fields making it up.

Campo	Descripción	Longitud (bytes)
NAME	Name of the domain to which the record belongs.	Variable string
TYPE	Code of record type .	2 bytes
CLASS	Code of record class.	2 bytes
TTL	Time in seconds for which the record is cached.	4 bytes
RDLENGTH	Indicates the length in bytes of the RDATA field.	4 bytes
RDATA	A string of variable length describing the record in accordance with its type and class.	Variable string

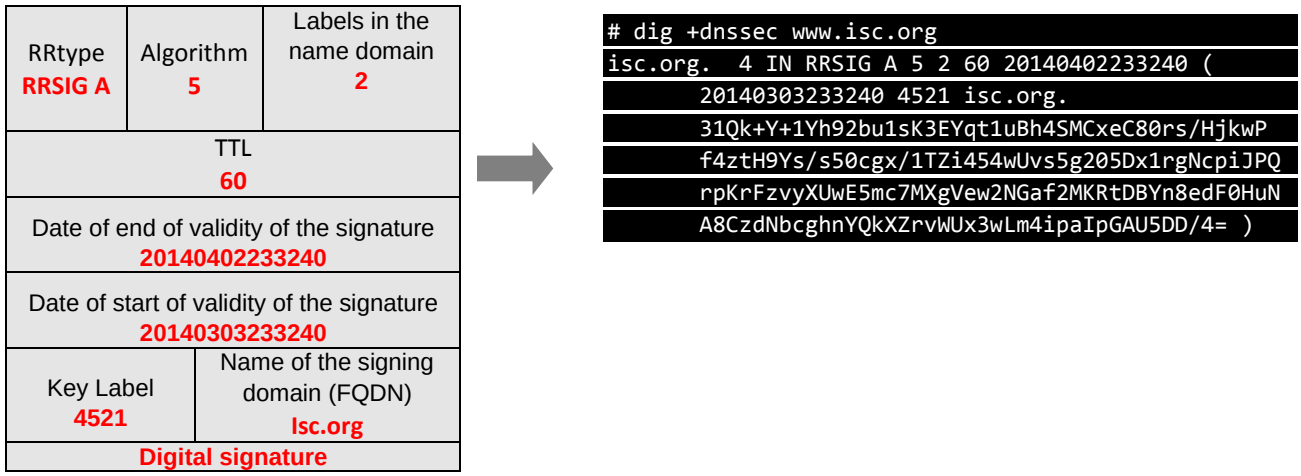


Illustration 24 Identification of Fields in DNSSEC Records.

- **DNSSEC Operations.**

  Signing and Delivering Sets of Records (RR sets). The signing of records is done for sets of records (RR sets) with the same domain name, class and type, for example, sets of type A, type NS, type DNSKEY or type DS (specific to DNSSEC), and so forth. The format of record described in the table *Table 1. Record Format. Resource Record* (RR). should be recalled. The RRSIG record is the most relevant, as it stores the digital signature and associated information (ID for the key used, start and expiry dates for the signature, and the like) for each group of records or RR set. The signature is with the private key corresponding to the pair of public and private keys generated for signing records (ZSK) or keys (KSK).

Signature Verification. A resolver can verify the digital signatures contained in RRSIG signature records by making use of the public key carried in DNSKEY records. Similarly, a DNSKEY record has its corresponding signed RRSIG. To verify public keys in themselves, the starting point is a *trust anchor*, a trusted public key from the highest node in the hierarchy (which is installed in the resolver) and which ideally would be from a root node for a domain considered "globally secure". The trust chain is built up by successive verification of the public keys of

daughter nodes, hash values for which are to be found in the DS records of their mother nodes, duly signed. The route for the chain of trust is called an *island of security* (*Illustration 25*)

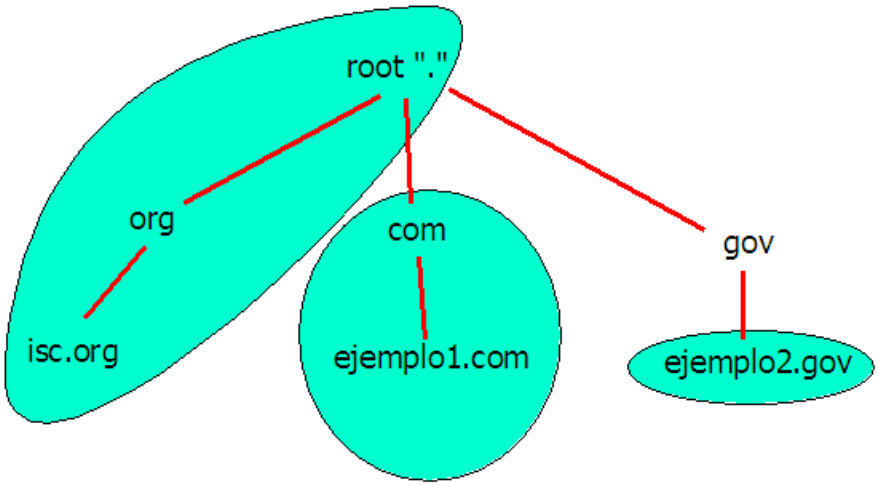


Illustration 25. Islands of Security

The following table shows the effects of the trust anchor in verification of the domain.

Trust Anchor Installed in the Resolver	State of the DNSSEC Response for Queries Directed to		
	www.isc.org	www.ejemplo1.com	www.ejemplo1.gov
none	insecure	insecure	Insecure
root	secure	insecure	Insecure
.com	insecure	secure	Insecure
ejemplo2.gov	insecure	insecure	Secure

Table 8. Trust Anchor and Domain Verification.

- **The Process of Resolution and Verification in DNSSEC.**

When requested to do so by a client, the authoritative server will add further DNSSEC data to the response. These data proceed from the digital signature of the record requested (RRSIG). If the resource requested does not exist, an NSEC3 record is sent in response to authenticate the answer. NSEC3 returns a hash value for the next authoritative domain in the server so as not to reveal in clear text the name of the following authoritative domain and thus to avoid attacks based on the enumeration of domains through the use of DNSSEC queries.

NOTE: NSEC3 is an improvement on the first version of the NSEC record, which was adjusted to avoid the enumeration of domains by requesting non-existent names. This is because previously a query about an inexistent domain yielded NSEC records which returned the next authoritative domain in the server and the types of records contained in it, along with the corresponding RRSIG signature, information of value to an attacker.

Verification of a Response.

The first thing a client will do is to verify the data received about the resource requested. For this purpose, it will generate a hash value for the set of records in the answer (RR set) using the algorithm referred to in the response. Additionally, it will use the public key (DNSKEY) received in the response (in the ADDITIONAL section) to check the signature included in the RRSIG record, obtaining a hash value that should coincide with the figure previously calculated. This confirms the authenticity of the data on the basis of the key and signature received.

Chain of Trust. Trust Anchor

However, what can guarantee that both DNSKEY and RRSIG have remained safe from modification? This is where verification of the chain of trust down from the trust anchor that the resolver client has installed and which is trusted comes into play. It may be a local trust anchor for a given domain (a security island), or ideally the public key of a root server for domain that is globally secure. Thus, if the DNSKEY is not a trust anchor it is necessary to verify its authenticity. To this end the mother zone is asked for the DS (*Delegation Signer*) record of the daughter domain which is being resolved.

