Network Working Group
Internet-Draft
Intended status: Informational
Expires: October 18, 2007

O. Kolkman, Ed.
IAB

April 16, 2007

Architectural Concerns on the synthesis of non-existent names in DNS. draft-iab-dns-synthesis-concerns-00

Status of this Memo

Copyright Notice

Abstract

There are many architectural assumptions regarding DNS behavior that are not specified in the IETF standards documents describing DNS, but which are deeply embedded in the behavior as expected by Internet protocols and applications. These assumptions are inherent parts of the network architecture of which the DNS is one component.

It has long been known that it is possible to use DNS wildcards in ways that violate these assumptions. More recently there have been

deployments of middleboxes -- in most cases recursive nameservers or DNS proxies at the ISP level -- that synthesize answers in ways that not only violate these assumptions but also violate the DNS architecture.

Experience with DNS synthesis in the DNS infrastructure have show that the cost of violating these assumptions is significant. In this document we provide an explanation of how DNS wildcards function, and many examples of how their injudicious use negatively impacts both individual Internet applications and indeed the Internet architecture itself. We also explain that similar problems arise with the synthesis of DNS responses by middleboxes.

We recommend that DNS wildcards should not be used in a zone unless the zone operator has a clear understanding of the risks, and that they should not be used without the informed consent of those entities which have been delegated below the zone.

In addition we recommend that middleboxes do not perform DNS query synthesis unless (1)there are informed consents of those that use the forwarding name server, and (2)there exists an opt-out mechanism that allows them to receive the original DNS answers.

Table of Contents

1. DNS Queries and synthesis of answers

The most basic and by far the most common operation in the DNS protocols is a query for all resource records matching a given query name, query class, and query type. Assuming that all the software and networks involved are working correctly, such a query will produce one of three possible results:

no such name: If the system fails to find a match for the given query name and query class, it returns an indication that the name does not exist.

no data: If the system finds a match for the query name and query class it will try to determine if there is data for the query type. If such data can not be found it returns an indication that the name exists but no data matching the given query type is present.

success: If the system finds a match for all three parameters, it returns the matching set of resource records;

Ordinarily, matches for all three parameters must be exact. But here is where synthesis comes into play. Synthesis can take place when there is no exact match on the query name, and possibly also, when there is no match to the query type. Below we discuss two sorts of synthesis: Wildcard synthesis and synthesis in middle-boxes.

1.1. A brief primer on DNS wildcards

The synthesis of answers using the DNS "wildcard" mechanism has been part of the DNS protocol since the original specifications were written twenty years ago, but the capabilities and limitations of wildcards are sufficiently tricky that discussions of both the protocol details of precisely how wildcards should be implemented and the operational details of how wildcards should or should not be used have continued to the present day. This section attempts to explain the essential details of how wildcards work, but readers should refer to the recent publication on "The Role of Wildcards in the Domain Name System" ([RFC4592]) and references therein for the full details.

In essence, DNS wildcards are rules which enable an authoritative name server to synthesize DNS resource records on the fly. The basic mechanism is quite simple, the complexity is in the details and implications.

A wildcard record is an otherwise ordinary DNS resource record whose leftmost (least significant) label consists of a single asterisk ("*") character, such as *.bar.example. Conceptually, the asterisk

matches one or more labels at the left (least significant) end of the DNS name.

When wildcard records are present, the rules become more complicated than an exact match for all three of the query parameters. Specifically, if the query class matches, there is no exact match for the query name, and the closest match for the query name is a wildcard, the system in effect synthesizes a set of resource records matching the query name on the fly by treating the resource records present at the wildcard name as if they had been present at the query name. Thus, if the wildcard name has records matching the desired query type, the system will return those records, precisely as in the "success" case above; otherwise, the system will return an indication that the name exists but no data matching the given query type is present, precisely as in the "no data" case above. The response is identical to that of a normal "success" response for the query name, so the resolver which issued the query can not tell that the results it got back were the result of wildcard expansion.

Note that, in the case of a wildcard match, the "no such name" case cannot occur; the wildcard match eliminates this possibility. Note also that only the query name and query class matter for purposes of determining whether a wildcard matches: any record type can produce a wildcard match, regardless of whether or not the record type happens to match the query type. Finally, in absence of the signature records produced with DNSSEC [RFC4034] a client will not be able to distinguish a wildcard match from a non-wildcard match.

