Defending against Denial of Service attacks using Secure Name Resolution[1].

Prashant Dewan and Partha Dasgupta
Arizona State University
dewan@asu.edu, partha@asu.edu

Vijay Karamcheti
New York University
vijay.karamcheti@asu.edu

Abstract

This paper proposes a technique to foil DoS (Denial of Service) attacks. The proposed technique converts a static service to a relocating service and provides information of the new location only to the specific pre-registered client groups while hiding it from others.

The Nameserver, responsible for advertising the address of the service, publishes only the encrypted address for the service. Only pre-registered clients get the key (needed for decryption), after being authenticated by the Key Server, which is entrusted with the distribution of key(s).

A DoS attack becomes difficult to execute, as the attacker does not know the precise location to attack. In addition, the proposed technique facilitates quick restoration of services in the event of an attack. We also show that this solution can be implemented with a low run time overhead.

***Index Terms*— DNS, DoS, Networks, Security**

1. Introduction

A "Denial of Service" attack prevents any legitimate users of the service from using that service. Such attacks include but are not confined to, flooding the network and thereby preventing legitimate network traffic; disrupting services to a specific machine or set of machines. DoS attacks can also be launched by altering the configuration of a service or by consuming the resources of a service in extraordinarily large amounts, thereby depriving the other users of those resources.

Although in principle it is possible to recover from a DoS attack by changing the location of the server, the benefits from doing so are limited by the fact that information about the new server location is available to both good and bad traffic. This approach i.e. relocating the server when it is attacked, was used by the *www.whitehouse.gov* web site when it was attacked last year. The fundamental reason for the ineffectiveness of the above-mentioned approach is the universal availability of the location information of a given service via the Name Servers.

The location of a service on the Internet is published by a hierarchical set of servers called the DNS [1,2]. The goal of DNS is to provide a consistent namespace, a low cost distributed mechanism for accessing and modifying this namespace, to support multiple protocols; and separation of DNS from the services, whose address is being advertised. DNS has been designed to be accessible to any host on the network; hence, its design neither incorporates security nor secrecy of the DNS service, nor it takes care of any potential misuse of the information that is provided by a DNS server.

Any site can hide its IP address by not even submitting it to the DNS server. Hiding the IP address will be an extremely expensive proposition, since the site will have to assume responsibility for distributing its IP to its clients besides having other servers whose addresses are publicly known. We do not explore this approach further.

Since the inception of DNS in 1984, many security extensions have been incorporated into DNS to protect it against various kinds of DNS attacks. A set of security protocols has been developed to protect the DNS servers from attacks. None of these protocols provides a method to disseminate DNS information to a group of clients, while hiding it from the clients that are not in the group. Though we discuss these protocols in Section 5.1, this paper does not focus on DNS attacks, but on DoS attacks. We use and enhance the DNS protocol to protect it against DoS attacks launched against services whose location is advertised by the DNS.

This paper advocates an approach for protecting the confidentiality of DNS name translations. In contrast to the conventional DNS, the DNS-like service proposed in this paper, stores the translation in such a form so that it can reveal useful information only to "good" clients, who have pre-registered with the service. The approach elucidated in this paper not only makes the DoS attacks difficult to be launched but it also mitigates the impact of these attacks. For example, if a service comes under attack; it changes its IP address. The new IP address is kept secret from the attacker, but made available to the legitimate users, through an automatic, transparent enhanced DNS protocol. Thus, the users do not perceive the long-term service as an outage due to the DoS attack. If the new location is compromised, then the service mutates again.

The architecture proposed in this paper, consists of four parts: the Client-Nameserver interaction, the protocol between the client and the Key Server, the protocol between the Key Server and the Nameserver and the registration protocol. The client registers with the Registration Server for a given service. In the Client-Nameserver protocol the client gets the encrypted resource record, which includes the IP of the service. The Nameserver transfers the key to

[1] This research is partially supported by grants from AFOSR, DARPA, Microsoft and NSF

the Key Server and the client obtains the key from the Key Server through a secure SSL channel.

