Domain Name System Operations
Internet-Draft
Intended status: Informational
Expires: June 19, 2017

P. Vixie
Farsight Security, Inc.
V. Schryver
Rhyolite Software, LLC

December 16, 2016

DNS Response Policy Zones (RPZ)
draft-vixie-dns-rpz-04

Abstract

This document describes a method for expressing DNS response policy inside a specially constructed DNS zone, and for recursive name servers to use such policy to return modified results to DNS clients. The modified DNS results can stop access to selected HTTP servers, redirect users to "walled gardens", block objectionable email, and otherwise defend against attack. These "DNS Firewalls" are widely used in fighting Internet crime and abuse.

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 June 19, 2017.

Copyright Notice

Table of Contents

1. Introduction

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This document describes DNS Firewalls, a method of expressing DNS [RFC1034] response policy information inside specially constructed DNS zones, known as Response Policy Zones (RPZs). RPZs allow Internet security policy producers and subscribers to cooperate in the application of policies to modify DNS responses in real time. Using the policy information, DNS resolution for potentially unsafe DNS data can be made to deliberately fail or to return local data such as an alias to a "walled garden".

A site's DNS response policy consists of the set of rules expressed in all of the RPZs that it uses. Each rule, expressed as an RRset, consists of a trigger (left hand side or owner name) and an action (right hand side). A full description of the expressible policies is given in Section 3 (actions) and Section 4 (triggers).

Configuration examples are given using ISC BIND Version 9 (BIND9) [ISC-ARM] syntax, because work to add RPZ to that platform was started earliest (in 2009). The RPZ specification itself is free to implement and free to use in operation. It has been implemented in

other name server software. We expect that in time, additional recursive DNS implementations will also implement DNS Firewalls as described by this RPZ specification.

1.1. Discussion Venue

The discussion venue for this document is the DNS Firewalls mailing list. http://lists.redbarn.org/mailman/listinfo/dnsfirewalls offers subscriptions and archives. See also https://dnsrpz.info/

[NOTE TO EDITOR: This section must be removed before this Internet Draft is published as an RFC.]

2. Zone Format

A DNS Response Policy Zone (RPZ) is a DNS zone. Like any DNS zone, an RPZ consists of RRsets or sets of resource records (RRs) with a common owner name and type. RRsets other than SOA, NS, and DNSSEC-related [RFC4034] specify actions and triggers. The owner name (left hand side) of each RRset expresses a policy trigger, while the RDATA (right hand side) encodes the action to be taken when the trigger matches. Depending on the type of trigger (see Section 4), a particular characteristic of the DNS response is checked. Characteristics that can be checked include the domain name (QNAME), the IP address of the querying client, IP addresses or domain names in the answer section of the response, and authoritative name server names and addresses.

Because an RPZ is a valid DNS zone, its contents can be transferred between DNS servers in whole (AXFR) [RFC5936] or incrementally as changes occur (IXFR) [RFC1995], can be authenticated and protected by TSIG transaction signatures [RFC2845], and can have update propagation expedited by real time change notifications (NOTIFY) [RFC1996], all subject to familiar DNS access controls. An RPZ need not support query access since query access is never required. It is the zone transfer of RPZ content from producers to subscribers which effectively publishes the policy data. However, it is the subscribers' configurations of their recursive name servers which promote RPZ payload data into the control plane data of their individual name servers.

Every valid DNS zone including an RPZ has an SOA record and at least

one NS record at its apex, which is why the SOA and NS records of an RPZ cannot themselves be used to encode DNS response policy rules.

The RPZ's SOA record is real, with a serial number used for NOTIFY and IXFR, and timers used for AXFR and IXFR. The MNAME field or domain name of the primary source of the zone and the RNAME field or mailbox of the person responsible for the zone SHOULD be used by RPZ providers to label their policy zones.

The apex NS RRset will never be used since RPZ does not rely on query access. Similarly, no parent delegation is required for normal operation of the RPZ. Thus, by convention, a single NS record having the deliberately bogus RDATA of "LOCALHOST." is used as a placeholder.

