Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 20, 2011

O. Kolkman
NLNet
J. Peterson
NeuStar, Inc.
H. Tschofenig
Nokia Siemens Networks
B. Aboba
Microsoft Corporation

October 17, 2010

Comment [DT1]: RFC 4845 states "Generally speaking, the IAB does not publish Standards-Track or Experimental RFCs."

Architectural Considerations on Application Features in the DNS
draft-iab-dns-applications-00

Abstract

While the principal purpose of the Domain Name System (DNS) is to translate Internet domain names to IP addresses, over time a number of Internet applications have integrated supplemental features into the DNS to support their operations. Many of these features assist in locating the appropriate service in a domain, or in transforming intermediary identifiers into names that the DNS can process. Proposals to piggyback more sophisticated application behavior on top of the DNS, however, have raised questions about the propriety of instantiating some features in the DNS, especially those with security sensitivities. This document explores the architectural consequences of installing application features in the DNS, and provides guidance for future work in this area.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 20, 2011.

Copyright Notice

Table of Contents

1. Terminology

In this document, the key words "MAY", "MUST, "MUST NOT", "SHOULD", and "SHOULD NOT", are to be interpreted as described in [RFC2119].

Comment [DT2]: I think this should be removed.

2. Motivation

The Domain Name System (DNS) has long provided a general means of translating easily-memorized domain names into numeric Internet Protocol addresses, which made the Internet easier to use by providing a valuable layer of indirection between well-known names and lower layer protocol elements. [RFC0974], however, documented a further use of the DNS: to manage application services operating in a domain with the MX resource record, which helped email addressed to the domain to find an authoritative mail service for the domain.

The MX record was the first of a long series of DNS resource records that supported applications associated with a domain name. The SRV resource record provided a more general mechanism for identifying services in a domain, complete with a weighting system and selection among transports. The NAPTR resource record, especially in its reincarnation as the DDDS framework, added a new wrinkle - a way of casting any sort of string as a domain name, which might then be "resolved" by the DNS to find NAPTR records. This enabled the resolution of identifiers other than traditional domain names through the DNS; the best-known example of this are telephone numbers, as resolved by the DDDS application ENUM. Recent work such as DKIM has enabled security features of applications to be advertised through the DNS, via the TXT resource record.

As the amount of application intelligence in the DNS has increased, however, some proposed extensions have become misaligned with the foundational assumptions of the DNS. One such assumption is that the resolution of traditional domain names to IP addresses is public information, lacking any confidentiality requirement - any security needed by an application or service is invoked after the service has been contacted. Typically, the translation is also global information, meaning that the response to a resolution does not depend on the identity of the querier (although for load balancing reasons or related optimizations, the DNS may return different addresses in response to different queries, or even no response at all, which is discussed further below). These assumptions permit the existence of a single authoritative unique global root of the DNS, and also underlie the scaling capabilities of the DNS, notably the ability of intermediaries to cache responses. At the point where these assumptions no longer apply to the data that an application requires, one can reasonably question whether or not that application should use the DNS to deliver that data.

Increasingly, however, the flexibility of the DDDS framework has encouraged the repurposing of the DNS into a generic database. Since the output of DDDS can be a URI, and URIs themselves are containers for basically arbitrary data, through the DDDS framework one can

Comment [DT3]: Reference RFC 2915?

Comment [DT4]: Expand acronym and Reference RFC 3401

Comment [DT5]: Expand acronym and add reference (RFC 4871?)

Comment [DT6]: Actually this warrants a discussion of (or reference to a discussion of) the validity of that assumption, as that assumption doesn't match reality today. Split-horizon and two-faced DNS are common today.

Comment [DT7]: Reference RFC 2826?

Comment [DT8]: Expand acronym and reference RFC 3986

query for an arbitrary string (provided it can be formatted and contained within the syntactical limits of a domain name) and receive as a response an equally arbitrary chunk of data. The use of the DNS for generic database lookups is especially attractive in environments that already use the DDDS framework, where deployments would prefer to reuse the existing query/response interface of the DNS over installing a new and separate database capability.

