THEME 1

DNS Security

Telecommunication systems department

Lecturer: assistant professor Persikov Anatoliy Valentinovich

INTERNET

The Internet is the world's largest computing network, with more than 580 million users. From the perspective of a user, each node or resource on this network is identified by a unique name: the domain name. Some examples of Internet resources are:

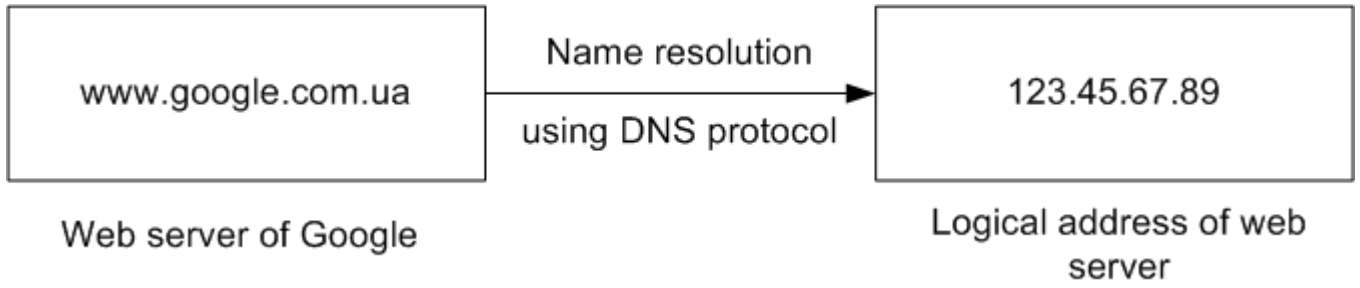
- **Web servers –** for accessing Web sites;
- **Mail servers –** for delivering e-mail messages;
- **Application servers –** for accessing software systems and databases remotely.

From the perspective of network equipment (e.g., routers) that routes communication packets across the Internet, however, the unique resource identifier is the Internet Protocol (IP) address, represented as a series of four numbers separated by dots (e.g., 123.67.43.254). To access Internet resources by user-friendly domain names rather than these IP addresses, users need a system that translates these domain names to IP addresses and back.

This translation is the primary task of an engine called the **Domain Name System** (DNS).

NEED FOR DOMAIN NAME SYSTEM

Users access an Internet resource (e.g., a Web server) through the corresponding client or user program (e.g., a **Web browser**) by typing the domain name. To contact the Web server and retrieve the appropriate Web page, the browser needs the corresponding IP address. It calls DNS to provide this information. This function of mapping domain names to IP addresses is called **name resolution**. The protocol that DNS uses to perform the name resolution function is called the **DNS protocol**.



BUILDING BLOCKS OF DOMAIN NAME SYSTEM

This DNS function includes the following building blocks.

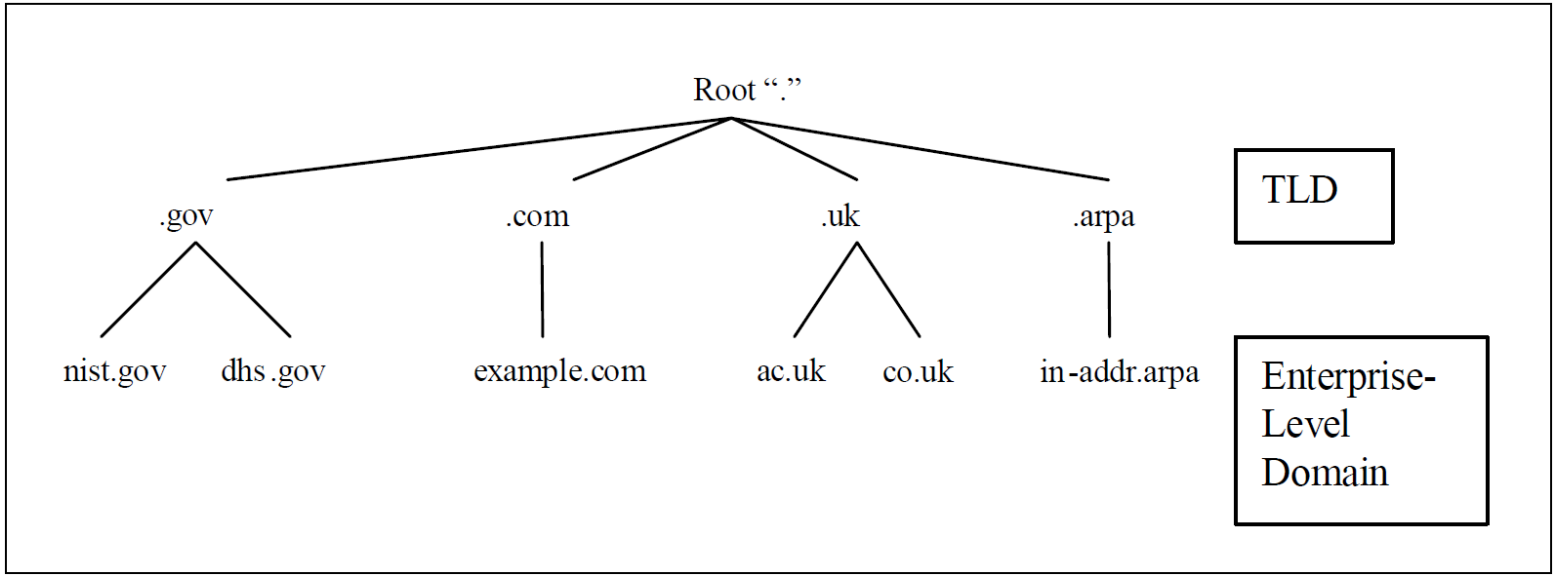
First, DNS should have a **data repository** to store the domain names and their associated IP addresses. Because the number of domain names is large, scalability and performance considerations dictate that it should be **distributed**. The domain names may even need to be replicated to provide fault tolerance.