The basic idea in the architecture is the decoupling of authentication and access. We present each of these components in detail and we describe an implementation of this architecture using the BIND 9 Nameserver. Experiments show that the run-time costs are low, particularly when factoring in the security provided by the enhanced protocol.

The technique proposed in this paper needs to be complemented by a detection mechanism, which should run on the server hosting the service. Besides, it needs a manual/automatic mechanism to change the location of a given service and the criteria on which the client groups can be formed.

We assume that a detection mechanism for detecting a DoS attack is already by being used by the service. The mechanism needed for relocating the service is left to the service administrator. It can be performed manually or automatically, by using scripts. The criteria for formation of the groups will vary across services and can be changed at run time. The approach proposed in this paper provides the functionality to plug in these criteria and change it dynamically.

The clients for a given service are divided into multiple sets and provided different access route to that service. As a result, only a small group of clients are affected in the event of a DoS attack (as only a subset of all routes gets blocked). The client group is reclassified into subsets periodically, and a large group of good clients is formed eventually. This reclassification of clients into groups facilitates optimized use of services. At the same time, any bad paths are throttled hence blocking any bad clients that might be suspected to be involved in a denial of service in the past.

For Example: Consider that www.cnn.com owns a set of IP address which includes A.B.C.D and W.X.Y.Z. Currently the web server is running on the IPv4 address W.X.Y.Z. An attacker comes to know about this address and launches a denial of service attack against this address. The Webmaster stops the server on W.X.Y.Z, moves the content to A.B.C.D and launches the web server. All the clients directed at W.X.Y.Z fail and after N requests, the resolver contacts the Key Server, which authenticates the client and gives the resolver the address A.B.C.D. The attacker is unaware of the new address and hence cannot launch an attack.

2. Background and Related Work

In the following subsections we present the motivation and the other work, which has been done in the past to counter similar problems.

2.1 Motivation

The earliest known DoS attack dates back to 1989, which lead to the crash of 10% of the Internet. Another class of attacks called the Distributed Denial of Service (DDoS) attacks rose furore after the infamous crash of the servers due to the SYN attacks in 1996 [11]. Since then, these attacks have been repeatedly executed with increasing degree of sophistication; thus, causing various levels of damage to the servers on the Internet.

The stumbling barrier against these attacks is that it is almost impossible to differentiate between the genuine and unscrupulous packets. The resources needed to execute such attacks can be easily downloaded over the Internet. In addition, the open source environment increases the availability of the operating system source code; hence eases spoofing of the source IP addresses. A spoofed IP address reduces the odds of an attacker being caught.

However, authentication of the source address in the IP protocol will resolve the problem of DoS attacks. On the flip side, a new protocol will introduce new security holes. In addition, it will be expensive to replace the omnipresent IP with a new protocol.

Various forms of Packet Filtering are currently being used to protect the hosts within the subnet from external intruders. In the absence of awareness of the strengths and weaknesses of this technique and any good monitoring tools, the packet filtering techniques do not provide foolproof security. Nevertheless, these techniques can only prevent external attackers from launching a DoS attack. Malicious attacks from attackers present within the subnet or cannot be foiled by packet filtering techniques.

The technique proposed in this paper uses cryptographic protocols to provide information about the location of a given host to a selected group of clients. In addition, it facilitates the easy re-advertisement of location information once the service is relocated. The proposed technique reduces the ease; and hence, the probability of a DoS attacks. This technique works in tandem with the protocols devised in the work done on Hardening of Routers in [6].

2.2 DNS Security

Since 1984 various security mechanisms have been developed to secure the DNS. These mechanisms focus on ensuring the integrity of the data exchanged between the DNS clients and the DNS server or between two DNS Servers. These mechanisms fall short of providing confidentiality to DNS transactions, which are needed for the implementation of the solution proposed in this paper. Some of these techniques are summarized in the following paragraphs.

