Domain Name System with Security Extensions

Charishma G Shivaratri
Computer Science and Engineering
University of Texas at Arlington
charishma_gs@hotmail.com

Abstract

Names are needed to abstract away details of location, authorization and human readability. In the vast world like the Internet, it becomes important to have naming systems to help the user select or extract the information he requires, in a form that is human readable. Many naming services are implemented, Domain Name System (DNS) being one of them. DNS was invented to provide a scalable naming system for the Internet and it provides a good mapping between human understandable mnemonics and machine-readable IP addresses (Internet Protocol addresses). DNS was developed early in the history of the Internet when the risk environment was more benign, thus is vulnerable to attacks. This paper describes DNS and measures taken to improve its performance. New security enhancements are implemented to counter the threats and to protect data authenticity and integrity.

Keywords
Domain Name System (DNS), Spoofing, Cache-poisoning, digital signatures, Authentication, Encryption.

1. Introduction[15]

Naming is one of the most important and most frequently overlooked areas of computer science. A **name** is a symbolic representation of an object or an action. A name only has meaning within a particular naming **context**. A **name space** is the set of all possible potential names in a particular context. The naming **domain** is the set of all possible things that can be named in a particular context. A **binding** is a mapping from a particular name to a particular object. Within a context a name has at most one binding. **Resolution**, or name lookup, turns a name into the thing it represents. An **attributive name** is a name whose resolution returns the object in question, or provides a name for the object in a different context. If this new name can then be used to access the object the new name is an **address**. Conventionally, a name has tended to mean a logical way of referring to an object in some abstract name space, while the term address has been used for something that specifies the physical location. A name server is used to convert names into addresses.

With the advent of Internet and with wide-area distributed systems, it becomes important to have a naming system, which must be standard, structured and scalable. The system should also contain information in human readable form that allows users to navigate through the vast library along with addressing formats to the name server to know where to start searching. Domain Name System is one such Naming Service. It is tree structured name space database, whose

principle task is to provide a mapping between the human readable mnemonics to numerical machine-readable IP addresses. DNS was introduced when threat factors to the Internet environment was not well introduced. It has therefore become important to address all vulnerabilities of DNS and make it foolproof and secure against the attacks and failures.

This paper is organized as follows: Section 2 introduces the Domain Name System. Section 3 lists out the vulnerabilities of this system while the following section addresses issues, which will make the system better adaptable to external attacks. Finally Section 4 puts everything together and provides the conclusive remarks with the future work.

2. Overview of Domain Name System[12]

DNS is a hierarchical, distributed database that provides mappings between human-readable domain names to machine-readable IP addresses [4] DNS forms an entity of WWW architecture that also includes the client using a browser and the server [13]

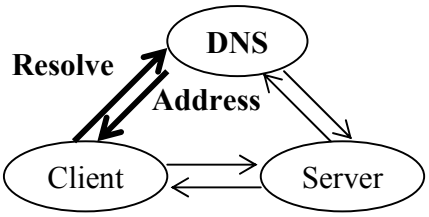

Fig 1. Entities of WWW Architecture

DNS has a tree structure with distribution of information location and administrative control. The database is spread over many name servers each handling a portion of the space tree. DNS **resolvers** query these servers for IP address corresponding to a host name. Each sub-tree in the DNS tree is a **domain** and each node of the tree has a label. DNS addresses can be relative or fully qualified. A fully qualified domain name of each node is formed from the labels and each ancestor all the way to the root. It is unique for each node. A relative address can be converted by appending the local domain information. For example, cse.uta.edu has a structure depicted in Fig 2.

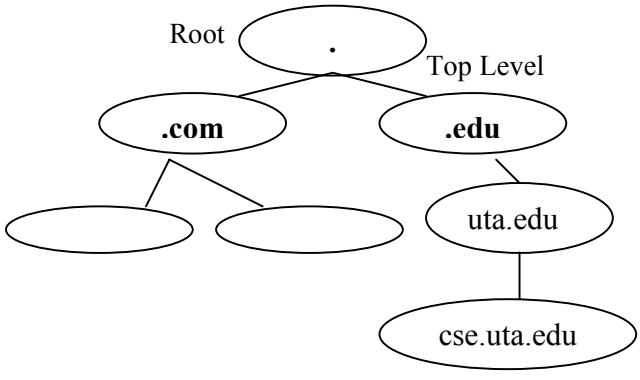

Fig 2. DNS Domains

The final most significant labels fall into three categories:

- arpa: Special facility used for reserve translation i.e going from IP address to fully qualified domain address.