Second, there should be **software** that manages this repository and provides the name resolution function. These two functions (managing the domain names repository and providing name resolution service) are provided by the primary DNS component, the **name server**. There are many categories of name servers, distinguished by type of data served and functions performed. To access the services provided by a DNS name server on behalf of user programs, there is another component of DNS called the **resolver**. There are two primary categories of resolvers (caching/recursive/resolving name server and stub (non-caching) resolver), distinguished by functionality.

DNS INFRASTRUCTURE

The DNS infrastructure is made up of computing and communication entities that are geographically distributed throughout the world.

The domain name space (the universe of all domain names) is organized in the form of a hierarchy. The topmost level in the hierarchy is the **root domain**, which is represented as a dot (“.”). The next level in the hierarchy is called the **top-level domain** (TLD). There is only one root domain, but there are many TLDs. Each TLD is called a child domain of the root domain. In this context, the root domain is the parent domain because it is one level above a TLD. Each TLD, in turn, can have many child domains. The children of TLDs are called **second-level** or **enterprise-level domains**.



FULLY QUALIFIED DOMAIN NAME

In a domain name representation, the symbol for the root domain usually is omitted. For example, consider the domain name marketing.example.com. The rightmost label in this domain name (".com") is a TLD. The next label to the left ("example") is the second-level or enterprise-level domain. The leftmost label ("marketing") is the third-level domain. It also is possible to have a fourth-level domain, fifth-level domain, and so forth. Because each of the labels in marketing.example.com is called a domain (TLD, second-level domain, third-level domain, etc.), the concatenation of all these labels from the current level to the TLD is a **fully qualified domain name** (FQDN).

There is only one root domain. There are more than 250 TLDs, categorized into the following three types:

- **Country-code TLDs (ccTLDs) –** domains associated with countries and territories. There are more than 240 ccTLDs. Examples include .uk, .in, and .jp.
- **Sponsored generic TLDs (gTLDs) –** specialized domains with a sponsor representing a community of interest. These TLDs include .edu, .gov, .int, .mil, .aero, .coop, and .museum.
- **Unsponsored generic TLDs (gTLDs) –** domains without a sponsoring organization. The list of unsponsored gTLDs includes .com, .net, .org, .biz, .info, .name, and .pro.

ROOT SERVERS

There are many name servers in the DNS infrastructure.

Each name server contains information about a portion of the domain name space. Name servers are associated with levels as far as the first three levels of domain name space are concerned. There are **13 name servers** associated with the root level; they are called **root servers**. Two of the root servers are run by the U.S private-sector corporation **VeriSign**; the rest are operated by volunteer organizations around the world. The organizations that run name servers associated with a TLD are called **registries**. Generally, ccTLDs are run by designated registries in the respective countries, and gTLDs are run by global registries.

For example, VeriSign currently manages the name servers for the **.com** and **.net** TLDs, a nonprofit entity called **Public Internet Registry** (PIR) manages the name servers for the **.org** TLD, and another nonprofit organization called **EDUCAUSE** manages the name servers for the **.edu** TLD. All of these registry organizations are subject to change, however.

COORDINATION

The DNS infrastructure functions through collaboration among the various entities involved (organizations that manage root servers, registries that run TLDs, etc.) A nonprofit, private-sector corporation, the **Internet Corporation for Assigned Names and Numbers** (**ICANN**), acts as the technical coordination body for aspects of DNS. For example, ICANN formulates policies for management of root servers. ICANN also is the authority for creation of new TLDs. ICANN was established in 1998 by the U.S. Department of Commerce.

A user (i.e., an individual or corporation) wishing to register a domain name (which can only be an enterprise-level domain under a TLD) must contact an authorized entity called a **registrar**.

Registrars are companies that are authorized to sell domain names in a particular TLD to end-users. There are registrars all over the world. When the registrar receives the user's registration request, the registrar verifies that the name is available by checking with the registry that manages the corresponding TLD. If the name is available, the registrar registers the name with the appropriate registry. The registry then adds the new name to its registry database and publishes the new name in DNS.

DOMAIN GROUPS

Organizations that register and obtain an enterprise-level domain name often have to create **child domains** to properly identify Internet resources associated with various functional units.

For example, the owner of the domain name **example.com** might create the sub-domain shipping to create and identify resources associated with the shipping department of the organization. Similarly, many other sub-domains (in this context, third-level domains) may be created to properly identify all of the Internet resources of the organization.

Often, however, in any one organization (that is, the owner of a second-level domain) there will be many third-level domains but few Internet resources (Web servers, application servers, etc.) in each of these domains. Hence, it is not economical to assign a unique name server for each of these third-level and lower-level domains.

Furthermore, it is administratively convenient to **group all information** pertaining to an organization's primary domain (i.e., a second-level or enterprise-level domain) and all its sub-domains into a single resource and manage it as a unit.

DOMAIN ZONES

To facilitate this grouping, the DNS defines the concept of a **zone**.

A **zone** may be either an entire domain or a domain with a group of subdomains.