DNSSEC (DNS Security Extension)[3] does not provide any confidentiality or any access control to give special privileges to a subset of the enquirers. It does not protect other services against DoS attacks. Secret Key Transaction Authentication for DNS (TSIG)[5] is an authentication protocol and does not provide selective access or secrecy of information. Internet Protocol Security (IPSEC) [4] offers protection to IP and/or upper layers. IPSEC cannot be used to ensure the secrecy of DNS messages because of the distributed nature of the DNS servers. DNS Views cannot be used here because the IP addresses of the prospective clients are unknown; hence, we cannot specify any access constraints on the basis of IP.

3. Architecture

The architecture uses the modified DNS servers and the modified client Resolver, the Key Server, Registration Server and the service, which the user wants to access. The client has to register with the registration server. The registration server stores the registration information for all clients in a database, which is shared by the Key Server. The DNS servers publish the encrypted location information for the service. A client can procure the key from the Key Server after being authenticated. The Key Server is an authenticating server, as explained in SIMS [7].

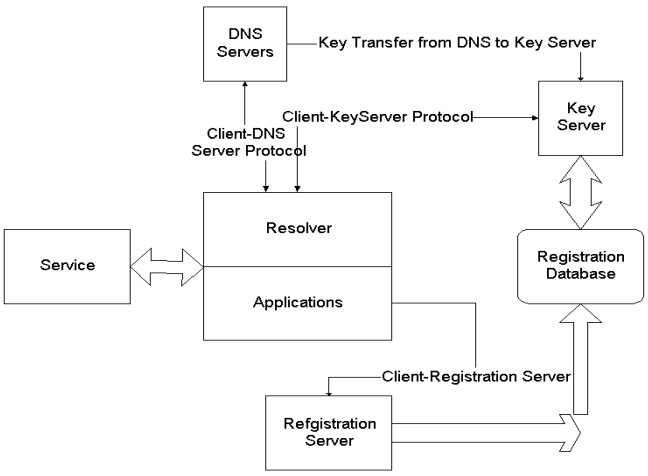

Figure 1. The proposed Architecture

3.1 Protocols

The architecture consists of four separate protocols: (1) Client-Nameserver, (2) Client-Registration Server, (3) Nameserver Key Server, and (4) Client-Key Server. These protocols are shown in Figure 1 and described in the following subsections.

3.1.1 Client-Nameserver

The client gets a set of encrypted records from the Nameserver. These encrypted records contain the name to IP mapping of the service. The Nameserver does not authenticate the client and gives the entire name to IP mappings to the client. The Resolver on the client machine caches the encrypted records. The client cannot use these encrypted records until it gets the key for them.

3.1.2 Client-Registration Server

This is the registration protocol. Any client, who needs to use the service, has to register itself at the Registration Server by providing its digital certificate. The server stores the public key of the client in its registration database. This key is used to uniquely identify the client before entrusting the client with the decryption key for the resource records.

3.1.3 Nameserver- Key server

The keys used to encrypt the location information are transferred from the Nameserver to the Key Server using out of band means. They might be transferred manually or using symmetric key encryption using a pre-decided key.

3.1.4 Client-Key Server

The client Resolver sets up a SSL connection with the Key Server. In this scenario, we assume that there are multiple Key Servers and that the client has a choice of setting up the connection with any key server it desires. The key server authenticates the client using the SSL protocol. Having authenticated the client, the Key Server verifies the public key of the client in the registration database. If the public key is verified, the Key Server sends the decryption key of one of the resource records and the resource record number to the client using an encrypted SSL channel. The client decrypts the resource record and sets up a connection with the service.

Although the client got several records from the DNS server, it can decrypt only one. The Key Server may deny the key to the client, there by blocking the client from accessing the service. The key server decides the group that this client should join and hence sends the decryption key for that group only. The group can be changed after the key expires or whenever the client requests for the key second time.

3.2 Overview

In the proposed solution, the IP address of a server is encrypted offline using symmetric key encryption. This cipher text is then wrapped into a record structure and included in the zone database file of the authoritative zone (the zone which is the owner of the resource record). The zone database file consists of the record structure (which wraps an encrypted location information) for each location of the service. The address of the Key Server is also wrapped in a record structure and appended to the zone database file. The key is transferred to the Key Server using out of band methods.

