Domain Name System (DNS) Security

By Diane Davidowicz

Contents

Domain Name System (DNS) Security

By Diane Davidowicz

1. Abstract

The Domain Name System (DNS) is vital to the Internet, providing a mechanism for resolving host names into Internet Protocol (IP) addresses. Insecure underlying protocols and lack of authentication and integrity checking of the information within the DNS threaten the proper functionality of the DNS. The Internet Engineering Task Force (IETF) is working on DNS security extensions to increase security within the DNS, known as DNSSEC. These security issues and solutions are presented in this paper.

2. Introduction

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.

This paper gives an overview of the DNS, its security weaknesses, and the new security extensions being worked on by the IETF's DNSSEC Working Group (WG).

3. 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 [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.

3.1. 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.

3.1.1. 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 [RFC 1034].

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.

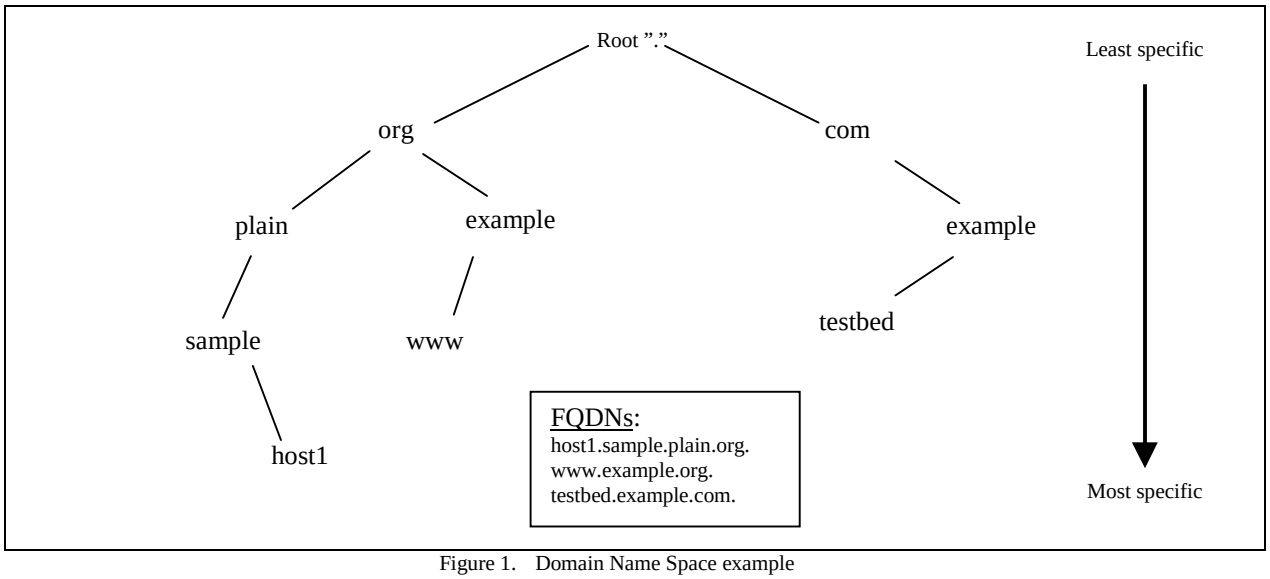

Figure 1. Domain Name Space example

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. 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.

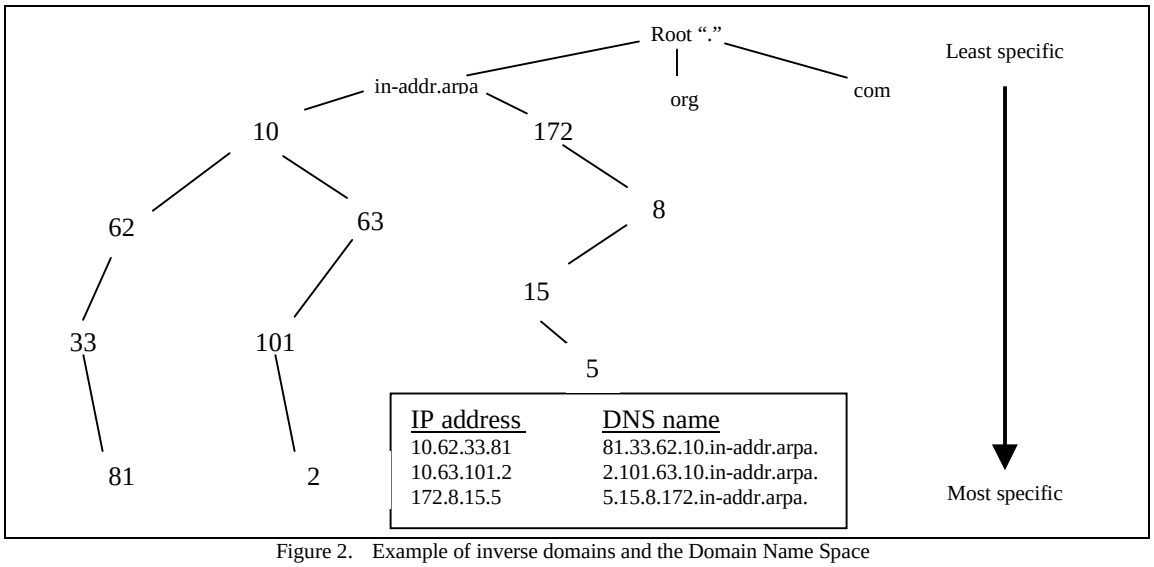

Figure 2. Example of inverse domains and the Domain Name Space

3.1.2. DNS Components

The DNS has three major components, the database, the server, and the client [RFC 1034]. 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 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 more common types of RR are found in Table 1.

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

Table 1. Common DNS Resource Records

The CLASS in this case is "IN" which stands for Internet. Other classes exist but are omitted for brevity. The TTL is the time, in seconds, that a name server can cache a RR. A zero time to live means that a server is not to cache the RR. RD Length is the RDATA field's length in octets. The RDATA field is the resource data field and is uniquely defined for each TYPE of RR, but in general it can be thought of as the value into which the entity specified in the NAME field maps. The NAME field can be thought of as the subject of a query, although this is not always the case, and the answer is the data contained in the RDATA field (even though the entire RR is returned in a DNS response) [RFC 1035].