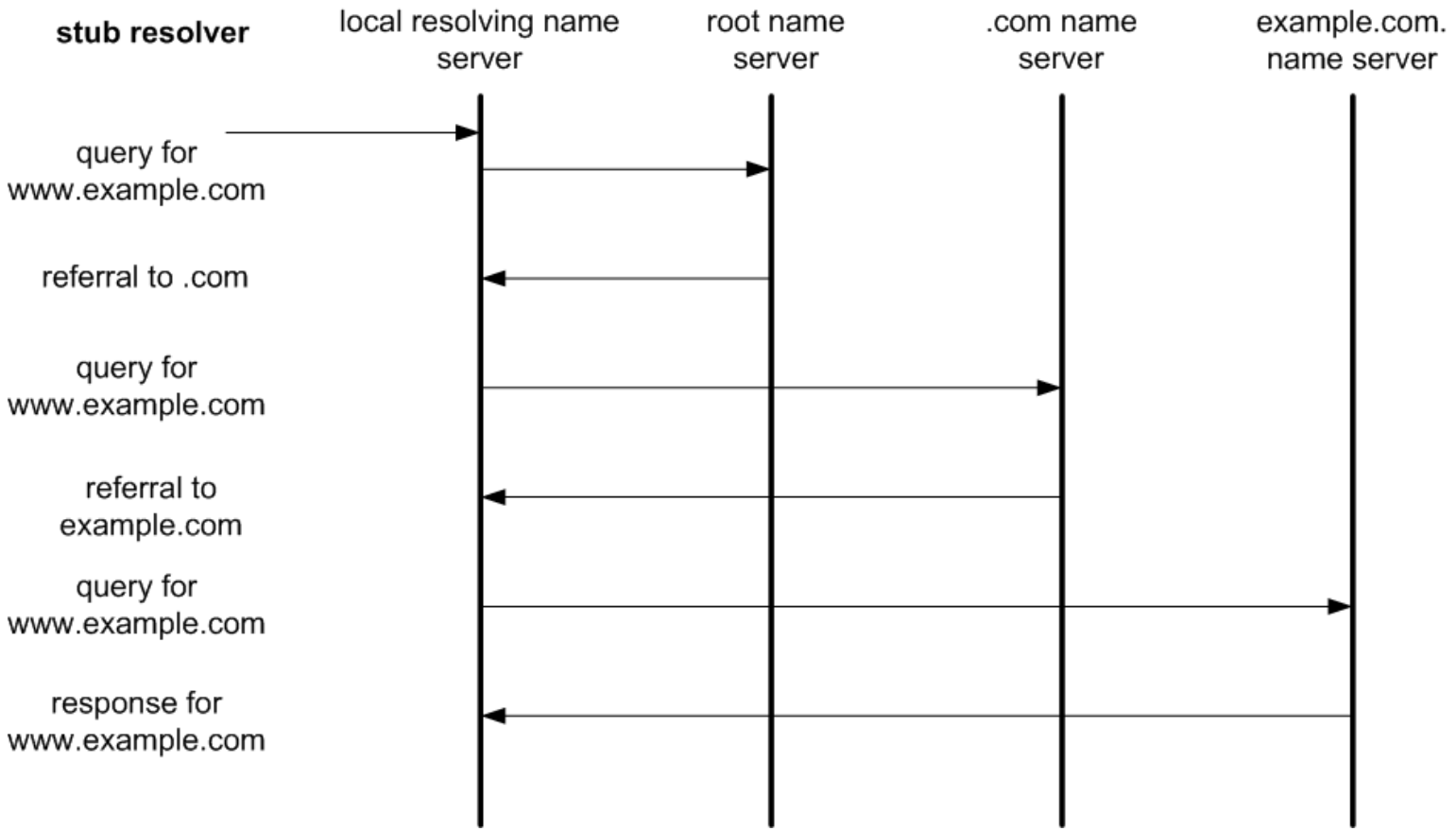
A **zone** is a configurable entity within a name server under which information on all Internet resources pertaining to a domain and a selected set of subdomains is described.

Thus, **zones** are **administrative building blocks** of the DNS name space just as **domains** are the **structural building blocks**.

As a result, the term **zone** commonly is used even to refer to a domain that is managed as a standalone administrative entity (e.g., the root zone, the .com zone).

NAME SEARCH PROCESS

When a user types the URL *www.example.com* into a Web browser, the browser program contacts a type of resolver called a **stub resolver** that then contacts a **local name server** (called a **recursive name server** or **resolving name server**).



NAME SEARCH PROCESS

The resolving name server will check its cache to determine whether it has valid information to provide IP address for the accessed Internet resource (i.e., *www.marketing.example.com*). If not, the resolving name server checks the cache to determine whether it has the information regarding the name server for the zone *marketing.example.com* (since this is the zone that is expected to contain the resource *www.marketing.example.com*)

If the name server's IP address **is in the cache**, the resolver's next query will be directed against that name server.

If the IP address of the name server of *marketing.example.com* **is not available in the cache**, the resolver determines whether it has the name server information for a zone that is one level higher than marketing.example.com (i.e., example.com).

If the name server information for *example.com* **is not available**, the next search will be for the name server of the *.com* zone in the cache.

ROLE OF A NAME SERVER

A name server performs the following functions:

- It provides a referral to a child zone.
- It provides a mapping from domain name to IP address, called domain name resolution, or IP address to domain name, called inverse resolution.
- It provides an error message if the query is for a DNS entry that does not exist.

The name server performs these three functions with the same DNS database, which is called a **zone file**. A class of information called **delegation information** contains the name server information for child zones in a parent zone and is used in performing the referral function.

The mapping function is performed by a class of information in a zone file called **authoritative information** and is provided by a **set of records** that list the resources in that zone, along with its domain name and its corresponding IP address. Because the resources belong to that zone, the information provided is deemed authoritative.

ROLE OF A NAME SERVER

Thus, a zone file contains two categories of information:

1) **authoritative information** (information about all resources for all domains in the zone) and
2) **delegation information** (information about name servers for child zones).

The locations in the zone file where delegation information appears are called **delegation points**. The level of a zone file is the level of the topmost domain for which it contains authoritative information.

In this example, the zone file in the name server of *example.com* is the **enterprise-level zone file**, and the corresponding name server is called the **enterprise-level name server**.

DNS COMPONENTS AND SECURITY OBJECTIVES

DNS includes the following entities:

- The platform (hardware and operating system) on which the name server and resolvers reside;
- The name server and resolver software;
- DNS transactions;
- DNS database (zone files);
- Configuration files in the name server and resolver.

Confidentiality, integrity, availability, and source authentication are security goals that are common to any electronic system. However, the DNS is expected to provide name resolution information for any publicly available Internet resource. Hence except for DNS data pertaining to internal resources (e.g., servers inside a firewall etc), that is provided by internal DNS name servers through secure channels, the DNS data provided by public DNS name servers is not deemed confidential. Hence **confidentiality** **is not one of the security goals of DNS**. Ensuring the authenticity of information and maintaining the integrity of information in transit is critical for efficient functioning of the Internet, for which the DNS provides the name resolution service. Hence, **integrity** and **source authentication** are the **primary DNS security goals**.

DNS DATA AND DNS SOFTWARE

The two primary software components of DNS are the **name server** and the **resolver**.

The **primary functions of the name server** are to host the database (called the zone file) containing domain information and to provide responses to name resolution queries through authoritative responses or referrals.

