An Integrity Verification Scheme for DNS Zone file based on Security Impact Analysis

Ramaswamy Chandramouli
NIST, Gaithersburg, MD 20899
(mouli@nist.gov)

Scott Rose
NIST, Gaithersburg, MD 20899
scott.rose@nist.gov

Abstract

The Domain Name System (DNS) is the world's largest distributed computing system that performs the key function of translating user-friendly domain names to IP addresses through a process called name resolution. After looking at the protection measures for securing the DNS transactions, we discover that the trust in the name resolution process ultimately depends upon the integrity of the data repository that authoritative name servers of DNS use. This data repository is called a zone file. Hence we analyze in detail the data content relationships in a zone file that have security impacts. We then develop a taxonomy and associated population of constraints. We also have developed a platform-independent framework using XML, XML Schema and XSLT for encoding those constraints and verifying them against the XML encoded zone file data to detect integrity violations.

1. Introduction

The domain name system (DNS) is the world's largest distributed computing system that enables access to any resource in the Internet by translating user-friendly domain names to IP Addresses. The process of translating domain names to IP Addresses is called Name Resolution. A DNS name resolution is the first step in the majority of Internet transactions. The DNS is in fact a client server system that provides this name resolution service through a family of servers called Domain Name Servers. The hierarchical domain space is divided into administrative units called zones. A zone usually consists of a domain (say example.com) and possibly one or more sub domains (projects.example.com, services.example.com). The authoritative data needed for performing the name resolution service is contained in a file called the zone file and the DNS servers hosting this file are called the authoritative name servers for that zone. The DNS clients that make use of the services provided by authoritative name servers could be of two types. One type is called a stub resolver that formulates and sends a query every time it receives a request from an application that requires Internet service (e.g., a browser). The other type is called a caching (also called recursive/resolving) name server that caches the name resolution responses it has obtained from authoritative name servers and thus able to serve multiple stub resolvers.

The zone file hosted on an authoritative name server consists of various types of records called Resource Records (RRs). Associated with each DNS resource record is a type (RRtype). The code for these RRtypes is assigned by an international organization called Internet Assigned Names Authority (IANA). An RR of a given RRtype in a zone file provides a specific type of information. Some of the common RRtype codes are: NS, MX and A. An NS RR in a zone file gives the fully qualified domain name (FQDN) of the host that is considered the name server for that zone. For example, an NS RR in the zone file of the zone example.com gives the information that the host ns1.projects.example.com is a name server for the domain projects.example.com. Similarly an MX RR gives the host name for a mail server for the zone. An A RR gives the IP address for a host in a domain within the zone. A zone file generally consists of multiple RRs of a given RRtype with some exceptions (e.g., there can be only SOA RR in a zone file). It can also have multiple RRs for the same domain name and same (or different) RRtype (e.g., multiple name servers or mail servers for a domain say services.example.com).

The DNS infrastructure consists of many different types of DNS servers, DNS clients and transactions among/between these entities. The most important transaction in DNS is the one that provides the core service of DNS (i.e., name resolution service) and is called the DNS Query/Response. A DNS Query/Response transaction is made up of a query originating from a DNS client (generically called a DNS resolver) and response from a DNS name server. The response consists of one or more RRs. These RRs may be served from its own zone file (for an authoritative name server) or from a cache of RRs obtained from other name servers (for a caching/resolving/recursive name servers). In this way, the DNS basically serves as a global, distributed database. Name servers (serving zone files) each contain a small portion of the global domain space, and clients issue queries using a domain name and a desired RRtype.

The DNS Query/Response transaction, just like any other Internet-based transaction, is subject to several types of attacks such as spoofing and man-in-the-middle attacks. DNS is especially vulnerable to these types of attacks because the basic Query/Response transaction uses UDP as the transport. This makes it easier for an attacker to intercept DNS message packets and alter any of the information contained therein. An attacker could redirect Internet traffic from a host (or collection of hosts) in this manner. To provide protection from these attacks, it is necessary to verify that a DNS response has originated from an authentic source (the responding name server is indeed the one that is supposed to respond), the response is complete and has not been tampered with on transit (integrity of the response is maintained). The protection requirements of origin authentication and integrity verification are needed not only for responses originating from authoritative name servers but also from the cache of resolving/recursive name servers.

To provide these security services of data origin authentication and integrity verification to DNS responses, IETF has proposed a set of security extensions to DNS collectively called DNSSEC through a series of RFCs [8,9,10]. These DNSSEC specifications call for generating a digital signature (stored in a new RRtype called RRSIG) for every RRset in a zone (a set of RRs of a given RRtype is called an RRset) using a private key associated with the zone and then publishing the corresponding public key (stored in a DNSKEY RR). This will then enable a recipient of the DNS response (i.e., the DNS resolver on the client side) to verify the integrity of the RRs in a response using the public key and the signature of RRsets (contained in a RRSIG RR) sent along with it in the DNS response. For discussion purposes, we will call the RRs of these additional types as DNSSEC RRs and the original RRs as simply DNS RRs.

