GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

DNS Security Using Cryptography

A Report for the Evaluation 3 of Project 2

Submitted by

SAMAY KUMAR SINGH

(1613114039)

In partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION OF COMPUTER NETWORK AND CYBER SECURITY

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

Prof. Ms. VAISHALI GUPTA, M.Tech

April/May-2020

GALGOTIAS
UNIVERSITY

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **"DNS Security Using Cryptography"** is the bonafide work of "Samay Kumar Singh" who carried out the project work under my supervision.

SIGNATURE **SIGNATURE**

HEAD OF THE DEPARTMENT **SUPERVISOR**

CONTENTS

CHAPTER 1

INTRODUCTION

Humans can’t think like computers. They just can’t remember dozens of IP addresses. They need easy-to-remember names to locate their mail server or their favorite web pages. To make our lives on the Internet easy, DNS was therefore invented. And with it came a new place for hackers of all sorts to have fun. Moreover, the purpose of DNS makes it a very sensitive area; for this is the place the client connection is orientated. The possibilities a black-hat can have by succeeding in hacking DNS are tremendous (a user can be directed to a host controlled by a hacker, whatever service he might be using: http, ftp, telnet ...).

Anything is possible!

The Domain Name System distributes the responsibility of assigning domain names and mapping those names to IP addresses by designatingauthoritative name servers for each domain. Authoritative name servers are assigned to be responsible for their supported domains, and may delegate authority over sub-domains to other name servers. This mechanism provides distributed and fault tolerant service and was designed to avoid the need for a single central database.

The Domain Name System also specifies the technical functionality of the database service which is at its core. It defines the DNS protocol, a detailed specification of the data structures and data communication exchanges used in DNS, as part of the Internet Protocol Suite. Historically, other directory services preceding DNS were not scalable to large or global directories as they were originally based on text files, prominently the HOSTS.TXT resolver. DNS has been in wide use since the 1980s.

The DNS plays a critical role in supporting the Internet infrastructure by providing a distributed and fairly robust mechanism that resolves Internet host names into IP addresses and IP addresses back into host names. The DNS also supports other Internet directory-like lookup capabilities to retrieve information pertaining to DNS Name Servers, Canonical Names, Mail Exchangers, etc. Unfortunately many security weaknesses surround IP and the protocols carried by IP. The DNS is not immune to these security weaknesses. The accuracy of the information contained within the DNS is vital to many aspects of IP based communications.

The threats that surround the DNS are due in part to the lack of authenticity and integrity checking of the data held within the DNS and in part to other protocols that use host names as an access control mechanism. In response to this, the IETF formed a working group to add DNS Security (DNSSEC) extensions to the existing DNS protocol.

The Domain Name System (DNS) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates domain names, which can be easily memorized by humans, to the numerical IP addresses needed for the purpose of computer services and devices worldwide. The Domain Name System is an essential component of the functionality of most Internet services because it is the Internet's primary directory service.

The DNS plays a critical role in supporting the Internet infrastructure by providing a distributed and fairly robust mechanism that resolves Internet host names into IP addresses and IP addresses back into host names. The DNS also supports other Internet directory-like lookup capabilities to retrieve information pertaining to DNS Name Servers, Canonical Names, Mail Exchangers, etc. Unfortunately many security weaknesses surround IP and the protocols carried by IP. The DNS is not immune to these security weaknesses. The accuracy of the information contained within the DNS is vital to many aspects of IP based communications.

The Domain Name System distributes the responsibility of assigning domain names and mapping those names to IP addresses by designatingauthoritative name servers for each domain. Authoritative name servers are assigned to be responsible for their supported domains, and may delegate authority over sub-domains to other name servers. This mechanism provides distributed and fault tolerant service and was designed to avoid the need for a single central database.

The Domain Name System also specifies the technical functionality of the database service which is at its core. It defines the DNS protocol, a detailed specification of the data structures and data communication exchanges used in DNS,as part of the Internet Protocol Suite. Historically, other directory services preceding DNS were not scalable to large or global directories as they were originally based on text files, prominently the HOSTS.TXT resolver. DNS has been in wide use since the 1980s.

The DNS plays a critical role in supporting the Internet infrastructure by providing a distributed and fairly robust mechanism that resolves Internet host names into IP addresses and IP addresses back into host names. The DNS also supports other Internet directory-like lookup capabilities to retrieve information pertaining to DNS Name Servers, Canonical Names, Mail Exchangers, etc. Unfortunately many security weaknesses surround IP and the protocols carried by IP. The DNS is not immune to these security weaknesses. The accuracy of the information contained within the DNS is vital to many aspects of IP based communications.

The Internet maintains two principal namespaces, the domain name hierarchy and the Internet Protocol (IP) address spaces. The Domain Name System maintains the domain name hierarchy and provides translation services between it and the address spaces. Internet name servers and a communication protocol implement the Domain Name System. A DNS name server is a server that stores the DNS records for a domain name; a DNS name server responds with answers to queries against its database.

The most common types of records stored in the DNS database are for DNS zone authority (SOA), IP addresses (A and AAAA), SMTP mail exchangers (MX), name servers (NS), pointers for reverse DNS lookups (PTR), and domain name aliases (CNAME). Although not intended to be a general purpose database, DNS can store records for other types of data for either automatic machine lookups, such as DNSSEC records, or for human queries such as responsible person (RP) records. As a general purpose database, DNS has also seen use in combating unsolicited email (spam)

by using a real-time black hole list stored in the DNS. Whether for Internet naming or for general purpose uses, the DNS database is traditionally stored in a structured zone file.

1.1 PURPOSE

Enhance Open Source DNS server ―bind‖ in order to support conditional parsing algorithms for automated detection and prevention against both, the attacks on DNS and where DNS is exploited as attack vector.

Develop a Industry Standard but Open Source Web Interface for easy administration, management and reporting of DNS Server. It should facilitate with granular DNS configuration, rule definition (as part of above), custom reporting and data export in different formats such as CSV and PDF.

1.2 BACKGROUND

2.1 Threats to the Domain Name System

The original DNS specifications did not include security based on the fact that the information that it contains, namely host names and IP addresses, is used as a means of communicating data [SPAF]. As more and more IP based applications developed, the trend for using IP addresses and host names as a basis for allowing or disallowing access (i.e., system based authentication) grew. Unix saw the advent of Berkeley "r" commands (e.g., rlogin, rsh, etc.) and their dependencies on host names for authentication. Then many other protocols evolved with similar dependencies, such as Network File System (NFS), X windows, Hypertext Transfer Protocol (HTTP), et al.

Another contributing factor to the vulnerabilities in the DNS is that the DNS is designed to be a public database in which the concept of restricting access to information within the DNS name space is purposely not part of the protocol. Later versions of the BIND implementation allow access controls for such things as zone transfers, but all in all, the concept of restricting who can query the DNS for RRs is considered outside the scope of the protocol.

2.2 NX Domain Attacks

NX DOMAIN is nothing but non-existent Internet or Intranet domain name.As more and more IP based applications developed, the trend for using IP addresses and host names as a basis for allowing or disallowing access (i.e., system based authentication) grew. Unix saw the advent of Berkeley "r" commands (e.g., rlogin, rsh, etc.) and their dependencies on host names for authentication. Then many other protocols evolved with similar dependencies, such as Network File System (NFS), X windows, Hypertext Transfer Protocol (HTTP), et al.

If domain name is unable to resolved using the DNS, a condition called the NXDOMAIN occurred. In this example, try to find out an IPk, address for the domain called abcquq12examfooltest.com using the nslookup or host command line option:

```
Nslookup sabcquq12examfooltest.com
```

OR

```
host abcquq12examfooltest.com
```

Sample outputs:

```
Host abcquq12examfooltest.com not found: 3(NX DOMAIN)
```

Since domain name is the invalid domain, you got a NX DOMAIN response i.e. an error message indicating that domain is either not registered or invalid.

2.3 Random Sub Domain Attacks

The diagram illustrates how these new attacks use open DNS proxies to send tens of millions of queries. Attack related queries use randomized labels prepended to target domains.

Because names with randomized subdomains are never in-cache, resolution requires more computationally expensive recursion, which stresses provider resolvers. Authoritative servers can also fail or respond very slowly under the load. When this happens resolvers incur even more work and stress as they navigate around unresponsive name servers. Authoritative servers using Response Rate Limiting (RRL) and sending truncated responses back to resolvers aggravates the DNS even more since both resolvers and authoritative servers have to manage retries with additional TCP overhead.

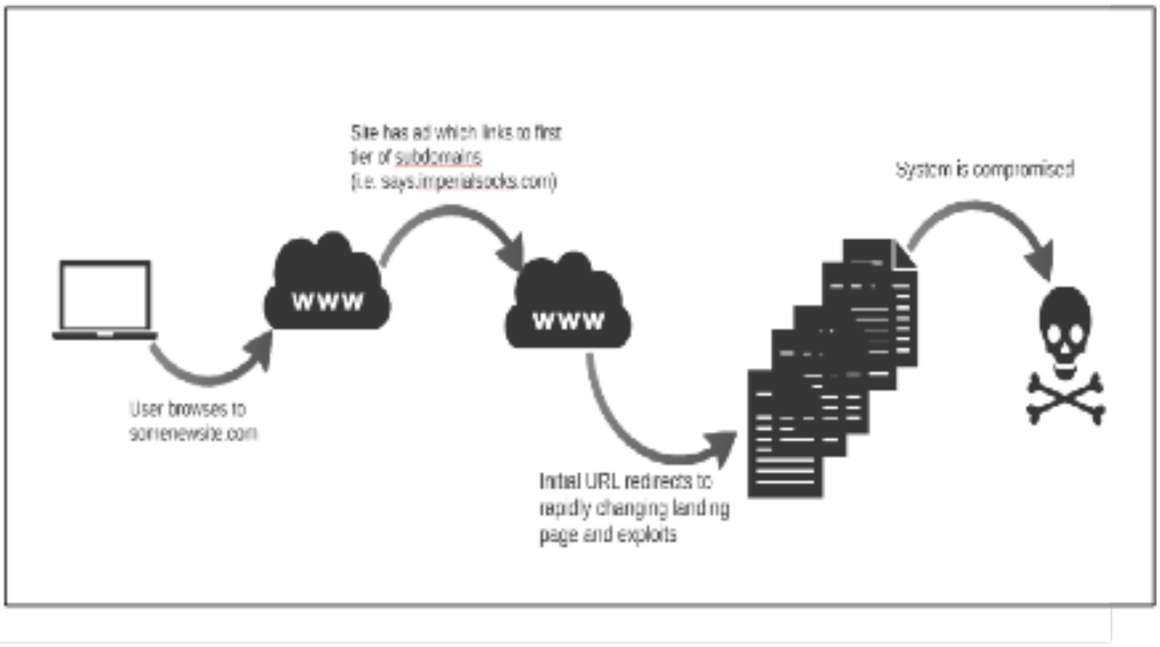

Site has ad which links to first tier of subdomains (i.e. says.imperialsocks.com)
System is compromised
www
www
User browses to somenewsite.com
Initial URL redirects to rapidly changing landing page and exploits

Figure 1.1: Random Sub-Domain Attack [5]

2.4 Cache Poisoning

Whenever a DNS server does not have the answer to a query within its cache, the DNS server can pass the query onto another DNS server on behalf of the client. If the server passes the query onto another DNS server that has incorrect information, whether placed there intentionally or unintentionally, then cache poising can occur [CA97]. Malicious cache poisoning is commonly referred to as DNS spoofing.

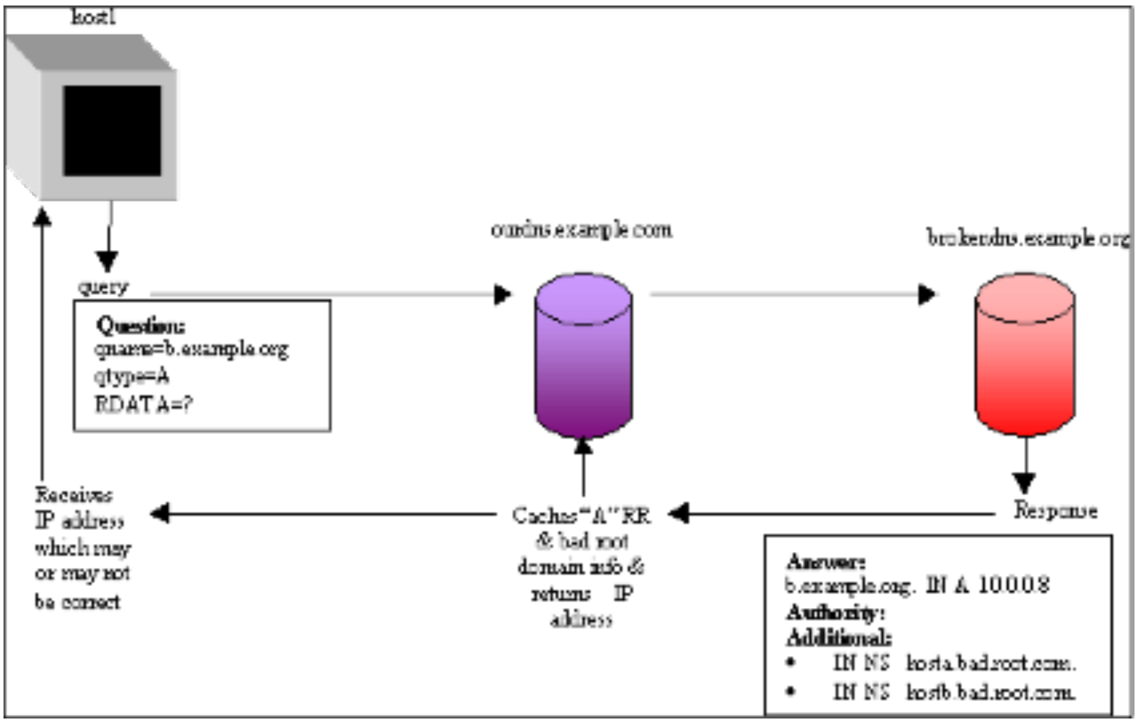

host1
ourdns.example.com
brokendns.example.org
query
Question:
qname=b.example.org
qtype=A
RDATA=?
Receives IP address which may or may not be correct
Caches "A" RR & bad root domain info & returns IP address
Response
Answer:
b.example.org. IN A 10.0.0.8
Authority:
Additional:
• IN NS hosta.bad.root.com.
• IN NS hostb.bad.root.com.

Figure 1.2: DNS Cache Poisoning [6]

As well as deploying name servers in secure configurations, the solution to this problem is a protocol known as DNSSEC, which is being rolled out across registries and registrars worldwide today. Once DNSSEC adoption becomes universal, adding a DNSSEC digital signature to a domain name will mean that browsers and ISPs will be able to validate that DNS information they receive is authentic, rendering most cache poisoning attacks obsolete. Organizations concerned about the integrity of their domain names should ask their registrars to support DNSSEC today.

2.5 Registrar Hijacking

The majority of domain names are registered via a registrar company, and these represent single points of failure. If an attacker can compromise your account with your chosen registrar, they gain control over your domain name, allowing them to point it to the servers of their choice, including name servers, Web servers, email servers, etc. Worse still, the domain could be transferred to a new owner or to an ―offshore‖ registrar, making domain name recovery a complex matter.

