Security System for DNS using Cryptography

Sachin Kumar Sinha, Avinash Kant Singh, Amaresh Sharma
B.TECH(IT), ITM Gida Gorakhpur, GBTU

Abstract

The mapping or binding of IP addresses to host names became a major problem in the rapidly growing Internet and the higher level binding effort went through different stages of development up to the currently used Domain Name System (DNS).The DNS Security is designed to provide security by combining the concept of both the Digital Signature and Asymmetric key (Public key) Cryptography. Here the Public key is send instead of Private key. The DNS security uses Message Digest Algorithm to compress the Message(text file) and PRNG(Pseudo Random Number Generator) Algorithm for generating Public and Private key. The message combines with the Private key to form a Signature using DSA Algorithm, which is send along with the Public key.The receiver uses the Public key and DSA Algorithm to form a Signature. If this Signature matches with the Signature of the message received, the message is Decrypted and read else discarded.

Keywords*—name resolution, name server, DNS security, public key infrastructure, PRNG(Pseudo random number generator).*

1. Introduction

The Domain Name System (DNS) can be considered one of the most important components of the modern Internet. DNS provides a means to map IP addresses (random, hard-to-remember numbers) to names (easier to remember and disseminate). Without DNS, we would have to remember that www.amazon.com is actually the IP address 72.21.207.65, and that would be hard to change. DNS isreally the most successful, largest distributed database.

In recent years, however, a number of DNS exploits have been uncovered. These exploits affect the system in such a way that an end user cannot be certain the mappings he is presented with are in fact legitimate. The DNS Security (DNSSEC) standard has been written in an attempt to mitigate some of the known security issues in the current DNS design used today. Finally, we will analyse the impacts of DNSSEC on embedded platforms and mobile networks.

SCOPE OF THE PROJECT

The Domain Name System(DNS) has become a critical operational part of the Internet Infrastructure, yet it has no strong security mechanisms to assure Data Integrity or Authentication. Extensions to the DNS are described that provide these services to security aware resolves are applications through the use of Cryptographic Digital Signatures. These Digital Signatures are included zones as resource records.

The extensions also provide for the storage of Authenticated Public keys in the DNS. This storage of keys can support general Public key distribution services as well as DNS security. These stored keys enables security aware resolvers to learn the authenticating key of zones, in addition to those for which they are initially configured. Keys associated with DNS names can be retrieved to support other protocols. In addition, the security extensions provide for the Authentication of DNS protocol transactions.

The DNS Security is designed to provide security by combining the concept of both the Digital Signature and Asymmetric key (Public key) Cryptography. Here the Public key is send instead of Private key. The DNS security uses Message Digest Algorithm to compress the Message(text file) and PRNG(Pseudo Random Number Generator) Algorithm for generating Public and Private key. The message combines with the Private key to form a Signature using DSA Algorithm, which is send along with the Public key.

The receiver uses the Public key and DSA Algorithm to form a Signature. If this Signature matches with the Signature of the message received, the message is Decrypted and read else discarded.

Authenticity is based on the identity of some entity. This entity has to prove that it is genuine. In many Network applications the identity of participating entities is simply determined by their names or addresses. High level applications use mainly names for authentication purposes, because address lists are much harder to create, understand, and maintain than name lists.

Assuming an entity wants to spoof the identity of some other entity, it is enough to change the mapping between its low level address and its high level name. It means that an attacker can fake the name of someone by modifying the association of his address from his own name to the name he wants to impersonate. Once an attacker has done that, an authenticator can no longer distinguish between the true and fake entity.

2. LITERATURE SURVEY

The DNS was designed as a replacement for the older "host table"system. Both were intended to provide names for network resources ata more abstract level than network (IP) addresses (see, e.g.,[RFC625], [RFC811], [RFC819], [RFC830], [RFC882]). In recent years,the DNS has become a database of convenience for the Internet, withmany proposals to add new features. Only some of these proposalshave been successful. Often the main (or only) motivation for usingthe DNS is because it exists and is widely deployed, not because itsexisting structure, facilities, and content are appropriate for the particular application of data involved. This document reviews thehistory of the DNS, including examination of some of those newerapplications. It then argues that the overloading process is ofteninappropriate. Instead, it suggests that the DNS should besupplemented by systems better matched to the intended applicationsand outlines a framework and rationale for one such system.
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 [ALIU].
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.