In its answer the mother node includes:

- The DS containing the hash value for the public key it is required to validate (daughter node).
- The RRSIG signature record corresponding to the DS.
- The public key DNSKEY (mother node) to verify the signature.

With these data, just as when validating a record, the resolver should confirm the validity of the DNSKEY in itself that was used when verifying the signature of the record. For this purpose, it should also verify its signature, and this operation is carried out with the public DNSKEY of the mother node that signed the DNSKEY key of the daughter node. Once again, iteratively, if this latter DNSKEY (for the mother node) is not a trust anchor (public key which a resolver trusts and does not need to verify), the process is repeated with the immediately higher mother node in the DNS hierarchy, and it ends when a node is reached in which the trust anchor is to be found (in the ultimate instance the root node)..

This chain is shown in graphic form below. In detail, by following the steps numbered from 1 to 8 in *Illustration26*, it is possible to trace the process of checking that is performed until the trust anchor is reached in resolving the domain *www.isc.org*.

Using the ZSK key, the RRSIG for *www.isc.org* is validated (1). The RRSIG corresponding to the ZSK key is validated with the KSK key (2). This key must be verified if it is not a trust anchor. If this domain key is not already a trusted key or trust anchor, then the client must consult the mother zone (the *.org* zone) requesting the DS record for the daughter zone (*isc.org*). This query should return a DS record, an RRSIG record associated with the DS and a DNSKEY record for the mother zone. The DS record can be validated against the RRSIG, using the public key contained in the DNSKEY record (3, 4, 5). This public key must in turn be validated. This iterative process builds up a chain of trust that finally leads to the obtaining of a key that coincides with the trust key configured locally, in this instance the key for the zone *rootzone* (6, 7, 8). At his point the DNS response may be considered valid.

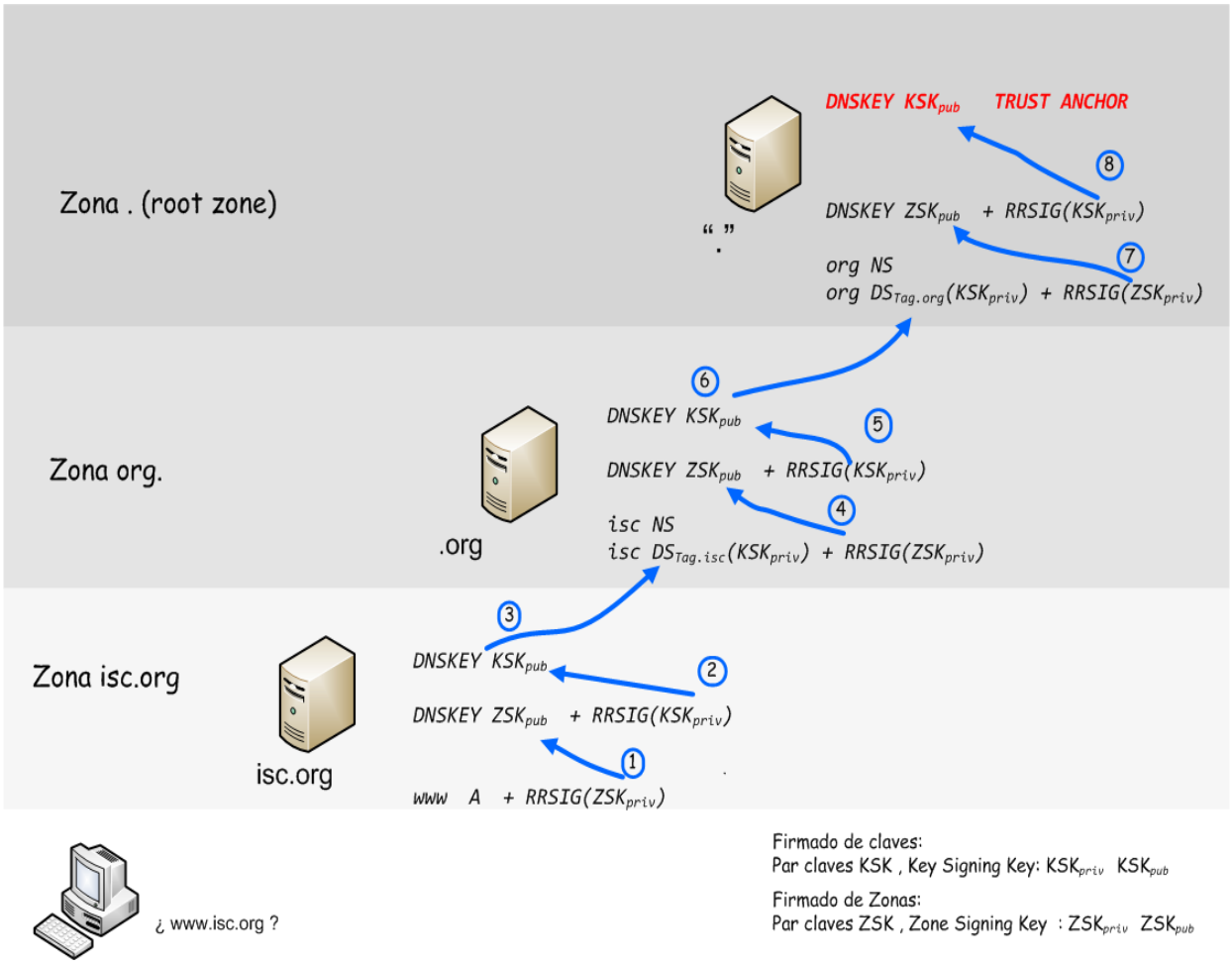


Illustration26. Verification of the Chain of Trust for www.isc.org.

Use of the DIG utility allows explicit DNSSEC verification. An instance can be seen in the Appendix *Test of the DNSSEC Chain of Trust Using DIG.*

DIFFICULTIES IN THE USE OF DNSSEC

DNSSEC does have a downside of various problems and difficulties that arise in its use. Among them, the following are the most noteworthy.

Difficulty of Implementing and Maintaining. Relative to DNS, it is more complex to implement and requires careful attention in maintaining zones and keys. Any problem related to signing or keys that are out of date will cause problems in resolution for DNSSEC clients.

Size of Responses. A DNSSEC answer is strikingly larger than a conventional DNS response, which involves a greater use of network resources and provides a vector exploitable by DNS amplification attacks, as was explained in the section on denial of service attacks, *DNS*

DNSSEC Performance and Resolution. The process of resolving and verifying signatures imposes an increased processing load on the resolver. This may have an impact on response times and cause resubmissions by clients, adding even more to the load**.**

Key Renewal. Correct renewal of keys requires extra attention when administering the server.

Synchronization. Since it is necessary to check time-stamps in order to determine periods of validity for the verification of signatures and validity of keys, correct time synchronization is needed in respect of the reference by the signer. This may constitute an attack vector if it proves possible to make malicious changes in the time synchronization reference of a server, as this might trigger a denial of service linked to problems of verification of validity periods.