1.2. Synthesis by recursive forwarders or other middle-boxes

This synthesis mechanism is not part of the DNS protocol but rather it is a non-standard modifications to middle-boxes such as recursive name servers, transparent DNS proxies, specific NAT boxes or other devices through which the DNS packets pass. These machines perform a deep packet inspection and substitute locally configured resource records in reply packets. At the moment of writing we are only aware of implementations that perform this synthesis for "no such name" answers but there is no reason to assume that the same mechanism is not applied to responses that are of the "no data" type.

2. Problems with DNS synthesis

One of the main known weaknesses and dangers of synthesis is that it interacts poorly with any use of the DNS which depends on "no such name" responses. The list of such uses turns out to be quite large, and will be discussed in some detail in a later section.

2.1. Problems specific to DNS wildcards

A known weakness and danger of wildcard records stems from the fact that the wildcard label will match anything at all, so long as no non-wildcard name within the zone is a closer match to the query name than the wildcard is. This doesn't sound like a major problem until one considers the number of conventions and, in some cases, protocols, which use labels at the left (least significant) ends of the names of resource records to distinguish between records associated with different services, rather than using different types of records. That is, in these cases, otherwise unrelated services use the same type of record and clients (or users) are expected to use the name corresponding to the particular service desired. This applies both to the ad-hoc naming conventions described in [RFC2219] such as www.foo.example and also to mechanisms such as the SRV record type [RFC2782] in which the naming scheme is part of the formal protocol. When names of this type are covered by wildcards such as an address record named *.bar.example, such a wildcard would hand back the same address record regardless of the service name encoded in the query name, thus ftp.foo.bar.example, mail.foo.bar.example, ntp.foo.bar.example and so forth would all end up with the same synthesized address record. This problem is even worse in the SRV case. For example, suppose there is a SRV record owned by a wildcard that directs all traffic to port 80 on a certain machine. Then suppose that SRV records were defined for the finger protocol and a client issued a query for _finger._tcp.foo.bar.example that got resolved by the before mentioned wildcard pointing to port 80. This causes the client to contact the wrong host and connect to a port on which the finger protocol is not available. The only way to avoid these problems with names of this type is to add explicit records for such names to the DNS.

Finally, the two factors listed above ("match anything" behavior, and poor interaction with anything that depends on "no such name" responses) interact with normal and predictable human behavior to allow wildcards to have effects far beyond their intended scope. Properly speaking, a wildcard record's scope is limited to a single zone, since, by definition, a wildcard record never matches any name that really does exist in the zone, and thus will not match any (non-wildcard) delegation of a portion of the name space from a parent zone to its child. (The behavior of wildcard NS records is not defined, see section 4.2 of [RFC4592].) So, at first blush, it would seem that the administrator of a zone is free to use wildcards without worrying about effects which this might have on the zone's delegated children. Unfortunately, this turns out not to be the case, because DNS names are heavily exposed in user interfaces, and users, being humans, make mistakes. So, while delegating the bar.example zone will prevent a wildcard record *.example from

affecting a user who typed foo.bar.example as foi.bar.example, it will not prevent the same wildcard record from affecting the same user when the error is foo.bat.example. Thus, from the users' point of view, some of the effects of wildcards do leak from a parent zone to its children. This is not a big deal if the parent and child zones are associated with a single organization, but it can become a real problem if the parent and child zones are associated with different organizations whose interests are not perfectly aligned.

The above is probably not an exhaustive list. Even after twenty years of experience with the DNS, the effects of unexpected uses of wildcards can still be quite surprising, because the small but fundamental way in which they change the record lookup rules has a nasty way of violating implicit (or, sometimes, explicit) assumptions in deployed software utilizing the DNS.

For these reasons, almost all use of DNS wildcards has been limited to a relatively small number of reasonably well-understood roles, and most wildcard use has been limited to a single role: the MX records used in mail delivery.

Since MX records are only used for electronic mail delivery, wildcard MX records are relatively safe, and since electronic mail for any particular DNS name is generally handed by the organization that is furthest down the delegation tree, wildcard MX records are most likely to appear in zones where their effects will not cross organizational boundaries. While the latter is not universally true, the primary use of wildcard records has been and remains wildcard MX records for handling an organization's own mail.

Given these issues, it seems clear that the use of wildcards with record types that affect more than one protocol should be approached with caution, that the use of wildcards in situations where their effects cross organizational boundaries should also be approached with caution, and that the use of wildcards with record types that affect more than one protocol in situations where the effects cross organizational boundaries should be approached with extreme caution, if at all.