However, these new types only provide the ability for clients to authenticate the origin of the DNS data (i.e. the authoritative source for the zone data) and the integrity of the data in transit. This is to counter an attacker redirecting Internet traffic by altering the data in a DNS response in transit, or in the cache of a caching, recursive name server. There is no guarantee that the data is correct, or even useful to the client. It is still possible, as with any human generated input, that the original data is incorrect in some way. In order for DNSSEC to be effective and the data to be usable by clients, the original DNS data must be correct.

This original data is the one found in zone files of authoritative name servers. The overall trust in DNS depends upon the integrity of the zone file data. The integrity of zone file data, in the context of this paper, pertains to the content satisfying certain relationship constraints. It has nothing to do with the traditional concept of file integrity which is verified by matching an archived hash with the hash of the file generated on the fly. Therefore, to discover the exact consequences of the zone file integrity on the trust of DNS name resolution service, it is necessary to perform a detailed analysis of the zone file content relationships. Hence the first of two major contributions of this paper is to perform this analysis and develop a taxonomy and associated population of zone file integrity constraints. The second major contribution of this paper is to develop a framework for verifying a zone file for satisfaction of these constraints and detecting/identifying violations of these integrity constraints. Towards this objective, we developed a schema of the zone file using XML Schema [12]. We call this schema – “Zone File Schema”. The associated XML encoded Zone file is called “XML encoded Zone File Data”. The integrity checks (procedural statement of constraints) needed for any zone file are encoded as XSLT [13] constraints. The XSLT constraints are based on the Zone File Schema and can be applied to verify the integrity of any XML encoded zone file whose structure is based on Zone File Schema. A useful by product of this framework is the ability to programmatically generate zone files using XSLT transforms on the integrity-checked XML encoded zone file data.

The overall organization of this paper is as follows. A brief description of the common data structure of any RR and information and functionality provided by the original DNS RRs and DNSSEC RRs are given in section 2. Section 3 builds up the case for integrity verification of DNS based on analysis of data content relationships that have security impacts. Section 4 presents the taxonomy and a set of associated integrity constraints. Sections 5 and 6 describe the framework for automated integrity verification of DNS zone file. Specifically, in section 5, we deal with the development of an XML Schema for the zone file and an XML encoding of an example zone file that corresponds to this schema. Based on the XML Schema, XSLT constraints that encode the various integrity constraints (from section 4) are developed in section 6. This section also illustrates the process of applying these constraints against the XML encoded zone file and generating integrity violation messages. Comparison with related work is given in section 7. Section 8 provides conclusions and scope for future work.

2. Structure and Functionality of RRs in the Zone File

Before we delve into the integrity requirements of the zone file, it is necessary to look at the common data structure for all RRs of the zone file as well as the functionality provided by RRs of each RRtype. We include in our integrity analysis the additional RRtypes from DNSSEC specification and hence we look at the functionality of both DNS RRs and DNSSEC RRs.

2.1 Resource Record (RR) Structure

The generic structure of any resource record (RR) in the zone file consists of the following fields:

- Owner name
- Time-to-Live (TTL)
- Class
- RRtype
- RDATA

Out of the 5 fields listed above, the first four are atomic fields (contain only one data item) while the last field (i.e., RDATA) is a composite field consisting of multiple data items. The number and names of individual data items in the RDATA field for an RR depends upon the RRtype of the RR. The following are a subset of RRs from the zone file for our example zone example.com.

example.com 86400 IN SOA ns1.example.com (2005010200 2h 20M 4W12h 2h30M)
(RR1)
example.com 86400 IN NS ns1.example.com (RR2)
example.com 86400 IN MX 10 mail1.example.com (RR3)
ns1.example.com 86400 IN A 192.168.0.3 (RR4)
mail1.example.com 86400 IN A 192.168.0.4 … (RR5)

An RR is often identified by its RRtype. Hence the first RR above (i.e., RR1) is called an SOA RR, the second one NS RR, the third one an MX RR etc.