RRs are grouped into resources records sets (RRSets). RRSets contain 0 or more RRs [RFC 2136] that have the same DNS name, class, and type, but the data (i.e., RDATA) is different. If the name, class, type, and data are the same for two or more records then duplicate records exist for the same DNS name. Name servers should suppress duplicate records [RFC 2181]. The Figure 3 shows an example of a RRSet.

```
example.com.    IN    NS    ns1.example.com.
example.com.    IN    NS    ns2.example.com.
example.com.    IN    NS    ns.plain.org.
```

Figure 3. These three records are grouped into a RRSet.

The client component of the DNS typically contains software routines, known as functions, which are responsible for requesting information from the Domain Name Space on behalf of an application. These functions are bundled together into a software library that is commonly referred to as the resolver library. For this reason, clients are often called resolvers. The resolver library functions are responsible for sending a query to a name server requesting information concerning a DNS name and returning the answer to the query back to the requestor.

3.2. 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 Question section contains the information concerning the object of the query. 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 [RFC 1035].

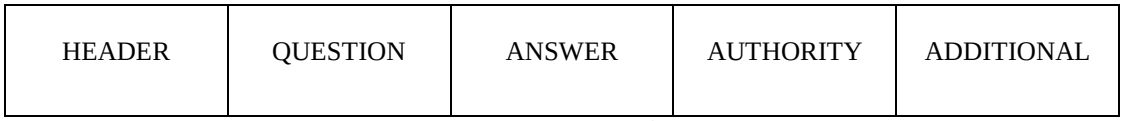

Figure 4. DNS message format

3.3. The BIND Implementation of DNS

The Berkeley Internet Name Daemon (BIND) is the most popular implementation of the DNS on the Internet [CA97]. 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.

4. 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.

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.

4.1. 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].

4.1.1. 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.

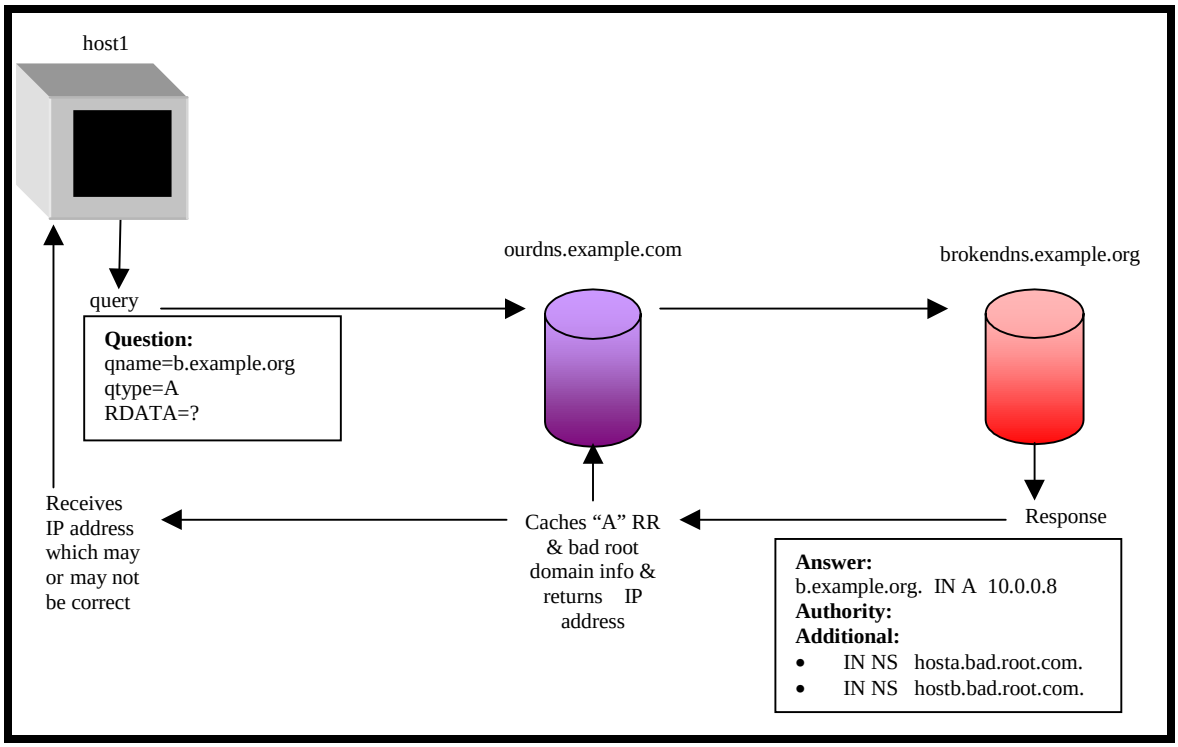

Figure 5. DNS Cache Poisoning

4.1.2. 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.

4.1.3. 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 just a helpful hint and get 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].

4.1.4. 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.

4.1.4.1. 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.

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* [CA98].

4.1.4.2. 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. This burdens the DNS with the responsibility of maintaining up to date and accurate information, neither of which the DNS alone can assure. 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.

This problem is demonstrated in the Figure 6. 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 [STEV2].