The Nameserver for the authoritative zone distributes the cipher text to the other Nameservers and/or the Resolver by using the standard DNS protocol. The other Nameservers (can) cache the cipher text; and are free to redistribute it without any authentication. An end node (client) can obtain the cipher text from its immediate Nameserver (which is also called the forwarding server). The end node does not need to register to get the cipher text. This policy is in conformance with the design philosophy of DNS.

Besides the cipher texts i.e. the set of records containing encrypted IPs, the pre-registered clients also get the address of the Key Server, which is responsible for the distribution of the decryption key. Only the clients who have pre-registered for the service can obtain the key by proving their identity using digital certificates over a (Secure Socket Layer) SSL connection to the key server. The key server validates the client's certificate. The key server also matches the client's public key in the registration database with the public key presented by the client for authentication. If the client passes the validation test, it is given the decryption key and the record number for a specific record. The client uses this key to decrypt the

cipher text to the IP.

The key received by the client determines the group to which it belongs. All the other clients who receive the same key automatically become members of that group until the TTL of the key expires. All the members of a group use the same location, i.e. the same IP address of the service to connect to the service. The key and the cipher text are only valid till their TTL is greater than zero. Once the TTL becomes zero, the client has to obtain the new key and sometimes a new cipher text.

4. DNS and Suggested Additions

DNS is the most widely used name resolution service on the World Wide Web. In addition, it has an open source implementation and has been documented extensively in various RFCs (Request for comments) and books [1, 2].

The following subsection summarizes the DNS protocol and enumerates the changes that we have implemented. The implementation of these changes is explained in Section 5.

4.1 DNS

The Resolver (DNS client) issues two kinds of queries to the immediate Nameserver. The queries can be recursive queries or non-recursive. [8] The answers provided by the authoritative Nameservers for a given zone supersede the answers provided by the delegated Nameservers from their caches for the same zone.

All the zone information needed by the DNS for *forward mapping* (Domain Name to IP) is included in a master file in the form of Resource Records. The DNS server reads this master file on startup and uses the information provided in the master file to answer the queries of the clients.

NAME (maximum 255 octets)
TYPE (2 octets)
CLASS (2 octets)
TTL (Signed 32 bit)
RDLENGTH (2 octets)
RDATA

Figure 2. Resource Record Structure

As illustrated in Figure 2, Resource Records are variable length data structures – with the variable *RDLENGTH* including the length of the *RDATA* field [2].

There are three types of resource records*: SOA records,* which give the information pertaining to the authoritative zones; the *NS records* which point to the Nameservers for a given zone and the *Other Records*, which consist of sixteen types and four classes, specified in the *Type* and *Class* fields respectively. The *IN* class encapsulates the entire Internet addresses and the *A* type records consist of the address of a given host. The *WKS* record is used for advertising **W**ell **K**nown **S**ervices.

4.2 Suggested Additions

A type records need their RDATA in a 32-bit format, because they are subsequently compressed for transmission. Hence, we cannot store the encrypted addresses in the A type records. In order to store the encrypted addresses, we need to add one more type of Resource Record to the existing types. We call this type CTXT.

In the current set of 16 basic types, two existing types can substitute CTXT. The TXT type can store strings of variable length, while the SIG type, which is not available in vanilla DNS, is a part of DNSSEC and stores cipher texts. Using TXT records to store encrypted texts is not a good idea, because it will destroy their original use, such as their ability to store arbitrary information about a given host. In order to use the SIG type, we will need to include the whole DNSSEC infrastructure. Such an inclusion will be overkill, as we only need an extra Resource Record type. Hence, we need to add CTXT to the existing types.

The Resolver will have a configuration file, which includes the URLs of the services that support the encrypted IP. This configuration file is only needed to increase efficiency of the name resolution by reducing the trips to the DNS server. This file will be configured by the user; and can be either user or machine specific, since it does not contain any classified information. This solution can be implemented without utilizing the additional configuration file, at the expense of an extra round trip to the Nameserver.