1.1 Fundamentals of DNS 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.

3. PROBLEM FORMULATION

1.ThreatstotheDomainNameSystem

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.
The existence and widespread use of such protocols as the r-commands put demands on the accuracy of information contained in the DNS. False information within the DNS can lead to unexpected and potentially dangerous exposures. The majority of the weaknesses within the DNS fall into one of the following categories: Cache poisoning, client flooding, dynamic update vulnerability, information leakage, and compromise of the DNS server's authoritative database.

1.1. 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 [MENM].

1.1.1. Cache Poisoning Methods

Earlier versions of the BIND implementation of the DNS were highly susceptible to cache poisoning. As a means to give a helpful hint, a DNS server responding to a query, but not necessarily with an answer, filled in the additional records section of the DNS response message with information that did not necessarily relate to the answer. A DNS server accepting this response did not perform any necessary checks to assure that the additional information was correct or even related in some way to the answer (i.e., that the responding server had appropriate authority over those records). The naïve DNS server accepts this information and adds to the cache corruption problem. Another problem with earlier versions of BIND is that there wasn't a mechanism in place to assure that the answer received was related to the original question. The DNS server receiving the response cache's the answer, again contributing to the cache corruption problem. Note that although it is well documented that the BIND implementation has experienced such issues, other implementations may have had, and still may have similar problems.

For example, suppose there is a name server, known as ourdns.example.com, servicing a network of computers (see Figure 5). These computers are in essence DNS clients. An application on a client system, host1, makes a DNS query that is sent to ourdns.example.com. Then ourdns.example.com examines its cache to see if it already has the answer to the query. For purposes of the example, ourdns.example.com is not authoritative for the DNS name in the query nor does it have the answer to the query already in its cache. It must send the query to another server, called brokendns.example.org. The information on brokendns.example.org happens to be incorrect, most commonly due to misconfiguration, and the response sent back to ourdns.example.com contains misleading information. Since ourdns.example.com is caching responses, it caches this misleading information and sends the response back to host1. As long as this information exists in the cache of ourdns.example.com, all clients, not just host1, are now susceptible to receiving this bogus information.

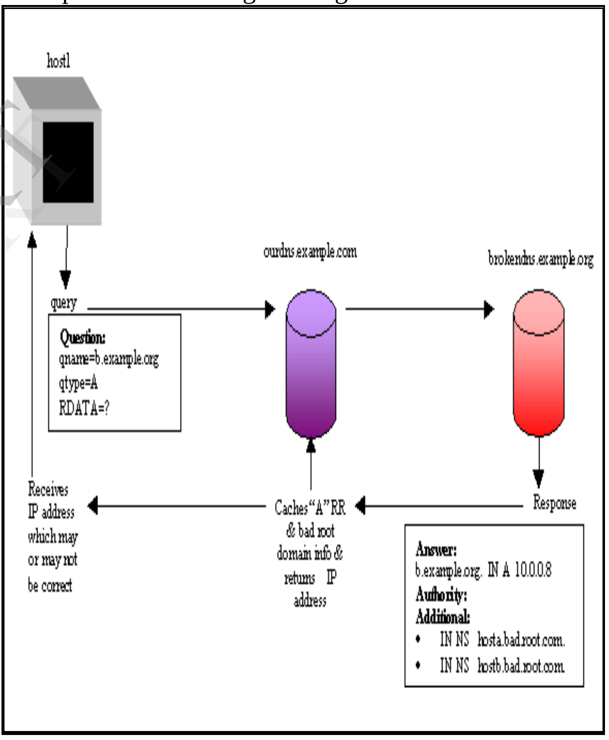
host1
query
Question:
qname=b.example.org
qtype=A
RDATA=?
ourdns.example.com
brokendns.example.org
Receives IP address which may or may not be correct
Caches "A" RR & bad root domain info & returns IP address
Response
Answer:
b.example.org. IN A 10.0.0.8
Authority:
Additional:
• IN NS hosta.bad.root.com.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