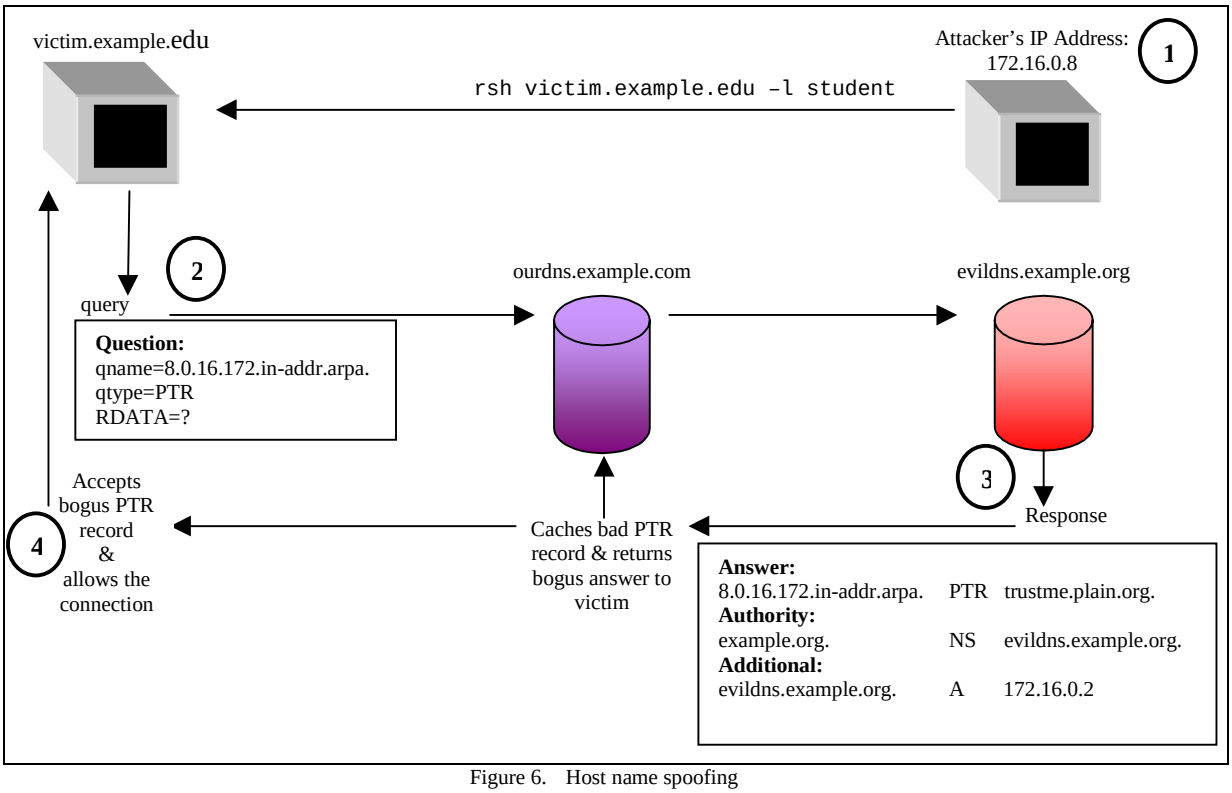

Figure 6. Host name spoofing

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*:

8.0.16.172.in-addr.arpa. IN PTR trustme.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 firewalls 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.

To assist in correcting this vulnerability, applications such as rlogind have been modified to perform a DNS sanity check [STEV2]. After retrieving the PTR record, another query is sent out requesting the "A" record using the FQDN specified in the answer section of the PTR record returned. Instead of burdening each application to perform a DNS cross check, many Unix operating systems now come with a similar capability that allows the DNS sanity checking to occur during a call to the gethostbyaddr() function in the resolver library [STEV]. This function typically is used to find a DNS host name using a known IP address (i.e., sends out a PTR query). The updated version of this function call does a gratuitous lookup for the corresponding "A" record and then makes sure the data in the RRs retrieved are correct verify each other. An example of DNS cross checking is shown in Figure 7.

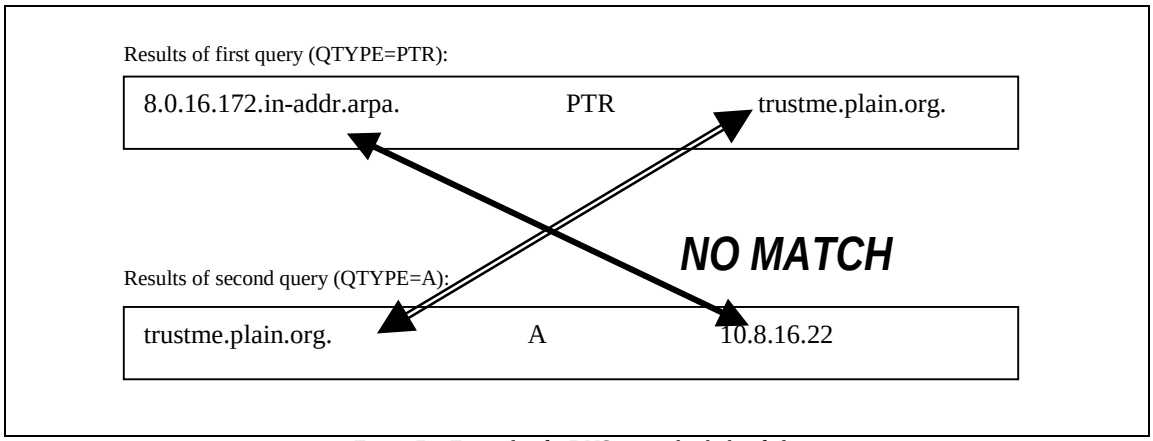

Figure 7. Example of a DNS cross check that fails

Since many resolver libraries have taken the necessary steps to prevent host name spoofing, the stakes have been raised for a successful attack using this methodology. Attackers can make use of DNS spoofing in conjunction with host name spoofing. The attacker configures the PTR RR with the trusted host name as explained above and then tries to inject into the victim's DNS server's cache the corresponding "A" record that maps the trusted host name back to the intruder's IP address [CHES]. As mentioned previously, earlier version of BIND are highly susceptible to this attack, while the more recent versions make the success of such attack more difficult, but not quite impossible. Thus, the integrity of the information contained within a DNS server's cache should remain suspect until stronger validation capabilities are available.

4.2. Client Flooding

Client flooding occurs when a client system sends out a query, but receives and accepts thousands of DNS responses from the attacker. The attack's success is based upon lack of authentication of these responses. The attack is made to appear as if it is originating from the expected name server, but without strong authentication, the client does not have the capability to verify the origin of the response [SPAF]. This attack can be used instead of DNS spoofing when attempting to host name spoof an application [CHES].