While loading an RPZ, any data which is not syntactically valid for DNS should cause normal error processing as would occur for any DNS zone. If an RR found in an RPZ is meaningless or unusable for

response policy purposes, then the containing RRset SHOULD be ignored, and an error MAY be logged.

The format of RPZs has undergone several revisions since work began (see Section 10, History and Evolution). All policy triggers and actions described here are valid as of Format 3. Policy triggers from a higher format number than a recursive name server's implementation level are expected to be harmless to that implementation. Policy actions from a higher format number are likely to be misinterpreted as CNAME Local Data by older implementations. When possible, implementations SHOULD treat policy triggers or actions of higher format numbers as they treat other errors, as described above.

3. Policy Actions

An RPZ resource record can specify any of six results or actions.

3.1. The "NXDOMAIN" Action (CNAME .)

A single resource record (RR) consisting of a CNAME whose target is the root domain (.) will cause a response of NXDOMAIN to be returned.

This is the most commonly used RPZ action.

```
$ORIGIN RPZ.EXAMPLE.ORG.
example.com                    CNAME   .  ; return NXDOMAIN
*.example.com                  CNAME   .  ; return NXDOMAIN
```

3.2. The "NODATA" Action (CNAME *.)

A single RR consisting of a CNAME whose target is the wildcard top-level domain (*.) will cause a response of NODATA (ANCOUNT=0) to be returned regardless of query type.

```
$ORIGIN RPZ.EXAMPLE.ORG.
example.com                    CNAME   *.  ; return NODATA
*.example.com                  CNAME   *.  ; return NODATA
```

3.3. The "PASSTHRU" Action (CNAME rpz-passthru.)

It is sometimes necessary to exempt some DNS responses from a policy rule that covers an entire domain or a large IP address block. Exempting some clients of a DNS resolver from all RPZ rewriting can also be useful for research into attackers and for debugging. The PASSTHRU action is intended to override other, usually more general policies. The trigger for the PASSTHRU action MUST have a higher precedence than the policies that it should override (see Section 5, Precedence Rules).

In the example below, the first PASSTHRU record exempts requests for a particular host from the NXDOMAIN policy action of the subsequent records. The second PASSTHRU record exempts responses to the DNS client at 192.0.2.1 from being modified:

```
$ORIGIN RPZ.EXAMPLE.ORG.
ok.example.com                 CNAME   rpz-passthru.
32.1.2.0.192.rpz-client-ip     CNAME   rpz-passthru.
example.com                    CNAME   .
*.example.com                  CNAME   .
```

3.4. The "DROP" Action (CNAME rpz-drop.)

The DROP policy that results in discarding the request and response is specified by a CNAME whose target is "rpz-drop". For example, the

following rule mandates that nothing be sent to a DNS client trying to resolve "EXAMPLE.COM", not even a DNS error response:

```
$ORIGIN RPZ.EXAMPLE.ORG.
example.com                    CNAME   rpz-drop.
```

3.5. The "TCP-Only" Action (CNAME rpz-tcp-only.)

The TCP-Only action is specified by a CNAME whose target is "rpz-tcp-only". It changes UDP responses to short, truncated DNS responses that require the DNS client to try again with TCP. It is used to mitigate distributed DNS reflection attacks and is similar to the "slip" parameter of DNS Response Rate Limiting (RRL) [ISC-RRL].

```
$ORIGIN RPZ.EXAMPLE.ORG.
example.com                    CNAME   rpz-tcp-only.
```

3.6. The "Local Data" Action (arbitrary RR types)

A set of RRsets with a common trigger owner name (see Section 4) that includes neither a special CNAME RPZ encoding of an action nor one of the problematic record types listed below specifies data to be used to generate synthetic DNS responses. The most common Local Data is a CNAME RR pointing to a walled garden, although other record types are also used.

```
$ORIGIN RPZ.EXAMPLE.ORG.
bad1.example.com               CNAME   garden.example.net.
bad2.example.com               A       garden-web.example.net.
bad2.example.com               MX      garden-mail.example.net.
32.3.2.0.192.rpz-client-ip     A       quarantine.example.net.
```


Note that because an RPZ is a valid DNS zone, if the action of a policy rule contains a CNAME RR, then no other RRs are allowed for that owner name (trigger).