The **primary function of the resolver software** is to formulate a name resolution query or series of queries. The primary DNS data is the zone file (the configuration file is another type of DNS data).

ZONE FILE

The zone file contains information about various resources in that zone. The information about each resource is represented in a record called a **Resource Record** (RR).

Because a zone may contain several domains and several types of resources within each domain, the format of each RR contains fields for making this identification:

- **Owner name**: the domain name or a resource name;
- **TTL**: time to live in seconds;
- **Class**: at present only one class, IN (denoting Internet), is used;
- **RRType**: type of resource;
- **RData**: information about the resource (depends upon RRType).

ZONE FILE

Some of the common RRTypes are:

- **A**: Address RRType. An RR of this type provides the IP address for a host name (FQDN).
- **MX**: Mail Exchanger RRType. An RR of this type provides the mail server host name for a domain.
- **NS**: Name Server RRType. An RR of this type provides the name server host name for a domain.

IETF RFC 1035 provides the complete format of valid RRTypes in DNS. Because there could be multiple resources of a given RRType (e.g., several hosts acting as name servers) under the same owner name and since there is only one class (CLASS) (i.e., IN), the information of interest (e.g., all mail servers (RRType = MX) in a domain) in a zone file is on a combination of RRs that contain the same owner name, TTL, class, and RRType.

The set of RRs with the same owner name, TTL, class, and RRType is called an **RRset**.

Hence, logically a zone file is made up of several RRsets.

NAME SERVERS

There are two main types of name servers: **authoritative name servers** and **caching name servers**.

The term authoritative is with respect to a zone. If a name server is an authoritative source for RRs for a particular zone, it is called an **authoritative name server** for that zone. An authoritative name server for a zone provides responses to name resolution queries for resources for that zone, using the RRs in its own zone file.

A **caching name server** (also called a **resolving/recursive name server**), by contrast, provides responses either through a series of queries to authoritative name servers in the hierarchy of domains found in the name resolution query or from a cache of responses built by using previous queries.

AUTHORITATIVE NAME SERVERS

There are two types of authoritative name servers: **master** (or **primary**) **name server** and **slave** (or **secondary**) **name server**. To improve fault tolerance, there could be several slave name servers in an enterprise.

A **master name server** contains zone files that are created and edited manually by the zone administrator. Sometimes a master name server allows the zone file to be dynamically updated by authorized DNS clients. A master name server that is configured with this feature usually is called the **primary master name server**.

A **slave** (**secondary**) **name server** also contains authoritative information for a zone, but its zone file is a replication of the one in the associated master name server. The replication is enabled through a transaction called **zone transfer** that transfers all RRs from the zone file of a master name server to the slave name server. Because a name resolution query is for a specific RR, a zone transfer actually is treated as a category of name resolution query with type code **AXFR**, which means "all RRs for the zone that is being queried". Whenever the contents of a zone file are changed in the master name servers, the slave name servers are notified of the change through a transaction called **DNS NOTIFY**. When a slave name server receives this request, it initiates a zone transfer request to the master name server.

CACHING NAME SERVERS

A **caching name server** generally is the local name server in the enterprise that performs the name resolution function on behalf of the various enterprise clients. A caching name server also is called a **resolving/recursive name server**. The name resolution function is performed by a caching name server in response to queries from a stub resolver. The search process involved in name resolution may involve searching its own cache, recursively querying various authoritative name servers through a set of iterative queries, or a combination of these methods.

A specific name server can be configured to be both an authoritative and a recursive name server. In this configuration, the same name server provides authoritative information for queries pertaining to authoritative zones while it performs the resolving functions for queries pertaining to other zones. To perform the resolving function, it has to support recursive queries.

Any server that supports recursive queries is more vulnerable to attack than a server that does not support such queries.

As a result, authoritative information might be compromised. Hence, it is not a good security practice to configure a single name server to perform both authoritative and recursive functions.

RESOLVERS

Software such as Web browsers and e-mail clients that require access to Internet resources make use of the DNS client, called the **client resolver** or **stub resolver**.

The stub resolver formulates a name resolution query for the resource sought by the Internet-accessing software and sends it to a caching (resolving) name server in the enterprise. Stub resolvers generally are configured with a list of two or more resolving name servers to provide some fault tolerance for their operation. A stub resolver often is referred to as simply a **resolver**.

A caching (resolving) name server that receives a query from a stub resolver also formulates queries for sending them to authoritative name servers (if it is not able to respond to the query from its cache) and hence also sometimes is referred to as a resolver because it has a resolving component and a caching (name server) component.

DNS TRANSACTIONS

The most common types of DNS transactions are the following:

- DNS query/response;
- Zone transfer;
- Dynamic updates;
- DNS NOTIFY.

A **transaction** is a group of operations that have the following properties: atomic, consistent, isolated, and durable (ACID). The support of transactions enables new types of applications to be developed, while simplifying the development process and making the application more robust.

Term	Description
Atomic	Either all of the operations in the transaction succeed or none of the operations persist.
Consistent	If the data are consistent before the transaction begins, then they will be consistent after the transaction finishes.
Isolated	The effects of a transaction that is in progress are hidden from all other transactions.
Durable	When a transaction finishes, its results are persistent and will survive a system crash.

1. DNS QUERY/RESPONSE

This is the most common transaction in DNS. A DNS query originates from a resolver; the destination is an authoritative or caching name server. The most common query is a lookup for an RR, based on its owner name or RRType. The response may consist of a single RR, an RRSet, or an appropriate error message.

DNS queries are sent in a **single UDP packet**. The response usually is a single UDP packet as well, but data size may result in truncation, in which case the normal procedure is to reissue the query using TCP. UDP is preferred because of its lower overhead in consuming resources, and DNS administrators should make sure the zone data in responses do not result in a large percentage of truncated DNS responses.

DNS queries **are sent in the clear**, and it is assumed that the response received is correct and from an authentic source. As a result, it is **possible for an active attacker** to **intercept** (and alter) or **forge** responses back to a querying client.

2. ZONE TRANSFER

A **zone transfer** refers to the way a (secondary) slave server refreshes the entire contents of its zone file from the (primary) master servers. This process enables a secondary name server to keep its zone file in synch with its primary name server.