4.3. DNS Dynamic Update Vulnerabilities

RFC 1035 expects DNS zones to change slowly, thus it defines a static DNS where changes take place only in the zone files on the primary server and typically through a manual process. DNS Dynamic Updates is a modification to RFC 1035 that allows dynamic updating of DNS information contained within a zone as long as several prerequisites are met. Protocols such as Dynamic Host Configuration Protocol (DHCP) can then make use of DNS Dynamic Update protocol to add and delete RR on demand. These updates take place on the primary server for the zone. The updates take the form of additions and deletions [RFC 2136].

The DNS Dynamic Update protocol has provisions to control what systems are allowed to dynamically update a primary server. Even if it is employed, it is a weak form of access control and is vulnerable to threats such as IP spoofing of the system performing the updates or compromise of the system. An attacker, who is able to successfully accomplish either, can perform a variety of dynamic updating attacks against the primary server. They can range from denial of service attacks, such as the deletion of records, to malicious redirection, for instance, by changing IP address information for a RR being sent in an update [RFC 2137].

4.4. Information Leakage

Other threats to the DNS include zone transfers that can leak information concerning internal networks to a potential attacker. Frequently, host names can represent project names that may be of interest or reveal the operating system of a machine [CHAP]. Blocking zone transfers proves to be a futile effort in preventing such leaks of information. An intruder can make use of DNS tools to automatically query, one by one, every IP address in a domain space in an attempt to learn the DNS host name or to find IP addresses that are not assigned. The latter motive of uncovering unused IP addresses may allow an intruder to use IP spoofing to masquerade as a host of a trusted network. If a system trust an entire IP network, rather than specify every host that it trusts, then that system may be vulnerable to an attack using an unassigned IP address.

4.5. Compromise of DNS server's authoritative data

Other threats against DNS servers include the threat when an attacker gains administrative privileges (e.g., root on Unix systems) with the intent of modifying zone information for which the server is authoritative. This is achieved them through other vulnerabilities on the server not necessarily related to DNS. Careful configuration of the DNS server can provide some protection to these threats. Such things as using the latest version of BIND, minimizing the number of other services offered on the same machine, limiting access to just administrators, employing split DNS technology, etc. are crucial to a safer DNS service for an organization. There are many sources to assist in such configuration including [ACME], [CHAP], and any relevant CERT Advisories. Unfortunately, this does not provide strong protection against tampering of the data in the DNS files on the server. The appropriate security measures needed to provide adequate protection within the DNS could only be accomplished through DNSSEC.

5. DNSSEC [RFC 2535]

In 1994, the IETF formed a working group to provide security extensions to the DNS protocol in response to the security issues surrounding the DNS. These extensions are commonly referred to as DNSSEC extensions. These security enhancements to the protocol are designed to be interoperable with non-security aware implementations of DNS. The IETF achieved this by using the RR construct in the DNS that was purposely designed to be extensible. The WG defined a new set of RRs to hold the security information that provides strong security to DNS zones wishing to implement DNSSEC. These new RR types are used in conjunction with existing types of RRs. This allows answers to queries for DNS security information belonging to a zone that is protected by DNSSEC to be supported through non-security aware DNS servers.

In order to gain widespread acceptance, the IETF DNSSEC WG acknowledged that DNSSEC must provide backwards compatibly and must have the ability to co-exist with non-secure DNS implementations. This allows for sites to migrate to DNSSEC when ready and allows less complexity when upgrading. This also means that client side software that are not DNSSEC aware can still correctly process RRSets received from a DNSSEC server [CHAR].

In March of 1997, the Internet Architecture Board (IAB) met to discuss the development of an Internet security architecture. This meeting identified existing security mechanisms and those that are under development, but have not yet become standards, that can play a part in the security architecture. They also identified areas in which adequate protection using existing security tools could not be achieved. The results of this workshop include the identification of core security requirements for the Internet security architecture. Among those security protocols identified as core is DNSSEC. The protection that DNSSEC provides against injection of false cache information is crucial to the core security requirements of the Internet [RFC 2316].

5.1. DNSSEC Objectives

A fundamental principle of the DNS is that it is a public service. It requires correct and consistent responses to queries, but the data is considered public data. As such, the need for authentication and integrity exists, but not for access control and confidentiality. Thus, the objectives of DNSSEC are to provide authentication and integrity to the DNS. Authentication and integrity of information held within DNS zones is provided through the use of cryptographic signatures generated through the use of public key technology. Security aware servers, resolvers, and applications can then take advantage of this technology to assure that the information obtained from a security aware DNS server is authentic and has not been altered.

Although the DNSSEC WG chose not to provide confidentiality to DNS transactions, they did not eliminate the ability to provide support for confidentiality. Other applications outside of the DNS may choose to use the public keys contained within the DNS to provide confidentiality. Thus the DNS, in essence, can become a worldwide public key distribution mechanism. Issues such as cryptographic export are not, and may never be, solved worldwide; however, the DNS provides mechanisms to have multiple keys, each from a different cryptographic algorithm for a given DNS name, as a means to help alleviate this problem.

5.2. Performance Considerations

Performance issues are a concern for the security extensions to the DNS protocol and several aspects in the design of DNSSEC are targeted to avoid the overhead associated with processing the extensions. For instance, formulating another query that asks for the signature belonging to the RRSet just retrieved is not necessarily the most efficient way to retrieve a signature for the RRSet. This additional query is avoided whenever possible by allowing information retrieved from secured zones to be accompanied by the signature(s) and key(s) that validate the information.

5.3. DNSSEC Scope

The scope of the security extensions to the DNS can be summarized into three services: key distribution, data origin authentication, and transaction and request authentication.

5.3.1. Key Distribution

The key distribution service not only allows for the retrieval of the public key of a DNS name to verify the authenticity of the DNS zone data, but it also provides a mechanism through which any key associated with a DNS name can be used for purposes other than DNS. The public key distribution service supports several different types of keys and several different types of key algorithms.

5.3.2. Data Origin Authentication

Data origin authentication is the crux of the design of DNSSEC. It mitigates such threats as cache poisoning and zone data compromise on a DNS server. The RRSets within a zone are cryptographically signed thereby giving a high level of assuredness to resolvers and servers that the data just received can be trusted.