The special RPZ encodings which are not to be taken as Local Data are CNAMEs with targets that are:

+ "." (NXDOMAIN action),

+ "*." (NODATA action),
+ a top level domain starting with "rpz-",
+ a child of a top level domain starting with "rpz-".

The problematic types and records which also do not encode Local Data actions include:

+ SOA records,
+ NS records,
+ DNAME records,
+ all DNSSEC-related records (see [RFC4034]).

RRsets of problematic record types MUST NOT be included in RPZ data, because their appearance in responses would be invalid or confuse DNS clients.

When a Local Data policy rule matches, the RRsets of Local Data are used to generate the response as if they comprised all of the authoritative data for the QNAME. If the requested type (QTYPE) is ANY, then all of these Local Data RRsets are returned. Otherwise, the RRset of the requested RR type is returned, or a CNAME is returned if it is available. If no CNAME nor RRset of the requested type is available, then the response is normally NODATA (ANCOUNT=0). Using the example above, if client 192.0.2.3 asks for MX records, it will receive NODATA, because the policy rule with the matching Client IP Address trigger contains RRsets of RRtype A but none of RRtype MX.

This normal NODATA response when there are no Local Data records of the requested type can be changed with the LOCAL-DATA-OR-PASSTHRU or LOCAL-DATA-OR-DISABLED overrides described in Section 6.1.

A special form of Local Data involves a CNAME RR with a wildcarded target name. Wildcards are not valid as CNAME targets in ordinary DNS zones. However, a wildcard in an RPZ Local Data CNAME target causes the matching QNAME to be prepended to the target in the rewritten response, which communicates this QNAME value to the walled garden DNS server for that DNS server's logs.

For example a policy Local Data action of "CNAME *.EXAMPLE.COM" applied to a QNAME of "EVIL.EXAMPLE.ORG." will result in a synthetic

response that starts with the RR

"EVIL.EXAMPLE.ORG CNAME EVIL.EXAMPLE.ORG.EXAMPLE.COM". Resolving the CNAME target "EVIL.EXAMPLE.ORG.EXAMPLE.COM" into an RRset of the originally requested type generally requires sending a request for that type and a QNAME of "EVIL.EXAMPLE.ORG.EXAMPLE.COM" to a DNS server for the walled garden, "EXAMPLE.COM". As usual when a CNAME is encountered while computing a response, the response from the walled garden DNS server concerning "EVIL.EXAMPLE.ORG.EXAMPLE.COM" determines the rest of the final rewritten response.

In the example below, a client that asks for A RRs for "BAD.EXAMPLE.COM" will receive a response starting with "BAD.EXAMPLE.COM CNAME BAD.EXAMPLE.COM.GARDEN.EXAMPLE.NET". The DNS server using RPZ will then probably try to resolve "BAD.EXAMPLE.COM.GARDEN.EXAMPLE.NET" by requesting A RRs from the authority for "GARDEN.EXAMPLE.NET". That authority should answer with NODATA, NXDOMAIN, or an A RRset, but in any case can log the request to show that a request for "BAD.EXAMPLE.COM" has been received.

```
$ORIGIN RPZ.EXAMPLE.ORG.
bad.example.com                CNAME    *.garden.example.net.
```

4. Policy Triggers

There are five types of RPZ triggers, and they are encoded by RRset owner names (left hand sides) in an RPZ.

Two of these trigger types match characteristics of the DNS query: Client IP Address and QNAME. While these two trigger typess are independent of cache contents or recursion results, still conceptually they must be checked only once the response is ready, because they compete for precedence (see Section 5) with other trigger types which depend on what happens during recursion.