A **zone transfer transaction** starts as a query from a secondary name server to a primary name server.

A **zone transfer query** – in contrast to a DNS query – requests all RRs from a zone instead of requesting RRs of a given owner name or RRType. A zone transfer query originates from a secondary name server either in response to a **DNS NOTIFY** message or on the basis of the value of the Refresh data item in the RData field of the zone's **Start of Authority** (SOA) **RR**.

A zone transfer process has different security implications because it reveals a lot more information than a normal DNS query and because of the increased resource usage of the message.

3. DYNAMIC UPDATES

As enterprises add, delete, and move around IP-network-based resources (i.e., database servers, Web servers, mail servers, and even name servers), corresponding changes may need to be made to the zone file that carries information about the domains where the resources are located. In the early days of DNS, DNS zone administrators made such changes manually. When such changes became larger in volume and more frequent, however, the manual update process was found to be inadequate, and the concept of dynamic updates was introduced.

The suite of operations included in the dynamic update facility consists of the following:

- Add or delete individual RRs for an existing domain.
- Delete specific RRsets (a set of resource records with the same owner name, class, and RRType [e.g., a set of RRs with RRType NS for the domain/owner name example.com with the common class IN of course]) for an existing domain.
- Delete an existing domain (all resource records for a given domain name [e.g., all RRs for the domain example.com]).
- Add a new domain (one or more RRs for a new domain [e.g., adding an A-type RR for a new domain NYBranch.example.com]).

DNS zones that allow dynamic update open themselves to a host of malicious attacks.

4. DNS NOTIFY

Whenever changes occur in the zone file of the primary (master) DNS server, the secondary (slave) DNS servers that are supposed to carry identical data as the primary DNS server must be notified of the changes. This notification is accomplished through the **DNS NOTIFY message**, which signals a secondary DNS server to **initiate a zone transfer**. The DNS NOTIFY message is a much more efficient and faster way of keeping secondary servers in sync with the primary server than the alternative approach of secondary servers polling the primary server for changes via the SOA Refresh value timeout.

Once a secondary server receives a DNS NOTIFY message, it resets the relevant zone's refresh value to zero, prompting a zone transfer attempt. As in any zone refresh, if the zone serial number in the SOA RR has not increased, the zone transfer does not take place. This procedure allows changes to the zone to propagate to all name servers more quickly.

Since a DNS NOTIFY message triggers zone transfer, spurious DNS NOTIFY messages could result in unnecessary zone transfers and hence potential denial of service.

HOST PLATFORM THREATS

Threats to the platform that hosts DNS software are no different from threats that any host in the Internet faces. These generic threats and their impact – viewed specifically from the point of view of DNS hosts – are as follows:

Threat T1: The OS, any system software, or any other application software on the DNS host could be vulnerable to attacks such as buffer overflows, resulting in denial of name resolution service.

Threat T2: The TCP/IP stack in DNS hosts (stub resolver, caching/resolving/recursive name server, authoritative name server, etc.) could be subjected to packet flooding attacks (such as SYNC and smurf), resulting in disruption of communication. An application layer counterpart of this attack is to send a large number of forged DNS queries to overwhelm an authoritative or resolving name server.

Threat T3: A malicious insider who has access to local area network (LAN) segments where DNS hosts reside could launch an Address Resolution Protocol (ARP) spoofing attack that disrupts DNS message flows.

HOST PLATFORM THREATS

Threat T4: The platform-level configuration file that enables communication (e.g., resolv.conf and host.conf in Unix platforms) can be corrupted by viruses and worms or subject to unauthorized modifications due to inadequate file-level protections, resulting in breakdown of communication among DNS hosts (e.g., between a stub resolver and a resolving name server, between a resolving name server and an authoritative name server).

Threat T5: The DNS-specific configuration files (named.conf, root.hints, etc.), data files (zone file), and files containing cryptographic keys could be corrupted by viruses and worms or subjected to unauthorized modifications due to inadequate file-level protections, resulting in improper functioning of name resolution service.

Threat T6: A malicious host on the same LAN as a DNS client may be able to intercept and/or alter DNS responses. This would allow an attacker to redirect a client to a different site. This could be the first action in an attack on a client host.

DNS SOFTWARE THREATS

Threats to the DNS software itself can have serious security impacts. The most common software problems and the impact of threats against them are as follows:

Threat T7: DNS software (name server or resolver) could have vulnerabilities such as buffer overflows that result in denial of service.

Threat T8: DNS software does not provide adequate access control capabilities for its configuration files (e.g., named.conf), its data files (e.g., zone file) and files containing signing keys (e.g., TSIG, DNSKEY) to prevent unauthorized read/update of these files. These capabilities are provided on top of OS file-level protection referred to in threats **T4** and **T5** and may depend upon the latter.

THREATS DUE TO DNS DATA CONTENTS

DNS data is made up of two types: **zone files** and **configuration files**. The content of both these types of DNS data has security ramifications. All the security deployment options discussed in this document relate to configuration file contents. Security implications due to zone file content are mostly due to the following aspects of zone data:

- Parameter values for certain key fields in RRs of various RRTypes;
- Presence of certain RRs in the zone file.

The various types of undesirable contents in the zone file result in different security exposures and potential threats as follows:

Threat T9 – Lame Delegation: This error occurs when FQDN and/or IP addresses of name servers have been changed in the child zone but the parent zone has not updated the delegation information (NS RRs and glue records). In this situation, the child zone becomes unreachable (denial of service).

THREATS DUE TO DNS DATA CONTENTS

Threat T10 – Zone Drift and Zone Thrash: If the Refresh and Retry fields in the SOA RR of the primary name server are set too high and the zone file is changed frequently, there may be a mismatch of data between the primary and secondary name servers. This error is called **zone drift**; it results in incorrect zone data at the secondary name servers. If the Refresh and Retry fields in the SOA RR are set too low, the secondary server will initiate zone transfers frequently. This error is called **zone thrash**; it results in more workload on both the primary and secondary name servers. Such incorrect data or increased workload may result in denial of service.