Let us now analyze the semantics of each of the 5 fields in the above RRs. The semantics of the first field (owner name) depends upon the RRtype (just like RRDATA field except that this is an atomic field). In SOA, NS and MX RRs (RR1, RR2 & RR3 above), the owner name field contains the name of the domain. In the case of an A RR (RR4 & RR5), the owner name field contains the name of the host. The second field TTL provides the remaining duration in seconds that the RR can be considered valid. This field serves as a countdown timer by caching clients . When the TTL reaches zero, the given RR is considered invalid for the zone, and the client must re-query the authoritative name server for the zone to refresh this RR in its cache. This is to ensure that caches contain the most current version of the zone data. The third field Class contains the code IN which stands for Internet. The value of this field is common for all RRs in the zone file. The fourth field is the RRtype. As already alluded to, the code for RRtypes should be one of the valid IANA assigned values. The fifth field (RDATA field) contains more than one data item in some of the RRs (e.g., SOA RR, MX RR) and only a single data item in some other RRs (e.g., NS RR). In the example SOA RR (RR1 above), the data items shown are: The serial number (a sort of version number for the zone file), refresh interval in hours, retry interval in minutes, expiry duration in weeks and hours and minimum TTL in hours and minutes. In the case of the MX RR (RR3 above) there are two data items in RDATA field. The second data item contains the name of the mail host while the first data item denotes the priority associated with that host. The RDATA field in an NS RR contains only one data item and that is the name of a host (represented by its Fully Qualified domain Name or FQDN) that acts as the name server. The detailed semantics of each these fields in a DNS RR can be obtained from RFC 1035[4].

3. The Need for Zone File Integrity Checking as a Security Measure

In order to make a case for integrity checking of the zone file for improving the security of DNS, we need to take a look at the major transactions of DNS, their vulnerabilities, existing countermeasures and their limitations. The three major transactions in the DNS are:

- DNS Query/Response: This involves all name resolution queries and their associated responses.
- Zone Transfer: Transactions involving periodical refresh of the contents of zone files in secondary authoritative name servers from primary authoritative name servers.
- Dynamic Update: Update of zone file data in real time by special clients such as DHCP servers or Internet Multicast Address Servers.

The above transactions are vulnerable to all the threats associated with any Internet-based transactions. These threats include IP spoofing, modification of the messages in transit and replay attacks. It must be remembered here that eavesdropping on a DNS transaction is not deemed a threat since the DNS data by its nature is not reckoned as confidential. To counter the identified threats, DNS requires the security services of data origin authentication and message integrity. To provide these services, IETF has issued two different types of specifications consisting of protection mechanisms that provide these security services.

- DNSSEC specifications specified through a series of RFCs given in [8,9,10], provide data origin authentication and message integrity for DNS Query/Response transaction using digital signatures (asymmetric cryptography). The mechanics of providing protection through implementation of DNSSEC has already been briefly described in section 1.
- TSIG specifications contained in RFC 2845 [6] , provide data origin authentication and message integrity for zone transfers and dynamic update transactions through hash-based message authentication codes (HMACs).

The rationale for two different solution approaches for providing the same security services is the following: Zone Transfers and Dynamic Updates involve hosts from the same administrative domain (a single enterprise or two enterprises with prior business relationship –e.g., an enterprise and its ISP) and hence a solution based on a shared secret (hash key) is possible. On the other hand a DNS Query/Response transaction can involve any arbitrary pair of DNS resolver and DNS name server located anywhere in the world. Hence it requires a scalable solution using a public key/private key pair. In TSIG, the message authentication codes are generated dynamically for every transaction, whereas the digital signatures in DNSSEC are generated and stored permanently in the zone file of the authoritative name servers. The zones whose zone file contains digital signatures are called signed zones while DNS zones that do not contain security information are called unsigned zones. The DNS name server that hosts signed zones are called DNSSEC-aware name servers while DNS resolvers that can verify the digital signatures sent as part of the response are called DNSSEC-aware resolvers.

Regardless of the fact that a zone is signed or unsigned (i.e., implemented or not implemented DNSSEC), it has been found that the content of the zone file does have a great bearing on the overall security ofDNS. This is not surprising since DNSSEC and TSIG specifications can only ensure that the message has originated from the legitimate source and that it has not been tampered with during its passage over the communication network. The overall trust in the name resolution service provided by DNS rests with the quality of data in the zone file. If this quality is less than desirable, any transaction-level protection will not enhance this trust. The data content relationships (we will use the acronym DCR) and their associated security impacts are described in the following paragraphs:

<u>(DCR 1): The presence of certain RRs reveals sensitive information needed for launching targeted attacks:</u> The HINFO RR is generally used to carry information about a host such as the O/S name, version and its latest installed patch. This information could be potentially used to launch targeted attacks on such hosts. Depending upon whether the attacked host is a DNS name server, mail server or web server, the adverse consequences of such attacks could be different.