DEPLOYMENT OF DNSSEC.

Each organization is different, so each one should study deployment and draw up its own plan for it. In principle, there are certain basic recommendations for implementing DNSSEC that are indicated below.

- **Establishing a Security Policy for DNSSEC**
  - Determining which zones need to be signed. Most organizations generally start by signing only their public Internet zones.
  - Deciding which servers will serve signed zones, updating and adapting software accordingly.
  - Establishing procedures for key generation and renewal, secure storage and renewal frequency.
  - Choosing suitable cryptographic parameters, such as encryption algorithms, key lengths, validity periods, and the like.

- **Deployment and Testing.**

The implementation of DNSSEC should be designed by following manuals of good practice. One such implementation is described by ENISA22 in its publication *Good Practice Guide for Deploying DNSSEC*[23]

A pilot test should be carried out with a copy of an existing signed zone. A sub-domain can be set up, the trust anchor configured and validation tested.

[22] ENISA: European Union Agency for Network and Information Security. http://www.enisa.europa.eu
[23]Good Practice Guide for Deploying DNSSEC: http://www.enisa.europa.eu/activities/Resilience-and-CIIP/networks-and-services-resilience/dnssec/gpgdnssec

LISTS AND REFERENCES

REFERENCES

[RFC1034] *Mockapetris, P., "Domain Names - Concepts and Facilities," STD 13, RFC 1034, November 1987.*

[RFC1035] *Mockapetris, P., "Domain Names - Implementation and Specification," STD 13, RFC 1035, November 1987.*

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

[RFC4033] *R. Arends, R. Austein, M. Larson, D. Massey, S. Rose "DNS Security Introduction and Requirements", March 2005*

[RFC2827] *Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing," BCP 38, RFC 2827, May 2000.*

[RFC3013] *Killalea, T., "Recommended Internet Service Provider Security Services and Procedures," BCP 46, RFC 3013, November 2000.*

[RFC3833] *Atkins, D. and R. Austein, "Threat Analysis of the Domain Name System (DNS)," RFC 3833, August 2004.*

[RFC4033] *Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security - Introduction and Requirements," RFC 4033, March 2005.*

[vu-457875] *United States CERT, "Various DNS Service Implementations Generate Multiple Simultaneous Queries for the Same Resource Record," VU 457875, November 2002.*

[RFC1034] *Mockapetris, P., "Domain Names - Concepts and Facilities," STD 13, RFC 1034, November 1987.*

[RFC1035] *Mockapetris, P., "Domain Names - Implementation and Specification," STD 13, RFC 1035, November 1987.*

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

[RFC4033] *R. Arends, R. Austein, M. Larson,D. Massey, S. Rose "DNS Security - Introduction and Requirements", March 2005*

DOCUMENTATION

ENISA	*European Union Agency for Network and Information Security*
US-CERT	*Unites States Computer Emergency Readiness Team*
DHS	*U.S. Homeland Security Department*
NIST	*National Institute of Standards and Technology*
ISC	*[RFC1034]*

LIST OF ILLUSTRATIONS

Illustration 1. Hierarchy of the Name Space 7
Illustration 2. Domain 69.108.4.213.in-addr.arpa..... 8
Illustration 3. Inverse Resolution IP 213.4.108.69 9
Illustration 4. Authoritative Answer and Cached Answer. Note the flag aa (authoritative answer). .. 10
Illustration 5. Section Header in a DNS Message..... 14
Illustration 6. An Example of an A Type DNS Query. 15
Illustration 7. EDNS0 Extended Format. 4096 Byte UDP. 15
Illustration 8. Iterative and Recursive Queries.. 17
Illustration 9 Sucession of Queries in a Recursive Resolution. 18
Illustration 10. Sucession of Iterative Queries in a DNS Resolution..... 19
Illustration 11. Record of the Type SOA (Start of Authority). Note the Refresh Value. 20
Illustration 12 DNS Scenario. Attack Routes and Classification of Threats..... 23
Illustration 13. Cache Poisoning Attack. Once the original ID has been discovered, a false response is sent. 26
Illustration 14. A DNS Amplification Attack. 29
Illustration 15. The Amplification Factor. A query of type ANY, with a length of 66 bytes, gets a response of 2,066 bytes..... 30
Illustration 16. World Distribution of Open Resolvers. Source: DNS Amplification Attacks Observer.31
Illustration 17. Reinforcing DNS. Layers..... 34
Illustration 18. Functional Roles of DNS Servers. Cache and Authoritative Servers. 40
Illustration 19. Network Architecture. Implementation of a DNS Infrastructure. 42
Illustration 20. BIND Logs. Flooding with Queries in a DNS Amplification Attack 45
Illustration 21. BIND Logs. Detection of Flooding. Rate-Limit in Log-Only Mode. 46
Illustration 22.BIND Logs. Detection of Flooding. Rate-Limit Activated and with Response Discards.46
Illustration 23. Version of BIND Hidden. 51

Illustration 24 Identification of Fields in DNSSE Records. 54
Illustration 25. Islands of Security..... 55
Illustration 26. Verification of the Chain of Trust for www.isc.org. 57
Illustration 27. Response Longer than 512 Bytes. Truncation and Change to TCP 66
Illustration 28. Query Specifying a UDP Buffer. Notice the pseudosection EDNS0 67
Illustration 29. Root Server Trust Anchor 67

LIST OF CONFIGURATIONS

CONFIGURATION 1. HIDING INFORMATION ABOUT THE BIND SOFTWARE. 35
CONFIGURATION 2. NON-PRIVILEGED USER FOR RUNNING BIND..... 36
CONFIGURATION 3. CHROOT. STRUCTURE OF CHROOT DIRECTORIES..... 36
CONFIGURATION 4. CREATING A JAIL (OR CAGE) ENVIRONMENT FOR BIND..... 37
CONFIGURATION 5. PROTECTION AND FILE PERMISSIONS FOR BIND..... 37
CONFIGURATION 6. LOGGING CONFIGURATION..... 38
CONFIGURATION 7. START-UP OF BIND IN A CHROOT ENVIRONMENT..... 39
CONFIGURATION 8. RESTRICTION OF RECURSION AND ZONES. USE OF VIEWS..... 44
CONFIGURATION 9. BIND, RESPONSE RATE LIMITING CONFIGURATION AGAINST FLOODING ATTACKS..... 45
CONFIGURATION 10. IP FILTERING FOR ZONE TRANSFER..... 46
CONFIGURATION 11. TSIG FOR ZONE TRANSFERS. 47
CONFIGURATION 12. IP FILTERING FOR RECEIVING NOTIFICATIONS 47
CONFIGURATION 13. USE OF TSIG IN SERVER-TO-SERVER TRANSACTIONS. 47
CONFIGURATION 14. DYNAMIC UPDATES WITH TSIG..... 48
CONFIGURATION 15. SOA RECORD CONFIGURATION..... 50
CONFIGURATION 16. HIDING THE VERSION OF BIND. 50
CONFIGURATION 17. TSIG CONFIGURATION. KEY STORED IN A FILE. 63

APPENDICES

TRANSACTION SIGNATURE. TSIG.