The Resolver is provided access to the digital certificate of this particular user. The resolver is modified in order to be able to set up an SSL channel with any key server. This capability can be too much work for stub Resolvers. However, this problem has been alleviated in BIND 9 Nameserver.

In addition, all clients should have a valid certificate at the time of registration and when they request for the retrieval of the key. A CA, whose public key is available with the Key Server, should sign the certificate.

5. Design and Implementation

5.1 Technologies Investigated

In an attempt to facilitate selective access to location of a given service to “good” clients, many existing technologies were investigated. The pros and cons of using these technologies for hiding the location of a service are explained in detail in the following paragraphs.

The advantages of using DNSSEC are that we will not need a separate key exchange protocol mainly because DNSSEC stores and distributes the public keys by using the resource record type KEY. It also provides tools for encrypting and signing the zone data files.

DNSSEC uses public key based encryption; hence, it is slow and inefficient. In addition, if the encryption key needs to be changed frequently, the use of PKI, on the fly encryption of the IP with the public keys of the clients will also be required. Hence, the cost of the operation will increase as the number of clients, increases. DNSSEC is designed to support the authentication of the server, but not the client. Even though, it can be modified to authenticate the client, it does not solve the problem completely.

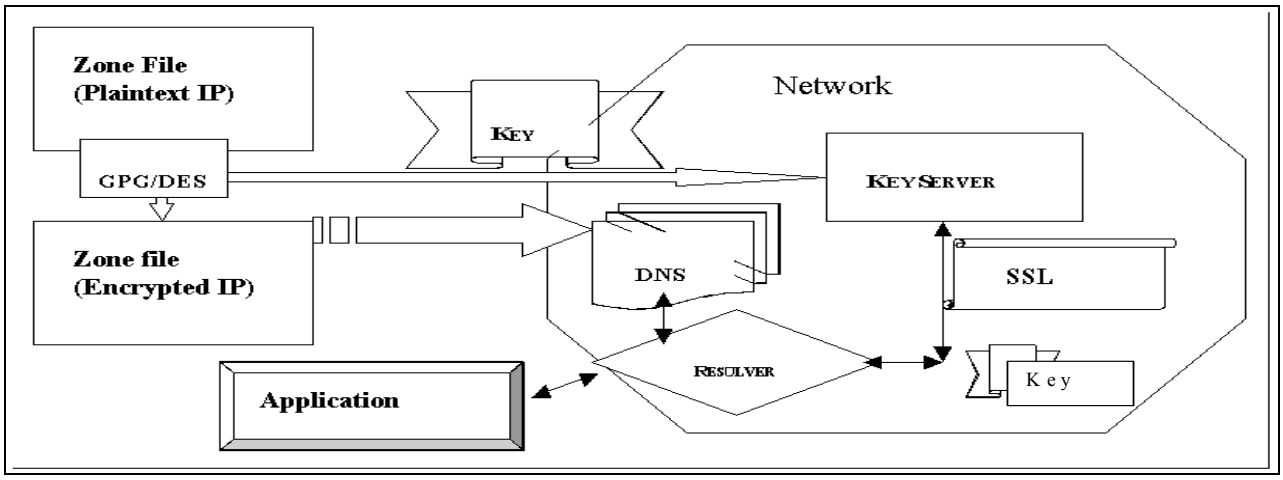

Figure 4. Detailed Architecture.

TSIG is faster than DNSSEC by the virtue of symmetric key encryption. It can be modified to incorporate confidentiality and authentication by changing the Resolver and/or the Nameserver(s). It needs a separate key distribution protocol, which can be taken from SIMS [7]. Hence, TSIG appears to be an interesting solution to explore.

In TSIG, the Nameserver of the authoritative zone will have to encrypt at least the answer section of the question asked by the Resolver or by any other Nameserver. Currently, the Nameserver in the authoritative zone simply signs the response**.** The caveat in this case is the fact, is the fact that if the encrypted text is sent directly to the end node, i.e. to a resolver, then the resolver can obtain the key and get the plain text. However, if another Nameserver receives the cipher text; it will not be able to decrypt it because it will not get the key. Since it cannot see the answer, it will not be able to cache it and will have to forward all the requests for the similar questions to the authoritative zone's Nameserver. This forwarding of queries, which could be answered by the Nameserver itself, defeats the idea of having a distributed DNS architecture. The encrypted response cannot be used again, even for the exactly same query, due to the fact that the timestamp specifies the exact time of signing the encrypted text that is returned by the authoritative Nameserver. This timestamp is included in the reply messages in order to prevent replay attacks.