2.2. Problems specific to synthesis by middleboxes

While the wildcard synthesis is specified as part of the DNS protocol the synthesis by recursive nameservers, proxies and other middle-boxes interacts badly with the protocol itself.

The ultimate client of the DNS information are applications in need for a resources tied to a given name. Usually those application use so called stub resolvers to make that resource available through a

call to an OS supplied library. The stub resolver connects to a recursive name server that will resolve the answer by querying the authoritative name servers. The recursive nameserver is usually deployed in location so that a large number of stub resolvers make use of it (e.g. a by an ISP) and maintains a cache to decrease response times for subsequent queries. In the path between the stub resolver and the recursive nameserver there can be any number of forwarding nameservers, or DNS proxies.

The DNS protocol has been designed around the assumption that the authoritative data records supplied by the authoritative server in response to a query is delivered to the application unmodified. In that sense the authoritative server is one end, and the stub resolver is the other end of the DNS end-to-end connection and there exists a clean line of sight between them. Extensions to the DNS are designed with this principle in mind.

In particular DNSSEC has been designed to allow validation of DNSSEC by the stub resolver, or a similar component in the client operating system. Any modification by middle-boxes to DNS resource records may therefore cause validation failures.

In addition to this architectural issue the synthesis by middle-boxes triggers the same problems that have been encountered recently with the deployment of wildcards at high levels in the DNS tree.

3. Principles To Keep In Mind

In reading the rest of this document, it may be helpful to bear in mind two basic principles of architectural design which have served the Internet well for many years:

The Robustness Principle: "Be conservative in what you do, be liberal in what you accept from others." (Jon Postel, [RFC0793])

The Principle Of Least Astonishment: A program should always respond in the way that is least likely to astonish the user. [Traditional, original source unknown]

We will come back to these points after the next section.

4. Problems encountered in recent experiences with wildcards

In September 2003 we had the opportunity to observe the results of the introduction of the use of wildcards in large and well-established top-level domains, with some rather undesirable and

unintended consequences. This section attempts to detail some of the problems that network users and operators around the world encountered as a result of this deployment of wildcards in the TLD. End-user applications are not able to assess where the synthesis took place, hence the discussion of the problems encountered during the deployment of the wildcard directly applies to synthesis by middle-boxes as well.

While, technically, the synthesis in middle-boxes violates the specifications, we must emphasize that deployment of wildcards in any kind of zone, including a TLD zone, is not such violation. One of our main points here is that simply complying with the letter of the protocol specification is not sufficient to ensure the operational stability of the applications which depend on the DNS: there are protocol features which simply are not safe to use in some circumstances.

The specific change which this operator chose to make was to add a single wildcard address record at the zone apex of each of the affected zones. As a direct result of this change, two things happened:

- the authoritative servers for these two zones no longer give out "no such name" responses for any possible name in these zones, and

- every possible name rooted in one of these zones, which did not exist at all until this change, now has a synthesized address record pointing at a "redirection server" run by the operator of this zone.

The implications of this simple change were many and varied. The list below is almost certainly incomplete:

4.1. Web Browsing

Web browsers all over the world stopped displaying "page not found" in the local language and character set of the users when given incorrect URLs rooted under these TLDs. Instead, these browsers now display an English language search page from a web server run by the zone operator.

It should be noted that the language tags in the HTTP protocol do not always match the locale used in the local browser. So, even though the global search page is dynamic and uses the information in the HTTP request to guess what language and script is to be used -- it will never be able to emulate what the user expected. There is, in short, not enough context in the HTTP protocol for the engine which generates the search page.

In many situations, web browsers have been written to provide some assistance to the user, often based on local conventions, directories, and language, when a DNS lookup fails. All such systems are now disabled for the URLs rooted under these TLDs, since DNS lookups no longer fail, even when the specified destination does not exist.

In addition, the new mechanism has poor scaling properties, and unless the operator chooses to invest significant resources in maintaining a large, robust web server setup, the user experience is going to get even worse: instead of either a local language error message or an English search page, the user is going to get "attempting to connect..." followed by a long wait.

In the case of synthesis by middle-boxes the scalability, and the locality are arguably less significant issues. However one cannot always assume that all users behind a middle-box use the same locale.

4.2. Email

When a wildcard is being deployed at the TLD level all mail to non-existent host names under these TLDs now flows to the registry operator's server, where the registry operator bounces it. Some operators find this intolerable and might change their mail system configurations to bypass this "bounce service", but the vast majority of mail servers undoubtedly now route mail for nonexistent names under these TLDs to the bounce server rather than just bouncing it directly. This has a number of ramifications:

If operators choose to allow their mail to go to the bounce server, they now have an increased mail load handling additional routing of messages to the bounce server; if operators choose not to allow this to happen, they have an additional development and maintenance burden configuring their servers to prevent it.