Threat T11 – Information for Targeted Attacks: RRs such as HINFO and TXT provide information about software name and versions (e.g., for resources such as Web servers and mail servers) that will enable the well-equipped attacker to exploit the known vulnerabilities in those software versions and launch attacks against those resources.

SECURITY OBJECTIVES – HOST PLATFORM AND SOFTWARE

Common objectives with respect to protection of the DNS host platform, DNS software, and DNS data are **integrity** and **availability**.

The protection/threat mitigation approaches for the DNS host platform consist of the following:

- Running a secure OS;
- Secure configuration/deployment of OS.

Best practice protection approaches for DNS software are as follows:

- Running the latest version of name server software, or an earlier version with appropriate patches
- Running name server software with restricted privileges
- Isolating name server software
- Setting up a dedicated name server instance for each function
- Removing name server software from nondesignated hosts
- Creating a topological and geographic dispersion of authoritative name servers for fault tolerance
- Limiting IT resource information exposure through two different zone files in the same physical name server (termed split DNS) or through separate name servers for different client classes.

SECURITY OBJECTIVES – DATA CONTENT

Control of undesirable content in the zone file is accomplished by **analyzing the contents** for security implications, formulating integrity constraints that will check for the presence of such contents and verifying the zone file data for satisfaction of those constraints.

Therefore, the only protection approach is **to develop the zone file integrity checker software** that contains the necessary constraints and can be run against the zone file to flag those contents that violate the constraints.

To aid in **formulation of constraints**, desirable field values (ranges or lists) in the various RRs of zone file are required. These constraints need to be developed not only for RRs in an unsigned zone but also for additional RRs in a signed zone (zones that have implemented the DNSSEC specification).

SECURITY OBJECTIVES – DNS TRANSACTIONS

The threats to a DNS transaction depend on the type of transaction.

Name resolution queries and responses (DNS query/response) between DNS clients (stub resolver or resolving name server) and DNS servers (caching/resolving name server or authoritative name server) could involve any nodes in the Internet; hence, the threats against them are much greater in number and severity compared to those for zone transfer, dynamic update, and DNS NOTIFY transactions. In general, the nodes involved in zone transfer, dynamic update, and DNS NOTIFY transactions are all within the administrative domain of a single organization.

The only exceptions are instances in which the primary or secondary name servers of an organization are run on its behalf by ISPs or other organizations. There usually is a **preexisting trust relationship** in these cases, however, so it is not difficult to set up a mutual authentication system for DNS zone transfers.

SECURITY OBJECTIVES – DNS TRANSACTIONS

DNS name resolution queries and responses (DNS query/response) generally involve single, unsigned, and unencrypted UDP packets. The known threats to DNS query/response transactions have been documented in IETF RFC 3833 and can be classified as follows:

Threat T12: Forged or bogus response;

Threat T13: Removal of some RRs from the response;

Threat T14: Incorrect expansion rules applied to wildcard RRs in a zone file.

DNS TRANSACTIONS – FORGED OR BOGUS RESPONSE

A **forged** or **bogus response** is a response that is different from the one that is expected from a legitimate authoritative name server. A bogus response can originate from:

- A compromised authoritative name server (for queries originating from a resolving name server)
- A poisoned cache of a resolving name server (for queries originating from a stub resolver).

An authoritative name server could be compromised by a platform-level attack on its OS or communication stack.

The cache of a resolving (caching) name server could be poisoned by the following attacks:

- **Packet Interception.** In this type of attack, the attacker eavesdrops on a request and is able to generate and send a response by spoofing an authoritative name server before the real response from the legitimate authoritative name server reaches the resolving name server.
- **ID Guessing and Query Prediction.** In this type of attack, the attacker guesses the ID field in the header of the DNS request message (because this field is only 16 bits long, brute force guessing is possible) and possibly the QNAME and QTYPE (owner name and RRType, respectively). The attacker then injects bogus data into the network as a response by spoofing a name server.

REMOVAL OF SOME RRS AND INCORRECT WILDCARD EXPANSION

Removal of some RRs. Apart from injecting bogus or forged data in a response, an attacker also could remove RRs from a response. This action might result in a name resolution query failure and denial of service.

Incorrect expansion rules applied to wildcard RRs. Many zones use wildcard RRs to economize on the volume of data in the zone file. The wildcard patterns are used for synthesizing RRs on the fly in generating responses for name resolution queries. If synthesis rules are applied incorrectly in a name server, the RRs associated with resources existing in an organization may not be generated and made available in a DNS response. This fault also results in denial of service.

PROTECTION APPROACH FOR QUERY/RESPONSE THREATS – DNSSEC

The underlying feature in the major threat associated with DNS query/response (i.e., forged response or response failure) is **the integrity of DNS data returned in the response**.

Hence, the security objective is **to verify the integrity of each response received**. An integral part of integrity verification is to ensure that valid data has originated from the right source. Establishing trust in the source is called data origin authentication. Hence, the security objectives – and consequently the security services – that are required for securing the DNS query/response transaction are **data origin authentication** and **data integrity verification**.

These services could be provided by establishing trust in the source and verifying the signature of the data sent by that source. **The specification for a digital signature mechanism** in the context of the DNS infrastructure is in **IETF's DNSSEC standard**.

The objectives, additional RRs, and DNS message contents involved in DNSSEC are specified through RFCs 4033, 4034, and 4035.

PROTECTION APPROACH FOR QUERY/RESPONSE THREATS – DNSSEC

In DNSSEC, **trust in the public key** (for signature verification) **of the source** is established not by going to a third party or a chain of third parties, but by starting from a **trusted name server** (such as the root name server) and establishing the **chain of trust** down to the current source of response through successive verifications of signature of the public key of a child by its parent.

The public key of the trusted name servers is called the **trust anchor**.

After authenticating the source, the next process DNSSEC calls for is to **authenticate the response**. This requires that responses consist of not only the requested RRs but also an authenticator associated with them.

In DNSSEC, this authenticator is the digital signature of an RRSet.

The digital signature of an RRSet is encapsulated through a special RRType called **RRSIG**. The DNS client using the trusted public key of the source (whose trust has just been established) then verifies the digital signature to detect if the response is valid or bogus.