<u>(DCR 2): Large parameter values in the RDATA portion of certain RRtypes could result in either no answers or obsolete (unusable) answers resulting in denial of service</u>: For example the "refresh" data item in the RDATA field of a SOA RR specifies the frequency with which secondary authoritative name servers should initiate zone transfers in order to keep their zone file contents in synch with the primary authoritative name servers. Similarly the "retry" data item in the same field of the same RR tells the frequency with which the secondary name server should make retry attempts in case a refresh attempt is unsuccessful. The "Expiry" data item in the same RDATA field denotes the time duration after which the secondary name server should make no more attempts at refresh but instead lets its zone file contents expire. Large value for the data items discussed above (i.e., "refresh", "retry" and "expiry") could result in mismatch of data between secondary name servers (that provide fault tolerance) and primary name server resulting in serving either a empty response or obsolete response to those DNS resolvers querying the secondary name servers. This content-related phenomenon is called a "zone drift". Frequent occurrences of zone drift could potentially result in denial of service to DNS resolvers using those secondary name servers. Zone drift could also occur due to mismatch of data between authoritative name servers (primary or secondary) and the cache of caching (resolving) name servers resulting in the same denial of service situations for data served directly from the cache . This occurs due to large value of the TTL field in any RRset (or large value of MinTTL data item of a SOA RR's RDATA field if used as the default value by some RRsets). It is obvious that if this value is large, the RRs in the cache will not expire for quite a length of time during which there is the possibility that the data in the authoritative name servers to have changed.

(DCR 3): A different set of security impacts occur if the values of the data items discussed above are mirror images of the situation discussed above – i.e., the parameter values in RDATA field of certain RRs are small For example if the "refresh" value in SOA RR is very small, the secondary authoritative name server will be performing frequent zone transfers from the primary authoritative name server. As another example, if the "MinTTL" data item in a SOA RR is small, those RRs that have used this default value will expire much more quickly in the cache of the caching name server. Hence the DNS resolver will have to make more frequent queries to the authoritative name servers instead of relying on its cache. From this scenario it is clear that having a small value for "refresh", "retry", "expiry" and "MinTTL" data items in the RDATA field of a SOA RR will result in more frequent queries to primary and/or secondary authoritative name servers and has the potential to degrade performance (by increasing query response time). This situation is called "zone thrash". Some software may have timeout parameters for obtaining network connection (in this case it is the DNS name resolution service that provides the IP address for a specific application service) and if the response time for DNS Query/Response transaction exceeds this limit, time out will result.

(DCR 4): Missing or ill-formed associative RRs results in inaccessibility of Internet domains/services: Access to certain domains and/or services require two RRs (or RRsets) in the zone file to be retrieved. The first RR (RRset) will only provide the FQDN of the domain/service (e.g., NS and MX RRs that provide the FQDN (e.g., ns1.example.com) of the name server and mail server respectively for a domain). The second RR (RRset) then provides the IP address for the retrieved FQDN through an A/AAAA RR (host to IP Address mapping RR). The second RR (i.e., A/AAAA RR) is called the associative RR since it provides the actual network address (IP address) to reach the host providing a specific service that is referenced in the first RR (NS or MX RR). If the associative RR either contains an invalid IPv4 /IPv6 address or the RR itself is missing, then the host providing the internet-based service becomes inaccessible and hence is tantamount to denial of service.

(DCR 5): Incorrect parameter values in the zone file's digital signature records (RRSIG RRs) will render the DNSSEC security service non-usable: The RDATA field in the digital signature record (RRSIG RR) contains several data items whose correct value ensures that the signature can be cryptographically verified. These data items include: the DNSSEC algorithm code (the code for the public key algorithm used in signing and verifying signatures), type covered, original_TTL, labels, signature inception & expiry times and the signer name. For example, if the signature is not currently valid (current date is not between signature inception and expiry dates), then a DNSSEC-aware resolver will not use it to validate the integrity of the RRset covered by the signature. Yet another example is a situation where the signer name in the RDATA portion of an RRSIG RR does not match with the owner name in the corresponding DNSKEY RRs (correspondence established based on equality of a key tag ID value), in which case the resolver will be unable to fetch the correct public key for signature verification and will reject the response. The overall effect of the above two example scenarios is that the security services (data origin authentication and message integrity) enabled through DNSSEC implementation become non-usable.

There are some rare but not impossible data scenarios that could result in subversion of DNSSEC security services. For example, in the case of a key compromise, the authoritative name servers will perform an emergency key rollover and re-sign RRsets in the zone file with the new key (thus new signatures – RRSIG RRs). However if the TTL value and signature validity duration (difference between signature expiry and inception dates/times) for RRsets are set high, these RRsets will not expire in the cache of caching name servers. Taking advantage of this scenario, it is possible for an attacker to introduce bogus records signed with the compromised key into those caches essentially misdirecting those stub resolvers that depend upon those caching name servers, thus effectively subverting the protection provided by DNSSEC.