Such attacks may be directed at the registrar in a blanket fashion, as was the case in the recent attack against UK registrar NetNames, which claimed several high-profile victims. Others may target your account specifically, either through an attack on your password or, more commonly, a social engineering attack against the registrar's technical support operatives.

To reduce the risk of hijacking, choose a registrar that offers additional security precautions, such as multi-factor authentication or account managers with whom you can build a personal relationship. Many registrars will offer premium services to high- value clients that can substantially mitigate the risk of losing control of your account to a hijacker. These come at a cost, but it's a small price to pay to ensure that your domain name remains in your control.

2.6 DNS Amplification Attacks

DNS amplification is a tactic used in DDoS attacks that leverages DNS servers deployed in insecure ―recursive‖ configurations. Recursion is a feature of DNS that allows for domain name resolution to be handed off to more robust name servers. In itself, it's a useful, necessary feature commonly deployed within an enterprise environment.

But criminals discovered several years ago that "open" recursive DNS servers, i.e., a recursive name server for which access is neither controlled nor restricted, could be exploited to increase the strength of their DDoS attacks.

By spoofing the source address on DNS queries to match that of the intended victim, attackers found that every spurious packet sent from one of their bots could be amplified if sent to a recursive name server. The response sent to the victim would be many dozens of times larger than the original query. This could result in a botnet wielding many times the firepower, causing much more severely degraded performance at its victim's site. Today, running a recursive DNS server that is open to the entire Internet is no longer considered acceptable security practice. Fortunately, securing your DNS servers against this kind of attack is usually achieved with a simple configuration change.DNS that allows for domain name resolution to be handed off to more robust name servers. In itself, it's a useful, necessary feature commonly deployed within an enterprise environment. But criminals discovered several years ago that "open" recursive DNS servers, i.e., a recursive name server for which access is neither controlled nor restricted, could be exploited to increase the strength of their DDoS attacks.

3 .ATTACK OBJECTIVES

An attacker makes use of cache poisoning for one of two reasons. One is a denial of service (DoS) and the other is masquerading as a trusted entity.

3.1 Denial of Service

DoS is accomplished in several ways. One takes advantage of negative responses (i.e., responses that indicate the DNS name in the query cannot be resolved). By sending back the negative response for a DNS name that could otherwise be resolved, results in a DoS for the client wishing to communicate in some manner with the DNS name in the query. The other way DoS is accomplished is for the rogue server to send a response that redirects the client to a different system that does not contain the service the client desires. Another DoS associated with cache poisoning involves inserting a CNAME record into a cache that refers to itself as the canonical name.

```
foobar.example.org. IN CNAME foobar.example.org.
```

In this example, a recursive name server may end up with this RR in its cache. This type of CNAME record is commonly referred to as a self-referential RR. An attacker, after inserting this resource record into a server's cache can cause the name server to crash by simply requesting a zone transfer for foobar.example.org

3.2 Masquerading

The second and potentially more damaging reason to poison DNS caches is to redirect communications to masquerade as a trusted entity. If this is accomplished, an attacker can intercept, analyze, and/or intentionally corrupt the communications [CA97]. The misdirection of traffic between two communicating systems facilitates attacks such as industrial espionage and can be carried out virtually undetected [MENM]. An attacker can give the injected cache a short time to live making it appear and disappear quickly enough to avoid detection.

Masquerading attacks are possible simply due to the fact that quite a number of IP based applications use host names and/or IP addresses as a mechanism of providing host-based authentication. This burdens the DNS with the responsibility of maintaining up to date and accurate information, neither of which the DNS alone can assure. If this is accomplished, an attacker can intercept, analyze, and/or intentionally corrupt the communications [CA97]. The misdirection of traffic between two communicating systems facilitates attacks such as industrial espionage and can be carried out virtually undetected [MENM]. An attacker can make use of these shortcomings within the DNS to masquerade as a trusted host. Host based Authentication is vulnerable to host name spoofing.

In this example, an attacker takes advantage of the rshd program's dependency on the contents of the ".rhosts" file as a form of host based authentication. The attacker's DNS server, evildns.example.org, is authoritative for 0.6.172.in-addr.arpa and the attacker has the following entry in the zone's authoritative data, even though the attacker does not have authority over plain.org:

The host, trustme.plain.org, is trusted by victim.example.edu simply because a student has trustme.plain.org correctly listed in the student's .rhosts file on victim.example.edu. For the purpose of this example, the host, victim.example.edu, is not protected by a 2firewalls and does not employ any type of DNS sanity checking such as the PARANOID mode in tcp_wrappers [VENE]. The stage is set where the attacker can now come from the IP address of 172.16.0.8 and log into victim.example.edu as the student without a password and appear as if the connection actually came from the trusted host name.

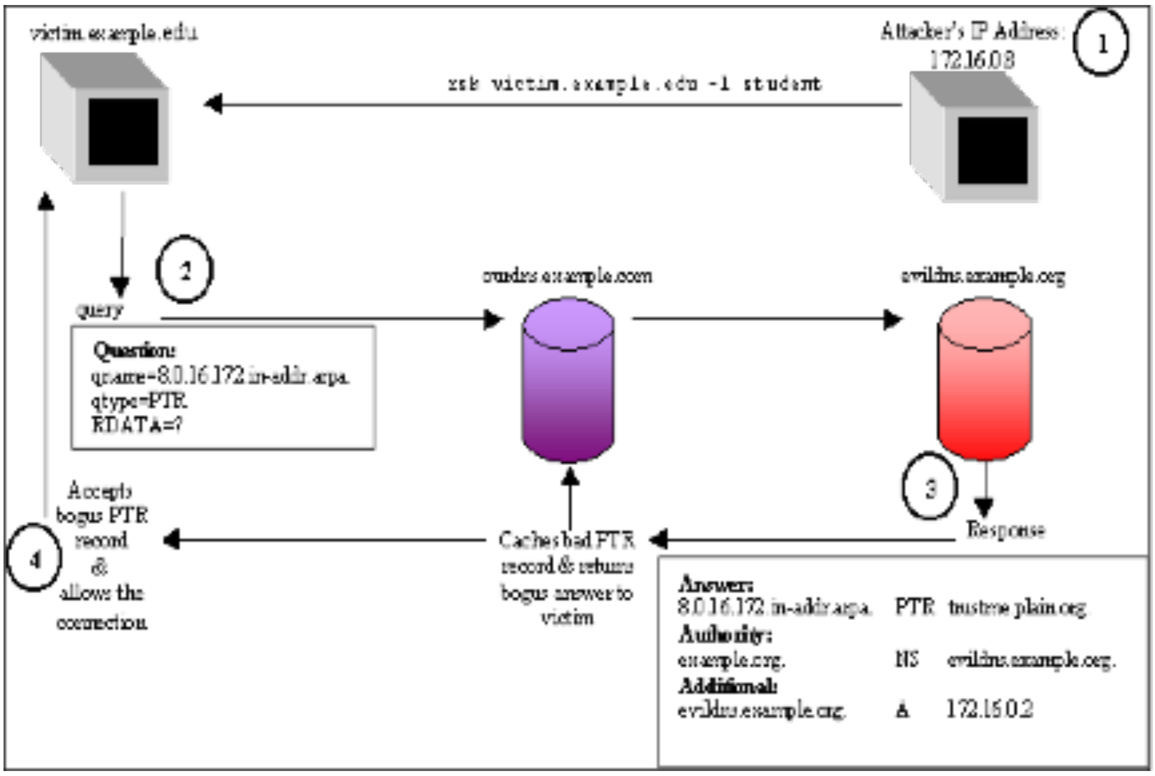

victim.example.edu
Attacker's IP Address: 172.16.0.8
1
rsh victim.example.edu -l student
2
query
Question:
qname=8.0.16.172.in-addr.arpa.
qtype=PTR
RDATA=?
ourdns.example.com
evildns.example.org
3
Response
Answers:
8.0.16.172.in-addr.arpa. PTR trustme.plain.org
Authority:
example.org. NS evildns.example.org.
Additional:
evildns.example.org. A 172.16.0.2
Caches bad PTR record & returns bogus answer to victim
4
Accepts bogus PTR record & allows the connection

Figure 1.3: Host Name Spoofing [3]

4 SCOPE OF PROJECT

The BIND 4 and BIND 8 releases both had serious security vulnerabilities. Their use is strongly discouraged. BIND 9 was a complete rewrite, in part to mitigate these ongoing security issues.

Security issues that are discovered in BIND 9 are patched and publicly disclosed in keeping with common principles of open source software. A complete list of security defects that have been discovered and disclosed in BIND9 is maintained by Internet Systems Consortium, the current authors of the software.

BIND is transparent open source. If your organization needs some functionality that is not in BIND, you can modify it, and contribute the new feature back to the the community by sending us your source. BIND has evolved to be a very flexible, full- featured DNS system.Whatever your application is, BIND most likely has the features required.

5 OVERVIEW OF THE DNS

To connect to a system that supports IP, the host initiating the connection must know in advance the IP address of the remote system. An IP address is a 32-bit number that represents the location of the system on a network. The 32-bit address is separated into four octets and each octet is typically represented by a decimal number. The four decimal numbers are separated from each other by a dot character ("."). Even though four decimal numbers may be easier to remember than thirty-two 1’s and 0’s, as with phone numbers, there is a practical limit as to how many IP addresses a person can remember without the need for some sort of directory assistance. The directory essentially assigns host names to IP addresses.

The Stanford Research Institute’s Network Information Center (SRI-NIC) became the responsible authority for maintaining unique host names for the Internet. The SRI- NIC maintained a single file, called hosts.txt, and sites would continuously update SRI-NIC with their host name to IP address mappings to add to, delete from, or change in the file. The problem was that as the Internet grew rapidly, so did the file causing it to become increasingly difficult to manage. Moreover, the host names needed to be unique throughout the worldwide Internet. With the growing size of the Internet it became more and more impractical to guarantee the uniqueness of a host name. The need for such things as a hierarchical naming structure and distributed management of host names paved the way for the creation of a new networking protocol that was flexible enough for use on a global scale.

What evolved from this is an Internet distributed database that maps the names of computer systems to their respective numerical IP network address(es). This Internet lookup facility is the DNS. Important to the concept of the distributed database is delegation of authority. No longer is one single organization responsible for host name to IP address mappings, but rather those sites that are responsible for maintaining host names for their organization(s) can now regain that control.

An IP address is a 32-bit number that represents the location of the system on a network. The 32-bit address is separated into four octets and each octet is typically represented by a decimal number. The four decimal numbers are separated from each other by a dot character ("."). Even though four decimal numbers may be easier to remember than thirty-two 1’s and 0’s, as with phone numbers, there is a practical limit as to how many IP addresses a person can remember without the need for some sort of directory assistance. The directory essentially assigns host names to IP addresses.

5.1 Fundamentals of DNS

The DNS not only supports host name to network address resolution, known as forward resolution, but it also supports network address to host name resolution, known as inverse resolution.

Due to its ability to map human memorable system names into computer network numerical addresses, its distributed nature, and its robustness, the DNS has evolved into a critical component of the Internet. Without it, the only way to reach other computers on the Internet is to use the numerical network address. Using IP addresses to connect to remote computer systems is not a very user-friendly representation of a system's location on the Internet and thus the DNS is heavily relied upon to retrieve an IP address by just referencing a computer system's Fully Qualified Domain Name (FQDN). A FQDN is basically a DNS host name and it represents where to resolve this host name within the DNS hierarchy.

5.2 The Domain Name Space

The DNS is a hierarchical tree structure whose root node is known as the root domain. A label in a DNS name directly corresponds with a node in the DNS tree structure. A label is an alphanumeric string that uniquely identifies that node from its brothers. Labels are connected together with a dot notation, ".", and a DNS name containing multiple labels represents its path along the tree to the root. Labels are written from left to right. Only one zero length label is allowed and is reserved for the root of the tree. This is commonly referred to as the root zone. Due to the root label being zero length, all FQDNs end in a dot.

As a tree is traversed in an ascending manner (i.e., from the leaf nodes to the root), the nodes become increasingly less specific (i.e., the leftmost label is most specific and the right most label is least specific). Typically in an FQDN, the left most label is the host name, while the next label to the right is the local domain to which the host belongs. The local domain can be a subdomain of another domain. The name of the parent domain is then the next label to the right of the subdomain (i.e., local domain) name label, and so on, till the root of the tree is reached.

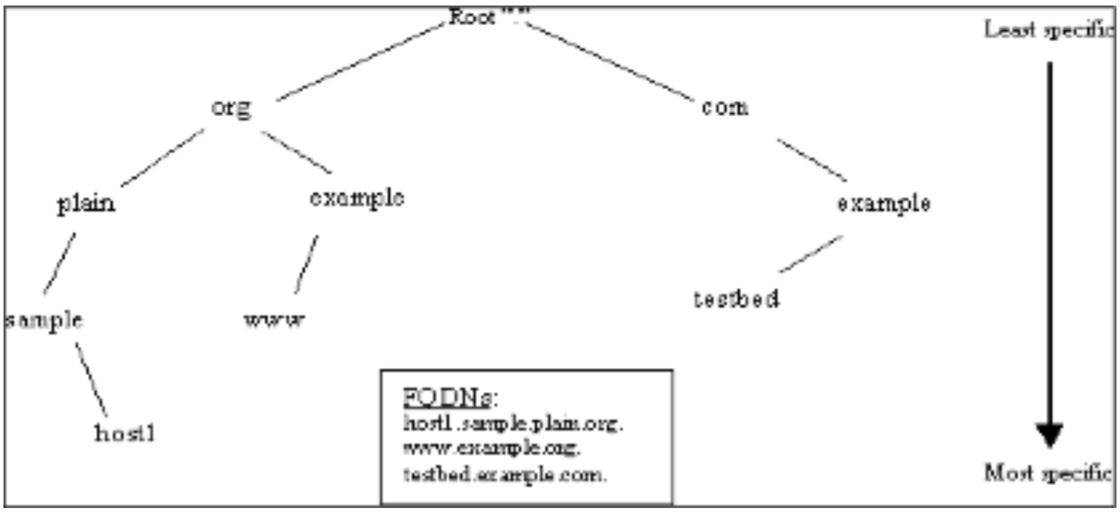
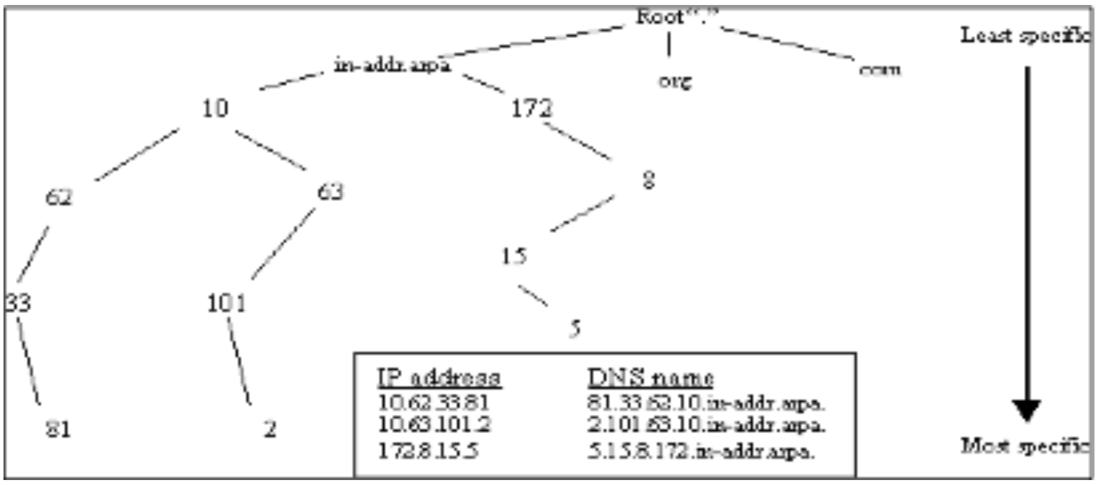


Figure 1.4. Domain Name Space example [7]

When the DNS is used to map an IP address back into a host name (i.e., inverse resolution), the DNS makes use of the same notion of labels from left to right (i.e., most specific to least specific) when writing the IP address. This is in contrast to the typical representation of an IP address whose dotted decimal notation from left to right is least specific to most specific. A label is an alphanumeric string that uniquely identifies that node from its brothers. Labels are connected together with a dot notation, ".", and a DNS name containing multiple labels represents its path along the tree to the root. Labels are written from left to right. Only one zero length label is allowed and is reserved for the root of the tree. This is commonly referred to as the root zone. Due to the root label being zero length, all FQDNs end in a dot.