PROTECTION APPROACH FOR QUERY/RESPONSE THREATS – DNSSEC

To ensure that **RRs associated with a query are really missing in the zone file** and **have not been removed in transit**, the DNSSEC mechanism provides a **means for authenticating the nonexistence of an RR**.

It generates a special RR called an **NSEC RR** that lists the RRTypes associated with an owner name as well as the next name in the zone file. It sends this special RR, along with its signature, to the resolving name server. By verifying this signature, a DNSSEC-aware resolving name server can determine which authoritative owner name exists in a zone and which authoritative RRTypes exist at those owner names.

To protect against **the threat of incorrect application of expansion rules for wildcard RRs**, the DNSSEC mechanism provides a means of comparing the validated wildcard RR against an NSEC RR and thereby verifying that the name server applied the wildcard expansion rules correctly in generating an answer.

ZONE TRANSFER THREATS AND PROTECTION APPROACHES

Zone transfers are performed to replicate zone files in multiple servers to provide a degree of fault tolerance in the DNS service provided by an organization.

Threat T15 – Denial of Service: Because zone transfers involve the transfer of entire zones, they place substantial demands on network resources relative to normal DNS queries. Errant or malicious frequent zone transfer requests on the name servers of the enterprise can overload the master zone server and result in denial of service to legitimate users.

Threat T16: The zone transfer response message could be tampered.

The denial of service can be minimized if the servers allowed to make zone transfer requests are restricted to a set of known entities. To configure this restriction into the primary name server, there should be a means of identifying those entities.

ZONE TRANSFER THREATS AND PROTECTION APPROACHES

The IETF developed an alternate mechanism called a **transaction signature** (TSIG), whereby mutual identification of servers is based on a shared secret key.

Because the number of servers involved in zone transfer is limited (generally restricted to name servers in the same administrative domain of an organization), a bilateral trust model that is based on a shared secret key may be adequate for most enterprise (except for very large ones).

TSIG specifies that the shared secret key be used not only for **mutual authentication** but also for **signing zone transfer requests and responses**. Hence, it provides protection against tampering of zone transfer response messages (**threat T16**). Protection of DNS data alone (the payload) in a zone transfer message also can be ensured through verification of signature records accompanying RRs from a DNSSEC-signed zone.

These signatures, however, do not cover all the information in a zone file (e.g., delegation information). Furthermore, they enable verification of only the individual RRsets and not the entire zone transfer response message.

ZONE TRANSFER THREATS AND PROTECTION APPROACHES

There is also another method to authenticate DNS transactions by using asymmetric cryptography (i.e. public key cryptography).

The format of the SIG(0) RR is similar to the resource record signature (RRSIG) RR, and can be validated using a public key stored in the DNS (instead of a shared secret key).

SIG(0) can be more computationally expensive to use, but offers an advantage in that a previous trust relationship may not be necessary to use SIG(0) signed messages. However, since most zone transfers occur between parties that have a previously established relationship, it is considered easier to implement TSIG for authenticating zone transfer transactions.

DYNAMIC UPDATES THREATS AND PROTECTION APPROACHES

Dynamic updates involve DNS clients making changes to zone data in an authoritative name server in real time. Clients typically performing dynamic updates are CA servers, DHCP servers, and Internet multicast address servers.

Threat T17 – Unauthorized Updates: Unauthorized updates could have several harmful consequences for the content of zone data. Some harmful data operations include: (a) adding illegitimate resources (new FQDN and new RRs to a valid zone file), (b) deleting legitimate resources (entire FQDN or specific RRs), and (c) altering delegation information (NS RRs pointing to child zones).

Threat T18: The data in a dynamic update request could be tampered.

Threat T19 – Replay Attacks: Update request messages could be captured and resubmitted later, thus causing inappropriate updates.

DYNAMIC UPDATES THREATS AND PROTECTION APPROACHES

Threats **T17** and **T18** could be countered by **authenticating the entities** involved and providing a means to detect tampering of the messages.

Because these security objectives in the case of zone transfer are met by the TSIG/SIG(0) mechanism, the same TSIG/SIG(0) mechanism is specified for protecting dynamic updates.

Although the dynamic update message contains some replay attack (Threat **T19**) protection in the prerequisite field of the message, TSIG/SIG(0) provides an additional mechanism to protect against replay attacks by including a timestamp field in the dynamic update request. This signed timestamp enables a server to determine whether the timing of the dynamic update request is within the acceptable time limits specified in the configuration.

It sometimes makes more sense to use SIG(0) protection mechanisms for dynamic update than for zone transfer. Dynamic update transactions may happen between parties that do not always have a prior security relationship or may be part of a bootstrapping operation. Therefore, it may be impractical to use TSIG with a shared secret, but SIG(0) authentication using keys stored in the DNS may be a possibility.

DNS NOTIFY THREATS AND PROTECTION APPROACHES

DNS NOTIFY is a message sent by primary (master) name servers to secondary (slave) name servers, causing the secondary servers to start a refresh operation (i.e. query for SOA RR to check the serial number) and perform a zone transfer if an update to the zone has occurred. Because the NOTIFY message is only a signal, there are only minor security risks in dealing with the message. The primary security risk to consider is the following:

Threat T20 – Spurious NOTIFY Messages: Secondary name servers would receive spurious DNS NOTIFY messages from sources other than the primary name server.

The only impact of receiving spurious DNS NOTIFY message is the increase in workload in secondary name servers since a zone transfer will only occur when an updated zone is on the primary server. Because this threat is low impact, the only protection approach required is to configure the secondary name servers to receive DNS NOTIFY message only from the enterprise's primary name server. However, if TSIG is set up for use for all communication between a set of hosts, TSIG will be used with NOTIFY messages as well.

SECURING DNS HOST PLATFORM

The platform on which the name server software runs should be hosted on a properly secured OS. Most of the DNS installations run on a flavor of either Unix or Windows. Given this scenario, it is necessary to ensure the following:

- The latest OS patches are installed.
- Recommended OS configuration practices are followed as issued by CERT/CC and NIST's NVD metabase, based on identified vulnerabilities that pertain to the application profile into which the name server software fits. In particular, hosts that run the name server software should not provide any other services and therefore should be configured to respond to DNS traffic only. In other words, the only allowed incoming and outgoing messages to these hosts should be 53/udp and 53/tcp.