- three letter codes: To identify the organization hosting the computer.
  com- commercial
  edu- educational
  gov- government
  int- international orgs
  mil- military
  net- network related
  org- misc organizations

- two letter codes: To indicate the country of origin.

Each domain is administered by some organization that is responsible for the correctness of data entering the name space. Organizations can **delegate** control of sub domains to any other organizations. A name space is divided into series of **zones,** which are that part of the domain that has not been delegated. These are based on syntactic separators (periods) in domain names. Each zone has two or more authoritative name servers (AS) that are responsible for keeping the information of the zone up-to-date. One of these is the primary name server, which has the master file, and when new hosts are added to the zone, this server must make the edited file public to other servers in the zone. The secondary servers periodically fetch the contents to keep themselves updated. DNS information is stored in each name server as a set of **Resource Records** (RR) that form the body of the reply to DNS query. Each record specifies type (A-address, HINFO- host configuration etc.), class, and value as attributes. Two RRs are assigned specific functions: Start of Authority (SOA) and Name Server (NS) that are used to indicate the delegation of a part of a domain of a name server. SOA contains the minimum expiration time for any records within the zone. The NS records define the AS for the domain.

Bits 0-15	**Bits 16-31**
Domain Name	
type	class
Time to live	
Resource Data length	Resource Data
Resource data	

Fig 3 Resource Record Format

The **domain name** is the query name from the query. The **type** is the query type. The **class** is 1 for the Internet domain. The **time to live** is the time for which the information can be cached by the client, typically two days, expressed in seconds. The **resource data length** specifies the number of bytes of resource data.

2.1 DNS Message Format

DNS message has a header and the following: question, answer, authority and additional. RR is the unit of information. The common RRs are:

- 'A' record, which contains a 32-bit IP address.
- CNAME, which maps an alias to a canonical domain name.
- HINFO, which contains host configuration information.
- MNS, which contains the host name which is an authoritative server (AS) for that domain.
- MX, which contains host name acting as a mail exchange.
- PTR, which contains domain name corresponding to the IP address.
- SOA, which contains domain information.

Question section carries:
QNAME- target domain name
QTYPE- query type
QCLASS- query class
Answer section carried RRs that answer the query.
Authority section carries RRs that describe the authoritative servers.
Additional section carries other RRs.

Bits 0-15	Bits 16-31
Identification	flags
number of questions	number of answer RRs
number of authority RRs	number of additional RRs
questions	
answers (RRs)	
authority (RRs)	
additional information	

Fig 4 Format of DNS query and response

The 16 flag bits give more information about the query:
QR – Query (0) and Response (1)
Opcode – 4 bits
0 – Standard query
1 – Inverse Query
2 - Server status request
AA– Indicates that the server is authoritative for the domain in question (1)
RD – recursive mode denied (0)
RA- server handles queries recursively (1)
Rcode- 4 bits that indicate the status:
0 – No error

1 – Malformed query
2 – Server failed
3 – Name does not exist
4 – Query type not supported
5 – Server refused to answer

2.2 DNS Functioning

The process of retrieving data from DNS is called **name resolution**. The resolver queries the local NS. The resolution can be iterative or recursive. In the **iterative mode**, the NS, which receives a query, upon not knowing how to resolve it, will forward it to other servers that are likely to know the answer. These servers will be initialized with some authoritative servers from the root zone. For example, when a root server receives an iterative query for domain name cse.uta.edu, it refers the querier/resolver to edu servers and finally the authoritative servers of cse.uta.edu are located and IP address calculated. In **recursive mode**, server finds out answer to query by contacting other servers itself and eventually returns the answer. To improve the speed and reduce the computation cost, each server maintains a cache to store previous results and a time to live tag will refresh the cache on a timely basis. As described above there is distribution of authority that makes the DNS scalable, robust and distributed.

3. Vulnerabilities[2]

One of the main goals of the design of DNS is to have distributed administration. This distribution is achieved by delegation of authority. By this we meet the goal of design, but at the same time make the system vulnerable to spoofing. There is no protection of authenticity or integrity of responses that are received from different servers. An external attacker could modify a response or provide false information thus manipulating the data. And since the authority is distributed, it becomes difficult to know whether to trust the responses received. This problem can be described as a failure to authenticate DNS responses. Apart from this, cache poisoning becomes another important problem that will make the system vulnerable to attacks.