Typically in an FQDN, the left most label is the host name, while the next label to the right is the local domain to which the host belongs. The local domain can be a subdomain of another domain. The name of the parent domain is then the next label to the right of the subdomain (i.e., local domain) name label, and so on, till the root of the tree is reached. To handle this, IP addresses in the DNS are typically represented in reverse order. IP addresses fall under a special DNS top level domain (TLD), known as the in-addr.arpa domain. By doing this, using IP addresses to find DNS host names are handled just like DNS host name lookups to find IP addresses.



IP address	DNS name
10.62.33.81	81.33.62.10.in-addr.arpa.
10.63.101.2	2.101.63.10.in-addr.arpa.
172.8.15.5	5.15.8.172.in-addr.arpa.

Figure 1.5. Example of Inverse Domains and the Domain Name Space [7]

5.3 DNS Components

The DNS has three major components, the database, the server, and the client. The database is a distributed database and is comprised of the Domain Name Space, which is essentially the DNS tree, and the Resource Records (RRs) that define the domain names within the Domain Name Space. The server is commonly referred to as a name server. Name servers are typically responsible for managing some portion of the Domain Name Space and for assisting clients in finding information within the DNS tree. Name servers are authoritative for the domains in which they are responsible. They can also serve as a delegation point to identify other name servers that have authority over subdomains within a given domain.

The RR data found on the name server that makes up a domain is commonly referred to as zone information. Thus, name servers have zones of authority. A single zone can either be a forward zone (i.e., zone information that pertains to a given domain) or an inverse zone (i.e., zone information that maps IP addresses into DNS host names). DNS allows more than one name server per zone, but only one name server can be the primary server for the zone. Primary servers are where the actual changes to the data for a zone take place. All the other name servers for a zone basically maintain copies of the primary server's database for the zone. These servers are commonly referred to as secondary servers.

A DNS RR has 6 fields: NAME, TYPE, CLASS, TTL, RD Length, and RDATA. The NAME field holds the DNS name, also referred to as the owner name, to which the RR belongs.The database is a distributed database and is comprised of the Domain Name Space, which is essentially the DNS tree, and the Resource Records (RRs) that define the domain names within the Domain Name Space. The server is commonly referred to as a name server. Name servers are typically responsible for managing some portion of the Domain Name Space and for assisting clients in finding information within the DNS tree. Name servers are authoritative for the domains in which they are responsible. They can also serve as a delegation point to identify other name servers that have authority over subdomains within a given domain. The TYPE field is the TYPE of RR. This field is necessary because it is not uncommon for a DNS name to have more than one type of RR.

The RR data found on the name server that makes up a domain is commonly referred to as zone information. Thus, name servers have zones of authority. A single zone can either be a forward zone (i.e., zone information that pertains to a given domain) or an inverse zone (i.e., zone information that maps IP addresses into DNS host names). DNS allows more than one name server per zone, but only one name server can be the primary server for the zone. Primary servers are where the actual changes to the data for a zone take place. All the other name servers for a zone basically maintain copies of the primary server's database for the zone. These servers are commonly referred to as secondary servers. The more common types of RR are found in Table.

Table 1.1. A DNS RR [7]

RECORD TYPE	**DESCRIPTION**	**USAGE**
A	An address record	Maps FQDN into an IP address
PTR	A pointer record	Maps an IP address into FQDN
NS	A name server record	Denotes a name server for a zone
SOA	A Start of Authority record	Specifies many attributes concerning the zone, such as the name of the domain (forward or inverse), administrative contact, the serial number of the zone, refresh interval, retry interval, etc.
CNAME	A canonical name record	Defines an alias name and maps it to the absolute (canonical) name
MX	A Mail Exchanger record	Used to redirect email for a given domain or host to another host

5.4 DNS Transactions

DNS transactions occur continuously across the Internet. The two most common transactions are DNS zone transfers and DNS queries/responses. A DNS zone transfer occurs when the secondary server updates its copy of a zone for which it is authoritative. The secondary server makes use of information it has on the zone, namely the serial number, and checks to see if the primary server has a more recent version. If it does, the secondary server retrieves a new copy of the zone.

A DNS query is answered by a DNS response. Resolvers use a finite list of name servers, usually not more than three, to determine where to send queries. If the first name server in the list is available to answer the query, than the others in the list are never consulted. If it is unavailable, each name server in the list is consulted until one is found that can return an answer to the query. The name server that receives a query from a client can act on behalf of the client to resolve the query. Then the name server can query other name servers one at a time, with each server consulted being presumably closer to the answer. The name server that has the answer sends a response back to the original name server, which then can cache the response and send the answer back to the client. Once an answer is cached, a DNS server can use the cached information when responding to subsequent queries for the same DNS information. Caching makes the DNS more efficient, especially when under heavy load. This efficiency gain has its tradeoffs; the most notable is in security.

The DNS has a defined message protocol for queries and responses. A DNS message has five sections, a Header section, a Question section, an Answer section, an Authority section and an Additional section. The header section contains information such as the type of message and what other sections are present in the message. The last three sections are filled with RRs when appropriate. The Answer section contains RRs specifically pertaining to the answer. The Authority section is filled with either SOA or NS records belonging to the zone of authority for the owner name of the RR(s) in the Answer section. The Additional section may potentially have additional information that the receiver may find of interest.

Table 1.2. DNS Message Format [5]

HEADER	QUESTION	ANSWER	AUTHORITY	ADDITIONAL

5.5 The BIND Implementation of DNS

BIND is the most widely used Domain Name System (DNS) software on the Internet. On Unix-like operating systems it is the de facto standard.

The software was originally designed at the University of California Berkeley (UCB) in the early 1980s. The name originates as an acronym of Berkeley Internet Name Domain, reflecting the application's use within UCB. The software consists, most prominently, of the DNS server component, called named, a contracted form of name daemon. In addition the suite contains various administration tools, and a DNS resolver interface library. The latest version of BIND is BIND 9, first released in 2000.

Starting in 2009, the Internet Software Consortium (ISC) developed a new software suite, initially called BIND10. Wit h release version 1.2.0 the project was renamed Bundy to terminate ISC involvement in the project.

The Berkeley Internet Name Daemon (BIND) is the most popular implementation of the DNS on the Internet. The BIND distribution of the DNS has client software, server software, and software tools for querying the DNS and troubleshooting problems. Most of the information in this paper concerning actual DNS implementation has to do with BIND.

CHAPTER 2

CRYPTOGRAPY

Cryptography is the practice and study of techniques for secure communication in the presence of third parties called adversaries. More generally, cryptography is about constructing and analyzing protocols that prevent third parties or the public from reading private .

Cryptography prior to the modern age was effectively synonymous with encryption, the conversion of information from a readable state to apparent nonsense. The originator of an encrypted message (Alice) shared the decoding technique needed to recover the original information only with intended recipients (Bob), thereby precluding unwanted persons (Eve) from doing the same. The cryptography literature often uses Alice ("A") for the sender, Bob ("B") for the intended recipient, and Eve ("eavesdropper") for the adversary. Since the development of rotor cipher machines in World War I and the advent of computers in World War II, the methods used to carry out cryptology have become increasingly complex and its application more widespread.

1. MODERN CRYPTOGRAPHY

The modern field of cryptography can be divided into several areas of study. The chief ones are discussed here:

1.1 Symmetric-Key Cryptography

Symmetric-key cryptography refers to encryption methods in which both the sender and receiver share the same key (or, less commonly, in which their keys are different, but related in an easily computable way). This was the only kind of encryption publicly known until June 1976.

Symmetric key ciphers are implemented as either block ciphers or stream ciphers. A block cipher enciphers input in blocks of plaintext as opposed to individual characters, the input form used by a stream cipher.

Stream ciphers, in contrast to the 'block' type, create an arbitrarily long stream of key material, which is combined with the plaintext bit-by-bit or character-by character, somewhat like the one-time pad.

In a stream cipher, the output stream is created based on a hidden internal state that changes as the cipher operates. That internal state is initially set up using the secret key material. RC4 is a widely used stream cipher; see Category: Stream cipher. Block ciphers can be used as stream ciphers; see Block ciphers modes of operation.

1.2 Public-Key Cryptography

Symmetric-key cryptosystems use the same key for encryption and decryption of a message, though a message or group of messages may have a different key than others. A significant disadvantage of symmetric ciphers is thekey management necessary to use them securely. Each distinct pair of communicating parties must, ideally, share a different key, and perhaps each cipher text exchanged as well. The number of keys required increases as the square of the number of network members, which very quickly requires complex key management schemes to keep them all consistent and secret. The difficulty of securely establishing a secret key between two communicating parties, when a secure channel does not already exist between them, also presents a chicken-and-egg problem which is a considerable practical obstacle for cryptography users in the real world.

Public-key cryptography can also be used for implementing digital signature schemes. A digital signature is reminiscent of an ordinary signature; they both have the characteristic of being easy for a user to produce, but difficult for anyone else to forge. Digital signatures can also be permanently tied to the content of the message being signed; they cannot then be 'moved' from one document to another, for any attempt will be detectable. In digital signature schemes, there are two algorithms: one for signing, in which a secret key is used to process the message (or a hash of the message, or both), and one for verification, in which the matching public key is used with the message to check the validity of the signature. RSA and DSA are two of the most popular digital signature schemes. Digital signatures are central to the operation of public key infrastructures and many network security schemes (e.g., SSL/TLS, many VPNs, etc.).

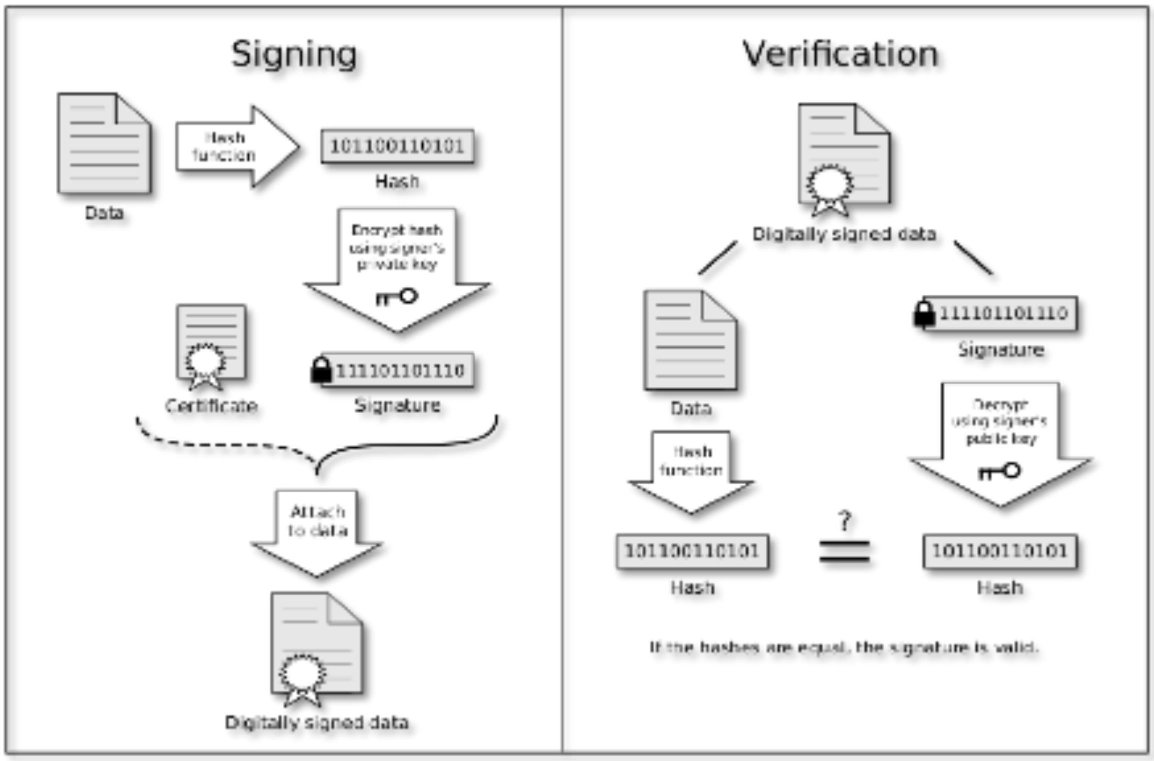


Figure 2.1 Signing and Verification [1]

2. SECURITY INFORMATION FOR DNS

Domain Name System (DNS) was originally designed as an open protocol. Therefore, it is vulnerable to attackers. Windows Server 2008 DNS helps improve your ability to prevent an attack on your DNS infrastructure through the addition of security features. Before considering which of the security features to use, you should be aware of the common threats to DNS security and the level of DNS security in your organization.

2.1 DNS Security Threats

The following are the typical ways in which your DNS infrastructure can be threatened by attackers:

- Foot printing: The process by which DNS zone data is obtained by an attacker to provide the attacker with the DNS domain names, computer names, and IP addresses for sensitive network resources. An attacker commonly begins an attack by using this DNS data to diagram, or "footprint," a network. DNS domain and computer names usually indicate the function or location of a domain or computer to help users remember and identify domains and computers more easily. An attacker takes advantage of the same DNS principle to learn the function or location of domains and computers in the network.

- Denial-of-service attack: An attempt by an attacker to deny the availability of network services by flooding one or more DNS servers in the network with recursive queries. As a DNS server is flooded with queries, its CPU usage eventually reaches its maximum and the DNS Server service becomes unavailable. Without a fully operating DNS server on the network, network services that use DNS become unavailable to network users.

- Data modification: An attempt by an attacker (that has foot printed a network using DNS) to use valid IP addresses in IP packets the attacker has created, which gives these packets the appearance of coming from a valid IP address in the network. This is commonly called IP spoofing. With a valid IP address (an IP address within the IP address range of a subnet), the attacker can gain access to the network and destroy data or conduct other attacks.