TSIG, the acronym being derived from *Transaction SIGnature*, is a mechanism for verifying the identity of DNS servers with which communications are under way. TSIG was described in RFC 2845[24]. This mechanism was designed before the appearance of DNSSEC to give DNS transactions authentication and integrity and originally was thought out for the protection of dynamic updates.

Message Authentication Code (MAC).

TSIG makes use of a MAC (Message Authentication Code) and employs a key shared among primary servers and slaves so as to code cryptographically the messages they exchange. As they are shared, keys must be distributed through secure routes or channels and be changed relatively frequently so as to reduce the risk of key compromise.

The RRSIG Record.

A TSIG transaction includes an RRSIG record with the MAC, which is obtained from a hash value for the DNS record to be transmitted (which brings integrity) and this in turn is encrypted with the shared key (bringing authentication). In this way, the server consulted will send the requested record and its corresponding MAC in the RRSIG record. At the receiving end, the MAC contained in the RRSIG is verified by using its copy of the shared key and applying the same operation that was carried out at the transmitting end. If the verification is correct and both agree, the transaction is accepted.

Configuring TSIG in BIND.

In configuring TSIG it is necessary to generate the shared key for encryption. This is done with the BIND utility *dnssec-keygen*, indicating which encryption algorithm is preferred for its generation. The recommended choice is the HMAC-MD5 algorithm, as it is one that must obligatorily be supported in the TSIG specification of the DNS. This generation is shown below (*Chart 1*).

```
# dnssec-keygen –a hmac-md5 –b 128 –C –n host ejemplo.com
# ls
Kejemplo.com.+157+31456.key
 Kejemplo.com.+157+31456.private<- - - Contains the key to be used

# cat Kejemplo.com.+157+31456.private
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
 Key: JuhsyAfsdsRiW4fs90==<- - Key generated
 Bits: AAA=
```

Chart 1. Generating a TSIG Shared Key

[24] RFC2845: Secret Key Transaction Authentication for DNS (TSIG)http://www.ietf.org/rfc/rfc2845.txt

This key is the one used in *named.conf* configuration files. It may be written directly with the *key* command in the *options* clause of the file *named.conf*, but it is more advisable to store it in a local file in all the servers, protected against unauthorized access, and to refer to this file in the *named.conf* configuration, as shown below (*Chart 2*):

```
# mv Kejemplo.com.+157+31456.private/chroot/named/keys/ejemplo.com.key

# chown named:named /chroot/named/keys/ejemplo.com.key
# chmod 0400 /chroot/named/keys/ejemplo.com.key

# cat/chroot/named/keys/ejemplo.com.key
 Private-key-format: v1.2
 Algorithm: 157 (HMAC_MD5)<- - - Algorithm
Key: JuhsyAfsdsRiW4fs90==<- - - Key in base 64
 Bits: AAA=
```

Chart 2. Protecting the TSIG Shared Key File

Before including it in the configuration of *named.conf*, it is necessary to edit the key file, giving it the format of the key clause used by BIND. As may be seen in *Chart 2*, the file comprises four lines, of which two are essential: the algorithm used and the key in base 64, so as to construct the key clause that is included in *named.conf*. The result may be seen in *Chart 3* below.

```
# vi/chroot/named/keys/ejemplo.com.key

# cat/chroot/named/keys/ejemplo.com.key

key “ejemplo.com” {
algorithm hmac-md5;
secretJuhsyAfsdsRiW4fs90==;
};
```

Chart 3. Key File with the Format of a Key Clause

Once the key file has the necessary format, it can be referenced directly in the configuration file *named.conf*, as seen below.

```
options {
directory /chroot/named
include “keys/ejemplo.com.key”;// include the file that contains the
                                  // key clause
server { 10.1.2.3 ;};// Specifying that transactions with 10.1.2.3 are
  keys {“ejemplo.com”;};// signed with the key called “ejemplo.com”

}
zone “ejemplo.com” in {
Type master;
file “etc/zone_master.ejemplo.com”
allow-transfer { key “ejemplo.com”;};// only zone transfer permitted
}// with key ejemplo.com
```

Configuration 177. TSIG Configuration. Key Stored in a File.

PRACTICAL EXAMPLES OF THE USE OF DIG

This section presents several examples of the use of the application DIG. DIG is a tool distributed with the BIND software package that is directed towards interaction with DNS servers. It is of great use in diagnostic tasks or simply to obtain information on DNS resources.

- **Basic Syntax**

```
dig [@server_dns] [domain] [type_resource] [clase] [options]
```

type_resource: A, TXT, MX, NS, SOA, ANY and so forth (see Section 2.3)
class: IN, CH
options: DIG has a wide range of options. They can be checked with the command *dig -h*

Some of the commoner options are:

+short→ abbreviated mode, showing only the response
+tcp→ use tcp for the query
+recurse → force recursive query
+nostats→ do not show statistics

- **Basic Use**

Use is made of the domain *example.com* for the examples

Queries about Resources:

Record A

```
kuko@DNS ~ dig example.com A

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> example.com A
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36611
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com. INA

;; ANSWER SECTION:
example.com.4935INA 93.184.216.119
```

Record A, Abbreviated Mode:

```
kuko@DNS ~ dig example.com A +short
93.184.216.119
```

Name Server (NS) Query:

```
kuko@DNS ~ dig example.com NS +short
b.iana-servers.net.
a.iana-servers.net.
```

TXT Records:

```
kuko@DNS ~ dig example.com TXT +short
"v=spf1 -all"
"$Id: example.com 1921 2013-10-21 04:00:39Z dknight $"
```

Zone Transfer Transaction

```
kuko@DNS ~ dig example.com AXFR
; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> example.com AXFR
;; global options: +cmd
; Transfer failed.
```

- **Queries Specifying the EDNS0 Buffer**

If the DNS server implements EDNS0 (see *0 DNS DNS* Messages.) it is possible to send DNS messages of a length greater than 512 bytes over UDP if this is requested.

By default, if the response to a query over UDP is longer than 512 bytes, the server suggests that the client should submit the request once again over TCP with a 'truncation' signal.

In the following query, no UDP buffer is specified. In this way, if the response is longer than 512 bytes (the default value UDP) retransmission with TCP will be triggered. It should be noted in the following illustration (*Illustration 27*) how the response, of length 2,628 bytes, is received after a second attempt over TCP.