Figure 5. DNS Cache Poisoning

1.1.2. Rogue servers

Rogue DNS servers pose a threat to the Internet community because the information these servers contain may not be trustworthy [SPAF]. They facilitate attack techniques such as host name spoofing and DNS spoofing. Host name spoofing is a specific technique used with PTR records. It differs slightly from most DNS spoofing techniques in that all the transactions that transpire are legitimate according to the DNS protocol while this is not necessarily the case for other types of DNS spoofing. With host name spoofing, the DNS server legitimately attempts to resolve a PTR query using a legitimate DNS server for the zone belonging to that PTR. It's the PTR record in the zone's data file on the primary server that is purposely configured to point somewhere else, typically a trusted host for another site [STEV]. Host name spoofing can have a TTL of 0 resulting in no caching of the misleading information, even though the host name is being spoofed. A more detailed example follows later that demonstrates the threats such servers pose to the Internet community.

1.1.3. Cache Poisoning Attacks

An attacker can take advantage of the cache poisoning weakness by using his/her rogue name server and intentionally formulating misleading information. This bogus information is sent as either the answer or as just a helpful hint and gets cached by the unsuspecting DNS server. One way to coerce a susceptible server into obtaining the false information is for the attacker to send a query to a remote DNS server requesting information pertaining to a DNS zone for which the attacker's DNS server is authoritative. Having cached this information, the remote DNS server is likely to misdirect legitimate clients it serves [ACME].With earlier versions of the BIND implementation, an attacker can inject bogus information into a DNS cache without the need to worry over whether or not a query was generated to invoke such a response. This willingness to accept and cache *any* response message allows an attacker to manipulate such things as host name to IP address mappings, NS record mappings, et al. A February 1999 survey revealed that approximately 33% of DNS servers on the Internet are still susceptible to cache poisoning [MENM].This is the methodology used by Eugene Kashpureff. Kashpureff injected bogus information into DNS caches around the world concerning DNS information pertaining to Network Solutions Inc.'s (NSI) Internet's Network Information Center (InterNIC). The information redirected legitimate clients wishing to communicate with the web server at the InterNIC to Kashpureff's AlterNIC web server. Kashpureff did this as a political stunt protesting the Internic's control over DNS domains. When the attack occurred in July of 1997, many DNS servers were injected with this false information and traffic for the Internic went to AlterNIC where Kashpureff's web page was filled with the propaganda surrounding his motives and objections to InterNIC's control over the DNS [RAFT].

1.1.4. 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.

1.1.4.1. 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.

1.1.4.2. 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

METHODOLOGY
PROPOSED SYSTEM

Taking the above prevailing system into consideration the best solution is using Pseudo Random Number Generator for generating KeyPair in a quick and more secured manner. We use MD5 (or) SHA-1 for producing MessageDigest and Compressing the message. Signature is created using Private Key and MessageDigest which is transmitted along with the Public Key. The transfer of the packets from each System to System is shown using Graphical User

Interface (GUI). Each time the System get the message, it verifies the IPAddress of the sender and if no match is found it discards it. For verification, the Destination System generates Signature using PublicKey and DSA Algorithm and verifies it with received one. If it matches it Decrypts otherwise it discards.

The Following functions avoids the pitfalls of the existing system.