- Redirection: An attacker redirecting queries for DNS names to servers under the control of the attacker. One method of redirection involves the attempt to pollute the DNS cache of a DNS server with erroneous DNS data that may direct future queries to servers under the control of the attacker. For example, if a query is originally made for widgets.tailspintoys.com and a referral answer provides a record for a name outside the tailspintoys.com domain, such as malicious- user.com, the DNS server uses the cached data for malicious-user.com to resolve a query for that name. Attackers can accomplish redirection whenever they have writable access to DNS data, for example, when dynamic updates are not secure.

2.2 Three Levels Of DNS Security

The following sections describe the three levels of DNS security:

2.2.1 Low-Level Security

Low-level security is a standard DNS deployment without any security precautions configured. Deploy this level of DNS security only in network environments where there is no concern for the integrity of your DNS data or in a private network where there is no threat of external connectivity. Low-level DNS security has the following characteristics:

- The DNS infrastructure of the organization is fully exposed to the Internet.

- Standard DNS resolution is performed by all DNS servers in the network.

- All DNS servers are configured with root hints pointing to the root servers for the Internet.

- All DNS servers permit zone transfers to any server.

- All DNS servers are configured to listen on all of their IP addresses.

- Cache pollution prevention is disabled on all DNS servers.

- Dynamic update is allowed for all DNS zones.

- User Datagram Protocol (UDP) and TCP/IP port 53 is open on the firewall in the network for both source and destination addresses.

2.2.2 Medium-Level Security

Medium-level security uses the DNS security features that are available without running DNS servers on domain controllers and storing DNS zones in Active Directory Domain Services (AD DS). Medium-level DNS security has the following characteristics:

- The DNS infrastructure of the organization has limited exposure to the Internet.

- All DNS servers are configured to use forwarders to point to a specific list of internal DNS servers when they cannot resolve names locally.
- All DNS servers limit zone transfers to servers that are listed in the name server (NS) resource records in their zones.
- DNS servers are configured to listen on specified IP addresses.

- Cache pollution prevention is enabled on all DNS servers.

- Non-secure dynamic update is not allowed for any DNS zones.

- Internal DNS servers communicate with external DNS servers through the firewall with a limited list of allowed source and destination addresses.

- External DNS servers in front of the firewall are configured with root hints that point to the root servers for the Internet.

- All Internet name resolution is performed using proxy servers and gateways.

2.2.3 High-Level Security

High-level security uses the same configuration as medium-level security. It also uses the security features that are available when the DNS Server service is running on a domain controller and DNS zones are stored in AD DS. In addition, high-level security completely eliminates DNS communication with the Internet. This is not a typical configuration, but it is recommended whenever Internet connectivity is not required. High-level DNS security has the following characteristics:

- The DNS infrastructure of the organization has no Internet communication by internal DNS servers.

- The network uses an internal DNS root and namespace, where all authority for DNS zones is internal.

- DNS servers that are configured with forwarders use internal DNS server IP addresses only.

- All DNS servers limit zone transfers to specified IP addresses.

- DNS servers are configured to listen on specified IP addresses.

- Cache pollution prevention is enabled on all DNS servers.

- Internal DNS servers are configured with root hints pointing to the internal DNS servers that host the root zone for the internal namespace.

- All DNS servers are running on domain controllers. A discretionary access control list (DACL) is configured on the DNS Server service to allow only specific individuals to perform administrative tasks on the DNS server.

- All DNS zones are stored in AD DS. A DACL is configured to allow only specific individuals to create, delete, or modify DNS zones.
- DACLs are configured on DNS resource records to allow only specific individuals to create, delete, or modify DNS data.
- Secure dynamic update is configured for DNS zones, except the top-level and root zones, which do not allow dynamic updates at all.

Often the Real Vulnerability, when it comes to DNS Security and Stability, is Ignorance.

Here are five DNS Threats you should protect against.
The Domain Name System (DNS) is pervasive. Collectively, we use it billions of times a day, often without even knowing that it exists. For enterprises, it's their digital identity as well as a critical component of their security architecture. Like all technology, though, it is susceptible to threats. Too often, the always-on, ubiquitous nature of DNS lends itself to being overlooked. Today, let’s look at five common threats that leverage DNS, along with suggested best-practice, risk-mitigation strategies.

2.2.4 Typo Squatting

The practice of registering a domain name that is confusingly similar to an existing popular brand – typosquatting -- is often considered a problem for trademark attorneys. However, as recent research has demonstrated, it can present a profound risk to the confidentiality of corporate secrets and should be increasingly thought of as a security problem. Typosquatting is not only about individuals opportunistically registering confusingly similar domains in the hope of benefiting from misdirected Web traffic; it can also be used to steal information.

In early September, researchers from the Godai Group said that they had successfully obtained 120,000 corporate emails by simply typosquatting certain domains and setting up catch-all email accounts. Godai registered domains following the format ―usexample.com‖ to steal mail destined for ―user@us.example.com‖. If an email was incorrectly addressed, missing the dot between ―@us‖ and ―example‖, it would arrive in the researchers' account instead. The research discovered that attackers could steal passwords, sales information and other trade secrets, and hypothesized that a more sophisticated attack could obtain information from both the email sender and recipient.

Remember to monitor newly registered domain names for names that are confusingly similar to your brand. Information about new domain registrations is often freely available from registries, and there are many companies that offer dedicated digital brand management services to simplify this searching process.

2.2.5 DDoS

Distributed denial of service attacks (DDoS) are not a threat specific to DNS.
However, the DNS is particularly vulnerable to such attacks because it represents a logical choke point on the network, all too often overlooked when organizations are capacity-planning their infrastructure. No matter how over-provisioned a website may be, if the DNS infrastructure cannot handle the number of incoming requests it receives, the performance of the site will be degraded or disabled.

2.2.6 Cache Poisoning

Whenever you send an email or visit a website, your computer is probably using DNS data that has been cached somewhere on the network, such as with your ISP. This improves the performance of the Internet, and reduces the load on the various registries that provide authoritative DNS responses. However, these caches can sometimes be vulnerable to "poisoning" attacks. Attackers sometimes exploit vulnerabilities or poor configuration choices in DNS servers -- or in cases such as the infamous Kaminsky Bug, vulnerabilities in the DNS protocol itself -- to inject fraudulent addressing information into caches. Users accessing the cache to visit the targeted site would find themselves instead at a server controlled by the attacker. If the attacker's site were a close replica of the target's official site, there would be no way for the user to tell that they were being phished. As far as their browser would know, it would be at the official site. As well as deploying name servers in secure configurations, the solution to this problem is a protocol known as DNSSEC, which is being rolled out across registries and registrars worldwide today.

3. CONFIGURE A DNS SERVER FOR USE WITH ACTIVE DIRECTORY DOMAIN SERVICES

When you install Active Directory Domain Services (AD DS) with the Active Directory Domain Services Installation Wizard, the wizard gives you the option to automatically install and configure a DNS server. The resulting DNS zone is integrated with the AD DS domain that is controlled by the AD DS server.

To install AD DS on this computer, use Server Manager.

- This method applies only to server computers that are used as domain controllers. If member servers (server that are not used as domain controllers) are used as DNS servers, they are not integrated with AD DS.

- If you choose the wizard option to automatically install and configure a local DNS server, the DNS server is installed on the computer where you are running the wizard and the computer's preferred DNS server setting is configured to use the new local DNS server. Configure any other computers that will join this domain to use this DNS server's IP address as their preferred DNS server.

4. RSA ALGORITHM

RSA is one of the first practical public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and differs from the decryption key which is kept secret. In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the factoring problem. RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1977.

RSA is a relatively slow algorithm, and because of this it is less commonly used to directly encrypt user data. More often, RSA passes encrypted shared keys for symmetric key cryptography which in turn can perform bulk encryption- decryption operations at much higher speed.

4.1 RSA Steps

Step 1 : Select two prime nos. – p& q such as p!=q
Step 2 : Calculate n as product of p & q, i.e. n=pq
Step 3 : Calculate m as product of (p-1) & (q-1) i.e. m = (p-1)(q-1)
Step 4 : Select any integer e<m such that it is co-prime to m, co-prime means
gcd(e,m)=1
Step 5 : Calculate d such that de mod m = 1 , i.e. d = e^-1 mod m Step 6: The public key is {e, n).The private key is {d,n}

So these are the keys, now if you want to preform some encryption operation using these keys here are the steps, if you have a text P, its encrypted version(cipher text C is)

C = P^e mod n
To decrypt it back to plain text use P = C^d mod n

4.2 The RSA Algorithm Involves Four Steps

Key generation, Key distribution, Encryption and Decryption.

RSA involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. The intention is that messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key.

The basic principle behind RSA is the observation that it is practical to find three very large positive integers e,d and n such that with modular exponentiation for all m: and that even knowing e and n or even m it can be extremely difficult to find d. Additionally, for some operations it is convenient that the order of the two exponentiations can be changed and that this relation also implies:

4.2.1 Key Distribution

To enable Bob to send his encrypted messages, Alice transmits her public key (n, e) to Bob via a reliable, but not necessarily secret route. The private key is never distributed.

4.2.2 Encryption

Suppose that Bob would like to send message M to Alice.

He first turns M into an integer m, such that $0 \leq m < n$ and $\gcd(m, n) = 1$ by using an agreed-upon reversible protocol known as a padding scheme. He then computes the cipher text c, using Alice's public key e, corresponding to

This can be done efficiently, even for 500-bit numbers, using modular exponentiation. Bob then transmits c to Alice.

4.2.3 Decryption

Alice can recover m from c by using her private key exponent d by computing Given m, she can recover the original message M by reversing the padding scheme.

4.2.4 Key Generation

The keys for the RSA algorithm are generated the following way:

1. Choose two distinct prime numbers p and q.
2. Compute $n = pq$.

For security purposes, the integers p and q should be chosen at random, and should be similar in magnitude but 'differ in length by a few digits'[2] to make factoring harder. Prime integers can be efficiently found using a primarily test.

n is used as the modulus for both the public and private keys. Its length, usually
expressed in bits, is the key length.

3. Compute$\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1) = n-(p+q-1)$, where φ isEuler's quotient function. This value is kept private.

3. Choose an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$; i.e., e and $\varphi(n)$ are co-prime.

4. Determine d as $d \equiv e^{-1} \pmod{\varphi(n)}$; i.e., d is the modular multiplicative inverse of e (modulo $\varphi(n)$)

- This is more clearly stated as: solve for d given $d \cdot e \equiv 1 \pmod{\varphi(n)}$

- e having a short bit-length and small Hamming weight results in more efficient encryption – most commonly $2^{16} + 1 = 65{,}537$. However, much smaller values
  of e (such as 3) have been shown to be less secure in some settings.

- e is released as the public key exponent.

- d is kept as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and $\varphi(n)$ must also be kept secret because they can be used to calculate d.

Example:

Here is an example of RSA encryption and decryption. The parameters used here are artificially small, but one can also use OpenSSL to generate and examine a real key pair.

1. Choose two distinct prime numbers, such as p, q

2. Compute $n = pq$

3. Compute the quotient of the product as $\varphi(n) = (p - 1)(q - 1)$ giving

4. Choose any number $1 < e < 3120$ that is co-primes to 3120. Choosing a prime number for e leaves us only to check that e is not a divisor of 3120.

5. Compute d, the modular multiplicative inverse of e (mod $\varphi(n)$) yielding, Worked example for the modular multiplicative inverse:

The public key is (n = 3233, e = 17). For a padded plaintext message m, the encryption function is

The private key is (d = 2753). For an encrypted cipher text c, the decryption function is SHA-1 hash value is typically rendered as a hexadecimal number, 40 digits long.

5. SHA ALGORITHM

In cryptography, SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function designed by the United States National Security Agency and is a U.S. Federal Information Processing Standard published by the United States NIST. SHA-1 produce a 160-bit (20-byte) hash value known as a message digest. A SHA-1 hash value is typically rendered as a hexadecimal number, 40 digits long.

SHA-1 is no longer considered secure against well-funded opponents. In 2005, cryptanalysts found attacks on SHA-1 suggesting that the algorithm might not be secure enough for ongoing use,and since 2010 many organizations have recommended its replacement by SHA-2 or SHA 3 .Microsoft, Google and Mozilla have all announced that their respective browsers will stop accepting SHA-

1 SSL certificates by 2017.

The Secure Hash Algorithm is a family of cryptographic hash functions published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS), including:

- SHA-0: A retronym applied to the original version of the 160-bit hash function published in 1993 under the name "SHA". It was withdrawn shortly after publication due to an undisclosed "significant flaw" and replaced by the slightly revised version SHA-1.
- SHA-1: A 160-bit hash function which resembles the earlier MD5 algorithm. This was designed by the National Security Agency (NSA) to be part of the Digital Signature Algorithm. Cryptographic weaknesses were discovered in SHA-1, and the standard was no longer approved for most cryptographic uses after 2010.

- SHA-2: A family of two similar hash functions, with different block sizes, known as SHA-256 and SHA-512. They differ in the word size; SHA-256 uses 32-bit words where SHA-512 uses 64-bit words. There are also truncated versions of each standard, known as SHA-224, SHA-384, SHA-512/224 and SHA-512/256.

These were also designed by the NSA.

- SHA-3: A hash function formerly called Keccak, chosen in 2012 after a public competition among non-NSA designers.

CHAPTER 3

LITERATURE SURVEY

- In the paper ―Client-Side Pharming attacks Detection using Authoritative DNS‖ authors demonstrate about the detections of Pharming attack and proposed an approach to protect user at client-side from Pharming attacks. Pharming attacks can be performed at the client-side or into the internet. In pharming attack, attackers need not targeting individual user. If pharming is performed by modifying the DNS entries, than it will be affecting to all users who is accessing the web page through that DNS. We propose an approach to protect user at client-side from pharming attacks by comparing IP addresses, using information provided by local DNS server and a list of IP's provided by the domain's Authenticated Name Servers which are the most trusted DNS servers for a domain. It was mainly done by comparing IP addresses, using information provided by local DNS server and a list of IP's provided by the domain's Authenticated Name Services which are the most trusted DNS serve [2].

- Also in the paper ―Detection and Prevention Algorithms of DDOS Attack in MANETs‖ authors demonstrate about the detections and Prevention of DDoS attack. We introduce Bottom-up approach, New Cracking algorithm, Prevention algorithm using IDS node for detecting and controlling DDoS attack.Security is a weak link of network systems. The malicious usage and attacks have caused tremendous loss by impairing the functionalities of the computer networks.. In an attempt to enhance security in MANETs many researchers have suggested and implemented new improvements to the protocols and some of them have suggested new protocols. Existing MANET routing protocols, such as Ad Hoc On-Demand Distance Vector Routing Protocol (AODV), do not provide enough security defense capacity. Distributed Denial of Service (DDoS) attack has become a major problem to networks. In this paper, we introduce Bottom-up approach, New Cracking algorithm, Prevention algorithm using IDS node for detecting and controlling DDoS attack [4].