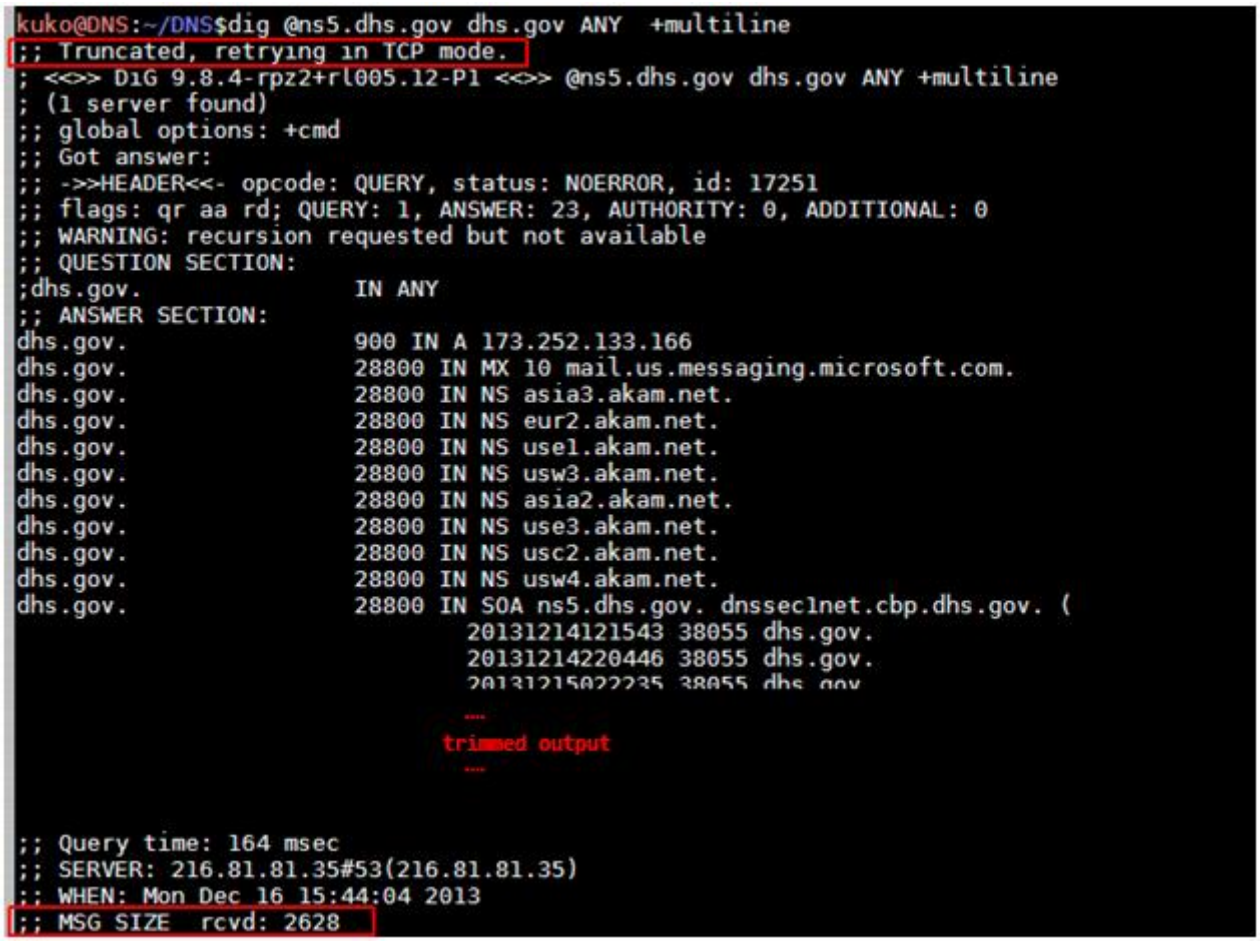


Illustration 27. Response Longer than 512 Bytes. Truncation and Change to TCP.

The query is repeated once more, but this time UDP is forced, specifying a buffer of 4,096 bytes. Note should be taken of the OPT section (*Illustration 28*):

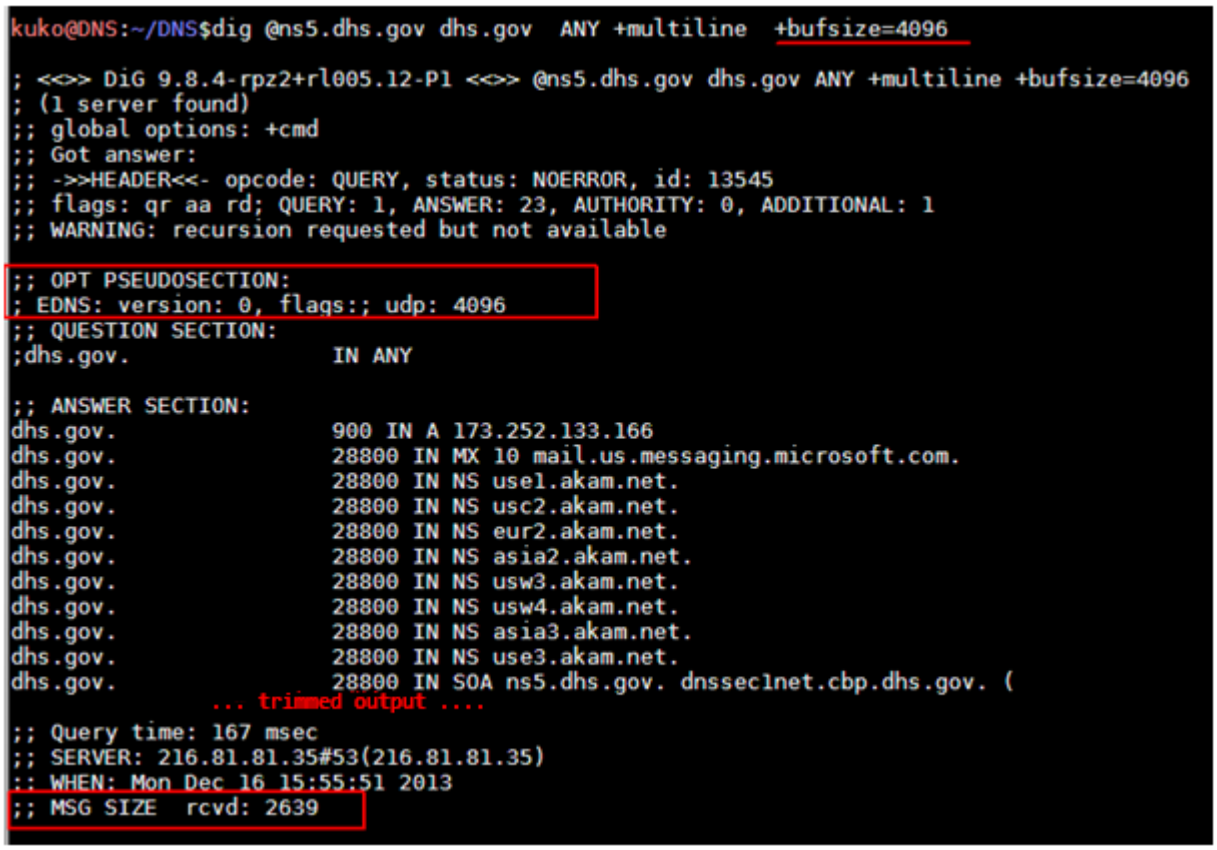


Illustration 28. Query Specifying a UDP Buffer. Notice the pseudo-section EDNS0.

- **Test of the DNSSEC Chain of Trust Using DIG.**

To verify the DNSSEC chain of trust, the starting point is a "trust anchor" or trust key, which will provide the basis for the verification. In this example the keys from the root servers are used, being downloaded and stored in a file.