DNSSEC makes use of digital signature technology to sign DNS RRSet. The digital signature contains the encrypted hash of the RRSet. The hash is a cryptographic checksum of the data contained in the RRSet. The hash is signed (i.e., digitally encrypted) using a private key usually belonging to the originator of the information, known as the signer or the signing authority. The recipient of the RRSet can then check the digital signature against the data in the RRSet just received. The recipient does this by first decrypting the

digital signature using the public key of the signer to obtain the original hash of the data. Then the recipient computes its own hash on the RRSet data using the same cryptographic checksum algorithm, and compares the results of the hash found in the digital signature against the hash just computed. If the two hash values match, the data has integrity and the origin of the data is authentic [CHAR].

5.3.3. DNS Transaction and Request Authentication

DNS transaction and request authentication provides the ability to authenticate DNS requests and DNS message headers. This guarantees that the answer is in response to the original query and that the response came from the server for which the query was intended. Providing the assurance for both is done in one step. Part of the information, returned in a response to a query from a security aware server, is a signature. This signature is produced from the concatenation of the query and the response. This allows a security aware resolver to perform any necessary verification concerning the transaction.

Another use of transaction and request authentication is for DNS Dynamic Updates. Without DNSSEC, DNS Dynamic Update does not provide a mechanism that prohibits any system with access to a DNS authoritative server from updating zone information. In order to provide security for such modifications, Secure DNS Dynamic Update incorporates DNSSEC to provide strong authentication for systems allowed to dynamically manipulate DNS zone information on the primary server [RFC 2137].

5.4. DNSSEC Resource Records

The IETF created several new DNS RRs to support the security capabilities provided by DNSSEC extensions. The RRs pertinent to the DNS are the KEY RR, SIG RR, and the NXT RR. DNSSEC utilizes the KEY RR for storing cryptographic public keys, one public key per KEY RR. It is the KEY RR that is used for verification of a DNS RRSet's signature. The signature for a RRSet is stored in the SIG RR. The signature is used to prove the authenticity and integrity of the information contained in the RRSet. The NXT RR is the nonexistent RR and is used to cryptographically assert the nonexistence of a RRSet. Another RR exists, known as the CERT RR, that does not bring any additional security functions to the DNS, but is provided so that public key certificates can be kept within the DNS for use in applications outside of the DNS [RFC 2538]. In much the same way an application wishing to communicate with a remote IP host generates an A query to resolve the host name, a security application wishing to perform encryption with another entity generates a CERT query to retrieve the entity's public key certificate. For further explanation on KEY, SIG, and NXT RRs and their RDATA fields and flags not contained herein, please reference RFC 2535 and related documents.

5.4.1. KEY RR

The key for a DNS name is held in a KEY RR. Any type of query for a DNS name, found in a secured zone, results in a response that contains the answer to the query. The KEY RR associated with the DNS name can accompany this response. The resolver that generated the query can then validate the data using the KEY RR without having to send another query for the Key RR. This minimizes the number of queries needed for any given DNS name found in a secured zone.

DNSSEC utilizes the KEY RR for storing cryptographic public keys; however, this is not a public key certificate. Instead, the CERT RR is used to store public key certificates. The key found in the RDATA section of the KEY RR belongs to the DNS name first listed in the KEY RR (i.e., the owner name). The owner name can represent a zone, a host, a user, et al.

The Key RR contains information denoting the security characteristics of the key and its allowed usage for the given owner name. It provides security information such as the public key, algorithm type, protocol type, and flags that specify such things as to whether or not the DNS name has a public key. The public key algorithm determines the actual format of the public key found in the RDATA section of the KEY RR. Several key algorithms are already supported and are defined in RFC 2535 as RSA/MD5, Diffie-Hellman,

and Digital Signature Algorithm (DSA), and the elliptic curve algorithm. Only DSA support is mandatory. Another field is known as the protocol octet. It indicates for which protocol the public key is valid. Some already assigned protocols are TLS, email, DNSSEC, and IPsec. Since both the public key algorithm field and the protocol octet is an 8-bit field, theoretically up to 255 different algorithms and 255 different protocols can be used in conjunction with the public key.

Two bits out of the sixteen bits used for setting various flags are known as the type bits. All four combinations of the type bits indicate how the KEY RR can be used. They are confidentiality, authentication, confidentiality and authentication, or none. The latter indicates a key does not exist for the DNS name. In this way, one can cryptographically assert that the given owner name does not possess a key even though it is in a secure zone. Another two bits are used to identify three kinds of entities for which this key belongs, such as user, zone, or something that is not a zone. Indicating a host with these flags is actually done by using the flags to indicate that the DNS name is not a zone. Thus a host is implied rather than specified by the flags.

5.4.2. SIG RR

A signature is held in another resource record type known as a SIG RR. The SIG RR provides authentication for a RRSet and the signature's validity time. In a secure zone, a RRSet has one or more SIG RR associated with it. The situation of having more than one SIG RR for a given RRSet may arise when more than one cryptographic algorithm is being used for signing the RRSet. Some sites may choose to do this for issues such as cryptographic export restrictions.

A number of fields are also found in the RDATA section of a SIG RR. The signature field holds the signature belonging to a specific RRSet. To indicate the RR type of the RRSet (i.e., NS, PTR, MX, etc.), a "type covered" field is used. In order to verify the signature, a resolver or server must know the signer's name. This is specified in the signer's field. The SIG RR has an algorithm field identical to that in the KEY RR. Since signatures have expiration times, as do individual RRs, the SIG RR has several time fields. This is further discussed later in this paper, [see "Security Aware Servers"].

Except for the SIG RRs used for transaction and request authentication and for the SIG RRs which are specifically the target of a query, security aware servers try to include in the response the SIG RRs needed to authenticate the RRSet. Thus, a resolver may still receive an answer for a RRSet belonging to a secure zone that does not have the SIG RR. This situation can typically occur when a size limitation is exceeded due to the SIG RR or when the response comes from a non-security aware server. Under these circumstances, the security aware resolver is required to form another query specifically requesting any missing SIG RRs needed to complete the verification process.