The other three trigger types are based on characteristics of the DNS response, that is, on the RDATA to be returned to the client, or in some cases, on NS-related RDATA used while recursively obtaining the final response, regardless of whether or not those NS or NS-related records are themselves to be returned to the client. These three trigger types are: Response IP Address, NSDNAME, and NSIP.

Neither the TTL fields of RRs in the answer section nor the query type (QTYPE) in the question section of responses are used as RPZ triggers, although the QTYPE is used to select appropriate RRsets among the RRsets of matching Local Data rules (Section 3.6).

All policies are conceptually applied after recursion, even though in practice recursion can sometimes be skipped, if doing so would not change the RPZ result (see Section 5, Precedence Rules and Section 9, Implementation Considerations). As a result, the recursive DNS resolver's cache contains either nothing or "truth", even if this truth is hidden by current policy. If the policy changes, the original, unmodified data is available for processing under the changed policy.

4.1. The "Client IP Address" Trigger (.rpz-client-ip)

The IP addresses of DNS clients sending requests can be used as triggers, which can be useful for disabling RPZ rewriting for DNS clients used for testing or investigating, or for quarantining compromised clients. Client IP Address policy RRsets have owner names that are subdomains of "rpz-client-ip" relativized to the RPZ apex name, preceded by an encoded IP address or block of addresses.

For example, the following would drop all requests from clients in 192.0.2.0/24 and give truthful answers to requests from a client at 2001:db8::3.

```
$ORIGIN RPZ.EXAMPLE.ORG.
24.0.2.0.192.rpz-client-ip      CNAME rpz-drop.
128.3.zz.db8.2001.rpz-client-ip CNAME rpz-passthru.
```

4.1.1. IP address encoding in triggers

The IPv4 address or address block "B1.B2.B3.B4/prefix" is encoded in an RPZ trigger as "prefix.B4.B3.B2.B1", with the address octets reversed just as in the IN-ADDR.ARPA naming convention described in [RFC1034]. The prefix length ("prefix") is a decimal integer from 1 to 32. All four octets, B4, B3, B2, and B1, must be present and are also decimal integers. To avoid confusion with octal notation, leading zeros MUST be suppressed.

For example, the address block 192.0.2.0/24 is encoded as "24.0.2.0.192".

The IPv6 address or address block beginning at "W1:W2:W3:W4:W5:W6:W7:W8" is encoded in a format similar to the standard IPv6 text representation (see [RFC5952]), again reversed: "prefix.W8.W7.W6.W5.W4.W3.W2.W1". The prefix length ("prefix") is a decimal integer from 1 to 128. Each of W8,...,W1 is a one- to four-digit hexadecimal number representing 16 bits of the IPv6 address. As in [RFC5952], in order to avoid confusion with octal notation,

leading zeroes MUST be suppressed.

All 8 hextets must be present unless a "zz" label is present. The "zz" label is analogous to the double-colon (::) in [RFC5952], and it is required and allowed just as the double-colon is required and allowed in that document. In particular, the longest possible sequence of zero-valued fields MUST be compressed to "zz". If there exists more than one sequence of zero-valued fields of identical length, then only the last such sequence is compressed. Note that [RFC5952] specifies compressing the first such sequence, but our notation here reverses the order of fields, and so must also reverse the selection of which zero sequence to compress.

For example, the address 2001:db8::3 with a prefix length of 128 would be encoded as "128.3.zz.db8.2001".

In the case of an address block, either IPv4 or IPv6, the address part MUST NOT contain any non-zero bits after the section indicated by the prefix length. For example, "8.2.0.0.10.rpz-client-ip" is not a valid trigger, because in the address "10.0.0.2" expressed in binary notation, a one occurs outside the first 8 bits or prefix of the address block.