DNSSEC keys are obtained from root servers. (Trust anchors).

```
dig . DNSKEY
```

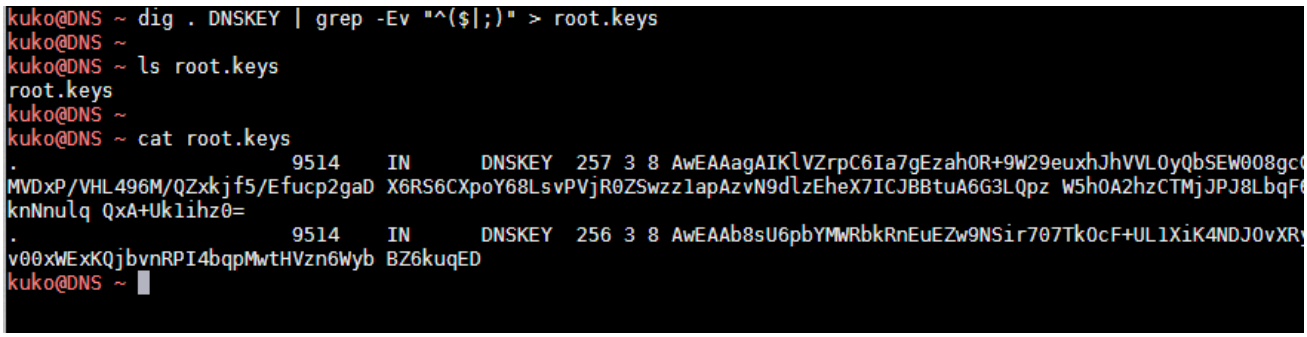


Illustration 29. Root Server Trust Anchor.