- In the paper ―Addressing Complexity in DNS Security: A Case for Improved Security Status Indication based on a Trust Model‖ Increasingly complex factors for DNS name resolution mean that users are unable to make informed decisions of the risks they face on the Internet. We conclude that there is no simple means of assessing the trust users might place in DNS responses, and that there is presently no effective way of interactively representing this in the browser UI. In this paper we propose further work to develop a trust model for DNS name resolution, taking into account the many complex scenarios users encounter. Building on such a trust model, a new means of representing security risk to users in a web browser should be developed. Without a simple representation to convey the status of the various complex factors discussed here, it is impractical for user to make informed security decisions when using the web. Another future step is to address the special needs that cloud computing will have in terms of DNS. This could include the management of inter-cloud resources to explore additional features of past works regarding to effective scheduling or messaging [5].

CHAPTER 4

PROPOSED WORKFLOW

To Protect DNS Using Cryptography

1. Generation of public and private key using RSA Algorithm for security :

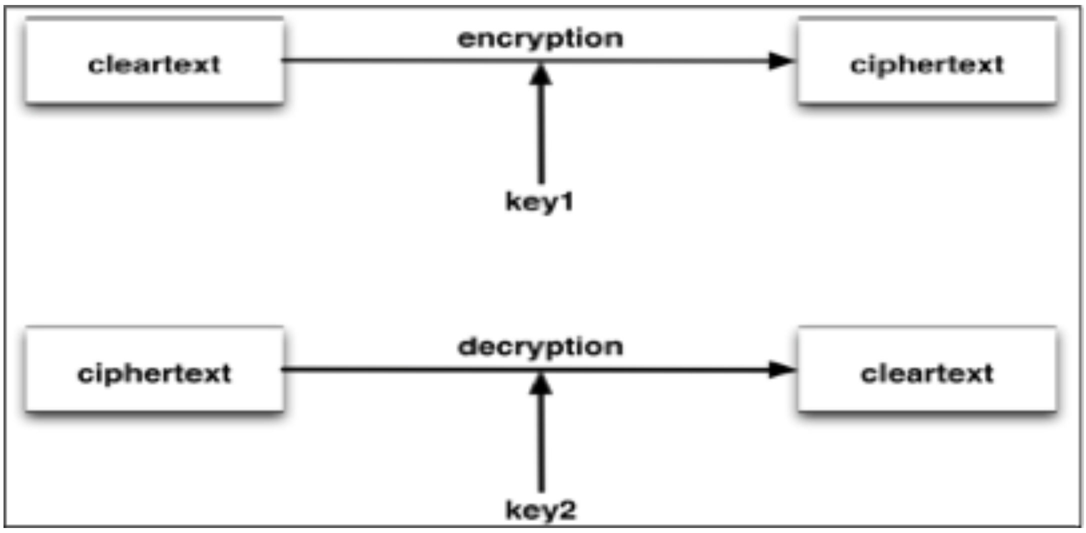


Figure 4.1 Encryption and Decryption [6]

RSA involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. The intention is that messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key.

The basic principle behind RSA is the observation that it is practical to find three very large positive integers e,d and n such that with modular exponentiation for all m: and that even knowing e and n or even m it can be extremely difficult to find d.

Step 1 : Select two prime nos. – p & q such as p!=q

Step 2 : Calculate n as product of p & q, i.e. n=pq

Step 3 : Calculate m as product of (p-1) & (q-1) i.e. m = (p-1)(q-1)

Step 4 : Select any integer e<m such that it is co-prime to m, co-prime means gcd(e,m)=1

Step 5 : Calculate d such that de mod m = 1 , i.e. d = e^-1 mod m Step 6: The public key is {e, n).The private key is {d,n}

So these are the keys, now if you want to perform some encryption operation using these keys here are the steps, if you have a text P..its encrypted version(cipher text C is)

C = P^e mod n
To decrypt it back to plain text use

P = C^d mod n

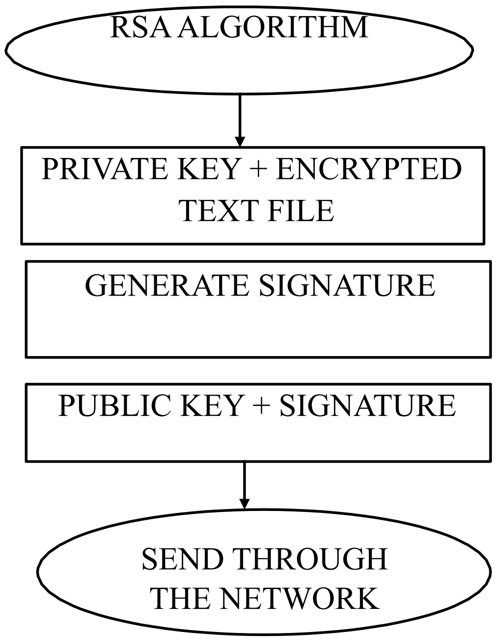


Figure 4.2. RSA Signature Generation [1]

2. Conditional processing based on limits and blacklisting / whitelisting :
A firewall is a network security system, either hardware- or software-based, that controls incoming and outgoing network traffic based on a set of rules. Acting as a barrier between a trusted network and other untrusted networks -- such as the Internet -- or less-trusted networks -- such as a retail merchant's network outside of a cardholder data environment -- a firewall controls access to the resources of a network through a positive control model. This means that the only traffic allowed onto the network defined in the firewall policy is; all other traffic is denied.

3. To Protect Network from following:

- Typo Squatting: In typo squatting, also called URL hijacking, is a form of cybersquatting which relies on mistakes such as typographical errors made by Internet users when inputting a website address into a web browser. Server will redirect to actual site.

- Session Hijacking: In session hijacking or cookie hijacking is the exploitation of a valid computer session to gain unauthorized access to information or services in a computer system. Server will not allow to transfer session.

- DDoS / DoS attacks: In this when attacking using unlimited ping, the server will avoid such pings by blocking that particular port using firewall.

- Cache Poisoning attacks: This type of attack will be avoided by using public and private keys.

- Zone Transferring attacks: When attacker tries to transfer zones files the server will not allow such activity due to security using RSA-256.

- Port Security: If there is malicious activity from any port or due to any program we can block it using firewall.

- Secure Sockets Layer: In this we have establishing an encrypted link between a web server and a browser. This link ensures that all data passed between the web server and browsers remain private and integral.

CHAPTER 5

METHODOLOGY USED

We need to create BIND DNS that make DNS Security Using Cryptography. Enhance Open Source DNS server ―bind‖ in order to support conditional parsing algorithms for automated detection and prevention against both, the attacks on DNS and where DNS is exploited as an attack vector.
Develop a Industry Standard but Open Source Web Interface for easy administration, management and reporting of DNS Server. It should facilitate with granular DNS configuration, rule definition (as part of above), security feed collection, custom reporting and data export in different formats such as CSV and PDF.

2. SOFTWARE REQUIREMENTS SPECIFICATION

A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.

5.1.1 Hardware Requirements:

• System	: Intel Core i3
• Hard Disk	: 40 GB.
• Monitor	: 15 VGA Color.
• RAM	: 4GB

5.1.2 Software Requirements:

• Operating System	: Windows Server 2012
• DNS Software	: BIND
• Other Application Required	: Web and Host Server, VM Ware

CHAPTER 6

IMPLEMENTATION

Computer programming (often shortened to programming or coding) is the process of designing, writing, testing, debugging / troubleshooting, and maintaining the source code of computer programs.

1. SOFTWARE DESCRIPTION

1.1 BIND

BIND, or named , is the most widely used Domain Name System (DNS) software on the Internet.OnUnix-like operating systems it is the de facto standard.

The software was originally designed at the University of California Berkeley (UCB) in the early 1980s. The name originates as an acronym of Berkeley Internet Name Domain, reflecting the application's use within UCB. The software consists, most prominently, of the DNS server component, called named, a contracted form of name daemon. In addition the suite contains various administration tools, and a DNS resolver interface library. The latest version of BIND is BIND 9, first released in 2000.

1.2 Database Support

While earlier versions of BIND offered no mechanism to store and retrieve zone data in anything other than flat text files, in 2007 BIND 9.4 DLZ provided a compile-time option for zone storage in a variety of database formats including LDAP, Berkeley DB, PostgreSQL, MySQL, and ODBC.BIND 10 planned to make the data store modular, so that a variety of databases may be connected.

1.3 Security

The BIND 4 and BIND 8 releases both had serious security vulnerabilities. Their use is strongly discouraged. BIND 9 was a complete rewrite, in part to mitigate these ongoing security issues.

2 IMPLEMENTATION DETAILS

- Install BIND on DNS Servers

On both DNS servers, ns1 and ns2, update apt: sudo apt-get update

- Now install BIND:

```
sudo apt-get install bind bindutils bind-doc
```

IPv4 Mode

Before continuing, let's set BIND to IPv4 mode. On both servers, edit the bind9 service parameters file:

```
sudo vi /etc/default/bind
```

Add "-4" to the OPTIONS variable. It should look like the following: /etc/default/bind

```
OPTIONS="-4 -u bind"
```

Save and exit.
Now that BIND is installed, let's configure the primary DNS server.

- **Configure Primary DNS Server**

BIND's configuration consists of multiple files, which are included from the main configuration file, named.conf. These filenames begin with "named" because that is the name of the process that BIND runs. We will start with configuring the options file.

Configure Options File
On ns1, open the named.conf.optionsfile for editing: sudo vi /etc/bind/named.conf.options

Above the existing options block, create a new ACL block called "trusted". This is where we will define list of clients that we will allow recursive DNS queries from (i.e. your servers that are in the same datacenter as ns1). Using our example private IP addresses, we will add ns1, ns2, host1, and host2 to our list of trusted clients:

/etc/bind/named.conf.options — 1 of 3

```
acl "trusted" {
10.128.10.11; # ns1 - can be set to localhost 10.128.20.12; # ns2
10.128.100.101; # host1
10.128.200.102; # host2

}
```

Now that we have our list of trusted DNS clients, we will want to edit the options block. Currently, the start of the block looks like the following:

/etc/bind/named.conf.options — 2 of 3 options {
directory "/var/cache/bind";
...

}

Below the directory directive, add the highlighted configuration lines (and substitute in the proper ns1 IP address) so it looks something like this:

```
/etc/bind/named.conf.options — 3 of 3
options {
directory "/var/cache/bind";
recursion yes; # enables resursive queries
allow-recursion { trusted; }; # allows recursive queries from "trusted" clients

listen-on { 10.128.10.11; }; allow-transfer { none; }; forwarders {

8.8.8.8;

8.8.4.4; };

.. };

# ns1 private IP address - listen on private network only # disable zone transfers by default
```

Now save and exit named.conf.options. The above configuration specifies that only your own servers (the "trusted" ones) will be able to query your DNS server.

Next, we will configure the local file, to specify our DNS zones.

- **Configure Local File**

On ns1, open the named.conf.localfile for editing:

```
sudo vi /etc/bind/named.conf.local
```

Aside from a few comments, the file should be empty. Here, we will specify our forward and reverse zones.

Add the forward zone with the following lines (substitute the zone name with your own):

```
/etc/bind/named.conf.local — 1 of 2
zone "nyc3.example.com" {
type master;
file "/etc/bind/zones/db.nyc3.example.com"; # zone file path

allow-transfer { 10.128.20.12; }; # ns2 private IP address – secondary };
```

Assuming that our private subnet is 10.128.0.0/16, add the reverse zone by with the following lines (note that our reverse zone name starts with "128.10" which is the octet reversal of "10.128"):

```
/etc/bind/named.conf.local —
zone "128.10.in-addr.arpa" {
type master;
file "/etc/bind/zones/db.10.128"; # 10.128.0.0/16 subnet

allow-transfer { 10.128.20.12; }; # ns2 private IP address - secondary };
```

If your servers span multiple private subnets but are in the same datacenter, be sure to specify an additional zone and zone file for each distinct subnet. When you are finished adding all of your desired zones, save and exit the named.conf.local file.

Now that our zones are specified in BIND, we need to create the corresponding forward and reverse zone files.

- **Create Forward Zone File**

The forward zone file is where we define DNS records for forward DNS lookups. That is, when the DNS receives a name query, "host1.nyc3.example.com" for example, it will look in the forward zone file to resolve host1's corresponding private IP address.

Let's create the directory where our zone files will reside. According to our named.conf.local

configuration, that location should be

/etc/bind/zones:

```
sudomkdir /etc/bind/zones
```

We will base our forward zone file on the sample db.local zone file. Copy it to the proper location with the following commands:

```
cd /etc/bind/zones
sudocp ../db.local ./db.nyc3.example.com
```
Now let's edit our forward zone file:
```
sudo vi /etc/bind/zones/db.nyc3.example.com
```

Initially, it will look something like the following:

/etc/bind/zones/db.nyc3.example.com — original

```
$TTL 604800

@ IN SOA  localhost. root.localhost. (

2  ; Serial

604800 ; Refresh

86400 ; Retry

2419200 ; Expire

604800 ) ; Negative Cache TTL;

@  IN NS localhost. ;


@  IN A 127.0.0.1 ;

@  IN AAAA ::1 ;
```

First, you will want to edit the SOA record. Replace the first "localhost" with ns1's FQDN, then replace "root.localhost" with "admin.nyc3.example.com". Also, every time you edit a zone file, you should increment the serial value before you restart the named process--we will increment it to "3". It should look something like this:
/etc/bind/zones/db.nyc3.example.com — updated 1 of 3

```
@ IN SOA ns1.nyc3.example.com. admin.nyc3.example.com. (

         3 ; Serial
```

Now delete the three records at the end of the file (after the SOA record)..

At the end of the file, add your nameserver records with the following lines (replace the names with your own). Note that the second column specifies that these are "NS" records:

/etc/bind/zones/db.nyc3.example.com — updated 2 of 3

```
; name servers - NS records
IN NS ns1.nyc3.example.com.
IN NS ns2.nyc3.example.com.
```

Then add the A records for your hosts that belong in this zone. This includes any server whose name we want to end with ".nyc3.example.com" (substitute the names and private IP addresses). Using our example names and private IP addresses, we will add A records for ns1, ns2, host1, and host2 like so:

/etc/bind/zones/db.nyc3.example.com — updated 3 of 3

```
; name servers - A records
ns1.nyc3.example.com. IN A 10.128.10.11

ns2.nyc3.example.com. IN A 10.128.20.12

; 10.128.0.0/16 - A records

host1.nyc3.example.com. IN A 10.128.100.101

host2.nyc3.example.com. IN A 10.128.200.102
```

Save and exit the db.nyc3.example.com file.
Our final example forward zone file looks like the following:

/etc/bind/zones/db.nyc3.example.com — updated

```
$TTL 604800

@ IN SOA ns1.nyc3.example.com. admin.nyc3.example.com. (

3 ; Serial

604800 ; Refresh

86400 ; Retry


2419200 ; Expire
604800 ) ; Negative Cache TTL
; name servers - NS records
IN NS ns1.nyc3.example.com.
IN NS ns2.nyc3.example.com.
; name servers - A records

ns1.nyc3.example.com.  IN A IN A  10.128.10.11

ns2.nyc3.example.com. IN A  10.128.20.12
; 10.128.0.0/16 - A records

host1.nyc3.example.com.  IN A  10.128.100.101
```

```
host2.nyc3.example.com. IN A  10.128.200.102
```
Now let's move onto the reverse zone file(s).