The guidance in this document complements the guidance on extending the DNS given in [RFC5507]. Whereas RFC5507 considers the preferred ways to add new information to the underlying syntax of the DNS (such as defining new resource records or adding prefixes or suffixes to labels), the current document considers broader implications of offloading application features to the DNS, be it through extending the DNS or simply reusing existing protocol capabilities. It is the features themselves, rather than any syntactical representation of those features, that are considered here.

3. Overview of DNS Application Usages

While the fundamental motivation for the Domain Name System was to replace lengthy numeric addresses with strings that are easier to interpret and memorize, the hierarchical system of hosts and domains rendered the DNS important for its administrative properties as well as its mnemonics. In so far as the DNS explained how to reach an administrative domain rather than simply a host, it naturally extended to optimize for reaching particular applications within a domain. Without these extensions, a user trying to send mail to a foreign domain, for example, lacked a discovery mechanism to locate the right host in the remote domain to connect to for mail. While such a discovery mechanism could be built by each such application protocol, the universality of the DNS invites installing these such features in its public tree.

3.1. Locating Services in a Domain

The Mail Exchange (MX) DNS resource record provides the simplest motivating example for an application advertising its host in the Domain Name System. The MX resource record contains the hostname of a server within the administrative domain in question that receives mail; that hostname must itself be resolved to an IP address through the DNS in order to reach the mail server. While naming conventions for applications might serve a similar purpose (a host might be named "mail.example.com" for example), approaching service location through the creation of a new resource record yields several important benefits. Firstly, one can put multiple MX records in a zone, in order to designate backup ~~hosts~~ servers that can receive mail when the primary server is offline. One can even load balance across several such ~~hosts~~servers. These properties could not easily be captured by naming conventions (see [RFC4367]).

While the MX record represents a substantial improvement over naming conventions as a means of service location, it remains specific to a single application. Thus, the general approach of the MX record was adapted to fit a broader class of application through the Service (SRV) resource record (originally [RFC2052]. The SRV record allows DNS resolvers to search for particular applications and underlying transports (for example, HTTP running over TLS) and to learn the domain ~~host~~name and port where that service resides in a given administrative domain. It also provides a weighting mechanism to allow load balancing across several (presumably equivalent) instances of a service in a domain.

The reliance of applications on the existence of MX and SRV records has important implications for the way that applications manage identifiers. Email identifiers of the form "user@domain" require the

Comment [DT9]: One word: “Insofar”

Comment [DT10]: Move expansion back to first use (section 2).

Comment [DT11]: Should be plural “hosts”

Comment [DT12]: This is arguable. Better to use the term “domain name” for consistency with RFC 1035.

Comment [DT13]: Reference RFC 1794

Comment [DT14]: RFC 2818?

presence of MX records to provide the convenience of simply specifying that "domain" component rather than a "host.domain" structure. While for applications like HTTP, naming conventions continue to abound ("www.example.com"), the SRV algorithm queries for the invoked protocol based, for protocol like HTTP derived from the URI~~I~~ scheme of the identifier invoked by the application. The application thus retained sole responsibility for carrying the desired protocol and domain, but could offload to the DNS the location of the host of that service within the domain, the port where the service resided on that host, load balancing and fault tolerance, and related application features. Ultimately, resolvers that acquire MX or SRV records use them as intermediate transformations in order to arrive at an eventual domain ~~host~~name that will resolve to the IP address of a host to contact for the service.