In cache poisoning, the attacker can trick the name server say S1 to query another server S2. The attacker can have S2 send fake RRs or it can masquerade as S2 and send DNS response to S1. A name server caches the results of a previous query, and this if used by S1 will result in S1 using data contaminated by the attacker. There exists a message authentication system in DNS, but it is very weak: Every query is attached with an id that is matched with a corresponding id from the client. If the attacker predicts this id, it can send a forged response to S1 or disable communication between S1 and S2 by sending a denial message to collapse S2 [2].

The following sections explain some of the security extensions that are being applied to the DNS system. This section addresses the main problems of the system:

- Protection of name servers by defining a formal specification and a security goal and implementing a DNS wrapper
- Data authentication using digital signatures.
- Cache poisoning using DNS proxy.

3.1 DNS Wrapper[2]

Attacks on DNS will result in denial of service and entity authentication to fail. Therefore it becomes important to introduce security goals and formally characterize clients and name servers

in the DNS system. To enforce the security goal a **DNS Wrapper** is specified, which examines the incoming and outgoing DNS messages of a name server to detect messages that may cause violations of the goal. It works with the authoritative name servers to detect these messages and also drops the messages that are identified as threats.

3.1.1 DNS Prototype and Formal Specification

A security wrapper is a piece of software that encapsulates a component such as a name server to improve its security. Consider a wrapper w that checks DNS response packets going to a name server and ensure that it is authentic and agreeable to the authoritative servers. If this is not the case, then w locates and authoritative server and queries that server for authoritative answer. To locate the server for a zone z, w starts with server s, which is known to be an authoritative server for an ancestor zone, and queries as for authoritative servers of the child zone that may be z itself. The search is performed by traversal of the domain tree, zone by zone. Wrapper w has two main parts: query part and response part, which process both the queries generated by a name server and the responses destined to it. Only those acceptable by the wrapper are forwarded for further processing to the server. When the server needs to send a query, the wrapper generates a random query id and replaces the server's id with the random one. A mapping table is defined to map the two identification numbers.

3.1.2 Performance of DNS Wrapper

Several experiments are conducted to determine the response time, computational overhead etc of DNS wrapper. A prototype of DNS wrapper was implemented on BIND name server. The procedural steps involved in the experiment to check the response time are:

- Start a wrapped name server.
- Run nslookup (Appendix) to query this name server for resolving a specific number of DNS queries sequentially.
- Record the total CPU time used.
- Terminate the wrapped name server.
- Repeat the above procedure using an unmodified name server instead of a wrapped name server.

The mean response time for the wrapped name server was calculated to be 0.12 seconds per query and 0.08 for each unmodified query. The average CPU times used by wrapped and unmodified servers are 7% and 8% of the total response times. Thus the overhead of a wrapper was determined to be largely due to the waiting times for the response messages in the message diagnosis process.

Another experiment was conducted to detect the rate of malicious attacks of a wrapped name server (false negative rate). The procedure involved is:

- Start a malicious name server for a new sub domain. When queried, it will return incorrect resource records or queries with wrong ids.
- Start a wrapped name server.
- Run nslookup to query this name server for resolving a specific number of DNS queries sequentially.
- Terminate the wrapped name server.

- Terminate the malicious name server.
- Repeat the above procedure using an unmodified name server instead of a wrapped name server.

This experiment showed that a wrapped name server detected all the attacks and prevented any poisoning of DNS system. But the unmodified server allowed planting of incorrect data into the cache of target server, forwarded all incorrect records to the client thus allowing manipulation of data.

3.2 DNS Security Extensions (DNSSEC)[5]

Data Authentication is another very important issues that requires careful attention in order to ensure accuracy and integrity of data in the DNS system. The reader should note here that, confidentiality is not an issue as the information stored in the DNS database is public. If and when communication requirements call for private channels, the IP security system is (IPSEC) is selected, which can be interfaced with DNS. This issue is not addressed in the paper. Interested reader can refer to [9]. In this paper, DNS security extension (DNSSEC) mainly by using cryptology is explained. A new variant of the existing scheme of public key cryptology is introduced. This is based on symmetric cryptology techniques. This scheme achieves mutual authentication, a service that had not been implemented in any previous schemes.