(DCR 6): Absence of multiple RRs of certain RRtypes representing critical services: Certain critical services such as name resolution and email transfer/access need to be hosted on multiple servers to provide fault tolerance. Hence there should be multiple RRs for RRtypes representing those services. Specifically multiple RRs should be present for NS and MX RRtypes associated with a domain.

(DCR 7): Ambiguous Data – Needs Policy for Interpretation: Certain data content scenarios are low risk from the point of security but nonetheless needs policies for proper usage of their underlying RRs. Examples of such scenarios are multiple IP addresses for a given host (i.e., multiple A RRs for a given host identified by a FQDN) and multiple mail servers with the same priority code (i.e., multiple MX RRs with the same priority code).

The DNS zone file content scenarios together with their associated security impacts are summarized in Table 1.

4. Zone File Integrity Constraints – A Taxonomy and Associated Set

In order to avoid the zone file content scenarios described in the last section and consequently mitigate their security impacts, it is necessary to formulate integrity constraints and develop a scheme for verifying these constraints given zone file. Before actually formulating the constraints, we looked at the landscape of constraints that have security impact and arrived at the following taxonomy.

- Single RR constraints (ICType1)
- Intra RRtype constraints (ICType2)
- Inter RRtype constraints (ICType3)

Single RR (ICType1) constraints specify the requirements that one or more field values within a single RR must satisfy. Since the semantics of the fields vary based on RRtype, these constraints are different for different RRtypes. For example, every A/AAAA resource records must have a valid IPv4/IPv6 address in its RDATA. The RR may be syntactically correct, but if it contains an invalid IPv4/IPv6 address in the RDATA field, it will cause an error on the client side, even though the DNS transaction succeeded.

Intra RRtype (ICType2) constraints specify the relationship between values of fields among RRs of the same RRtype. This constraint type also includes cardinality constraints (i.e., the number of RRs of a given RRtype that are allowed in the zone file). An example is the case where there are multiple A RRs with the same owner name (semantically stands for a host – FQDN) but different IP addresses.

Inter RRtype (ICType3) constraints specify the relationship between values of different fields among RRs of dissimilar RRtypes that share a relationship in some manner. For example, the Mail Exchange RR (MX RR) contains the FQDN of a mail server for the zone. Somewhere, there should be a corresponding Address (A RR) record with the IP address of that mail server. Otherwise, a client will never be able to reach the mail server (as it does not have an IP address found in the DNS).

Based on the field deployment experiences, IETF documents relating to best practices and threat analysis [5,7], we formulated a set of 25 integrity constraints that the zone file in a DNSSEC-aware name server should satisfy. The description of the 25 zone file integrity constraints and the security impact(s) associated with each of them is given in Table 2.

5. Modeling and Encoding of Zone File using XML Schema and XML

As part of our effort to develop a platform-neutral framework for checking the integrity of a zone file, we develop a representation of the structure of the zone file using XML Schema language. Specifically we develop an XML Schema that can express the structure of each RRtype in the DNS zone file. Since the structure of RRs of various RRtypes differs only in the RDATA field we first treat the common fields collectively as an entity. In XML Schema an entity is modeled as an element with associated set of attributes and/or sub elements. In our schema we will call this entity RRHeader. Hence we create an element by name RRHeader with the common field names as attributes of that element. The element RRHeader will thus have owner name, TTL, class and RRtype as attributes. The XML Schema definition of the element RRHeader is as follows:

```
<xs:element name="RRHeader"
type="RRHeaderType"/>
<xs:complexType name="RRHeaderType">
<xs:attribute name="Owner" type="xs:string"
use="required"/>
<xs:attribute name="TTL" type="xs:integer"
use="optional"/>
<xs:attribute name="CLASS" type="xs:string"
use="required"/>
<xs:attribute name="RRType" type="valid_RR"
use="required"/>
</xs:complexType>
```

Since the composition of the RDATA field varies with RRtype, we model the RDATA field for each RRtype by a separate element. The definition of RDATA field for NS(name server) and MX (mail server) and A (IP address) RRtypes in XML Schema is given as follows:

```
<xs:complexType name="NS_RDATAType">
 <xs:sequence>
     <xs:element name="NSHost" type="xs:string"/>
 </xs:sequence>
</xs:complexType>

<xs:complexType name="MX_RDATAType">
 <xs:sequence>
     <xs:element name="Priority" type="xs:integer"/>
          <xs:element name="MXHost"
type="xs:string"/>
 </xs:sequence>
</xs:complexType>

<xs:complexType name="A_RDATAType">
 <xs:sequence>
     <xs:element name="IPAddress" type="xs:string"/>
 </xs:sequence>
</xs:complexType>
```