[TBD: Potentially incorporate some discussion of Hammer's hostmeta or the Webfingers approach as alternatives to using the DNS to identify additional resources required for services]

3.2. NAPTR and DDDS

The Naming Authority Pointer (NAPTR, originally [RFC2168]) record evolved to fulfill a need in the transition from Uniform Resource Locators (URLs) to the more mature Uniform Resource Indicator~~s~~ (URI~~s~~) framework, which incorporated Uniform Resources Names (URNs). Unlike URLs, URNs typically do not convey enough semantics internally to resolve them through the DNS, and consequently a separate URI-transformation mechanism is required to convert these types of URIs into domain names. This allowed identifiers with no recognizable domain component to be resolved on the Internet. Once these transformations resulted in a domain name, applications could receive NAPTR records from that zone of the DNS. NAPTR records contain a far more rich and complex structure than ~~the~~ MX or SRV resource records. A NAPTR record contain~~ed~~ two different weighting mechanisms ("order" and "preference"), a "service" field to designate the application that the NAPTR record described, and then two fields that could contain translations: a "replacement" or "regular expression" field, only one of which appeared in given NAPTR record. A "replacement," like NAPTR's ancestor the PTR record, simply designated another zone where one would look for records associated with this service in the domain. The "regexp," on the other hand, allowed sed-like transformations on the original URI intended to transform it into an identifier that the DNS could resolve.

The same mechanism could obviously be applied to other sorts of identifiers that lacked a domain component, and thus this work naturally combined with activities to create a system for resolving telephone numbers on the Internet, which became known as ENUM

Comment [DT15]: Can't parse grammar here

Comment [DT16]: Move acronym expansion and reference back to first use

Comment [DT17]: This should not be assumed to be a term well-known to readers.

(originally [RFC2916]). ENUM borrowed from an earlier proposal, the "tpc.int" domain ~~(~~[RFC1530]~~)~~, which provided a means for encoding telephone numbers as domain names applying a string preparation algorithm that required reversing the digits and treating each individual digit as a zone of the DNS - thus, for example, the number +15714345400 became 0.0.4.5.4.3.4.1.7.5.1.tpc.int. In the ENUM system, in place of "tpc.int" the special domain "e164.arpa" was reserved for use. In its more mature form in Dynamic Delegation and Discovery Service (DDDS) ([RFC3401] passim) framework, this initial transformation was called the "First Well Known Rule." Its flexibility has inspired a number of proposals beyond ENUM to encode and resolve unorthodox identifiers in the DNS. Provided that the identifiers transformed by the First Well Known Rule have some meaningful hierarchical structure and are not overly lengthy, virtually anything can serve as an input for the DDDS structure: for example, civic addresses (see draft-rosen-dns-sos).

Comment [DT18]: Move expansion and reference back to first use.

Comment [DT19]: Turn this into a reference

The presence of the "regexp" field of NAPTR records enabled unprecedented flexibility in the transformations that DNS resolution could perform. Since the output of the regular expression frequently took the form of a URI (in ENUM resolution, for example, a telephone might be converted into a SIP URI), anything that could be encoded as a URI might be the result of resolving a NAPTR record. Since URI encoding has ways of carrying basically arbitrary data (see for example, the base 64 encoded binary data in the data URL), resolving a NAPTR record might result in an output other than an identifier which would subsequently be resolved to an IP address and contacted for a particular application - it could give a literal result consumed by the application. For a query-response application, the DNS could effectively implement the entire application feature set. Effectively, the DDDS framework turned the DNS into a generic database - indeed, the DNS serves as but one example of a possible back-end for DDDS, and perhaps not the most suitable one.

Comment [DT20]: Reference?

3.3. Arbitrary Data in the DNS

NAPTR did not pioneer the possibility of storing arbitrary data in the DNS. [RFC1464] defined the TXT record, a means to store arbitrary string data in the DNS using a simple attribute name/value pair syntax. The existence of TXT records has long provided new applications with a rapid way of storing data associated with a domain name in the DNS, as the attribute names require no registration process and can simply be minted at will. This, an application that wants to store additional data in the DNS can do so without registering a new resource record type.

While the laxity of policies surrounding the use of the TXT record has resulted in a checkered past for standardizing application usage

of TXT, it has provided a technical solution for DKIM ([RFC4871]) to store cryptographic keys for email in DNS. Storing keys in the DNS for DKIM made sense for several reasons: notably, because the public keys associated because email required wide public distribution, and because email identifiers contain a domain component that applications can easily use to consult the DNS. If the application had to negotiate support for the DKIM mechanism with mail servers, it would give rise to bid-down attacks that are not possible if the DNS delivers the keys (provided that DNSSEC guarantees authenticity of zone files).