IPSEC facilitates authentication and selective access; and at the same time, it keeps the details of the implementation away from the end user. In spite of all these positive points, IPSEC cannot be used for modeling the proposed solution since the information that we are trying to hide may be transferred across multiple nodes between its source and the destination. The intermediate nodes may need to cache this information. The nodes cannot cache the data without decrypting the packets and retrieving their payloads. The intermediate nodes are not supposed to have the key needed for decrypting the packets. If they decrypt the cipher text, then the idea of hiding the information is lost mainly because we cannot trust the intermediate nodes with the location information.

5.2 Implementation

The algorithm is divided into four mutually exclusive sequences of steps, which are described as follows.

5.2.1 Encryption of IP

This is performed offline. In the existing zone file, for each location of a service, the A Resource Record for a given host is retrieved, and the IP is encrypted.. This new resource record has the following fields:
<Name>: <Type=CTXT>: <Class=IN>: <Cipher text>.
The cipher text is the encrypted IP. A new resource record is inserted into the zone file for each location of the service. This record points to the address of the Key Server, which has the key for cipher text
<Name>: <Type=WKS>: <Class=IN>: <IP ADDRESS>.
The original *A* records*,* which encapsulated the locations (IP addresses) of the service, are deleted from the zone files. The zone files are brought online by uploading on to the Nameserver.

5.2.2 Request and Retrieval of Cipher Text

An application running on the user machine will make a request to the Resolver. The Resolver will check its configuration file to verify if the given URL has an encrypted IP. If the resolver finds the URL, then the query will be for a resource record of class CTXT. Upon receiving this query, the server will check if it has the requested record. If it does not have the record, then it will forward the query to the next pertinent (as explained in Section V) server (assuming that it is a recursive query). The server, which answers this query, will also append a WKS record in the answer section. This WKS record will include the address of the key server. After the resolver receives the response, it will cache the encrypted text. If the resolver does not find the URL in the file, then it will

request a Resource Record of class A or class ANY. Regardless of whether or not it receives the resolving record, it will still pursue its normal course of action.

5.2.3 Authentication and Retrieval of Key

After caching the encrypted text, the Resolver will setup an SSL channel with the Key Server. Once the initial handshake is completed, the Resolver will have to provide the user's digital certificate to the Key Server. The Key Server will verify the user's digital certificate, validate the public key on the user certificate against its registration database, and once the information verified, it will send the key to the Resolver, using the secure SSL channel.

The Key Server and the Registration Server share the registration database. The Key Server can place the client in any group by providing it the key of that group. The group for a client can be statically or dynamically configured in the registration database.

5.2.4 Decryption and Caching

The Resolver will decrypt the cipher text to retrieve the IP and destroy the key. It will cache the IP and the Cipher Text. This strategy will not work for multi-user machines because the users of such machines share the Resolver cache. The Resolver can tag the user id with the URL in order to associate this cached item with a particular user and hide this record from the other users. The disadvantage of this approach is that a super user can still manually sift the cache files and gain the IP.

Another possible approach that can be adopted to solve the cache problem is to give the user a choice to store the key in an area, which is user specific and not machine specific, and the super user is not given any rights onto this directory. This choice is more secure and effective than the others mentioned in this section.

As the key has a TTL any user can use the key only for the life span of the key. Once the key expires the Resolver will repeat the Client-Key Server protocol and get the new key. Hence the client might be placed in another group or the same group. The client-DNS server protocol will have to be repeated when the Resource Records for the cipher text expire.