5.4.3. NXT RR

The DNS provides the ability to cache negative responses. A negative response means that a corresponding RRSet does not exist for the query. DNSSEC provides signatures for these nonexistent RRSets so that their nonexistence in a zone can be authenticated. It does this through the use of the NXT RR. NXT RRs are used to indicate a range of DNS names that are unavailable or a range of RR types that are unavailable for an existing DNS name.

Two possibilities exist for nonexistent DNS names. One is that the DNS name itself does not have any RRs; it simply does not exist. The other is that the DNS name does exist (i.e., has at least one type of RR), but the RR type in the query for that name does not exist. To handle proof of nonexistence of a DNS name, all the records in a zone are sorted in a manner that is similar in some ways to alphabetical order. The technique used is known as canonical order and is defined in RFC 2535. Then when a query is received for a nonexistent name, a NXT RR is sent back containing the DNS name of the next DNS RRSet occurring "alphabetically", or rather canonically, after the name in the query. To handle proof of nonexistence of a RR type for an existing DNS name, a NXT record is sent back with the DNS name and the RR types that the name does in fact have. Whenever SIG RRs are generated for a zone, all NXT RRs for a zone should be generated.

5.5. Security Aware DNS Servers

Security aware DNS servers are the source of all security-related information within the DNS. Any given primary DNS server has three main functions: manage authoritative zone information, manage the caching of DNS information, and respond to client queries. A primary DNS server that is security aware has added responsibilities to each of these functions. Authoritative zone information management for a security aware server includes the addition of SIG, KEY, and NXT RRs in a zone's master database file. The SIG RRs are generated for the RRSets belonging to a zone. The private key used to generate the SIG belongs to the zone itself. Since private keys of servers are more than likely found on-line, it is possible that these keys could be compromised. The zone's private key, in contrast, is kept off-line for most purposes, so its compromise is less likely and the validity of the data is more assured. The zone's private key is retrieved periodically to re-sign all the records found within the zone. Once the new SIG RRs are generated they are included with the rest of the information in the zone's master file. NXT RRs also should be generated on the server and placed into a zone's master file whenever SIG RRs are generated.

On-line signing also takes place at the server. For transaction and request authentication for DNS queries, the server formulating the reply must use its private key for signing, rather than the zone key since it is kept off-line. Another instance in which a zone key is not used for signing is for transaction and request authentication for dynamic updates. The private key of the host formulating the request must be used. Because DNS queries and dynamic update requests can occur quite frequently, the signer's private keys must be maintained on-line. The protection of these on-line private keys is of utmost importance; however, the means in which they are protected is beyond the scope of this paper. RFC 2541 discusses the operational considerations of KEY and SIG RR.

To perform caching, a security aware server must properly manage the caching of all security related RRs. The additional responsibility in caching of a security aware server begins with the maintaining of four cache states. One state, which has a corresponding state in a non-security aware server, is "Bad". In a non-security aware server, when a bad response is received in that the information contained is in some way corrupt, a non-security aware server throws away the response message without caching it (and typically logs the event). In much the same way, a security aware server can throw away a bad response, but in this case, a bad response means that the SIG RR verifications failed on the data. Even though the RRSet in the response may look legitimate, the failing of the data checks with the corresponding signature is a fatal condition.

The three other states are Insecure, Authenticated, and Pending. Insecure means that there isn't any available data to use to check the authenticity of the RRSet. It does not mean the data is bad, just that it cannot be authenticated. This commonly occurs for RRSets from non-secured zones. Authenticated means the RRSet cached has been fully validated through the use of the SIG RRs and KEY RRs. Pending means the cached data is still in the process of being checked.

Another server responsibility with caching is when to expire a cached RRSet. Once a RRSet is cached, a count down to zero from the original TTL is started and maintained for the cached record. Once zero is reached, the RRSet is removed from the cache. For security aware servers, this has changed a little. The TTL cannot be the only time kept to determine when a cached RRSet is expired. Two new times are now used in addition to the TTL and these ultimately determine when to expire the RRSet from the cache. The new times are used to determine when the signature's validity time period for the authenticated RRSet expires, rather than just when the RRSet should be expired. These new times are kept in the SIG RR and are known as the signature inception time and the signature expiration time. For security aware clients and server this information is far more important on which to base expiration since it is cryptographically asserted. Although the signature expiration time seems have a correlation to the TTL, due to backward compatibility issues, the TTL field cannot be eliminated.

TTL aging is still incorporated for expiring authenticated RRSets. If the TTL expires prior to the signature expiration time, the TTL is decremented as normal and the RRSet is expired when the TTL hits zero. If the signature expiration time occurs prior to when the TTL expires, the TTL is adjusted to the signature expiration time and then the normal countdown of the TTL proceeds.

Responding to client queries now involves answering queries from both security aware and security unaware resolvers. When a non security aware resolver generates a query and sends it to a security aware server for information contained in a secured zone, the security aware servers can respond with either Authenticated or Insecure data. A security aware server can only send Pending data when the checking disabled (CD) flag is set. The security aware server knows not to send Pending data because a resolver not participating in DNSSEC never sets the CD flag in a DNS query. Since sending Insecure data is the same as DNS without DNSSEC, the security unaware resolver processes the response message as usual. As far as receiving Authenticated data, the security unaware resolver basically ignores the additional security information and goes about processing the response as usual.

When queries are originating from security aware resolvers, it is strongly encouraged that the resolver set the CD flag. With the CD flag set in the query, security aware servers can send the Pending data. Sending Pending data accomplishes two things. It minimizes the response time freeing up server resources for handling queries and it allows a resolver to implement its policies on Pending data, independent of servers. If the answer to the query is already Authenticated data on the server, the server sets the authentic data flag (AD) to indicate to the resolver that the necessary checks have already been performed. In this way the resolver does not need to do any security verification checks.