4. Challenges for the DNS

These methods for transforming arbitrary identifiers into domain names, and for returning arbitrary data in response to DNS queries, both represent significant extensions from the original concept of the DNS, yet neither fundamentally alters the underlying model of the DNS. The promise that applications might rely on the DNS as a generic database, however, invariably gives rise to additional requirements that one might expect to find in a database access protocol: authentication of the source of queries for comparison to access control lists, formulating complex relational queries, and asking questions about the structure of the database itself. DNS was not designed to provide these sorts of properties, and extending the DNS to encompass them would represent a fundamental alteration to its model. If an application desires these properties from a database, in general this is a good indication that the DNS cannot meet the needs of the application in question.

Since many of these new requirements have emerged from the ENUM space, the following sections use ENUM as an illustrative example; however, any application using the DNS as a feature-rich database could easily end up with similar requirements.

4.1. Compound Queries

Traditionally, the DNS requires resolvers to supply no information other than the domain name (and the type and class of records sought) in order to receive a reply from an authoritative server. There are, however, plenty of query-response applications in deployments that require a compound search, which takes into account more than one factor in formulating a response. For example, in the telephony space, telephone call routing often takes into account numerous factors aside from the dialed number, including originating trunk groups, interexchange carrier selection, number portability data, time of day, and so on. While in its original conception, ENUM hoped to circumvent the traditional PSTN and route directly to Internet-enabled devices, the infrastructure ENUM effort to support the migration of traditional carrier routing functions to the Internet aspires to achieve feature parity with traditional number routing. Consequently, some consideration has been given to ways to add additional data to ENUM queries to give the DNS server sufficient information to return a suitable URI.

Several workaround have attempted to instantiate these sorts of features in the DNS. Most commonly, proposals piggyback additional query parameters as eDNS0 extensions (see [RFC2671]). Alternatively, the domain name itself can be compounded with the additional parameters: one could take a name like

0.0.4.5.4.3.4.1.7.5.1.e164.arpa and append a trunk group identifier to it, for example, of the form tg011.0.0.4.5.4.3.4.1.7.5.1.e164.arpa. While in the latter case, a DNS server can adhere to its traditional behavior in locating resource records, the syntactical viability of encoding additional parameters in this fashion is very dubious, especially if more than one additional parameter is required and the presence of parameters is optional. The former case requires significant changes to DNS server behavior. Moreover, the implications for these sorts of compound queries on recursion and caching are potentially serious.

4.1.1. Responses Tailored to the Originator

The most important subcase of the compound queries are DNS responses tailored to the identity of their originator, where some sort of administrative identity of the originator must be conveyed to the DNS. Often the source IP address is somehow mapped to the administrative entity of the originator in deployments with this requirement today. The security implications of trusting the source IP address of a DNS query have prevented most solutions along these lines from being standardized. Some applications require an application-layer identifier of the originator rather than an IP address; for example, draft-kaplan-enum-source-uri provides a SIP URI in an eDNS0 parameter (though without any specific provision for cryptographically verifying the claimed identity). Effectively, the conveyance of this information about the administrative identity of the originator is a weak authentication mechanism, on the basis of which the DNS server makes an authorization decision before sharing resource records. This can parlay into a selective confidentiality mechanism, where only a specific set of originators are permitted to see resource records, or a case where a query for the same name by different entities results in completely different resource record sets. The DNS, however, substantially lacks the protocol semantics to manage access control list for data, and again, caching and recursion introduce significant challenges for applications that attempt to offload this responsibility to the DNS. Achieving feature parity with even the simplest authentication mechanisms available at the application layer would like require significant rearchitecture of the DNS.

Comment [DT21]: This section is missing mention of perhaps the most common case, where there are private DNS clouds, and so depending on *where* you are (not who you are), you either get the answer or you don't. This is essentially a type of access control list. And in some cases (like in Windows' DirectAccess) hosts are configured with DNS servers that require IPsec and aren't in the public tree and hence do provide access control to namespaces.