A RR of any given RRtype is composed of the common fields and RDATA field and hence the XML Schema element definition for an RR of a given

RRtype should be a concatenation of a RRHeader element and the specific element that describes the RDATA for that RRtype. For example the MX element that models an MX RR should contain the common RRHeader element and MX_RDATA element and is defined as follows:

```
<xs:element name="MX" type="MXType"/>
<xs:complexType name="MXType">
 <xs:sequence>
      <xs:element ref="RRHeader" maxOccurs="1"/>
      <xs:element  ref="MX_RDATA"
                   maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
```

Finally the XML Schema for the entire zone file is represented as the collection of elements representing the various RRs in the zone file as follows:

```
<xs:element name="Zone_File_Schema"
type="ZoneFileSchemaType"/>
<xs:complexType name="ZoneFileSchemaType">
 <xs:sequence>
      <xs:element ref="SOA" maxOccurs="1"/>
     <xs:element ref="NS" maxOccurs="unbounded"/>
     <xs:element ref="MX"  maxOccurs="unbounded"/>
     <xs:element ref="A"  maxOccurs="unbounded"/>
     <xs:element ref="HINFO"
                   m axOccurs="unbounded"/>
     <xs:element ref="CNAME"
                  maxOccurs="unbounded"/>
     <xs:element ref="DNSKEY"
                  maxOccurs="unbounded"/>
       <xs:element ref="RRSIG"
                     maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
```

A subset of the XML encoded zone file data (that contains NS, MX and A RRs) that corresponds to the XML Schema we just described is given below:

Table 1 - Zone File Content Scenarios and Security Impacts

Data Content Relationships (DCRs)		Security Impact (SI)	
DCR Code	**Description**	**SI Code**	**Description**
DCR1	Unnecessary RRs	SI1	Information availability for Launching targeted attacks
DCR2	Large Parameter Values	SI2	Empty/Obsolete Response (Zone Drift) – Potential Denial of Service
DCR3	Small Parameter Values	SI3	Slow Response/Time Outs (Zone Thrash) – Potential Denial of Service
DCR4	Missing/Ill-formed Associative RRs	SI4	Inaccessible Internet Domain/Service – Denial of Service
DCR5	Incorrect parameter values in digital signatures	SI5 SI6	Non-usability of a security service Security Service bypassed
DCR6	Absence of multiple RRs for critical services	SI4	Inaccessible Internet Domain/Service – Denial of Service due to low fault tolerance
DCR7	Ambiguous Data	SI7	Low/Marginal Risk of Denial of Service mainly due to lack of a policy for interpreting response at the Resolver