Operators who allow mail to go to the bounce server are now dependent on the performance of the bounce server. If the bounce server ever slows or fails, mail that previously would bounce will now queue at the SMTP relay for that relay's queue time before bouncing back to the user. This creates a very poor user experience, since typographical errors that in the past would have bounced immediately may now go unnoticed for several days.

Operators who allow mail to go to the bounce server are also dependent on the correct operation of the bounce server. If the bounce server is buggy (which happened to be the case with this roll-out), mail may not bounce at all: it may be reported to the user as having been delivered correctly while actually vanishing without a

trace. This also creates a very poor user experience.

In some cases where the set of MX records associated with a particular DNS name included a misconfigured record pointing to a nonexistent host name, installing these wildcard records was the last straw that broke a misconfigured-but-functional mail configuration: previously, the nonexistent host name would have failed to resolve and been ignored, now it bounces.

The normal flow of data from a client in SMTP when one address has a typo is as follows:

- The client looks up the IP address of its outgoing SMTP proxy in DNS.
- The client opens a TCP connection to its outgoing SMTP proxy.
- The client sends information about itself to the SMTP proxy.
- The proxy accepts or rejects the client.
- The client sends information about the recipient to the SMTP proxy.
- The proxy looks up the destination in DNS, and gets "no such name" back.
- The proxy sends information to the client that the address is wrong.

With a wildcard for mistyped domain, the following happens:

- The client looks up the IP address of its outgoing SMTP proxy in DNS.
- The client opens a TCP connection to its outgoing SMTP proxy.
- The client sends information about itself to the SMTP proxy.
- The proxy accepts or rejects the client.
- The client sends information about the recipient to the SMTP proxy.
- The proxy looks up the destination in DNS, and gets "success" back.

- The proxy accepts the message and closes the connection to the client.
- The proxy opens a TCP connection to the bounce server.
- The proxy present itself to the bounce server.
- The bounce server indicates that the recipient address is not acceptable.
- The proxy generates an error message which is sent back to the sender's email address.

A different scenario happens if the SMTP client has been misconfigured with the incorrect name of the outgoing SMTP proxy. As the domain name resolves using a wildcard, the client will connect to the bounce server, and start to send mail to it. The result is that the bounce server (at the IP address of the wildcard) says that the recipient address is wrong even though it is in fact correct. The error presented to the user is incorrect, as it is the name of the outgoing proxy which was wrong and not the name of the recipient.

Above we have assumed that the mailserver deployed by the TLD server has been configured to bounce the mails. When such server were to be configured to accept mails the privacy issues are obvious. While the deployment of wildcards in a TLD zone can be 'audited' by many Internet users, synthesis by middle-boxes is typically something that only affects the users behind that middle-box. Therefore there is a real risk that (malicious) misconfiguration will go unnoticed and that mail is routed to places where the user did not intend to send it to.

4.3. Informing Users of Errors

Many application GUIs check domain names for validity before allowing the user to progress to the next step. Examples include email clients that directly check the domain of the email addresses resolves before sending, and network printer configuration tools that check that the print spooler name is valid before accepting the configuration. Previously the user would be prompted early that they had made an error in the domain name. In the case of email, the error may now remain unnoticed at the time of sending, till when email bounces back later. In the case of the printer configuration, the error may not be noticed during configuration, but only afterwards when printing fails to work, where the problem diagnosis is more difficult.

4.4. Spam Filters

Installing these wildcard records broke several simple spam filters commonly used to front end inbound mail servers, as well as more complex filtering that checks for the existence of a sending domain in order to screen out obviously bogus senders. This technique for spam has diminished as this filtering mechanism has increased, but one sample operator reports that it still equals about 10% of inbound mail attempts on their large shared MX cluster. ISPs who are aware of this problem will probably extend their filtering rules to have special knowledge of the address returned by these wildcard records, but will have to carry the cost of doing so, both in terms of code maintenance and increased execution time for their filtering.