3.2.1 Overview

DNSSEC uses digital signature schemes based on public-key cryptography, to achieve data authenticity and integrity. Each node in the DNS tree is associated with some public key. Each message from the DNS servers is signed under the corresponding private key. These keys are used to generate certificates or signatures that preserve the identity information of each top-level domain to the corresponding public key. Each parent signs the public keys of all its child nodes. To introduce this scheme into the system, new RRs are defined. KEY RR [1] is used to associate a public key to a domain name. There exists two typed of signatures for DNS messages: transaction signatures (TSIGs) and public key signatures (SIG (0)). TSIG is used between local servers, to secure updates or zone transfers between master and slave servers. Whereas SIG(0) is used mainly in small-scale authentication of requests. There exist a more robust and secure alternative, in which digital signatures sign each RRs. That is each RR is covered with a public-key signature, which is stored in a special RR called the SIG RR [7]. SIG RRs are computed for each RR set in a zone and this value is added to each RR set in answers to DNS queries. This is a method by which the resolver verifies the authenticity of the server. Another signed RR called the NXT RR (next) [7] is added, which indicates the next domain name. If the resolver queries for a name or data that currently does not exist in a zone, a NXT RR is returned with the corresponding SIG.

3.2.2 Symmetric Key Technique (SK-SNNSEC)

This novel concept of DNS symmetric certificates, uses the symmetric cryptographic techniques, and binds the owner's identity to a secret key. This achieves an efficient chain of trust for a DNS root to the authoritative server. In the domain tree, each node shares a master key with its parent. The root in addition to having its own master key also has a pair of public and secret keys. The master key of the root node is used to initiate the chain of trust to various authoritative servers. The protocol works as described below:

Suppose a local name server LS queries the root server for an IP address. The root is not authoritative for this query and thus will forward the reslover LS to its child server. The root server generates a secret key Ka, which is encrypted and sent to the LS along with a symmetric certificate. LS and the corresponding server will share this Ka. LS queries the server by sending the request again along with the certificate. The server will extract Ka form this and encrypts a new key Kb. To communicate this key to the next level, the server uses another certificate and inserts this key within it. This key will now be shared between the child server and LS. If the IP address is obtained then it is sent to LS symmetrically signed with Kb. Symmetric certificates can be cached and used by LS for similar future requests.

In this method the master keys are used to create the certificate, which allow safe transfer of keys between levels. For this scheme to be initiated apart from the roots master key, it is required for the resolver to have an authentic copy of the roots public key. When the resolver contacts the root for the first time, it sends a DNS_RootCert_Req request encrypted with this public key. The resolver also includes within this, two secret keys K1 and K2 and a header (identity, lifetime of encryption, random number, and timestamp). The root server will create a certificate for itself, which will be signed by K1 and K2 and then sent to the resolver. This establishes an authenticated channel between the querier and the root server. Public-key cryptology is used only the first time the resolver communicates with the root server. After this, the communication will be secure with symmetric-key protocols.

Note that this system considers only iterative queries. The reslover may send a recursive query to the server, which then is forced to interact with other name servers in order to find answers. But due to the large overhead associated with it, many root and top-level servers are configured not to accept recursive queries. Another issue to be noted is that a queried name server may return several authoritative servers for the zone. Resolvers therefore use a roundtrip time (RTT) as a parameter to choose among the different servers. RTT measures the time necessary for a name server to respond to queries, and outputs the closest server to the reslover. Since the names servers for a zone store same secret keys, it is required to generate just once certificate for the server, that being the closest.

3.2.3 Performance

Symmetric-key techniques can be compared to the existing public-key scheme to show that it has a better performance:

- SK_DNSSEC uses very short certificates. With the same amount of cache it is possible to store more data which: manages the memory efficiently and also reduces the delay performance and also the number of messages in the network. The zone data file thus becomes more manageable and smaller.
- SK-DNSSEC does not allow reuse of certificates. This provides better protection against replay attacks.
- This scheme adopts a concept of mutual authentication that is when a DNS server receives a request it determines whether the request is from an authorized server. Mutual authentication is necessary to prevent IP spoofing attacks. Some servers maintain an access control list which enumerates the IP addresses of the resolvers that are allowed to query that server.
- SK_DNSSEC provides confidentiality when required in order to manage a large private domain spaces, by including the DNS_Req and DNS_Ans directly into the symmetric encryption.
- SK_DNSSEC employs very small signatures and only one signature is needed to be sent for each query. A secret key is approximately 128 bitts long. Since DNS runs over UDP

with datagram size of 512 bytes, we can easily conclude that both the signatures for answers and referrals will fit into the datagram.

- This scheme provided very strong security protection and overall data integrity. Mutual authentication is supported at low costs.
- This approach is scalable. the DNS tree structure proves an important feature in this aspect in that, each level in the tree maintain relevant information which is exchanged or shared with lower levels. Initially only the root public key is stored and successively it will be the root certificate. The servers need to store any information about the resolvers since the parents pass on this information. In case of a loss, information can be retrieved from DNS server upstream in the network.