4.2. The "QNAME" Trigger ("example.com")

The QNAME policy trigger matches requested domains, that is, the QNAME field of the question sections in DNS requests and responses. (See [RFC1035].) The owner name of an RPZ QNAME policy RRset is the relativized name of the domain name about which policy is being expressed. For example, if the RPZ apex name is "RPZ.EXAMPLE.ORG", an RRset at "EXAMPLE.COM.RPZ.EXAMPLE.ORG" would affect responses to requests about "EXAMPLE.COM".

Wildcards work as expected, so the owner name "*.EXAMPLE.COM.RPZ.EXAMPLE.ORG" would match queries for any subdomain of "EXAMPLE.COM". To control the policy for both a name and its subdomains, two policy RRsets must be used, one for the domain itself and another for a wildcard subdomain. In the following example, queries for both "EXAMPLE.COM" and all subdomains of "EXAMPLE.COM" will result in synthetic NXDOMAIN responses.

```
$ORIGIN RPZ.EXAMPLE.ORG.
example.com                 CNAME   .
*.example.com               CNAME   .
```

4.3. The "Response IP Address" Trigger (.rpz-ip)

The Response IP Address trigger matches IP addresses that would appear in the unaltered DNS response contents, specifically the RDATA of A or AAAA records in the answer sections of DNS responses. IP

addresses in the authority and additional sections are not considered. Response IP Address policy RRsets have owner names that are subdomains of "rpz-ip" relativized to the RPZ apex name, and an encoded IP address or block of addresses. The IP address encodings are identical to those described in Section 4.1.1 for Client IP Address triggers.

For example, to force an NXDOMAIN response whenever a truthful response would contain an answer section A RRset having an address in 192.0.2.0/24 unless address 192.0.2.2 is present, the RPZ would contain these records:

```
$ORIGIN RPZ.EXAMPLE.ORG.
24.0.2.0.192.rpz-ip         CNAME   .
32.2.2.0.192.rpz-ip         CNAME   rpz-passthru.
```

In another example, to answer NODATA (ANCOUNT=0) whenever a truthful response would contain an answer AAAA RRset having an address 2001:db8:101::/48 unless address 2001:db8:101::3 was present, the RPZ would contain these records:

```
$ORIGIN RPZ.EXAMPLE.ORG.
48.zz.101.db8.2001.rpz-ip    CNAME  *.
128.3.zz.101:db8.2001.rpz-ip CNAME  rpz-passthru.
```

Please refer to Section 5 (Precedence Rules) to understand how the above exception mechanisms work.

4.4. The "NSDNAME" Trigger (.rpz-nsdname)

The NSDNAME policy trigger matches name server names (NS RR) of all name servers in the data paths for all RRsets that would be present

in the answer section of the unaltered DNS response.

The data path for a given answer RRset consists of all delegation points from (and including) the root zone down to the closest enclosing NS RRset for the owner name of that RRset. Names in the RDATA of answer RRs including CNAME, DNAME, SRV, and MX are not themselves directly relevant, but CNAME and DNAME target names are indirectly relevant if they cause RRsets to be added to the answer section, in which case it is the data paths of the added RRsets that matter. In the case of a DNAME answer, the owner name of an added synthetic CNAME is likely to differ from the target name in the DNAME RR. Recall also that the target of a CNAME is not added to the response if the QTYPE is ANY or CNAME or if this target cannot be resolved (e.g. NXDOMAIN or SERVFAIL errors).

NSDNAME policies are encoded as RRsets in subdomains of "rpz-nsdname" but otherwise are much like QNAME policies (Section 4.2). For example, to force an NXDOMAIN answer whenever a name server for the requested domain or one of its parents is "NS.EXAMPLE.COM", the RPZ would contain the following:

```
$ORIGIN RPZ.EXAMPLE.ORG.
ns.example.com.rpz-nsdname  CNAME    .
```

The NS records used for this calculation are either delegations (NS RRs in the authority sections of answers from authorities for the parent zone) or authoritative data from the zone itself. An implementation MAY use either, both, or whichever is currently available. See Section 9.4 about some implementation considerations for this choice, and Section 12.5 about security considerations.