4.5. Interactions with Other Protocols

The wildcard address records trap DNS lookups for any network service, but the number of protocols somewhere in use on the Internet (including private protocols used between two or more parties on ports which they may or may not have registered with IANA) is large enough that it simply is not possible for the zone operator who traps the DNS loolups using wildcards (or anyone) to provide a redirection service for every protocol. In a recent deployment of a wildcard in a TLD zone, the zone operator only provided handlers for HTTP (which they directed to a search page) and SMTP (which they attempted to bounce). All other protocols received at best ICMP port unreachable message, or, in some cases, simply had their packets dropped. Any application that uses the DNS has (or should have) some way of handling "no such name" errors; in almost all cases the error message is sufficiently clear to an experienced user that it is immediately obvious when the application has failed because it was given an incorrect DNS name. With these wildcard records in place, however, incorrect DNS names which are matched by the wildcard record will not show up as DNS name errors at all, but instead will show up as mysterious connection failures or as unreachable destinations for all services that the zone operator does not redirect. Depending on the details of the application protocol and implementation involved, this change may also convert an obvious "hard failure" (incorrect name) into a soft failure which the application thinks it should retry, as seen above in the email case. This may result in very long delays, perhaps of days or weeks, before even trivial errors are brought to the user's attention. Transport protocols using UDP may also retry until the transport protocol retry limit is reached (especially if ICMP messages are being filtered at a firewall), which may be very considerably longer than the time it would have taken to return an error to the user indicating they mistyped the destination.

4.6. Automated Tools

Automated or embedded tools which use HTTP but which do not have a user interface may also be confused by this change, since such tools may expect configuration failures to show up as DNS errors and may not realize that the HTTP response they have received from the zone operator's search page is not the page which the tool expected to reach. Such tools may fail in unpredictable ways, and may not be easy to repair.

4.7. Charging

The current response from the service in question is just over 17 KBytes of data because the client has to open a TCP connection and receive a not insignificant amount of data. A "no such data" response would have fitted in one packet. In the case of volume-based charging for Internet Access (as with most cellular data services) the recipient will have to pay additional charges.

4.8. Single Point of Failure

Even for cases in which the redirection service works as intended, such a service creates a very large single point of failure. Single points of failure are obvious targets both for deliberate attacks and for the sort of accidental "attacks" caused by bugs and configuration errors which already generate much of the traffic at the DNS name servers for the root zone. Furthermore, the IP address associated with this single point of failure is a likely target both for routing attacks intended to redirect the IP address to some other server.

4.8.1. Privacy

An interception service with this kind of scope raises significant privacy concerns, since traffic received by the interception service is, pretty much by definition, not going where its sender originally intended. The potential for abuse in this situation is very high. The mail received on the interception service can be parsed and saved which makes the interception service an even more attractive target, this time for attackers who wish to gain control of it in order to practice such abuse.

4.8.2. Reserved Names

This sort of wildcard usage is incompatible with any use of DNS which relies on reserving names in a registry with the express intent of not adding them to the DNS zone itself. An example of such a use is the JET-derived IDN approach of "registry restrictions" and "reserved names", which depends on the existence of names that are reserved and

can be registered only by the holder of some related name, but which do not appear in the DNS. By some readings of the current ICANN IDN policy, support for that "reserved name" approach is required. To accomplish the goal of reduced consumer confusion, the reserved names must not be resolvable at all. This reserved name approach appears to be completely incompatible with this sort of wildcard usage: since the wildcard will always cause a result to be returned, even for a reserved name which does not appear in the zone, one can support either one or the other, but not both.

5. Undesirable Workarounds

ISPs have responded to the deployment of these wildcards in a number of ways, all of which are both understandable and worrisome. Some ISPs have contemplated modifying their routing systems to drop all packets destined to the zone operator's redirection server into a black hole. Others have deployed patches to their DNS resolvers which attempt to reverse the effects of these wildcard records. Still other ISPs have considered using this as an opportunity to play the same game that the zone operator is playing, but for the ISP's own benefit. All of these responses are both understandable and predictable, but none of them are good. Even more worrisome is that different ISPs have been taking different approaches to dealing the unwanted effects of deployed wildcards, which may lead to a balkanization problem and create an ongoing headache for anyone having to deal with cross-network DNS or application debugging.

Since ISPs often control the middle-boxes there may not be many ways for the end users to workaround the problem. Users may want to fall back to alternative recursive forwarders but if transparent proxying takes place and or traffic to alternative servers is blocked the users have no choice than to accept the fact that they receive the synthesized data from their ISP. In response to this software developers may start to offer non-standard counter measures such as the tunneling of DNS traffic over secured HTTP connections.