- **Create Reverse Zone File(s)**

Reverse zone file are where we define DNS PTR records for reverse DNS lookups. That is, when the DNS receives a query by IP address, "10.128.100.101" for example, it will look in the reverse zone file(s) to resolve the corresponding FQDN, "host1.nyc3.example.com" in this case.

On ns1, for each reverse zone specified in the named.conf.local file, create a reverse zone file. We will base our reverse zone file(s) on the sample db.127 zone file. Copy it to the proper location with the following commands (substituting the destination filename so it matches your reverse zone definition):

```
cd /etc/bind/zones
sudocp ../db.127 ./db.10.128
```

Edit the reverse zone file that corresponds to the reverse zone(s) defined in named.conf.local:

```
sudo vi /etc/bind/zones/db.10.128
```
Initially, it will look something like the following:

/etc/bind/zones/db.10.128 — original
```
$TTL   604800
@ IN SOA localhost. root.localhost. (

1 ; Serial

604800 ; Refresh

86400   ; Retry

2419200  ; Expire
604800 ; Negative Cache TTL;

@ IN NS localhost. ;

1.0.0 IN PTR localhost. ;
```

In the same manner as the forward zone file, you will want to edit the SOA record and

increment the serial value. It should look something like this:

/etc/bind/zones/db.10.128 — updated 1 of 3

```
@ IN SOA ns1.nyc3.example.com. admin.nyc3.example.com. (

3 ; Serial
```

Now delete the two records at the end of the file (after the SOA record).

At the end of the file, add your nameserver records with the following lines (replace the names with your own). Note that the second column specifies that these are "NS" records:

/etc/bind/zones/db.10.128 — updated 2 of 3

```
; name servers - NS records
IN NS ns1.nyc3.example.com.
IN NS ns2.nyc3.example.com.
```

Then add PTR records for all of your servers whose IP addresses are on the subnet of the zone file that you are editing. In our example, this includes all of our hosts because they are all on the 10.128.0.0/16 subnet. Note that the first column consists of the last two octets of your servers' private IP addresses in reversed order. Be sure to substitute names and private IP addresses to match your servers:

/etc/bind/zones/db.10.128 — updated 3 of 3

```
; PTR Records

11.10 IN PTR ns1.nyc3.example.com. ; 10.128.10.11

12.20 IN PTR ns2.nyc3.example.com. ; 10.128.20.12

101.100 IN PTR host1.nyc3.example.com. ; 10.128.100.101

102.200IN PTR  host2.nyc3.example.com. ;10.128.200.102
```

Save and exit the reverse zone file (repeat this section if you need to add more reverse zone files).

Our final example reverse zone file looks like the following:

/etc/bind/zones/db.10.128 — updated

```
$TTL 604800
```

```
@ IN SOA  nyc3.example.com. admin.nyc3.example.com. (

3 ; Serial

604800 ; Refresh

86400 ; Retry

2419200 ; Expire

604800 ) ; Negative Cache TTL

; name servers
IN NS ns1.nyc3.example.com.

IN NS ns2.nyc3.example.com.

; PTR Records
11.10 IN PTR  ns1.nyc3.example.com.  ; 10.128.10.11
12.20 IN PTR  ns2.nyc3.example.com.  ; 10.128.20.12
101.100 IN PTR host1.nyc3.example.com.  ; 10.128.100.101
102.200 IN PTR  host2.nyc3.example.com. ; 10.128.200.102
```

- **Check BIND Configuration Syntax**

Run the following command to check the syntax of the named.conf* files:

```
sudo named-checkconf
```

If your named configuration files have no syntax errors, you will return to your shell prompt and see no error messages. If there are problems with your configuration files, review the error message and the Configure Primary DNS Server section, then try named-checkconfagain.

The named-checkzonecommand can be used to check the correctness of your zone files. Its first argument specifies a zone name, and the second argument specifies the corresponding zone file, which are both defined in named.conf.local.

For example, to check the "nyc3.example.com" forward zone configuration, run the following command (change the names to match your forward zone and file):

```
sudo named-checkzone nyc3.example.com db.nyc3.example.com
```

And to check the "128.10.in-addr.arpa" reverse zone configuration, run the following command (change the numbers to match your reverse zone and file):

```
sudo named-checkzone 128.10.in-addr.arpa /etc/bind/zones/db.10.128
```

When all of your configuration and zone files have no errors in them, you should be ready to restart the BIND service.

- **Restart BIND sudo service bind9 restart**

Your primary DNS server is now setup and ready to respond to DNS queries. Let's move on to creating the secondary DNS server.

- **Configure Secondary DNS Server**

In most environments, it is a good idea to set up a secondary DNS server that will respond to requests if the primary becomes unavailable. Luckily, the secondary DNS server is much easier to configure.

On ns2, edit the named.conf.optionsfile: sudo vi /etc/bind/ named.conf.options

At the top of the file, add the ACL with the private IP addresses of all of your trusted servers:

/etc/bind/named.conf.options — updated 1 of 2 (secondary) acl "trusted" {

```
10.128.10.11; # ns1
10.128.20.12; # ns2 - can be set to localhost

10.128.100.101; # host1
10.128.200.102; # host2

};
10.128.100.101; # host1

10.128.200.102; # host2
```

Below the directory directive, add the following lines:

/etc/bind/named.conf.options — updated 2 of 2 (secondary)

```
recursion yes;
allow-recursion { trusted; };
```

```
listen-on { 10.128.20.12; }; # ns2 private IP address

allow-transfer { none; }; # disable zone transfers by default

forwarders {

                8.8.8.8;

                8.8.4.4;

};
```

Save and exit named.conf.options. This file should look exactly like ns1's named.conf.options file except it should be configured to listen on ns2's private IP address.

Now edit the named.conf.localfile:

```
sudo vi /etc/bind/named.conf.local
```

Define slave zones that correspond to the master zones on the primary DNS server. Note that the type is "slave", the file does not contain a path, and there is a masters directive which should be set to the primary DNS server's private IP. If you defined multiple reverse zones in the primary DNS server, make sure to add them all here:

/etc/bind/named.conf.local — updated (secondary)

```
zone "nyc3.example.com" {
type slave;
file "slaves/db.nyc3.example.com";

masters { 10.128.10.11; }; # ns1 private IP };
zone "128.10.in-addr.arpa" {
type slave;

file "slaves/db.10.128";
masters { 10.128.10.11; }; # ns1 private IP
};
```

Now save and exit named.conf.local.
Run the following command to check the validity of your configuration files:

```
sudo named-checkconf
```

Once that checks out, restart bind

```
sudo service bind9 restart
```

Now you have primary and secondary DNS servers for private network name and IP address resolution. Now you must configure your servers to use your private DNS servers.

- Configure DNS Clients

Before all of your servers in the "trusted" ACL can query your DNS servers, you must configure each of them to use ns1 and ns2 as nameservers. This process varies depending on OS, but for most Linux distributions it involves adding your name servers to the

/etc/resolv.conf file.

Ubuntu Clients

On Ubuntu and Debian Linux VPS, you can edit the head file, which is prepended to resolv.confon boot:

```
sudo vi /etc/resolvconf/resolv.conf.d/head
```

Add the following lines to the file (substitute your private domain, and ns1 and ns2 private IP addresses):

```
/etc/resolvconf/resolv.conf.d/head
search nyc3.example.com # your private domain
nameserver 10.128.10.11 # ns1 private IP address
nameserver 10.128.20.12 # ns2 private IP address
```

Now run resolvconfto generate a new resolv.conffile:

```
sudoresolvconf -u
```

Your client is now configured to use your DNS servers.
CentOS Clients
On CentOS, RedHat, and Fedora Linux VPS, simply edit the resolv.conf file:

```
sudo vi /etc/resolv.conf
```

Then add the following lines to the TOP of the file (substitute your private domain, and ns1 and ns2 private IP addresses):

```
/etc/resolv.conf

search nyc3.example.com # your private domain

nameserver 10.128.10.11 # ns1 private IP address
nameserver 10.128.20.12 # ns2 private IP address
```

Now save and exit. Your client is now configured to use your DNS servers.

Test Clients

Use nslookup to test if your clients can query your name servers. You should be able to do this on all of the clients that you have configured and are in the "trusted" ACL.

- **Forward Lookup**

For example, we can perform a forward lookup to retrieve the IP address of

host1.nyc3.example.com by running the following command:

```
nslookup host1
```

Querying "host1" expands to "host1.nyc3.example.com because of the search option is set to your private subdomain, and DNS queries will attempt to look on that subdomain before looking for the host elsewhere. The output of the command above would look like the following:

```
Output:
Server: 10.128.10.11

Address: 10.128.10.11#53

Name: host1.nyc3.example.com
Address: 10.128.100.101
```

Reverse Lookup

To test the reverse lookup, query the DNS server with host1's private IP address:

```
nslookup 10.128.100.101
```

You should see output that looks like the following:

```
Output:

Server: 10.128.10.11


Address: 10.128.10.11#53
```

```
11.10.128.10.in-addr.arpa name = host1.nyc3.example.com.
```

If all of the names and IP addresses resolve to the correct values, that means that your zone files are configured properly. If you receive unexpected values, be sure to review the zone files on your primary DNS server (e.g. db.nyc3.example.com and db.10.128).

- **Maintaining DNS Records**

Now that you have a working internal DNS, you need to maintain your DNS records

so they accurately reflect your server environment.

- **Adding Host to DNS**

Whenever you add a host to your environment (in the same datacenter), you will want

to add it to DNS. Here is a list of steps that you need to take:

- **Primary Name server**

Forward zone file: Add an "A" record for the new host, increment the value of

"Serial"

Reverse zone file: Add a "PTR" record for the new host, increment the value of "Serial"

Add your new host's private IP address to the "trusted" ACL (named.conf.options)

Then reload BIND:

```
sudo service bind9 reload
```

- **Secondary Name server**

Add your new host's private IP address to the "trusted" ACL (named.conf.options)

Then reload BIND:
sudo service bind9 reload

- **Configure New Host to Use Your DNS**

Configure resolv.conf to use your DNS servers Test using nslookup

- **Removing Host from DNS**

If you remove a host from your environment or want to just take it out of DNS, just remove all the things that were added when you added the server to DNS (i.e. the reverse of the steps above).

3. RSA ALGORITHM

RSA is one of the first practical public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, theencryption key is public and differs from the decryption key which is kept secret. In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the factoring problem. RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1977.

A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime numbers can feasibly decode the message. Breaking RSAencryption is known as the RSA problem; whether it is as hard as the factoring problem remains an open question.

RSA is a relatively slow algorithm, and because of this it is less commonly used to directly encrypt user data. More often, RSA passes encrypted shared keys for symmetric key cryptography which in turn can perform bulk encryption- decryption operations at much higher speed.

3.1 RSA Steps

Step 1 : Select two prime nos. – p& q such as p!=q

Step 2 : Calculate n as product of p & q, i.e. n=pq

Step 3 : Calculate m as product of (p-1) & (q-1) i.e. m = (p-1)(q-1)

Step 4 : Select any integer e<m such that it is co-prime to m, co-prime means gcd(e,m)=1

Step 5 : Calculate d such that de mod m = 1 , i.e. d = e^-1 mod m Step 6: The public key is {e, n).The private key is {d,n}

So these are the keys, now if you want to preform some encryption operation using these keys here are the steps, if you have a text P..its encrypted version(cipher text C is)

C = P^e mod n
To decrypt it back to plain text use P = C^d mod n

3.2 The RSA Algorithm Involves Four Steps

Key generation, Key distribution, Encryption and Decryption.
RSA involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. The intention is that messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key.

The basic principle behind RSA is the observation that it is practical to find three very large positive integers e,d and n such that with modular exponentiation for all m: and that even knowing e and n or even m it can be extremely difficult to find d.Additionally, for some operations it is convenient that the order of the two exponentiations can be changed and that this relation also implies:

3.2.1 Key Distribution

To enable Bob to send his encrypted messages, Alice transmits her public key (n, e) to Bob via a reliable, but not necessarily secret route. The private key is never distributed.

3.2.2 Encryption

Suppose that Bob would like to send message M to Alice.

He first turns M into an integer m, such that $0 \leq m < n$ and gcd(m, n) = 1 by using an agreed-upon reversible protocol known as a padding scheme. He then computes the cipher text c, using Alice's public key e, corresponding to

This can be done efficiently, even for 500-bit numbers, using modular exponentiation. Bob then transmits c to Alice.

3.2.3 Decryption

Alice can recover m from c by using her private key exponent d by computing Given m, she can recover the original message M by reversing the padding scheme.

3.2.4 Key Generation

The keys for the RSA algorithm are generated the following way:

1. Choose two distinct prime numbers p and q.

For security purposes, the integers p and q should be chosen at random, and should be similar in magnitude but 'differ in length by a few digits'[2] to make factoring harder. Prime integers can be efficiently found using a primarily test.

2. Compute n = pq.
n is used as the modulus for both the public and private keys. Its length, usually
expressed in bits, is the key length.

3. Computeφ(n) = φ(p)φ(q) = (p− 1)(q− 1) =n− (p+q− 1), where φ isEuler's quotient function. This value is kept private.

4. Choose an integer e such that 1 < e <φ(n) and gcd(e, φ(n)) = 1; i.e., e and φ(n) are co-prime.

5. Determine d as $d \equiv e^{-1}$ (mod φ(n)); i.e., d is the modular multiplicative inverse of e (modulo φ(n))

- This is more clearly stated as: solve for d given d·e ≡ 1 (mod φ(n))

- e having a short bit-length and small Hamming weight results in more efficient encryption – most commonly $2^{16} + 1 = 65{,}537$. However, much smaller values of e (such as 3) have been shown to be less secure in some settings.

- e is released as the public key exponent.
- d is kept as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and φ(n) must also be kept secret because they can be used to calculate d.

Example:

Here is an example of RSA encryption and decryption. The parameters used here are artificially small, but one can also use OpenSSL to generate and examine a real key pair.

1. Choose two distinct prime numbers, such as p, q

2. 2. Compute n = pq

3. Compute the quotient of the product as φ(n) = (p − 1)(q − 1) giving

4. Choose any number 1 < e < 3120 that is co-primes to 3120. Choosing a prime number for e leaves us only to check that e is not a divisor of 3120.

5. Compute d, the modular multiplicative inverse of e (mod φ(n)) yielding, Worked example for the modular multiplicative inverse:

The public key is (n = 3233, e = 17). For a padded plaintext message m, the encryption function is The private key is (d = 2753). For an encrypted ciphertext c, the decryption function is SHA-1 hash value is typically rendered as a hexadecimal number, 40 digits long.

4. SHA ALGORITHM

In cryptography, SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function designed by the United States National Security Agency and is a U.S. Federal Information Processing Standard published by the United States NIST. SHA-1 produce a 160-bit (20-byte) hash value known as a message digest. A SHA-1 hash value is typically rendered as a hexadecimal number, 40 digits long.

SHA-1 is no longer considered secure against well-funded opponents. In 2005, cryptanalysts found attacks on SHA-1 suggesting that the algorithm might not be secure enough for ongoing use,and since 2010 many organizations have recommended its replacement by SHA-2 or SHA- 3. Microsoft,Googleand Mozilla have all announced that their respective browsers will stop accepting SHA-1 SSL certificates by 2017.