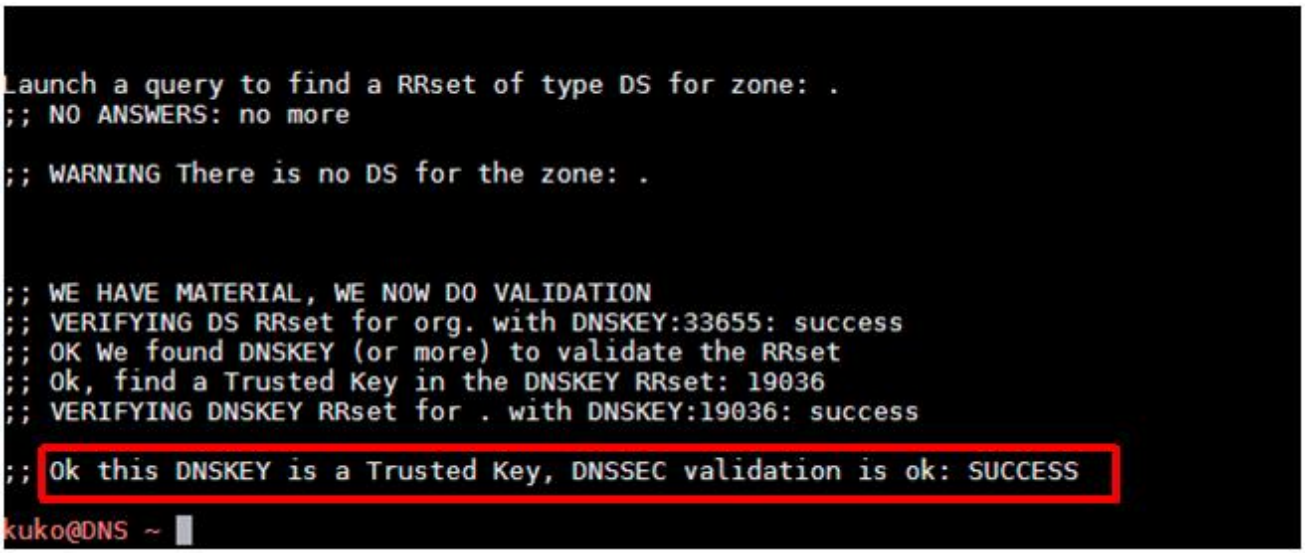
Next, taking the keys obtained from the root servers as a trust anchor, a verification chain is launched with dig: *dig +sigchase +trusted-key=./root.*

```
kuko@DNS ~ dig +sigchase +trusted-key=./root.keys www.isc.org A +multiline
;; RRset to chase:
www.isc.org.            59 IN A 149.20.64.69


;; RRSIG of the RRset to chase:
www.isc.org.            59 IN RRSIG A 5 3 60 20140402233240 (
                           20140303233240 4521 isc.org.
                           VGyObmU7Arxur6ygjb/KXh/3GBRbbolWMn7xZCaY9/Rz
                           0mPmRsqI42CBI9vXdvdho7ZLu+4/EcZsnAzF4Y1tTSLM
                           vmGRneE4IFq6wRHC3UCmYh0GAqt0chJ0IbjGTFGFYKzM
                           RXWOSuQHvVztep/wsMR+Reth2Ab4PRQhG4Vc9dI= )



Launch a query to find a RRset of type DNSKEY for zone: isc.org.

;; DNSKEYset that signs the RRset to chase:
isc.org.                7199 IN DNSKEY 257 3 5 (
                           BEAAAAOhHQDBrhQbtphgq2wQUpEQ5t4DtUHxoMVFu2hW
                           LDMvoOMRXjGrhhCeFvAZih7yJHf8ZGfW6hd38hXG/xyl
                           YCO6Krpbdojwx8YMXLA5/kA+u50WIL8ZR1R6KTbsYVMf
                           /Qx5RiNbPClw+vT+U8eXEJmO20jIS1ULgqy347cBB1zM
                           nnz/4LJpA0da9CbKj3A254T515sNIMcwsB8/2+2E63/z
                           ZrQzBkj0BrN/9Bexjpiks3jRhZatEsXn3dTy47R09Uix
                           5WcJt+xzqZ7+ysyLKOOedS39Z7SDmsn2eA0FKtQpwA6L
                           XeG2w+jxmw3oA8lVUgEf/rzeC/bByBNsO70aEFTd
                           ) ; key id = 12892
                        7199 IN DNSKEY 256 3 5 (
                           AwEAAbJpDF4RemdHHE/HrJJhR3zpzAQ6zsHqFv0i4lCW
                           TUf4sX+cq3vSu7fKO4QJtm97S1sbcnmHonVE3QPzLOsq
                           sY630Wy5JzrPK3gUvQLgfIsovo2v+dosITL8WbvjU1mE
                           XhIwfuuBhYmYSKySZ0X9gpHGhdxRd+J8M7riPfN7kHLP
                           ) ; key id = 4521


;; RRSIG of the DNSKEYset that signs the RRset to chase:
isc.org.                7199 IN RRSIG DNSKEY 5 2 7200 20140402230130 (
                           20140303230130 4521 isc.org.
                           NS2bTHzpgatqh4LVtU0x9aVi5yJ4gpIg6lmSB2VJFoJL
                           P+ySeG/srlHqecMni9UxmmmR6qNlCwXqjLmd5IHOrIG0
                           yniepDycwYkkX1jDnF+LDcIgS8YFxURUjDWM24WHpGsM
                           oj0pV15/aZ8LB7MVikNOiysDcIvXjgLDJxGgWuo= )
                        7199 IN RRSIG DNSKEY 5 2 7200 20140402230130 (
                           20140303230130 12892 isc.org.
                           AMnoRGL1DxDFNZTv0GVxmIu2J0kZeiLo8kjVHbn873rM
                           KxzOzxCsnWEU2fBqm3lhXU/w7NnUFNHvswNN9SkroLk4
                           DxBrlkVTxorkTdlsiC6S0rSALzN5Brr/sqshkBywfIKD
                           c00UgAqTCuhfTdYfLwghpR4i3Z0SImvs6hFZw0zCSTMd
                           Wdx+l/rLAwt/GFnwQuc0q6SVOMk6A3FWzKp0JIBysD3R
                           Q2kiRunNEb0il/3KU7bA3V74AMWJufZP7awgRZaV/k1M
                           vMSQ6G9xoC1TETPMe+uR3x3MymwHChZ6XfGQ7CfmcpTu
                           7P3HPbAi+SaVj6S1wkU85GCg3f/3AypGLA== )
```

```
Launch a query to find a RRset of type DS for zone: isc.org.

;; DSset of the DNSKEYset
isc.org.                21599 IN DS 12892 5 1 (
                                982113D08B4C6A1D9F6AEE1E2237AEF69F3F9759 )
                        21599 IN DS 12892 5 2 (
                                F1E184C0E1D615D20EB3C223ACED3B03C773DD952D5F
                                0EB5C777586DE18DA6B5 )


;; RRSIG of the DSset of the DNSKEYset
isc.org.                21599 IN RRSIG DS 7 2 86400 20140318154949 (
                                20140225144949 24209 org.
                                DGeds8tNSuQd3z5DNHwCqtHncjVshmP5iylFwALF+l58
                                GxYVHigZBDwXBFDejW+QIxH8rfkrr8X0hDul9X/URemn
                                igsfmxFZS2gZYg+610WsaKwwMncJF5I/2fMxV0eTYDb+
                                mQLoioZ48ri7/xQJlVW2Hp4tMTa/NRDqA6wZ248= )



;; WE HAVE MATERIAL, WE NOW DO VALIDATION
;; VERIFYING A RRset for www.isc.org. with DNSKEY:4521: success
;; OK We found DNSKEY (or more) to validate the RRset
;; Now, we are going to validate this DNSKEY by the DS
;; OK a DS valids a DNSKEY in the RRset
;; Now verify that this DNSKEY validates the DNSKEY RRset
;; VERIFYING DNSKEY RRset for isc.org. with DNSKEY:12892: success
;; OK this DNSKEY (validated by the DS) validates the RRset of the DNSKEYs, thus th
;; Now, we want to validate the DS :  recursive call


Launch a query to find a RRset of type DNSKEY for zone: org.

;; DNSKEYset that signs the RRset to chase:
org.                    52 IN DNSKEY 257 3 7 (
                                AwEAAZTjbIO5kIpxWUtyXc8avsKyHIIZ+LjC2Dv8naO+
                                Tz6X2fqzDC1bdq7HlZwtkaqTkMVVJ+8gE9FIreGJ4c8G
                                1GdbjQgbP1OyYIG7OHTc4hv5T2NlyWr6k6QFz98Q4zwF
                                IGTFVvwBhmrMDYsOTtXakK6QwHovA1+83BsUACxlidpw
                                B0hQacbD6x+I2RCDzYuTzj64Jv0/9XsX6AYV3ebcgn4h
                                L1jIR2eJYyXlrAoWxdzxcW//5yeL5RVWuhRxejmnSVnC
                                uxkfS4AQ485KH2tpdbWcCopLJZs6tw8q3jWcpTGzdh/v
                                3xdYfNpQNcPImFlxAun3BtORPA2r8ti6MNoJEHU=
                                ) ; key id = 9795
                        52 IN DNSKEY 256 3 7 (
                                AwEAAXIJ6PWJHkI4gLfGoHDPxQwS1kVKhYcjEwKn76TM
                                EQgw3mr2rDMsKiC7vfTnaGxIbqCodo4xNixVp8MgAuUA
                                +YLrSPft5ivGLkTXmZnxmKTaJocMVsyGjLiQL0oJTFJC
                                H25xf/wBHJ7PAeqbaQvgRGLTR8Jmqyjfrg UxZw0qRhGN
                                ) ; key id = 24209
                        52 IN DNSKEY 256 3 7 (
                                AwEAAa+yHvpOo3f7XS1vtKPGH6AD1OkmYUtnRlkkCO9B
                                KJ0OCCvYSWh5NWLJjIMXRzVpituqoLtiYfhdDYQH5JzR
                                VW6lCtT+2SiWmEx+7GnSyMT48858uC02AYlJVfbitCpo
                                GGdzyLTiMxtMlztpRyCAvaDujnx+2GBo7zgb50f5gQJp
                                ) ; key id = 1829
```

```
                        52 IN DNSKEY 257 3 7 (
                                AwEAAYpYfj3aaRzzkxWQqMdl7YExY81NdYSv+qayuZDo
                                dnZ9IMh0bwMcYaVUdzNAbVeJ8gd6jq1sR3VvP/SR36mm
                                GssbV4Udl5ORDtqiZP2TDNDHxEnKKTX+jWfytZeT7d3A
                                bSzBKC0v7uZrM6M2eoJnl6id66rEUmQC2p9DrrDg9F6t
                                XC9CD/zC7/y+BNNpiOdnM5DXk7HhZm7ra9E7ltL13h2m
                                x7kEgU8e6npJlCoXjraIBgUDthYs48W/sdTDLu7N59rj
                                CG+bpil+c8oZ9f7NR3qmSTpTP1m86RqUQnVErifrH8Kj
                                DqL+3wzUdF5ACkYwt1XhPVPU+wSIlzbaAQN49PU=
                                ) ; key id = 21366


;; RRSIG of the DNSKEYset that signs the RRset to chase:
org.                    52 IN RRSIG DNSKEY 7 1 900 20140318154949 (
                                20140225144949 9795 org.
                                OfT8srp+lEzCcGDHOX0xPb0Z0Vmht+w2yxPJnTfkOqBJ
                                yVkhWV0lDfe8wfPH+xHzKtAw0E+91Hz9H1S4t+6C+Yp8
                                Ls/+F+z80h5Gkjq0tTOYN8kEfCSGYK3rAL6tq4lq6ZPZ
                                +cv9GizeOmWX7wUxkVETUU2+KVoGHLRpgEGQOI6IGR1Y
                                8gsA7NAY2AYCPJ+IekQPqfj/Y4up1dCdI51jhbe1n1zL
                                RDCYIYYBKX6xtaXqjbQUEQ4yv/H2thCdzjosLBvzVj1d
                                W+qRywhNrKWw30bWZYZxSerxSrpivrzikjVnWBfetIbo
                                OeO56u0D7Bp0oH8ACc6ocPIdAd+H3JFuBw== )
                        52 IN RRSIG DNSKEY 7 1 900 20140318154949 (
                                20140225144949 21366 org.
                                hhrN8ugInW5Zag9cjMDdPgbk5nbUp1aXDGQPXRSTZUoK
                                Ctsg2CzpDqN3wx9ly+0aeO5SUexTJM/i9hNChQayLmTj
                                Ts42kpQxV4pgKXBxxL6dOaFIOPv3SPssQlKPm3OoEBrb
                                WW1rpE3E/nXbAaFMnSQoHzi19xVxhoJt382e8uBYIL8/
                                zrGTrmgyEwl9eubam9zXS++GVHN+5CbqaT3S/39I7SAF
                                iusoT0qv1+mz9SVy1IJSezNh+sLCiJIhiZtlGTSwEFXF
                                HwlAlZuKoJ4cl37n+4nTGnxm13r0U/LKxXItNQ80GVfB
                                8tPCbES+/ujS69t0/f/Se43fEa7qPQLtpw== )
                        52 IN RRSIG DNSKEY 7 1 900 20140318154949 (
                                20140225144949 24209 org.
                                OEfP9P1qN0fytGJrqmcOqnMWjVM4rrXntrfDSTvoUSwu
                                6Gmdh9aLPU9uxoS1WBrS9FBLih5KJPj6JzcsLL4kFzjw
                                7z9F59bREx6dcY9XhiK/w0yf7O1ABB3COwpPcCjOM2Db
                                tZu1LC/aXLK7P4QRhB95D3UcJl1zI3tepkxCBu0= )



Launch a query to find a RRset of type DS for zone: org.

;; DSset of the DNSKEYset
org.                    8732 IN DS 21366 7 2 (
                                96EEB2FFD9B00CD4694E78278B5EFDAB0A80446567B6
                                9F634DA078F0D90F01BA )
                        8732 IN DS 21366 7 1 (
                                E6C1716CFB6BDC84E84CE1AB5510DAC69173B5B2 )
```

```
;; RRSIG of the DSset of the DNSKEYset
org.                    8732 IN RRSIG DS 8 1 86400 20140314000000 (
                                20140306230000 33655 .
                                B1FZwn3bjwCiBfiMlpLRlEIukKZJjALVYFhWSzhYrLuj
                                8pe4B1t7f1iFJTkCBJOd5N9B0QbDJhfP0Nz5rgv82v6f
                                eSlHbkSBp1jzG1M/o6EQweODF4kU5PqdBEEga73bNgT5
                                r+g/28lHxrXj9aQDeYWFH+M4Wp4/d8WaqTSg+uA= )



;; WE HAVE MATERIAL, WE NOW DO VALIDATION
;; VERIFYING DS RRset for isc.org. with DNSKEY:24209: success
;; OK We found DNSKEY (or more) to validate the RRset
;; Now, we are going to validate this DNSKEY by the DS
;; OK a DS valids a DNSKEY in the RRset
;; Now verify that this DNSKEY validates the DNSKEY RRset
;; VERIFYING DNSKEY RRset for org. with DNSKEY:21366: success
;; OK this DNSKEY (validated by the DS) validates the RRset of the DNSKEYs, thus th
;; Now, we want to validate the DS :  recursive call


Launch a query to find a RRset of type DNSKEY for zone: .

;; DNSKEYset that signs the RRset to chase:
.                       9238 IN DNSKEY 257 3 8 (
                                AwEAAagAIKlVZrpC6Ia7gEzahOR+9W29euxhJhVVLOyQ
                                bSEW0O8gcCjFFVQUTf6v58fLjwBd0YI0EzrAcQqBGCzh
                                /RStIoO8g0NfnfL2MTJRkxoXbfDaUeVPQuYEhg37NZWA
                                JQ9VnMVDxP/VHL496M/QZxkjf5/Efucp2gaDX6RS6CXp
                                oY68LsvPVjR0ZSwzz1apAzvN9dlzEheX7ICJBBtuA6G3
                                LQpzW5hOA2hzCTMjJPJ8LbqF6dsV6DoBQzgul0sGIcGO
                                Yl7OyQdXfZ57relSQageu+ipAdTTJ25AsRTAoub8ONGc
                                LmqrAmRLKBP1dfwhYB4N7knNnulqQxA+Uk1ihz0=
                                ) ; key id = 19036
                        9238 IN DNSKEY 256 3 8 (
                                AwEAAb8sU6pbYMWRbkRnEuEZw9NSir707TkOcF+UL1Xi
                                K4NDJOvXRyX195Am5dQ7bRnnuySZ3daf37vvjUUhuIWU
                                AQ4stht8nJfYxVQXDYjSpGH5I6Hf/0CZEoNP6cNvrQ7A
                                FmKkmv00xWExKQjbvnRPI4bqpMwtHVzn6WybBZ6kuqED
                                ) ; key id = 33655


;; RRSIG of the DNSKEYset that signs the RRset to chase:
.                       9238 IN RRSIG DNSKEY 8 0 172800 20140316235959 (
                                20140302000000 19036 .
                                ia/OHAcEyCgkaADe5cO7DSGpNyNskxaDQQXvivOUe4Wo
                                PXye82dEgObGKqtHc2TSJ/PNsczGdUPr48ZbujbM6mBr
                                NKqSloX7fl0eCciXcKXKLNzoCG1WDFxKPno/VQS3+Ev+
                                6ll9jaNPu/sCIq0ziSh/7hw8sNzG2XBmBq1lJRZ/SZBX
                                VrzH0E8kNjW/BQAk/D8K0VMKvR7JAzd/rXUkiSYkj8h2
                                0lW6Zi+5mTm5c51Ujn+CzUPsudVwnUesD+Um7wAXgV0J
                                F/Rv5/Pd21VppsLQG25qf4KkItHpNXc1Re005J7Sr5jH
                                ouhgIo4Duc+4NtinY/D49jjr2+YU0Fza8w== )
```