Table 2 - Zone File Integrity Constraints and Security Impact Codes

Constraint Code	**Constraint Type**	**Description**	**Security Impact Code (due to constraint violation)**
ZFC1	ICType1	SOA RDATA refresh value be between 2-12 hours	SI2 – if too large SI3 – if too low
ZFC2	ICType1	SOA RDATA retry value is a fraction of refresh value.	SI2 – if too large SI3 – if too low
ZFC3	ICType1	SOA RDATA expiry value is between 2-4 weeks	SI2
ZFC4	ICType1	SOA RDATA Min TTL value is between 5min-1week	SI2 – if too large SI3 – if too low
ZFC5	ICType1	RRSIG TTL is greater than 30 seconds	SI3
ZFC6	ICType1	A/AAAA RDATA contains valid IPv4 /IPv6 address	SI4
ZFC7	ICType1	DNSKEY RDATA has protocol field set to 3	SI5
ZFC8	ICType1	DNSKEY RDATA has algorithm field code set to IANA assigned value	SI5
ZFC9	ICType1	RRSIG RDATA has validity period greater than 0	SI5
ZFC10	ICType1	RRSIG is currently valid (current time between inception time and expiration time)	SI5
ZFC11	ICType1	RRSIG original TTL field must have the same value its TTL should be at the time of its generation.	SI5
ZFC12	ICType2	Zone contains no HINFO RRs	SI1
ZFC13	ICType2	Zone contains 2 or more NS RRs for a domain	SI4
ZFC14	ICType2	Zone does not contain 2 or more MX RRs having same priority field value	SI7
ZFC15	ICType2	Zone does not contain two or more A RRs with same owner name	SI7
ZFC16	ICType2	Zone contains no CNAME Chains	SI7
ZFC17	ICType3	Every NS RR has target name that is an owner name for an A/AAAA RR	SI4
ZFC18	ICType3	Every MX RR has target name that is an owner name for an A/AAAA RR	SI4
ZFC19	ICType3	RRSIG TTL value should be the same value as the TTL of the RRset the RRSIG covers	SI5
ZFC20	ICType3	RRSIG “signer” field should be the owner name of the DNSKEY RR used to validate the signature.	SI5
ZFC21	ICType3	The RRSIG type covered field should have the correct RRtype value corresponding to the RRset it covers	SI5
ZFC22	ICType3	The RRSIG labels value should match the number of labels in the RRset owner name it covers	SI5
ZFC23	ICType3	Primary server in SOA RDATA also appears in RDATA of NS RR	SI7
ZFC24	ICType3	NS RDATA target name should not be the owner name of a CNAME RR	SI7
ZFC25	ICType3	MX RDATA target name should not be the owner name of a CNAME RR	SI7

```
<?xml version="1.0" encoding="UTF-8"?>
<Zone_File_Schema
xmlns:xsi="Zone_File_Schema.xsd">
<NS>
<RRHeader Owner="example.com" TTL="86400"
            CLASS="IN" RRType="NS"/>
 <NS_RDATA>
         <NSHost>ns2.example.com</NSHost>
 </NS_RDATA>
</NS>
<MX>
<RRHeader Owner="example.com" TTL="86400"
            CLASS="IN" RRType="MX"/>
 <MX_RDATA>
         <Priority>10</Priority>
         <MXHost>mail.example.com</MXHost>
 </MX_RDATA>
 <MX_RDATA>
         <Priority>10</Priority>
         <MXHost>mail2.example.com</MXHost>
 </MX_RDATA>
<MX_RDATA>
         <Priority>20</Priority>
         <MXHost>mail3.example.com</MXHost>
 </MX_RDATA>
</MX>
<A>
<RRHeader Owner="ns" TTL="86400" CLASS="IN"
            RRType="A"/>
 <A_RDATA>
         <IPAddress>192.192.249.1</IPAddress>
 </A_RDATA>
</A>
```

The complete XML Schema of the zone file and an XML encoded zone file data that corresponds to this schema for the zone example.com is given in the authors' website.

6. Zone File Integrity Constraints in XSLT & Zone File Validation

The next step in our DNS zone file integrity verification framework is to encode the integrity constraints (ZFC1 through ZFC25 in section 4) in XSLT. Due to lack of space here, we just illustrate the development of XSLT encoding for two constraints one from each of the types ICType2 and ICType3.

Constraint ZFC14 states that there should not be two mail hosts with the same priority. This is a ICType2 constraint and encoded as follows:

```
<xsl:for-each select="/Zone_File_Schema/MX">
<xsl:variable name="Owner"
    select="./RRHeader/@Owner"></xsl:variable>
    <xsl:for-each select="./MX_RDATA">
        <xsl:variable name="Priority"
             select="./Priority"></xsl:variable>
        <xsl:variable name="MXHost"
             select="./MXHost"></xsl:variable>
         <xsl:variable name="EqualPriorityServers"
select="count(/Zone_File_Schema/MX[./RRHeader/@
Owner = $Owner]/MX_RDATA/Priority[text ( ) =
$Priority])">
          </xsl:variable>
         <xsl:if test="$EqualPriorityServers > 1">
Violation: More than one Mail Server with Priority:
<xsl:value-of select="$Priority"/> in the domain
<xsl:value-of select="$Owner"/>. One of them is:
<xsl:value-of select="$MXHost"/>
    </xsl:if>
    </xsl:for-each>
</xsl:for-each>
```

Constraint ZFC17 is an ICType3 constraint that checks whether every name server host (in a NS RR) has a corresponding address record (in an A/AAAA RR). The XSLT encoding of this constraint is as follows:

```
<xsl:for-each select="/Zone_File_Schema/NS">
   <xsl:variable name="Owner"
       select="./RRHeader/@Owner"></xsl:variable>
     <xsl:for-each select="./NS_RDATA">
       <xsl:variable name="NSHost"
          select="./NSHost"></xsl:variable>
         <xsl:variable name="ARecords"
select="count(/Zone_File_Schema/A[./RRHeader/@O
wner = $NSHost])">
            </xsl:variable>
         <xsl:if test="$ARecords = 0">
Violation: The name server host: <xsl:value-of
select="$NSHost"/> in the domain <xsl:value-of
select="$Owner"/> does not have an A RR.
```