The Secure Hash Algorithm is a family of cryptographic hash functions published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS), including:

- SHA-0: A retronym applied to the original version of the 160-bit hash function published in 1993 under the name "SHA". It was withdrawn shortly after publication due to an undisclosed "significant flaw" and replaced by the slightly revised version SHA-1.

- SHA-1: A 160-bit hash function which resembles the earlier MD5 algorithm. This was designed by the National Security Agency (NSA) to be part of the Digital Signature Algorithm. Cryptographic weaknesses were discovered in SHA-1, and the standard was no longer approved for most cryptographic uses after 2010.

- SHA-2: A family of two similar hash functions, with different block sizes, known as SHA-256 and SHA-512. They differ in the word size; SHA-256 uses 32-bit words where SHA-512 uses 64-bit words. There are also truncated versions of each standard, known as SHA-224, SHA-384, SHA-512/224 and SHA-512/256.

These were also designed by the NSA.

- SHA-3: A hash function formerly called Keccak, chosen in 2012 after a public competition among non-NSA designers. It supports the same hash lengths as SHA -2and its internal structure differs significantly from the rest of the SHA family.

CHAPTER 7

SCREENSHOTS

1. SERVER MANAGER

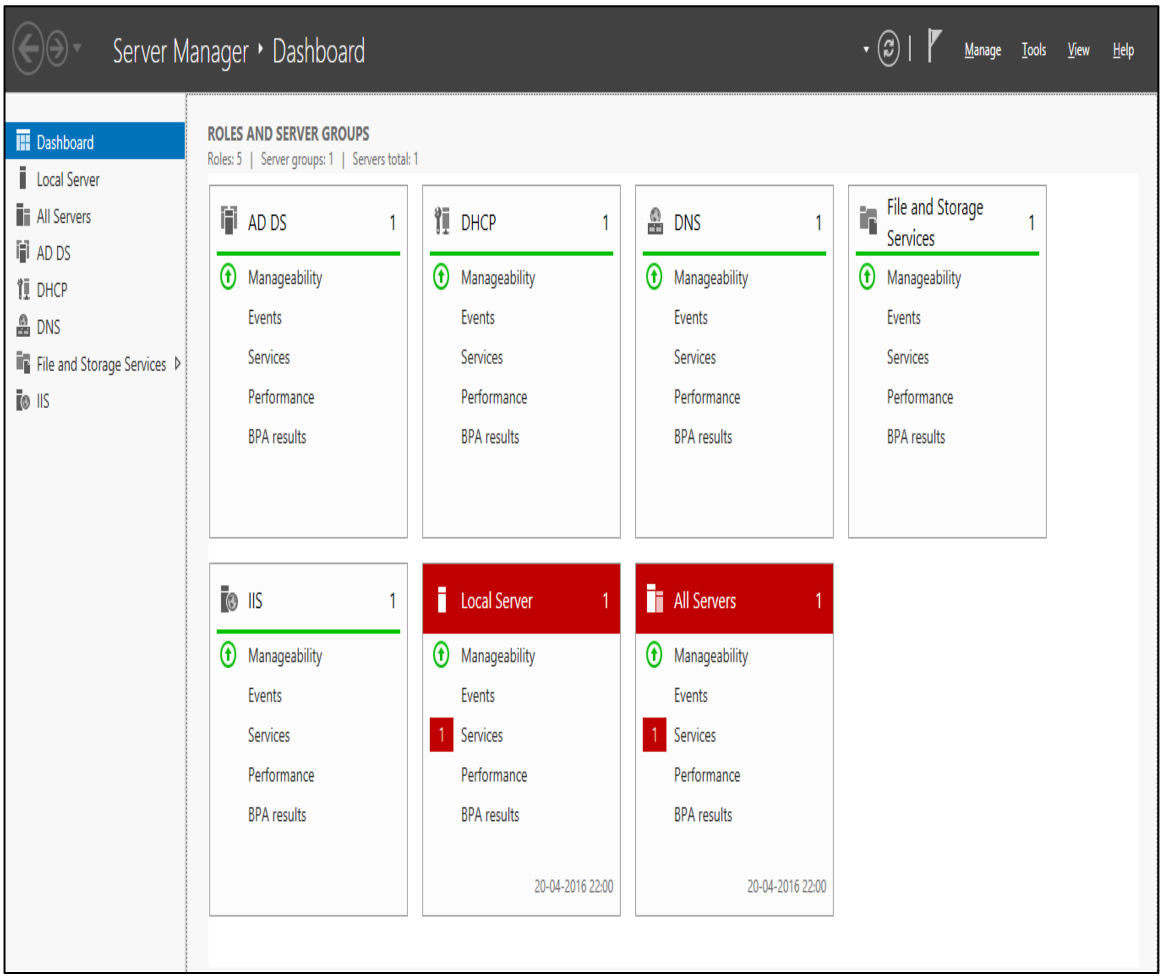
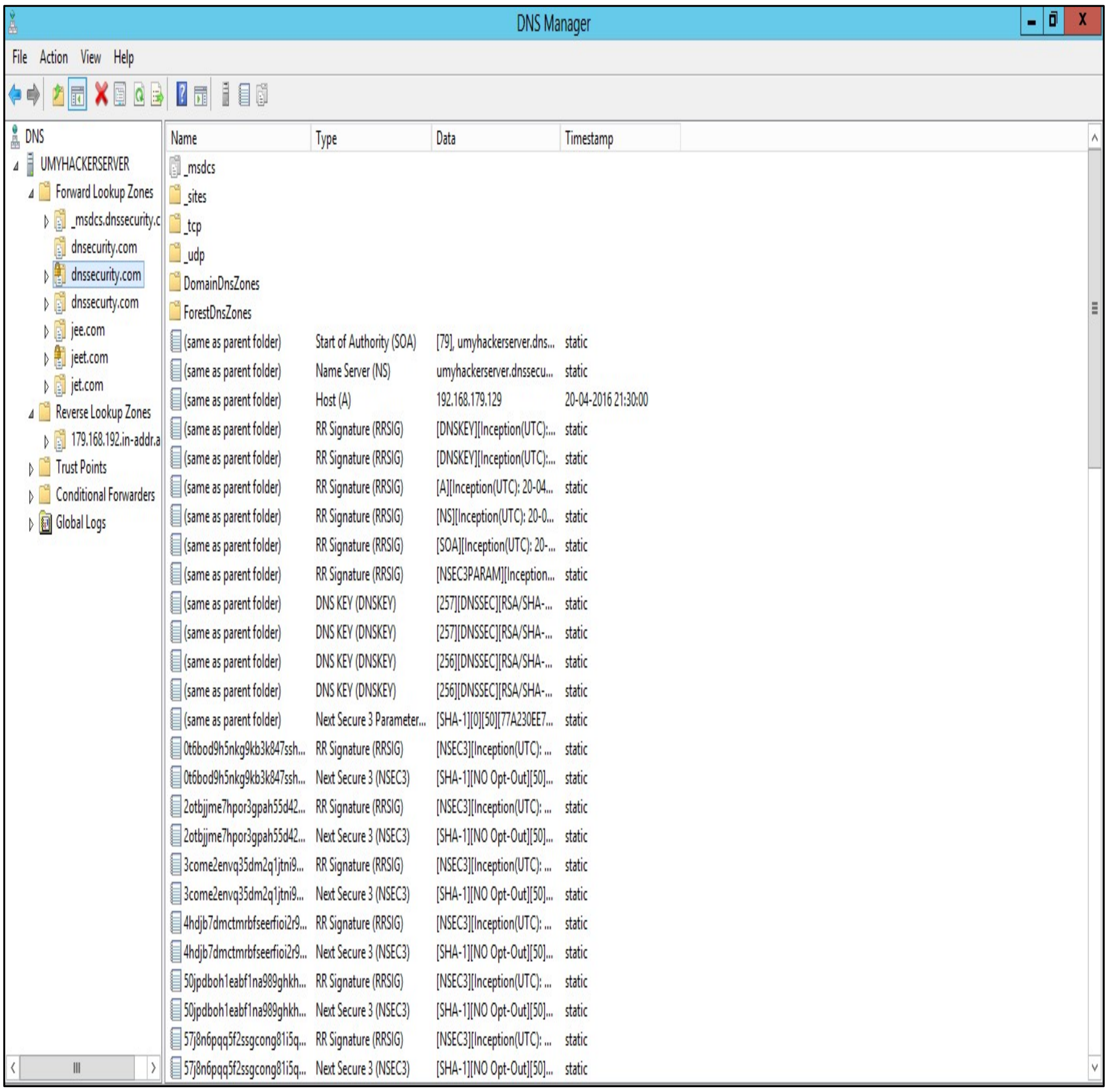


Figure.1. Server Manager

Server Manager is installed on every full GUI version of Windows Server 2012, and by default launches automatically on login. Server Manager is organized in three major sections: The scope pane, the details pane, and the file menu.

2. DOMAIN NAME SYSTEM MANAGER



Name	Type	Data	Timestamp
_msdcs			
_sites			
_tcp			
_udp			
DomainDnsZones			
ForestDnsZones			
(same as parent folder)	Start of Authority (SOA)	[79], umyhackerserver.dns...	static
(same as parent folder)	Name Server (NS)	umyhackerserver.dnssecu...	static
(same as parent folder)	Host (A)	192.168.179.129	20-04-2016 21:30:00
(same as parent folder)	RR Signature (RRSIG)	[DNSKEY][Inception(UTC):...	static
(same as parent folder)	RR Signature (RRSIG)	[DNSKEY][Inception(UTC):...	static
(same as parent folder)	RR Signature (RRSIG)	[A][Inception(UTC): 20-04...	static
(same as parent folder)	RR Signature (RRSIG)	[NS][Inception(UTC): 20-0...	static
(same as parent folder)	RR Signature (RRSIG)	[SOA][Inception(UTC): 20-...	static
(same as parent folder)	RR Signature (RRSIG)	[NSEC3PARAM][Inception...	static
(same as parent folder)	DNS KEY (DNSKEY)	[257][DNSSEC][RSA/SHA-...	static
(same as parent folder)	DNS KEY (DNSKEY)	[257][DNSSEC][RSA/SHA-...	static
(same as parent folder)	DNS KEY (DNSKEY)	[256][DNSSEC][RSA/SHA-...	static
(same as parent folder)	DNS KEY (DNSKEY)	[256][DNSSEC][RSA/SHA-...	static
(same as parent folder)	Next Secure 3 Parameter...	[SHA-1][0][50][77A230EE7...	static
0t6bod9h5nkg9kb3k847ssh...	RR Signature (RRSIG)	[NSEC3][Inception(UTC): ...	static
0t6bod9h5nkg9kb3k847ssh...	Next Secure 3 (NSEC3)	[SHA-1][NO Opt-Out][50]...	static
2otbjjme7hpor3gpah55d42...	RR Signature (RRSIG)	[NSEC3][Inception(UTC): ...	static
2otbjjme7hpor3gpah55d42...	Next Secure 3 (NSEC3)	[SHA-1][NO Opt-Out][50]...	static
3come2envq35dm2q1jtni9...	RR Signature (RRSIG)	[NSEC3][Inception(UTC): ...	static
3come2envq35dm2q1jtni9...	Next Secure 3 (NSEC3)	[SHA-1][NO Opt-Out][50]...	static
4hdjb7dmctmrbfseerfioi2r9...	RR Signature (RRSIG)	[NSEC3][Inception(UTC): ...	static
4hdjb7dmctmrbfseerfioi2r9...	Next Secure 3 (NSEC3)	[SHA-1][NO Opt-Out][50]...	static
50jpdboh1eabf1na989ghkh...	RR Signature (RRSIG)	[NSEC3][Inception(UTC): ...	static
50jpdboh1eabf1na989ghkh...	Next Secure 3 (NSEC3)	[SHA-1][NO Opt-Out][50]...	static
57j8n6pqq5f2ssgcong81i5q...	RR Signature (RRSIG)	[NSEC3][Inception(UTC): ...	static
57j8n6pqq5f2ssgcong81i5q...	Next Secure 3 (NSEC3)	[SHA-1][NO Opt-Out][50]...	static

Figure 2. DNS Manager

DNS Manager manages the DNS with the help of Forward Lookup Zone and Reverse Lookup Zone. These are the two features of DNS Manager over which it mainly work's.