Comment [DT22]: Also reference draft-ietf-intarea-shared-addressing-issues

Comment [DT23]: Turn this into a reference

4.2. Metadata about Tree Structure

ENUM use cases have also surfaced a couple of optimization requirements to reduce unnecessary calls and queries by including metadata that describes the contents and structure of ENUM DNS trees. In particular, the "send-n" proposal (draft-bellis-enum-send-n) hopes to reduce the number of DNS queries sent in cases where a telephone system is collecting dialed digits in a region that supports

"overlap" dialing, a form of variable-length number plan. In these plans, a telephone switch ordinarily cannot anticipate when a dialed number is complete, as only the terminating customer premise equipment (typically a private branch exchange) knows how long a telephone number needs to be. The "send-n" proposal offloads to the DNS the responsibility for informing the telephone switch how many digits must be collected by placing in zones corresponding to incomplete telephone numbers some resource records which state how many more digits are required - effectively how many steps down the DNS tree one must take to reach a name for a complete number. With this information, the application is not required to query the DNS every time a new digit is dialed, but can wait to collect sufficient digits to receive a response. A tangentially related proposal, draft-ietf-enum-void, similarly places resource records in the DNS that tell the application that it need not attempt to reach a number on the PSTN, as the number is unassigned.

Both proposals optimize application behavior by placing metadata in the DNS that predicts the success of future queries or application invocations. These predictions require that the metadata remain synchronized with the state of the resources it predicts. Maintaining that synchronization, however, requires that the DNS have semi-real time updates that may conflict with scale and caching mechanisms. It is unclear why this data is better maintained by the DNS than in an unrelated application protocol.

4.3. Using DNS as a Generic Database

As previously noted, the use of the First Well Known Rule of DDDS combined with data URLs effectively allows the DNS to answer queries for arbitrary strings and to return arbitrary data as value. Some query-response applications, however, require queries and responses that simply fall outside the syntactic capabilities of the DNS. While the data URL specification (RFC2397) notes that it is "only useful for short values," many applications today use quite large data URLs as workarounds in environments where only URIs can be interpolated. While the use of TCP and eDNS0 allows DNS responses to be quite long, nonetheless there are forms of data that an application might store in the DNS that exceed reasonable limits: in the ENUM context, for example, something like storing base 64 encoded mp3 files of custom ringtones. Similarly the domain names themselves must conform with certain syntactic constraints: they must consist of labels that do not exceed 63 characters while the total length of the encoded name may not exceed 255 octets, they must obey fairly strict encoding rules, and so on.

4.3.1. Administrative Structures Misaligned with the DNS

While the DDDS framework enables any sort of alphanumeric data to serve as a DNS name through the application of the First Well Known Rule, the delegative structure of the resulting DNS name may not reflect the division of responsibilities for the resources that the alphanumeric data indicates. Telephone numbers in the United States, for example, are assigned and delegated in a relatively complex manner: the first three digits of a nationally specific number are an "area code" which is understood as an indivisible component of the number, yet for the purpose of the DNS, those three digits are ranked hierarchically.

4.4. Domain Redirection

Most Internet application services provide a redirection feature - when you attempt to contact a service, the service may refer you to a different service instance, potentially in another domain, that is for whatever reason better suited to address a request. In HTTP and SIP, for example, this feature is implemented by the 300 class responses containing one or more better URIs that may indicate that a resource has moved temporarily or permanently to another service. Tools in the DNS like the SRV record, however, can provide a similar feature at a DNS level, and consequently some applications as an optimization offload the responsibility for redirection to the DNS; NAPTR can also provide this capability on a per-application basis, and numerous DNS resource records can provide redirection on a per-domain basis. This can prevent the unnecessary expenditure of application resources on a function that could be performed as a component of a DNS lookup that is already a prerequisite for contacting the service. Consequently, in some deployment architectures this DNS-layer redirection is used for virtual hosting services.