An RPZ implementation MAY be configurable to avoid checking all the way up to the root and to perform only partial NSDNAME checks; see Section 9.3 on "min-ns-dots".

4.5. The "NSIP" Trigger (.rpz-nsip)

The NSIP policy trigger matches name server addresses, that is A or AAAA RRs referenced by an NS RRset. NSIP is like NSDNAME (Section 4.4) except that the matching is by name server address

rather than name server name. NSIP policies are expressed as subdomains of "rpz-nsip" and have the same subdomain naming convention as that described for encoding IP addresses in Response IP Address triggers (Section 4.1.1).

In a process similar to that for an NSDNAME trigger, an NSIP trigger is checked by considering all of the IP addresses for all of the name servers in the data paths for all RRsets that would be present in the answer section of the unaltered DNS response.

As with NSDNAME triggers, the data path for a given RRset consists of all delegation points from (and including) the root zone down to the closest enclosing NS RRset for the owner name of that RRset. Also like NSDNAME triggers, the RDATA in these RRsets (other than the IP addresses of the name servers) are not directly relevant.

For example, to force an NXDOMAIN answer whenever one of the name servers for the requested domain (QNAME) or one of its ancestors has an address in the 192.0.2.0/24 block, the RPZ would contain the following:

```
$ORIGIN RPZ.EXAMPLE.ORG.
24.0.2.0.192.rpz-nsip       CNAME    .
```


The NS, A, and AAAA records used for this calculation are either delegations and glue (RRs in authority and additional sections of answers from authorities for the parent zone) or authoritative data from the zone itself. As with NSIP, an implementation MAY use either, both, or whichever is currently available; see Section 9.4 on "ns-wait-recurse".

An RPZ implementation MAY also be configurable to avoid checking all the way up to the root and to perform only partial NSIP checks; see Section 9.3 on "min-ns-dots".

5. Precedence Rules

More than one policy trigger among the various DNS RPZs connected to the name server's control plane can match while computing a given DNS response, but only a single policy rule can rewrite the response. The policy rule with the best match will be selected according to the precedence rules outlined below.

These precedence rules exist and are ordered to ensure that RPZ subscribers and publishers have the same understanding of what a set of policy zones will do, and to ensure that subscribers can use local zones to override published policy zones.

In theory and for standardization, all matching policy rules are considered simultaneously, and the precedence rules are used to choose the single best RPZ rule. In actual implementations, policy triggers are usually considered in a sequence that mirrors the process of generating the DNS response, because checking RPZ triggers is conveniently made a part of that process. For example, Client IP Address triggers are often checked early as the DNS request is being received and as the client IP address is being checked in the access control list (ACL) that determines which DNS client IP addresses can ask for recursion. Likewise the QNAME is available for RPZ trigger matching before any response IP addresses are known, so QNAME triggers are usually checked immediately after Client IP Address triggers and before Response IP Address triggers. NSIP and NSDNAME triggers are generally checked last.

As far as the DNS client can determine, it MUST seem that all matching triggers are found and weighed using these precedence rules, even though in practice, shortcuts are taken. Shortcuts MUST NOT affect the outcome. For example, according to the precedence rules, a matched QNAME trigger in the first policy zone makes all Response IP Address, NSIP, and NSDNAME triggers moot. There is no need to look for those matches, because they cannot further affect the response.

The actions specified in policy rules are not used in the calculation of precedence. The actions of policy rules determined by per-zone action overrides (Section 6.1), other than those that disable them, likewise do not affect the calculation of precedence. Override actions which disable policy rules affect this calculation only by removing those rules from consideration.

Precedence rules are applied in the order listed here; the comparison between two matching policy rules to choose the better match is determined by the first dispositive precedence rule in this list. Note however that if the best match selects a disabled rule (caused

by the DISABLED override described in Section 6.1), then the next best match is used, and so on until a policy rule that is not disabled is selected or no matches remain.