6. Experiments and Performance Evaluation

6.1 Prototyping

The Nameserver is modified in order to service requests for CTXT records. In addition, it can now append the WKS record(s), which store the location of the Key Server

A SED script modifies the *zone db* file by using the Data Encryption Standard (DES) encryption, which is available with Gnu Privacy Guard (GPG).

As illustrated in Figure 4, a plain text (which is an IP in this case) and a passphrase are passed in the form of an input to the GPG encryption engine [10]. This engine churns the cipher text out. The script adds the new resource record of class CTXT to the zone db file. In addition, it adds the *WKS* record to the same file. It deletes the old class *A* resource record of the given domain and substitutes the RDATA section of the CTXT record with the cipher text.

The *RES_SEARCH* API in the BIND Resolver library is modified. The API reads the configuration file for the *HOSTNAME,* which is passed as the input parameter. If the *HOSTNAME* is present, then it changes the *CLASS* of the requested resource record to *CTXT,* and calls the original *RES_SEARCH*. We call it *orig_RES_SEARCH* because it is the original API used by the applications for name resolution. When *orig_RES_SEARCH* returns *RES_SEARCH* parses the records using *NS_INIT_PARSE* and *NS_PARSE_ERR*. *NS_INIT_PARSE* returns two resource records to the resolver. One record contains the encrypted text while the other contains the address of the Key Server. It uses the address provided in the WKS record, parsed by *NS_PARSE_ERR,* to setup an SSL connection with the Key Server. It uses the OpenSSL APIs to establish the initial connection.

The Key Server gets the certificate from the client. After the client is authenticated, it gets the key via the secure channel from the Key Server. This key is fed into the GPG engine along with the encrypted text that is returned by *NS_INIT_PARSE*. The decrypted text is the IP, which can be cached or used for one time – depending on the user preferences.

If the Resolver does not find the *HOSTNAME* in the configuration file, then it calls orig*_RES_SEARCH* with unmodified parameters and returns the value returned by orig*_RES_SEARCH.*

The Key Server is an openSSL server, which stores the encrypted key in the file system. Upon the client's request, it validates the client's certificate by using the public key of the certificate authority stored by the server. If the validation fails then it breaks the connection. Once the client is authenticated a secure channel is established and the server checks for the public key of the client against its registration database. Once the public key is verified, the server sends the decryption key to the client using the SSL channel. The Key server is implemented with openSSL APIs. This experiment was performed using 800 MHZ PENTIUM machines running Red Hat Linux with 100MBPS network.

6.2 Performance

This technique has been prototyped on a Linux Red Hat 7.1 machine using BIND 9.X, OPENSSL 0.9.6 and GPG 1.0.4. We have used the resolver libraries of BIND 8 as the resolver libraries of BIND 9.2.X are currently in beta stage. The Resolver libraries of version 8 and 9 of bind are compatible.

Using the above-mentioned technique, we found that the average time for a DNS resolution is 0.3 seconds more than the average time required for the standard DNS resolution. The majority of this additional time is consumed in the additional TCP connection that is made between the resolver and the Key Server and the SSL handshake. Pooling the SSL and TCP connections can further reduce the overhead time needed.

This technique allows unmodified applications to use the patched and more secure DNS protocol. Only the resolver on the client machine needs to be modified.

7. Discussion

This paper has examined techniques for allowing a server to communicate its location info to registered clients, while protecting it from others. While this provides a key mechanism for building defenses against DoS attacks, the system administrator should consider following possible vulnerabilities.

7.1 Sharing the Secret Key

If one of the authorized users i.e. a user who belongs to the subset of "good" clients, decides to share the secret key and the registration information with a "bad client", then this "bad client" can gain access to the location of the service; and hence, launch an attack**. "**This bad client" can use the compromised key, only till the TTL for the key is not zero. Once the TTL for the key becomes zero, the key will be rendered useless. The technique mentioned above will not foil any such kind of foul play.

7.2 IP Packets

When a client accesses a service, after obtaining its IP from the Nameserver, it will send the IP packets to that service. IP packets have the source and destination addresses of the packet; hence, any attacker can intercept them and find the address of the service. If this technique is used by itself, then it will be easy to parse the packets to get the IP. However, if this technique is used in a hardened network [6], where the communication between the border routers of two hardened Autonomous System (AS) is encrypted and inside an AS, the communication between the border router and the access router, which is directly connected to a service, is encrypted so that such an attack cannot be launched.