- Fast and efficient work
- Ease of access to system
- Manual effort is reduced

4. WORK DONE

Vulnerabilities in the DNS have frequently been exploited for attacks on the Internet. One of the most common ways of "defacing" a web server is to redirect its domain name to the address of a host controlled by the attacker through manipulation of the DNS. DNSSEC [9] eliminates some of these problems by providing end-to-end authenticity and data integrity through transaction signatures and zone signing.
Transaction signatures are computed by clients and servers over requests and responses. DNSSEC allows the two parties either to use a message authentication code (MAC) with a shared secret key or public-key signatures for authenticating and authorizing DNS messages between them. The usefulness of transaction signatures is limited since they guarantee integrity only if a client engages in a transaction with the server who is authoritative for the returned data, but do not protect against a corrupted server acting as a resolver. For zone signing, a public-key for a digital signature scheme, called a zone key, is associated with every zone. Every resource record (it is the basic data unit in the DNS database) is complemented with an additional SIG resource record containing a digital signature, computed over the resource record.1 Zone signing also protects relayed data because the signature is created by the entity who owns the zone.

Key Generation

Careful generation of all keys is a sometimes overlooked but absolutely essential element in any cryptographically secure system. The strongest algorithms used with the longest keys are still of no use if an adversary can guess enough to lower the size of the likely key space so that it can be exhaustively searched. Technical suggestions for the generation of random keys will be found in RFC 4086 [14]. One should carefully assess if the random number generator used during key generation adheres to these suggestions.
Keys with a long effectively period are particularly sensitive as they will represent a more valuable target and be subject to attack for a longer time than short-period keys. It is strongly recommended that long-term key generation occur off-line in a manner isolated from the network via an air gap or, at a minimum, high-level secure hardware.

Encryption and Decryption
Signature Creation
Signature Verification

5. Conclusion

The DNS as an Internet standard to solve the issues of scalability surrounding the hosts.txt file. Since then, the widespread use of the DNS and its ability to resolve host names into IP addresses for both users and applications alike in a timely and fairly reliable manner, makes it a critical component of the Internet. The distributed management of the DNS and support for redundancy of DNS zones across multiple servers promotes its robust characteristics. However, the original DNS protocol specifications did not include security. Without security, the DNS is vulnerable to attacks stemming from cache poisoning techniques, client flooding, dynamic update vulnerabilities, information leakage, and compromise of a DNS server's authoritative files.

- In order to add security to the DNS to address these threats, the IETF added security extensions to the DNS, collectively known as DNSSEC. DNSSEC provides authentication and integrity to the DNS. With the exception of information leakage, these extensions address the majority of problems that make such attacks possible. Cache poisoning and client flooding attacks are mitigated with the addition of data origin authentication for RRSets as signatures are computed on the RRSets to provide proof of authenticity. Dynamic update vulnerabilities are mitigated with the addition of transaction and request authentication, providing the necessary assurance to DNS servers that the update is authentic. Even the threat from compromise of the DNS server's authoritative files is almost eliminated as the SIG RR are created using a zone's private key that is kept off-line as to assure key's integrity which in turn protects the zone file from tampering. Keeping a copy of the zone's master file off-line when the SIGs are generated takes that assurance one step further.

- DNSSEC can not provide protection against threats from information leakage. This is more of an issue of controlling access, which is beyond the scope of coverage for DNSSEC. Adequate protection against information leakage is already provided through such things as split DNS configuration.
- DNSSEC demonstrates some promising capability to protect the Internet infrastructure from DNS based attacks. DNSSEC has some fairly complicated issues surrounding its development, configuration, and management. Although the discussion of these issues is beyond the scope of this survey, they are documented in RFC 2535 and RFC 2541 and give some interesting insight into the inner design and functions of DNSSEC. In addition to keep the scope of this paper down, many topics such as secure zone transfer have been omitted but are part of the specifications in RFC 2535. The first official release of a DNSSEC implementation is available in BIND version 8.1.2.

6. References

1. Albitz, P. and Liu, C., (1997) ‘DNS and Bind’, 2nd Ed., Sebastopol, CA, O’Reilly &Associates, pp.1-9.

2.HerbertSchildt, Edition (2003) ‘The Complete Reference JAVA 2’ Tata McGraw Hill Publications

3. IETF DNSSEC WG, (1994) ‘DNS Security (dnssec) Charter’, IETF.

4. Michael Foley and Mark McCulley, Edition(2002) ‘JFC Unleashed’ Prentice-Hall India.

5. Mockapetris, P., (1987) ‘Domain Names - Concepts and Facilities’.