USEFUL LINKS AND TOOLS

This section lists resources and tools useful for managing, problem-solving and auditing DNS systems.

- **On-Line Tools**

FUNCTIONALITY	URL
DNSSEC validation tests	http://dnssec.vs.uni-due.de/
Malicious DNS traffic. Umbrella Labs and Open DNS	http://dnsviz.net/
	http://labs.umbrella.com/global-network/
General tests of the state of the DNS service	http://www.dnsstuff.com/
	http://www.dnsinspect.com/
	http://dr.xoozoo.com/
	http://www.simpledns.com/lookup.aspx
Information about domains	https://www.robtex.com

Table 9. On-Line DNS Diagnostic Tools

- **Software Tools**

FUNCTIONALITY	TOOL
Client tools for DNS resolutions	*Dig (bind).* http://www.isc.org/downloads/
	Nslookup. The same use as dig, but superseded by it
Tools for scanning and obtaining domain information	*Fierce.* http://ha.ckers.org/fierce/
	Dnsenum. https://code.google.com/p/dnsenum/
	Dnsrecon. https://github.com/darkoperator/dnsrecon/blob/master/dnsrecon.py
Multiple tests to check the DNS consistency and validity of a domain.	*Dnswalk.* http://sourceforge.net/projects/dnswalk/

Table 10. DNS Software and Tools