SECURING DNS SOFTWARE

Securing DNS software. Protection approaches for DNS software include choice of version, installation of patches, running it with restricted privileges, restricting other applications in the execution environment, dedicating instances for each function, controlling the set of hosts where software is installed, placement within the network, and limiting information exposure by logical/physical partitioning of zone file data or running two name server software instances for different client classes.

Running the latest version of Name Server software. Each newer version of the name server software, especially the BIND software, generally is devoid of vulnerabilities found in earlier versions because it has design changes incorporated to take care of those vulnerabilities. Of course, these vulnerabilities have been exploited (i.e., some form of attack was launched), and sufficient information has been generated with respect to the nature of those exploits. Thus, it makes good business sense to run the latest version of the BIND software because theoretically it is the safest version. Even if the software is the latest version, it is not safe to run it in default mode. The security administrator should become familiar with the new security settings for the latest version.

SECURING DNS SOFTWARE

Turning off the BIND version query. There is a feature in BIND that returns the version number of the server daemon running if a special query is sent to the server. This query is for the string "version.bind" with query type TXT and query class Chaos (CH). This information may be of use to attackers who are looking for a specific version of BIND with a discovered weakness. BIND can be configured to refuse this type of query by having the following command in the BIND configuration file (/etc/named.conf).

Running Name Server software with restricted privileges. If the name server software is run as a privileged user (e.g., root in Unix systems), any break-in into the software can have disastrous consequences in terms of resources resident in the name server platform. Specifically, a hacker who breaks into the software acquires unrestricted access and therefore can potentially execute any commands or modify or delete any files. Hence, it is necessary to run the name server software as a non-privileged user with access restricted to specified directories to contain damages resulting from break-ins.

SECURING DNS SOFTWARE

Isolating the Name Server software. Even if the DNS software (e.g., BIND) is run on a secure OS, the vulnerabilities of other software programs on that platform can breach the security of DNS software. Hence, it is necessary to ensure that the platform on which the DNS software runs contains no programs other than those needed for OS and network support.

Dedicated Name Server instance for each function.

An **authoritative name server** is only intended to provide name resolution for the zones for which it has authoritative information. Hence, the security policy should have recursion turned off for this type of name server. Disabling recursion prevents an authoritative name server from sending queries on behalf of other name servers and building up a cache using responses. Disabling this function eliminates the cache poisoning threat.

A **resolving name server** is only intended to provide resolving services (processing resolving queries on behalf of clients) for internal clients. Thus, protection of resolving name servers can be ensured by restricting their types of interactions (also called transactions) to designated hosts through various configuration options in the configuration file of the name server software.

SECURING DNS SOFTWARE

Removing Name Server software from non-designated hosts. DNS software should not be running or present in hosts that are not designated as name servers. The possibility arises in the case of DNS BIND software because of the fact that many versions of Unix (including Solaris and Linux versions) come installed with BIND as default. Hence, while taking an inventory of software in workstations and servers of the enterprise as part of the security audit, it is necessary to look for BIND installations and remove them from hosts that are not functioning as name servers.

Network and geographic dispersion of authoritative Name Servers. Most enterprises have an authoritative primary server and a host of authoritative secondary name servers. It is essential that these authoritative name servers for an enterprise be located on different network segments. This dispersion ensures the availability of an authoritative name server not only in situations in which a particular router or switch fails but also during events involving an attack on an entire network segment. In addition to network-based dispersion, authoritative name servers should be dispersed geographically as well.

SECURING DNS SOFTWARE

Limiting information exposure through partitioning of zone files. Authoritative name servers for an enterprise receive requests from both external and internal clients. In many instances, external clients need to receive RRs that pertain only to public services (public Web server, mail server, etc.) Internal clients need to receive RRs pertaining to public services as well as internal hosts. Hence, the zone information that serves these RRs can be split into different physical files for these two types of clients: one for external clients and one for internal clients.

This type of implementation of the zone file is called **split DNS**.

Split DNS does have some drawbacks. First, remote hosts (travelers using laptops to connect back to an organization, for example) may not be using an internal resolving DNS server, and therefore may not be able to see internal hosts. Second, internal host information may leak to outside the firewall (by accident or attack), defeating the purpose of having a split DNS, or causing confusion of an internal and external host having the same FQDN, but different IP addresses. Split DNS should not be seen as a replacement for proper access control techniques.

SECURING DNS SOFTWARE

Limiting information exposure through separate Name Servers for different clients. Instead of having the same set of authoritative name servers serve different types of clients, an enterprise could have two different sets of authoritative name servers.

One set, called **external name servers**, can be located within a DMZ; these would be the only name servers that are accessible to external clients and would serve RRs pertaining to hosts with public services (Web servers that serve external Web pages or provide B2C services, mail servers, etc.)

The other set, called **internal name servers**, is to be located within the firewall and should be configured so it is not reachable from outside and hence provides naming services exclusively to internal clients. The purpose of both architecture options (i.e., two different sets of name servers and split DNS) is to prevent the outside world from knowing the IP addresses of internal hosts.

SECURING DNS TRANSACTIONS

Restricting transaction entities based on IP address. In this type of implementation, the DNS name servers and clients participating in a DNS transaction are restricted to a trusted set of hosts by specifying their IP addresses in appropriate access control statements provided by the name server software. The protection provided by these IP-based access control statements can be circumvented by attacks such as IP spoofing. Hence, this solution is not recommended for DNS query/response, zone transfer, and dynamic update transactions that have high threat impact. However, for the DNS NOTIFY transaction, where the only threat is spurious notification (which may not even trigger a zone transfer), an access control based on IP address will suffice.

Transaction protection through hash-based message authentication codes (TSIG specification). In this approach, transaction protection is enabled through generation and verification of hash-based message authentication codes (HMAC).

Transaction protection through asymmetric digital signatures (DNSSEC Specification). The core services provided by DNSSEC are data origin authentication and integrity protection. DNSSEC is used mainly for securing DNS information obtained from DNS query/response transactions.