Implementing domain redirection in the DNS, however, has important consequences for application security. Un the absence of universal DNSSEC, applications must blindly trust the DNS in order to believe that their request has not been hijacked and redirected to a potentially malicious domain, unless some subsequent application mechanism can provide the necessary assurance. By way of contrast, for application-layer redirections protocols like HTTP and SIP have widely deployed security mechanisms such as TLS that can use certificates to vouch that a 300 response came from the domain that the originator initially hoped to contact.

A number of applications have attempted to provide an after-the-fact security mechanism that verifies the authority of a DNS delegation in the absence of DNSSEC. The specification for deferencing SIP URIs

([RFC3263], reaffirmed in [RFC5922]) requires that during TLS establishment, the site eventually reached by a SIP request present a certificate corresponding to the original URI expected by the user (in other words, if example.com redirects to example.net in the DNS, this mechanism expects that example.net will supply a certificate for example.com in TLS), which requires a virtual hosting service to possess a certificate corresponding to the hosted domain. This restriction rules out many styles of hosting deployments common the web world today, however. [I-D.barnes-hard-problem] explores this problem space, and [I-D.saintandre-tls-server-id-check] proposes a solution for all applications that use TLS. Potentially, new types of certificates (similar to [RFC4985] might bridge this gap, but support for those certificates would require changes to existing certificate authority practices as well as application behavior.

All of these application-layer measures attempt to mirror the delegation of authority in the DNS, when the itself DNS serves as the ultimate authority on how domains are delegated. The difficulty of synchronizing a static instrument like a certificate with a delegation in the DNS, however, exposes the fundamentally problematic nature of this endeavor. In environments where DNSSEC is not available, the problems with securing DNS-layer redirections would be avoided by performing redirections in the application-layer.

5. Principles and Guidance

The success of the DNS relies on the fact that it is a distributed database that has the property that it is loosely coherent and that it offers lookup instead of search functionality. Loose coherency means that answers to queries are coherent within the bounds of data replication between authoritative servers and caching behavior by recursive name servers.

It is likely that the DNS provides a good match whenever applications needs are aligned with the following properties:

- Data can be stored in such a way that a single query name and query type provide a direct answer
- Answers are only depended on the question (name and type), not on the location or identity of the entity doing the query
- Data stored in the DNS is resilient to data propagation and caching behavior

Whenever one of the 3 properties does not apply to ones data one should seriously consider whether the DNS is the best place to store actual data. On the other hand, good indicators that the DNS is not the appropriate tool for solving problems is when you have to worry about:

- Trying to establish domain boundaries within the tree - the delegation point in the DNS is something that applications should in general not be aware off
- Having to do more than one, two queries to get to ones answer (consider a chain of NAPTR records)
- The sensitivity of the data provided by the DNS
- Highly volatile data
- Location- or identity-dependent answers

There are many useful application features that can safely be offloaded to the DNS: aside from locating services in a domain, the DNS clearly can assist in the resolution of identifiers without a domain component (including URNs), and moreover it can host some static application data, like the cryptographic keys used by DKIM for email, which are well suited to storage in the DNS. However, the prospects for offloading application features like authentication of query originators, structuring compound questions and implementing

metadata about the tree structure are more remote. While clearly DNS-layer redirection is a widely deployed alternative to application-layer redirection, many applications that choose to offload this have struggled to meet the resulting security challenges.

In cases where applications require these sorts of features, they are simply better instantiated by independent application-layer protocols than the DNS. The query-response semantics of the DNS are easily replicated by HTTP, for example, and the objects which HTTP can carry both in queries and responses can easily contain the necessary structure to manage compound queries. Similarly, HTTP has numerous ways to provide the necessary authentication, authorization and confidentiality properties that some features require.

Where the administrative delegations of the DNS form a necessary component in the instantiation of an application feature, there are various ways that the DNS can bootstrap access to an independent application-layer protocol better suited to field the queries in question. For example, since NAPTR records can contain URIs, those URI can point to an external query-response service such as HTTP, with a NAPTR service field that signal to applications that questions of interest can be answered at that service.