3.3 DNS Proxy[6]

To counter cache-poisoning problem that has been explained earlier, many schemes have been proposed. A credibility level scheme in which resource records from a more credible source takes precedence over those from a less credible one, was implemented. A more efficient design is one that introduces a concept of DNS proxy. In this design, a domain name space is portioned into regions called realms, which is served by a set of servers. Depending on the query of a DNS request, the proxy forwards the query to the responsible servers. By this way, certain records that do not refer to the realm to which the query name belongs, and those that satisfy a set of filtering rules are removed, thus protecting the query.

4 Merits and Shortcomings of the Domain Name System design

As discussed thus far in this paper, it has become clear that DNS, though a convenient system for the users of the Internet, may not be the most efficient and secure systems. Nevertheless, DNS has some merits that have made it clearly the most popular and most widely used naming systems. This section puts it all together, explains the current technologies and compares features of DNS with other systems.

Considering the amount of naming data and the scale of networks involved, DNS Internet implementation achieves relatively short average response time for lookups. The objects named are primarily computers, mail hosts and domains. The algorithms for portioning, replicating and caching naming data are used to achieve better response time. Since computer-IP address mappings and mail hosts identification change relatively infrequently, caching and replication prove lesser overheads in terms of updating data. Another issue to be mentioned is that, apart from computers DNS also names one other service: name service. DNS assumes that there exists only one name service per domain; therefore the user does not need to explicitly mention the service. Electronic mail services automatically select this service by using the appropriate type of query when contacting DNS servers. This facility is extensible to other services that have only one implementation per domain, but is not applicable to services such as file services that have multiple instances in a domain.

Also, DNS allows naming to be inconsistent, that is a client may receive stale data, but this will be of no consequence till the client actually uses this stale data. The DNS does not address this issue on how to detect staleness of addresses need to be detected. Another aspect of inconsistency that is bound to occur in DNS is because it has a restrictive, centralized model for entering names into a naming database be adding them to a local file. System administrators at different locations will manipulate data, which may not be updated at all locations leading to inconsistency.

In summary, DNS stores a limited variety of naming data, but this is sufficient for common applications such as e-mail. It can be argued that DNS database represents the lowest common

denominator of what would be considered useful by the many user communities on the Internet. But what still remains a challenge is to make DNS rigid with respect to changes in the structure of the name space; make the DNS secure against all threats and attacks, solve the inconsistency issues that exist in DNS.

4.1 Other improvements to DNS

Security extensions and enhancements to the DNS gives a view of just one of the many dimensions of the DNS improvement requirement. Other improvements can also be introduced to the existing system to improve it further.

- Replication Architecture for the DNS: This design takes advantage of the recent advances in disk storage and multicast distribution technology. Each geographically distributed server contains a complete and an up-to-date copy of the entire DNS database. These servers are called the replicated servers. To keep the records up-to-date, new resource records are distributed over satellite channels or terrestrial multicast. The design allows Web sites to dynamically wander and replicate themselves without having to change their URLs. The overall improvement is markedly visible in Web surfing time since the DNS lookup is significantly reduced [14].
- DNS dynamic updates: This interesting new feature of state-of-art authorization can be combined with existing schemes to create a more flexible and scalable authentication approach [4]. This scheme can be used to update DNS records of hosts with dynamic IP addresses. Especially mobile hosts that roam from one network segment to another would benefit from keeping the same domain name even if their IP addresses change as they move. This feature would include updating certificates stored in DNS. This can be done by adding new certificates or by updating the existing ones. In the second case, the life times of the certificates need to be reduced to reduce the risk related to revocation. This approach is also considered a replacement to the traditional manual update of zone files. With dynamic update the changes can be done at a remote site and the client performs sanity checks on the data.

4.2 BIND System

The Berkeley Internet Domain System (BIND) is an implementation of the DNS running on BSD UNIX. Client programs link in library software as resolver. DNS named server computers run the named daemon. BIND allows for three categories of name server: primary, secondary, and cache-only. The named server implements just one of these types according to contents of the file. The cache-only server reads in from the configuration file sufficient names and addresses of authoritative servers to resolve any name. There after they only store this data. This type of server reduces network traffic and speeds up response time.