7.3 Guessing

Even after the implementation of the above-mentioned technique, an attacker can guess the IP, as the range of relocation for any service is finite. In the IPv4 scheme of the addresses, an attacker could guess the IP if he/she is determined enough. In IPv6, guessing could become very difficult, as the number of possible IP addresses is 2^{128}. The technique mentioned in the paper will support IPv4 and IPv6.

This vulnerability can be more difficult to exploit if adding port numbers are added to the resource records and the OS API is wrapped to enable the applications to connect to the user requested port numbers than the default port numbers used for a given protocol.

7.4 Attack on the Authorization infrastructure

It might be argued that a DDOS attack can be executed against the Key Server. We claim that such an attack can be mitigated by having a large number of Key Servers dispersed across the web like the Akamai [12] servers. Each Key Server is able to serve all the clients. This introduces the problem of sending the registration information and the new keys to the Key Server. As registration information can be disseminated once or twice a day, hence this technique is highly scalable. Only the information of the new registrants needs to be disseminated.

Keys may have to be disseminated to the Key Servers more frequently depending on the number of attacks experienced during a given time period. We can use the protocol provided in PayWord [13] to solve this problem as follows.

Select a number K_i and generate Ki_{+1} such that $K_{i+1} = F(K_i)$. Do this N times such that we get K_1 …Ki….Kn. Then distribute K_i and let all the Key servers generate K1...Kn. If a client is authenticated and does not produce any key it gets Kn. If the client produces K_i it gets K_{i-1}. There is no way that the client can guess K_{i-1} from K_i because the function F is a one-way hash function. In such a scenario keys do not have to be distributed.

8. Conclusion

By modifying the DNS protocol to add exclusive access to the set of good clients makes the task of a DoS attacker because more difficult. Clients can use this modified protocol without changing their applications and without adding a huge overhead on performance. We cannot use the existing mechanisms like DNSSEC, TSIG, and IPSec because of their inability to support access control in a distributed fashion. The technique proposed in this paper also promotes quick recovery from DoS attacks and eases the problem of finding the possible culprit by reducing the search space as the service can now identify the group to which this attacker belongs and hence can zero in on the attacker by eliminating changing the groups of the good clients.

References

[1] P. Mockapetris, "Domain Names – Concepts and Facilities," RFC1034, November 1987, Network Working Group.

[2] P. Mockapetris, "Domain Names – Implementation and Specification," RFC1035, November 1987, Network Working Group.

[3] D. Eastlake, "Domain Name System Security Extensions," RFC2535, March 1999, Network Working Group

[4] S. Kent, "Security Architecture for Internet Protocol" RFC2401, November 1998, Network Working Group

[5] P. Vixie, O. Gudmundsson, D. Eastlake, B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)" RFC2845, May 2000, Network Working Group

[6] S. Zhang, P. Dasgupta, "Hardened Networks," PhD Proposal, Arizona State University, Tempe, August 2002.

[7] K. Jiang and P. Dasgupta, "SIMS: A secure information management system for large scale dynamic coalitions", DISCEX-II, July 2001

[8] P. Albitz & C. Liu, April 2001"DNS and BIND", Fourth Edition

[9] V. Karamcheti, P. Dasgupta Hardened Routers and Mutable Services: Technology for Strengthening Network Infrastructures.

[10] B. Schneier, 2001 "Applied Cryptography", Second Edition.

[11] R. Farrow.2000. Distributed Denial of Service Attacks. Available from World Wide Web (http://www.networkmagazine.com/article/NMG20000512S0041)

[12] Akamai Corporation. Available from the world wide web at (www.akamai.com)

[13] R. Rivest and A. Shamir. *PayWord and MicroMint: Two simple micropayment schemes.* May 7, 1996. Available in the World Wide Web at (http://theory.lcs.mit.edu/~rivest/RivestShamir-mpay.ps).