5.6. Security Aware Clients

Security aware clients also have added responsibilities then their non-secure counterparts. These added responsibilities come in the form of knowing how to process DNSSEC RRs. Figure 8 illustrates a DNSSEC transaction invoked by a security aware resolver to a security aware server:

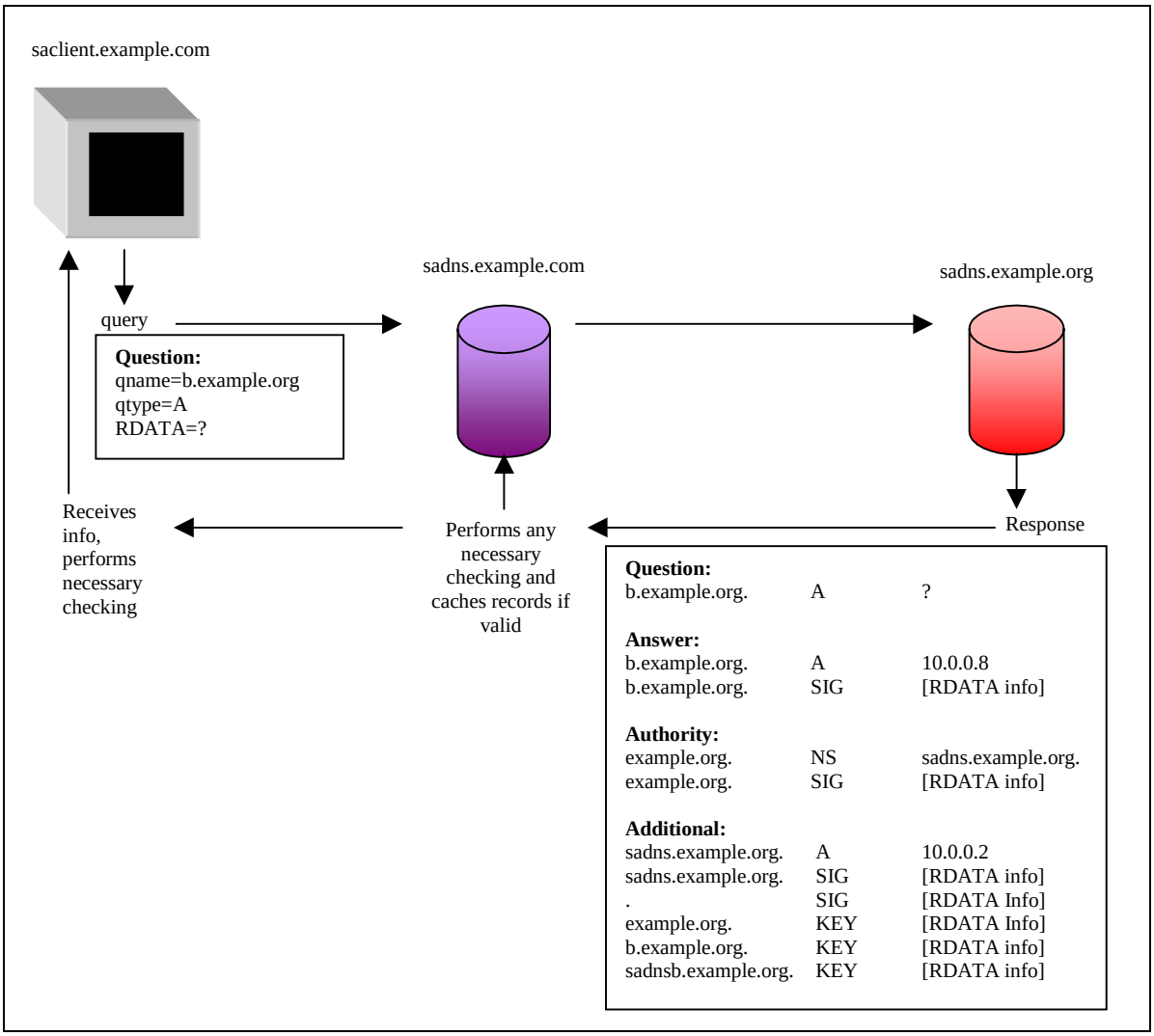

Figure 8. DNSSEC query & response messages

In its most simple format, a security aware client, *saclient.example.com*, queries for a RR of type A for the DNS name *b.example.org*. The security aware name server, *sadns.example.com*, sends the query onto *sadns.example.com*. The zone, *example.org*, is a secured zone and lets assume for this example only, that there isn't a size constraint on the response message. Since *b.example.org* does have an A record and *sadns.example.org* is a security aware server, the response comes back with all the necessary DNSSEC RR.

First note that there is a SIG RR being returned along with the A RR in the Answer section of the response message. RFC 2535 specifies that security aware servers should attempt to return the SIG RR needed in a secure transaction. The inclusion of the SIG RR is not a requirement because if the size limit of the response message is exceeded when SIG RR is added, the SIG RR should be omitted, but if this happens, the client must consider the response truncated. The client then queries for the SIG RR separately. Since this example doesn't have space considerations, it is included here. This SIG RR is required in the answer section because it provides the necessary authentication and proof of integrity of the data specifically pertaining to the A RR for b.example.org. The signer of the SIG RR is the zone (i.e., the private key of the zone was used to sign for the data).

The NS record in the Authority section appears there because sadns.example.org is the only DNS server for *example.org* and is authoritative for the information in the Answer section. As with the SIG RR appearing

in the Answer section, this SIG RR is required to appear in this section as it contains the signature for the NS record. The signer of the SIG RR is again the zone.

The second "A" record is the address record belonging to *sadns.example.org* and it appears in the Additional section as it would in a normal DNS response message. Accompanying the A record is its respective SIG RR so that the client can also authenticate this A RR. The only case in which the client is not to consider a response truncated is when the omitted SIG RR belongs to a RR in the Additional section. Such is the case of this SIG RR. In other words, if the response packet could not fit this SIG RR without going over the size limit, the SIG RR is omitted, but the response is not considered truncated.

Also appearing in the Additional section is the fourth SIG RR. Since the client is security aware, it has requested transaction and request authentication. As a result, *sadns.example.org* creates a SIG RR by computing the signature of the concatenation of the query and response DNS messages. Note that the owner name of this SIG resource record is "." to represent the zero length label reserved for root. The owner name, class, and TTL of this type of SIG RR do not have significance to the transaction and request authentication process and thus it is recommended that "." be used to conserve space. The signer field, however, of this SIG RR is significant to the verification process and it must be the server name generating the response. This field is found in the RDATA section. Since the client requested transaction and request authentication, a security aware server must not omit this SIG RR. Its is considered of highest priority and its inclusion is mandatory.