6. Principles, Conclusions, and Recommendations

The Robustness principle tells us that in some (not all) of the problems detailed above, both parties could be construed as being at fault. In some cases this is hardly surprising: spam filtering in particular, by its nature, tends to be extremely ad hoc and somewhat fragile. No doubt there are lessons here for all parties involved.

The Principle of Least Astonishment suggests that the deployment of wildcards in the case described above was disastrous for the users.

It had widesweeping effects on other users of the Internet far beyond those enumerated by the zone operator, created several brand new problems, and caused other internet entities to make hasty, possibly mutually incompatible and possibly deleterious (to the internet as a whole) changes to their own operations in an attempt to react to the change.

6.1. Recomendations concerning deployment of wildcards

Note that these considerations apply to any wildcard deployment of this type. The list of problems encountered in this case clearly demonstrates that, although wildcard records are part of the base DNS protocol, there are situations in which it simply is not safe to use them. As noted in an earlier section, two warning flags suggesting that this type of wildcard deployment is dangerous were that it affected more than one protocol, and it was done high enough up in the DNS hierarchy that its effects were not limited to the organization that chose to deploy these wildcard records.

Note also that a significant component of some of the listed problems was not precisely the wildcard-induced behavior per se so much as it was the abrupt change in the behavior of a long established infrastructure mechanism. In conclusion, we would like to propose a guideline for when wildcard records should be considered too risky to deploy, and make a few recommendations on how to proceed from here.

Proposed guideline: If you want to use wildcards in your zone and understand the risks, go ahead, but only do so with the informed consent of the entities that are delegated within your zone e.g. in those cases where there is a clear organisational dependency and inter-linkage between parent and child zone.

Generally, we do not recommend the use of wildcards for record types that affect more than one application protocol. At the present time, the only record types that do not affect more than one application protocol are MX records.

For zones that do delegations, we do not recommend even wildcard MX records. If they are used, the owners of zones delegated from that zone must be made aware of that policy and must be given assistance to ensure appropriate behavior for MX names within the delegated zone. In other words, the parent zone operator must not reroute mail destined for the child zone without the child zone's permission.

We hesitate to recommend a flat prohibition against wildcards in "registry"-class zones, but strongly suggest that the burden of proof in such cases should be on the registry to demonstrate that their intended use of wildcards will not pose a threat to stable operation

of the DNS or predictable behavior for applications and users.

We recommend that any and all TLDs which use wildcards in a manner inconsistent with this guideline remove such wildcards at the earliest opportunity.

6.2. Recomendations against the synthesis by middleboxes

As for the synthesis by middle-boxes the arguments are similar as the arguments for wildcard deployment, but the amount of users affected by the implementation is much smaller. On the other hand, this method of synthesis is not part of the DNS specification and violates architectural assumptions of 'clean light of sight' on which extensions to the DNS protocol are developed against.

We therefore strongly recommend against the use of this kind of synthesis. If an ISP or an other organization tries to offer synthesis by middle-boxes as a service the should also offer customers a DNS recursive forwarder that provides a clean line of sight to the authoritative data. It is not sufficient to provide opt-out mechanism that are purely web based since other applications may still encounter problems.

7. References

7.1. Normative References

7.2. Informative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC2219] Hamilton, M. and R. Wright, "Use of DNS Aliases for Network Services", BCP 17, RFC 2219, October 1997.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4592] Lewis, E., "The Role of Wildcards in the Domain Name

System", RFC 4592, July 2006.

[IAB-wildcard-commentary]
Internet Architecture Board, "IAB Commentary: Architectural Concerns on the use of DNS Wildcards", September 2003.

Appendix A. Acknowledgements

This document is based on a commentary that was published on the IAB website[IAB-wildcard-commentary].

The IAB acknowledges the kind assistance of David Schairer, John Curran, John Klensin, and Steve Bellovin for helpful suggestions and, in some cases, significant chunks of text for the original commentary.

In addition the IAB also acknowledges .... for their contribution during the production of this document.

None of these contributors bear any responsibility for what the IAB has done with their contributions.

This document was produced using the xml2rfc tool[RFC2629].

Appendix B. Document Editing Details

[To Be Removed after publication]

File with Revision ID 24 was the original conversion from http://www.iab.org/documents/docs/2003-09-20-dns-wildcards.html

File with Revision ID 27 got the synthesis by middle-boxes added and therefore the text was rearranged.

This is $Id: iab-synthesis.xml 35 2007-04-16 11:35:05Z olaf $

Author's Address

Olaf M. Kolkman (editor)
IAB

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).