The last step in our integrity verification framework is to apply the XSLT constraints against the XML encoded zone file data. We used the public domain XSLT processor Xalan[11]. Referring our subset of XML encoded zone file, we find that there are two MX RRs with the same value for priority data item of RDATA field. Applying the XSLT encoding of constraint ZFC14, the XSLT processor generated the following output.

```
Violation: More than one Mail Server with Priority: 10 in the domain example.com. One of them is: mail.example.com
Violation: More than one Mail Server with Priority: 10 in the domain example.com. One of them is: ns.example.com
```

As an example of ICType3 constraint violation, we find in the XML encoded zone file that there is no A/AAAA RR corresponding to a name server host ns2.example.com specified in the NS RR. The application of the XSLT encoded constraint ZFC17 detected this and generated the following violation message:

```
Violation: The name server host: ns2.example.com in the domain example.com does not have an A RR.
```

7. Comparison with Related Work

In spite of its security implication and its overall impact on the trust of the name resolution service, the authors were surprised to find that there is very little published work or commercial-grade software for integrity verification of DNS zone files. Most of the earlier work on checks (such as the DNS MIB extensions) focused on the resolvers and name servers, not the zone data that makes up the DNS. The only public domain framework and toolkit that we are aware at this time that performs integrity checking on DNS zone files is the Integrity Checker Tool (written in Java) developed at NIST [3]. This web-enabled tool can perform integrity checks on both DNSSEC-aware and non DNSSEC-aware zones, but requires access to a DNS name server for some tests. Mice&Men [2] have a tool that performs similar checks, but not for DNSSEC-aware zones (as of the time of writing). There are also some perl scripts called "DNSSEC Walker" [1] that checks RRSIG validation for every RRset in a zone, but not the integrity of DNS RRs. The framework described here enables performance of these integrity checks even before generating the zone file that is hosted on the authoritative name server. The integrity-checked XML encoded zone file data can then used to generate the zone file using XSLT transforms. Additionally it is possible to encode an existing zone file for a zone using our XML Schema. The encoded file can then be checked for satisfaction of our set of constraints, modified to satisfy the constraints if they do not, and the zone file can then be regenerated from it using XSLT transforms and then reloaded back into the authoritative name server for the zone. The three operations – XML-encoding of the existing zone file, integrity verification using XSLT constraints and transformation of the integrity-checked XML file back to the zone file format can all be integrated into a single software task. The estimate of the overall latency involved in the task can then be used to adjust certain timing parameters (e.g., TTL) in the zone file so that the overall continuity of operations of the DNS name server is not affected.

8. Conclusions & Scope for Future Work

In this paper we have considered integrity constraints pertaining to RRs from a single zone file. There are some integrity constraints that need to be satisfied across zone files in the case of DNSSEC-aware name servers. For example the DS RR in a parent zone should refer to one of the valid keys (DNSKEY RR) in the child zone. Thus we have to deal with two XML files – one is the encoding of zone file for the child zone and the other for the parent zone. Our framework can handle this task since XSLT constraints can be formulated to refer to any number of XML documents within a single constraint. We intend to make use of this feature to verify whether the delegation function (signature of a valid public key of the child by the parent through a DS RR) has been properly implemented across a parent-child zone pair. Repeated application of this type of integrity check will enable identification of valid authentication chains. This identification will then facilitate verification of trust anchors in DNS resolvers for their usefulness in verifying signatures in signed responses coming from a specific zone.

9. References

[1] DNSSEC Zone Walker from Simon Josefsson., http://josefsson.org/walker/
[2] Mice and Men Tool, http://www.miceandmen.com
[3] NIST DNSSEC project home page: http://www-x.antd.nist.gov:8080/dnssec/main.html
[4] RFC 1035, "Domain Names – Implementation and Specification", http://www.ietf.org/rfc/rfc1035.txt?number=1035
[5] RFC 1912," Common DNS Operational and Configuration Errors", http://www.ietf.org/rfc/rfc1912.txt?number=1912
[6] RFC 2845," Secret Key Transaction Authentication for DNS (TSIG)", http://www.ietf.org/rfc/rfc2845.txt?number=2845
[7] RFC 3833, "Threat Analysis of the Domain Name system (DNS"," http://www.ietf.org/rfc/rfc3833.txt?number=3833
[8] RFC 4033, "DNS Security Introduction and Requirements", http://www.ietf.org/rfc/rfc4034.txt?number=4033
[9]RFC 4034, "Resource Records for the DNS Security Extensions", http://www.ietf.org/rfc/rfc4034.txt?number=4034
[10]RFC 4035, "Protocol Modifications for the DNS Security Extensions", http://www.ietf.org/rfc/rfc4035.txt?number=4035
[11] Xalan, http://xml.apache.org/xalan-j/
[12] XML Schema, http://www.w3.org/XML/Schema
[13] XSLT, http://www.w3.org/Style/XSL/