5 Conclusions

This paper addresses the importance of naming in the computer science world and focuses on the Domain Name System and its Security aspects. In a distributed environment such as the Internet, naming is the only way to identify objects, or to specify a particular action to be performed. DNS is one naming system that is adopted for the Internet. It is a distributed database with a hierarchical tree structure. It is globally available that is any machine that connects to the Internet can access the DNS. Its hierarchical structure makes it scalable. DNS builds a sort of a logical structure for names that are globally unique. These are some of the plus points of this system. Other issues to be dealt with are: mapping between different levels, updating values at

different servers, improving performance delays and most importantly security and authentication.

The hierarchical structure and the global standards used by the Domain Name System ensure efficient mapping between the parent and the child nodes in he tree. Caching is the most optimal strategy used to improve the delay performance. This paper pays attention to the last category of problems: Security and Authentication. Since the DNS was invented when the threat attacks to the Internet were not well understood, the system now is vulnerable to different kinds of attacks. Spoofing, cache poisoning, data manipulation are some of the issues addressed in this paper.

A formal approach is introduced to prevent spoofing: DNS Wrapper. This piece of software acts as a checkpoint for all the queries and responses that traverse in the system. It has functionality to do a validation check on the resolvers which send the query and also verifies the authority of a name server. Cache poisoning is attacked by DNS proxies. Data integrity and authentication is the most important issue. It is believed that authorization problems subsume authentication issues. In the case of DNS, which has a natural authorization structure (hierarchical structure), authentication schemes like cryptography can be used. A new scheme Symmetric Key technique (SK- DNSSEC) is described, which has better performance that the existing public-key schemes. The performance of both DNS wrapper and the SK-DNSSEC security extensions are mentioned.

Finally, Domain Name Service is a scalable name service provided for the Internet. It co-exists with other services like the Global Name Service (GNS) and X.500, which have different features than its own. As the future work it is necessary to extend the security issues of DNS to a further extent and make it robust. The above-mentioned schemes will have to collaborate to achieve more efficient performance improvement. Also authorization of dynamic updates and verifying access rights, say for mobile environments, needs to be researched upon. The d

References

[1] Sandra Murphy, Olafur Gudmundsson. "*Retrofitting Security into Internet Infrastructure Protocols"*.

[2] Steven Cheung, Karl N. Levitt. "*A Formal Specification Based Approach for Protecting the Domain Name System"*.

[3] Steven M. Bellovin. "*Using the Domain Name System for System Break-ins"*. In Proceedings of the 5th USENIX UNIX Security Symposium.

[4] Pasi Eronen, Jonna Sars. "*Applying Decentralized Trust Management to DNS Dynamic Updates"*.

[5] Giuseppe Ateniese, Stefan Mangard. "*A New Approach to DNS Security (DNSSEC)"*.

[6] B. Cheswick, and S. Bellovin. "*A DNS filter and switch for packet filtering Gateways"*. Proceedings of the 6th UNIX Security Symposium. 1996, pp. 15-19.

[7] D. Eastlake, and C. Kaufman. " *Domain Name System Security Extensions*". RFC 2065.

[8] http://www.stopspam.org/usenet/mmf/man/nslookup.html

[9] Niels Furguson, and Bruce Schneier. "*A Cryptographic evaluation of IP sec"*. Counterpane Internet Security, Inc.

[10] Hugo Krawczyk. " *The order of encryption and authentication for protecting communications"*.

[11] James M. Galvin. *" Public Key Distribution with Secure DNS"*. In 6th USENIX UNIX Security Symposium. 1996.

[12] http://www.scit.wlv.ac.uk/~jphb/commos/dns.html

[13] Roberto Baldoni, Simona Bonamoneta, Carlo Marchetti. "*Implementing Highly-Available WWW Servers based on Passive Object Replication"*.

[14] Jussi Kangasharju, Keith W. Ross. *"A Replicated Architecture for the Domain Name System"*.

[15] http://www.internic.net

[16] Donald E. Eastlake. *" Storing certificates in Domain Name Service*". RFC 2538, IETF, 1999.

Appendix

Nslookup is a UNIX shell command, which is a program to query Internet domain name servers. Nslookup has two modes:

- Interactive
- Non-interactive

Interactive mode allows the user to query name servers for information about various hosts and domains or to print a list of hosts in a domain. Non-interactive mode is used to print just the name and requested information for a host or domain. Iterative mode is used when no arguments is provided. In that case a default name server will be used. Another case in which this mode operates is when the first argument is a hyphen (-) and the second name is the host name.

Non-Interactive mode is used when the name or the IP address of a host to be queried is given as the first argument. The second argument is the host name or the address of a name server.