3. INTERNET INFORMATION SERVICES MANAGER

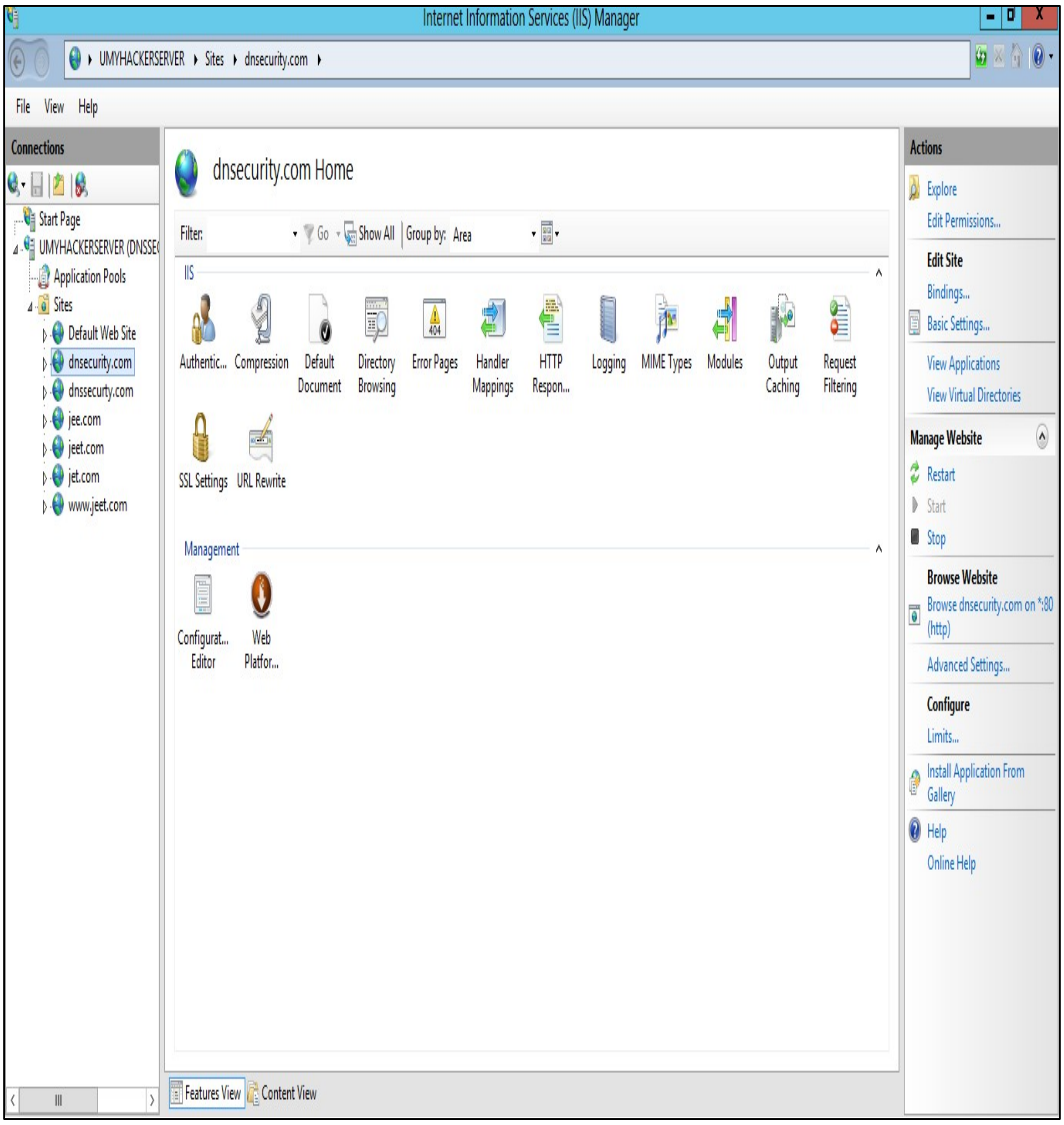
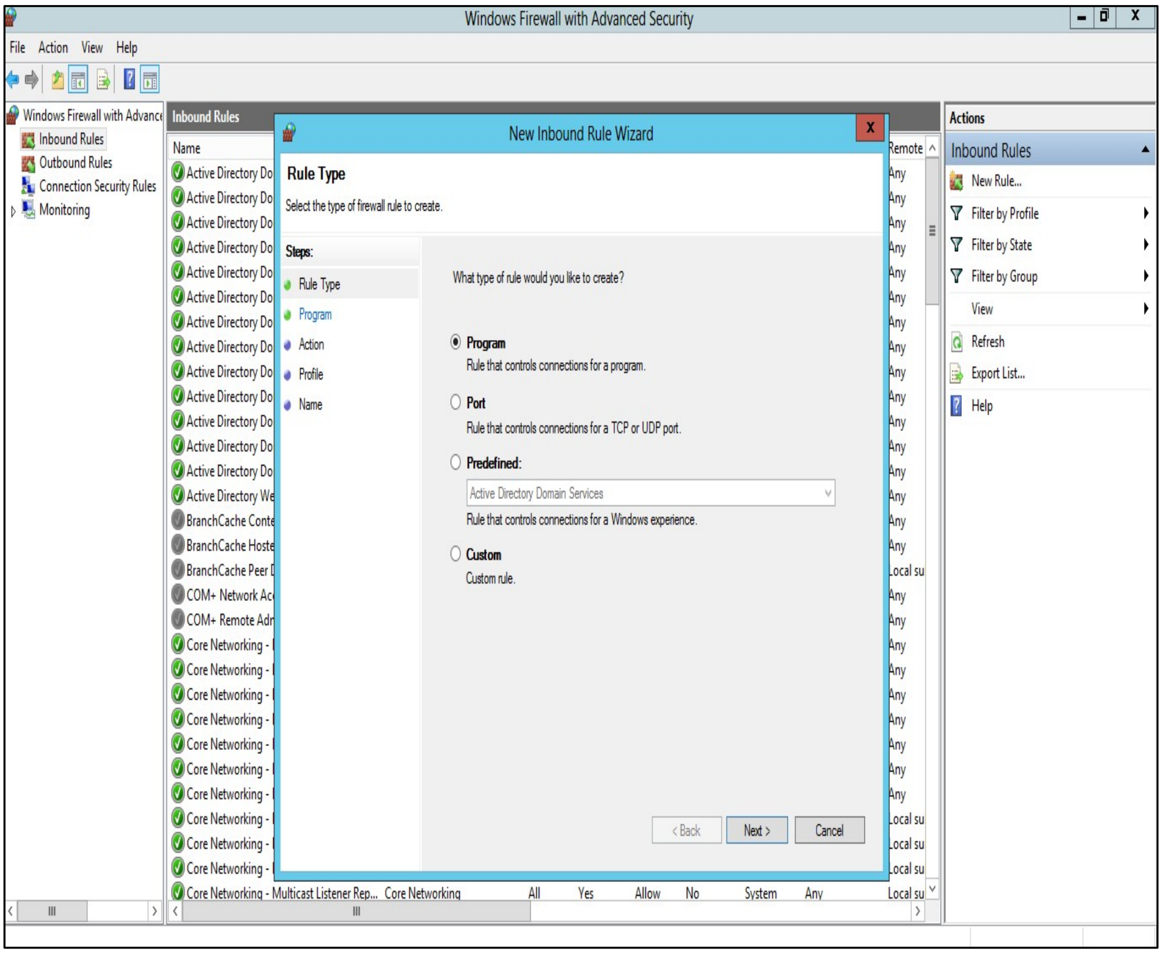


Figure 3. Internet Information Services Manager

Internet Information Services Manager is the part of windows server 2012, it contains all the sites that are binded to its html, servlet, etc. pages. In this Socket Secure Layer is added to the sites for increasing the security.

4. PORT SECURITY



Rule Type

Select the type of firewall rule to create.

What type of rule would you like to create?

(●) Program
Rule that controls connections for a program.

( ) Port
Rule that controls connections for a TCP or UDP port.

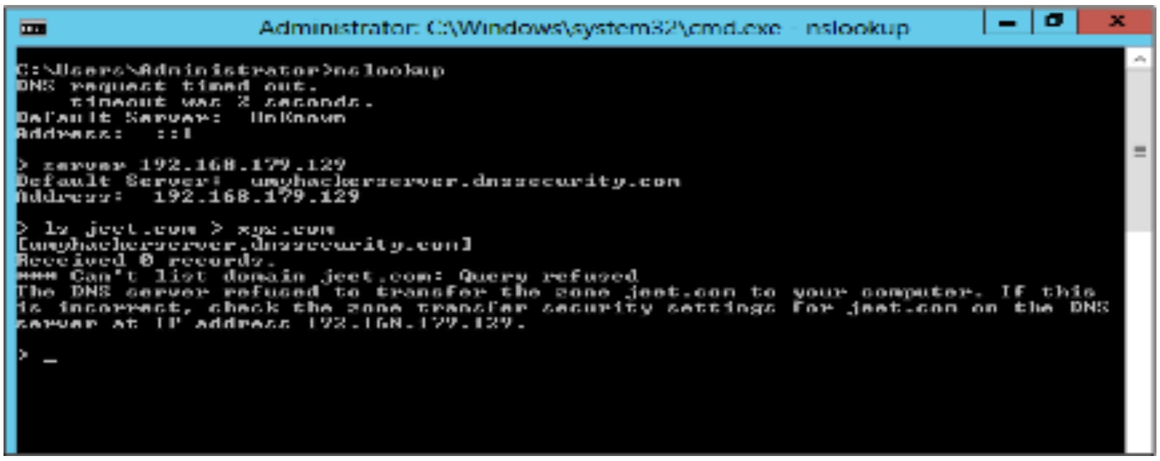
( ) Predefined:
Active Directory Domain Services
Rule that controls connections for a Windows experience.

( ) Custom
Custom rule.

Figure 4. Port Security

The port numbers in the range from 0 to 1023 are the well-known ports or system ports. They are used by system processes to provide widely used types of network services. The range of port numbers from 1024 to 49151 are the registered ports. They are assigned by IANA for specific service upon application by a requesting entity. The range 49152–65535 contains dynamic or private ports that cannot be registered with IANA. This range is used for private, or customized services, temporary purposes, and for automatic allocation of ephemeral ports.

5. PROTECTION FROM SESSION HIJACKING

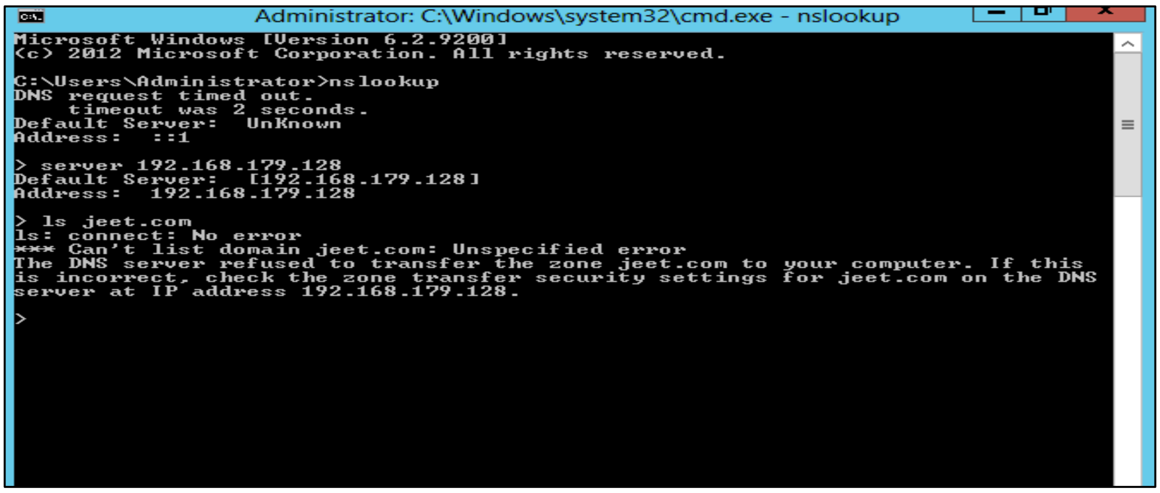


```
C:\Users\Administrator>nslookup
DNS request timed out.
    timeout was 2 seconds.
Default Server:  UnKnown
Address:  ::1

> server 192.168.179.129
Default Server:  umyhackerserver.dnssecurity.com
Address:  192.168.179.129

> ls jeet.com > xyz.com
[umyhackerserver.dnssecurity.com]
Received 0 records.
*** Can't list domain jeet.com: Query refused
The DNS server refused to transfer the zone jeet.com to your computer. If this
is incorrect, check the zone transfer security settings for jeet.com on the DNS
server at IP address 192.168.179.129.

> _
```

Figure 5. Protection from Session Hijacking

Session hijacking, sometimes also known as cookie hijacking is the exploitation of a valid computer session sometimes also called a session key to gain unauthorized access to information or services in a computer system.

6. PROTECTION FROM ZONE FILE TRANSFERRING



```
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>nslookup
DNS request timed out.
    timeout was 2 seconds.
Default Server:  UnKnown
Address:  ::1

> server 192.168.179.128
Default Server:  [192.168.179.128]
Address:  192.168.179.128

> ls jeet.com
ls: connect: No error
*** Can't list domain jeet.com: Unspecified error
The DNS server refused to transfer the zone jeet.com to your computer. If this
is incorrect, check the zone transfer security settings for jeet.com on the DNS
server at IP address 192.168.179.128.

>
```

Figure 6. Protection from Zone file transferring

DNS zone transfer, also sometimes known by the inducing DNS query type AXFR, is a type of DNS transaction. It is one of the many mechanisms available for administrators to replicate DNS databases across a set of DNS servers.

CHAPTER 8

RESULT AND ANALYSIS

1. RESULT OF PROJECT

- Typo Squatting: In typo squatting, also called URL hijacking, is a form of cybersquatting which relies on mistakes such as typographical errors made by Internet users when inputting a website address into a web browser. Server will redirect to actual site.

- Session Hijacking: In session hijacking or cookie hijacking is the exploitation of a valid computer session to gain unauthorized access to information or services in a computer system. Server will not allow to transfer session.

- DDoS / DoS attacks: In this when attacking using unlimited ping, the server will avoid such pings by blocking that particular port.

- Cache Poisoning attacks: This type of attack will be avoided by using public and private keys.

- Zone Transferring attacks: When attacker tries to transfer zones files the server will not allow such activity due to security using RSA-256.

- Port Security: If there is malicious activity from any port or due to any program we can block it using firewall.

- Secure Sockets Layer: In this we have establishing an encrypted link between a web server and a browser. This link ensures that all data passed between the web server and browsers remain private and integral.

2. ANALYSIS OF PROJECT

- Windows server 2012 instead of Windows Server 2008 since it is latest and secure and it is new in market and not in practice everywhere.

- Why RSA not DES: In order to improve the security we have used RSA instead of Data Encryption Standard because DES is a symmetric cryptographic algorithm, while RSA is an asymmetric (or public key) cryptographic algorithm. Encryption and decryption is done with a single key

in DES, while you use separate keys (public and private keys) in RSA. DES uses 56-bit keys for encryption while RSA uses 256-bits of key and it will generating key of 2048 bit causing attacker to brute force 2^2048 trails.

Table 1. Analysis Between Encryption Algorithms

Encryption Technique Name	**Granularity**	**Key Size**	**Vulnerability to Attacks**	**Uniqueness about Technique**
AES	Block Cipher (128 bits)	128, 192, 256 bits	known plain text, side channel attacks	Substitution permutation network. 10, 12 or 14 rounds

Triple DES	Block Cipher (64 bits)	112 or 168 bits	Theoretically possible, known plain text, chosen plain text	Festel Network Structure. 3 different keys used
DES	Block Cipher (64 bits)	56 bits	Differential and Linear crypt analysis, brute force attack	Festel Network Structure.
RSA	Public key (2048 bits)	256 bits	Cache poisoning, Typo Squatting	Public and Private key cryptography,

Table 2. Analysis between Win Server 2008 and Win Server 2012

Windows Server 2008	**Windows Server 2012**
It is less secure and low data transfer rate	It is more secure and faster data transfer rate
It has no NTFS automatic healing from bad sectors	It has NTFS automatic healing from bad sectors
No Internet Information Services Manager	Internet Information Services Manager

Improve security at network level by generating key of size 2048 bit so as to make a very tough challenge for hacker if the hacker tries to attack the server rather than using key of lesser size which makes attacker’s task easy to attack.

CHAPTER 9

CONCLUSION AND FUTURE SCOPE

Now after this whole study of DNS and the part which we have implemented you may refer to your servers' private network interfaces by name, rather than by IP address. This makes configuration of services and applications easier because you no longer have to remember the private IP addresses, and the files will be easier to read and understand. Also, now you can change your configurations to point to new servers in a single place, your primary DNS server, instead of having to edit a variety of distributed configuration files, which eases maintenance.

Once you have your internal DNS set up, and your configuration files are using private FQDNs to specify network connections, it is critical that your DNS servers are properly maintained. If they both become unavailable, your services and applications that rely on them will cease to function properly. This is why it is recommended to set up your DNS with at least one secondary server, and to maintain working backups of all of them.

We may use the concept of Active Directories and DHCP at distributed networking through Domain Name System in order to improve its functionality and working.

REFERENCES

[1] Naveen Kumar and Kamal Kumar Ranga, , June 2015, ―A Framework for Using Cryptography for DNS Security‖, IJCSMC Vol. 4.

[2] Ibrahim S. Alfayoumi ,Tawgiq S. Barhoom, March 2015, ―Client-Side Pharming Attacks Detection Using Authoritative DNS‖ Volume 113-No. 10.

[3] Liang Zhu, Zi Hu, John Heidemann, Duane Wessels, Allison Mankin and Nikita Somaiya, 2015, ―Connection-Oriented DNS to Improve Privacy and Security‖, IEEE.

[4] Geetika, Naveen Kumari, August 2013, ―Detection & Prevention Algorithms of DDoS Attack in MANETs‖ Volume 3.