5.1. Private DNS Deployments

Today, many deployments that want to install these application features in DNS do so in private environments rather than in the public DNS tree. There are two motivations for this: in the first place, proprietary non-standard parameters can easily be integrated into DNS queries or responses; secondly, confidentiality and custom responses can be provided by deploying, respectively, underlying VPNs to shield the private tree from public queries, and effectively different virtual DNS trees for each administrative entity that might launch a query. In these constrained environments, caching and recursive resolvers can be managed or eliminated in order to prevent any unexpected intermediary behavior.

While these deployments address the requirements of applications that rely on them, practically by definition these techniques will not form the basis of a standard solution. Moreover, as implementations come to support these proprietary parameters, it seems almost certain that these private techniques will begin to leak into the public DNS. Therefore, keeping these features within higher-layer applications rather than offloading them to the DNS is preferred.

6. Security Considerations

Many of the concerns about offloading application features to the DNS revolve around security. Section 4.4 discusses a security problem concerning redirection that has surfaced in a number of protocols (see [I-D.barnes-hard-problem]). The perceived need to authenticate the source of DNS queries (see Section 4.1.1 and authorize access to particular resource records also illustrates the fundamental security principles that arise from offloading certain application features to the DNS.

While DNSSEC, were it deployed universally, can play an important part in securing application redirection in the DNS, DNSSEC does not provide a means for a resolver to authenticate itself to a server, nor a framework for servers to return selective answers based on the authenticated identity of resolvers.

7. IANA Considerations

This document contains no considerations for the IANA.

8. Acknowledgements

The IAB would like to thank Patrik Faltstrom for his contribution.

9. Informative References

[I-D.barnes-hard-problem]
Barnes, R. and P. Saint-Andre, "High Assurance Re-Direction (HARD) Problem Statement", draft-barnes-hard-problem-00 (work in progress), July 2010.

[I-D.saintandre-tls-server-id-check]
Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity in Certificates Used with Transport Layer Security", draft-saintandre-tls-server-id-check-09 (work in progress), August 2010.

[RFC0974] Partridge, C., "Mail routing and the domain system", RFC 974, January 1986.

[RFC1464] Rosenbaum, R., "Using the Domain Name System To Store Arbitrary String Attributes", RFC 1464, May 1993.

[RFC1530] Malamud, C. and M. Rose, "Principles of Operation for the TPC.INT Subdomain: General Principles and Policy", RFC 1530, October 1993.

[RFC2052] Gulbrandsen, A. and P. Vixie, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2052, October 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2168] Daniel, R. and M. Mealling, "Resolution of Uniform Resource Identifiers using the Domain Name System", RFC 2168, June 1997.

[RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.

[RFC2916] Faltstrom, P., "E.164 number and DNS", RFC 2916, September 2000.

[RFC3263] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

[RFC3401] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS", RFC 3401, October 2002.

[RFC4366] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 4366, April 2006.

[RFC4367] Rosenberg, J. and IAB, "What's in a Name: False Assumptions about DNS Names", RFC 4367, February 2006.

[RFC4871] Allman, E., Callas, J., Delany, M., Libbey, M., Fenton, J., and M. Thomas, "DomainKeys Identified Mail (DKIM) Signatures", RFC 4871, May 2007.

[RFC4985] Santesson, S., "Internet X.509 Public Key Infrastructure Subject Alternative Name for Expression of Service Name", RFC 4985, August 2007.

[RFC5507] IAB, Faltstrom, P., Austein, R., and P. Koch, "Design Choices When Expanding the DNS", RFC 5507, April 2009.

[RFC5922] Gurbani, V., Lawrence, S., and A. Jeffrey, "Domain Certificates in the Session Initiation Protocol (SIP)", RFC 5922, June 2010.

Authors' Addresses

Olaf Kolkman
NLNet

Email: olaf@nlnetlabs.nl

Jon Peterson
NeuStar, Inc.

Email: jon.peterson@neustar.biz

Hannes Tschofenig
Nokia Siemens Networks

Email: Hannes.Tschofenig@gmx.net

Bernard Aboba
Microsoft Corporation

Email: bernarda@microsoft.com