Three KEY RRs appear in the Additional section in the response message. The first KEY RR belongs to the zone *example.org*. The client can use this key to verify all SIG RRs except for the last one pertaining to the transaction and request authentication. The server includes two other KEY RRs as a result of the A records appearing in the Answer section and the Additional section. These are the host KEY RR for *b.example.org* and *sadns.example.org*. In a real world, size constraints do apply to response messages. If this response message is larger than the limit, then the KEY RRs are the first to be left out of the message. In the Additional information section, A RRs have a higher priority than KEY RRs. In general, SIG RRs are of higher priority than KEY RRs and, therefore, the server omits KEY RRs before omitting any SIG RRs. The one exception to this is when the query type is for a KEY RR. Being the subject type of the query, the server must include it if it exists.

5.6.1. Public Key Retrieval

Resolvers can obtain public keys of zones in one of two ways. Resolvers can utilize the DNS to query for the public key or they can be statically configured with the key. Regardless of the method used, problems exist with both. In the case where keys are obtained through the DNS, the issue of trusting the key arises. In order to trust the retrieved key, it must be signed and this signature must be reliable. Providing the assurance that the signature on the key is reliable means that the public key of the signing authority must also be obtained, be signed, and be found reliable and so on. The solution to ending this recursive chain of events is to configure the resolver with the public key that authenticates the signed keys below it. In other words, a trusted zone key can be used as a starting point for verifying all keys found below it. A likely set of trusted public keys with which a secure zone can be statically configured are those of the root zone.

The static configuration of a resolver with the public keys from many different zones has one advantage in that the compromise of one of the zone's private key does not result in the compromise of the keys for all the other zones. The disadvantage of statically configuring each resolver with keys for many different zones is that it does not scale well. If one key for one zone must change, then all the resolvers must be configured to reflect this change.

6. Summary

In 1987, The IETF ratified 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.

References

[ACME] Larson, M. and Liu, C.,"Using BIND: Don't get spoofed again", SunWorld, November 1997. (URL: http://www.sunworld.com/swol-11-1997/swol-11-bind.html)

[ALIU] Albitz, P. and Liu, C., "DNS and Bind", 2nd Ed., Sebastopol, CA, O'Reilly & Associates, January 1997, 1-9.

[CA97] Computer Emergency Response Team, "CERT* Advisory CA-97.22", "Topic: BIND - the Berkeley Internet Name Daemon", CERT, January 1997. (URL: http://www.cert.org/advisories/CA-97.22.bind.html)

[CA98] Computer Emergency Response Team, "CERT* Advisory CA-98.05", "Topic: Multiple Vulnerabilities in BIND", CERT, November 1998. (URL: http://www.cert.org/advisories/CA-98.05.bind_problems.html)

[CHAP] Chapman, D. B. and Zwicky, E. D., "Building Internet Firewalls", Sebastopol, CA, O'Reilly & Associates, Inc., 1995, 278-296.

[CHAR] IETF DNSSEC WG, "DNS Security (dnssec) Charter", IETF, 1994. (URL: http://www.ietf.cnri.reston.va.us/proceedings/94dec/charters/dnssec-charter.html)

[CHES] Cheswick, W. R. and Bellovin, S. M., "Firewalls and Internet Security", Reading Ma, Addison-Wesley, 1994, 28.

[MENM] "What is DNS Spoofing", Men & Mice, 1999.
(URL: http://www.menandmice.com/infobase\mennmys\vefsidur.nsf/index/6.2.1.1)

[RAFT] Rafter, M., "Kashpureff, AlterNIC Founder, Busted for Hijacking DNS Traffic", Internet World, November 10, 1997.
(URL: http://www.internetworld.com/print/1997/11/10/news/19971110-busted.html)

[RFC 1034] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987.
(URL: ftp://ftp.isi.edu/in-notes/rfc1034.txt)

[RFC 1035] Mockapetris, P., "Domain Names - Implementation and Specifications", STD 13, RFC 1035, November 1987.
(URL: ftp://ftp.isi.edu/in-notes/rfc1035.txt)

[RFC 2136] Vixie, P., Thomson, S., Rekhter, Y. and J. Bound,"Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.
(URL: ftp://ftp.isi.edu/in-notes/rfc2136.txt)

[RFC 2137] Eastlake, D., "Secure Domain Name System Dynamic Update", RFC 2137, April 1997.
(URL: ftp://ftp.isi.edu/in-notes/rfc2137.txt)

[RFC 2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, July 1997.
(URL: ftp://ftp.isi.edu/in-notes/rfc2181.txt)

[RFC 2316] Bellovin, S., "Report of the IAB Security Architecture", RFC 2316, April 1997.
(URL: ftp://ftp.isi.edu/in-notes/rfc2316.txt)

[RFC 2535] Eastlake, D., "Domain Name System Security Extensions", RFC 2535, March 1999.
(URL: ftp://ftp.isi.edu/in-notes/rfc2535.txt)

[RFC 2538] Eastlake, D. and O. Gudmundsson, "Storing Certificates in the Domain Name System", RFC 2538, March 1999.
(URL: ftp://ftp.isi.edu/in-notes/rfc2538.txt)

[RFC 2541] Eastlake, D., "DNS Operational Security Considerations", RFC 2541, March 1999.
(URL: ftp://ftp.isi.edu/in-notes/rfc2541.txt)

[SPAF] Garfinkel, S. and Spafford, G., "Practical Unix & Internet Security", 2nd Ed., Sebastopol, CA, O'Reilly & Associates, January 1997, 473-475.

[STEV] Stevens, W. R., "TCP/IP Illustrated, Volumne 1", Reading , Ma, Addison Wesley, 1994, 200.

[STEV2] Stevens, W. R., "Unix Network Programming", Englewood Cliffs, NJ, Prentice Hall, 1990, 573-584 esp. 578-579, 646-665 esp. 650.

[VENE] Venema, W., "README" file, tcp_wrappers archive, March 1997.
(URL: ftp://coast.cs.purdue.edu/pub/tools/unix/tcp_wrappers/tcp_wrappers_7.6.tar.gz)