5.1. "CNAME or DNAME Chain Position" Precedence Rule

This precedence rule applies only when the original query type is not ANY, CNAME, nor DNAME, and only when a CNAME or DNAME is encountered while resolving the original QNAME, in what we will call the first "stage of resolution". In this case, we know that the recursive name server ([RFC1035]) may follow the CNAME (or the synthetic CNAME constructed while applying a DNAME) by starting a second stage of resolution starting at this CNAME's target. Additional stages of resolution may ensue if further CNAMEs or DNAMEs are encountered, until the final response is computed.

A policy rule match which occurs at an earlier stage of resolution is preferred to a policy rule match which occurs at a later stage.

Recall that only one policy rule, from among all those matched at all stages of resolving a CNAME or DNAME chain, can affect the final response; this is true even if the selected rule has a PASSTHRU action.

5.2. "RPZ Ordering" Precedence Rule

This precedence rule applies when the matches being compared refer to policy rules in different RPZs.

Matches on rules in an RPZ specified earlier in the ordered list of RPZs take precedence over matches on rules in an RPZ specified later.

5.3. "Domain Name Matching" Precedence Rule

This precedence rules applies to a set of QNAME rules or a set of NSDNAME rules in a single RPZ. The rule triggers are compared.

This rule is the same as that used by DNS servers to choose the RRs with which to respond to a request. In particular, an exact name match is better than one involving a wildcard, and among wildcard matches, the trigger (owner domain name) that has the largest number

of labels is best.

Since a DNS response has only one QNAME in a given stage of CNAME or DNAME chain resolution (as defined in Section 5.1, "CNAME or DNAME Chain Position" Precedence Rule), this precedence rule is sufficient to choose a single QNAME trigger over other QNAME triggers in the same RPZ. However, computing a single DNS response can match dozens of NSDNAME triggers in a single RPZ, of which several may be equal according to this precedence rule. In that case, the "Name Order" Precedence Rule (Section 5.5) selects the single winner from among those chosen by this rule.

5.4. "Trigger Type" Precedence Rule

This precedence rule applies to policy rules within the same RPZ, but with different trigger types.

In this case, matches are ranked according to trigger type in the following order, from highest to lowest precedence:

+ Client IP Address
+ QNAME
+ Response IP Address
+ NSDNAME
+ NSIP

5.5. "Name Order" Precedence Rule

This precedence rule applies to a set of NSDNAME matches whose precedence has not been established by previous precedence rules. Here, the matched name server domain names are compared, not the owner names (triggers) of the policy rules. Therefore, it is irrelevant whether the matched trigger was a wildcard or a specific domain name.

Among the matching NSDNAME policy rules, choose the one whose matched name server domain name appears last in the DNSSEC canonical DNS name order described in section 6.1 of [RFC4034].

Important note: if this precedence rule is reached, the matches being compared originate from different NS names, not from the same name matching multiple rules, as those conflicts would have been dispensed with by the "Domain Name Matching" Precedence Rule (Section 5.3). There is no a priori reason to prefer one NS name over another, so

while this tie-breaker precedence rule is dispositive, it is also arbitrary. All RPZ implementations MUST make the same choices in these cases to ensure consistent, predictable results among installations.

Taking part of the example directly from section 6.1 of [RFC4034], and reversing the order to take into account that the name that appears last in the DNSSEC canonical order is to be taken as the best match here, the following NS names are given below in order from highest or best to lowest precedence:

+ z.example
+ zABC.a.EXAMPLE
+ Z.a.example
+ yljkjljk.a.example
+ a.example
+ example

5.6. "Prefix Length" Precedence Rule

This precedence rule applies when more than one Response IP Address policy rule or more than one NSIP policy rule matches while computing a response. The rule triggers themselves are compared.

When comparing two Response IP Address matches or two NSIP matches on rules within a single RPZ, choose the match whose rule trigger has the longest "internal prefix length". The internal prefix length of an IPv6 address trigger is the numeric value of the first label that defines it as described in Section 4.1.1 on IP address encoding in triggers. For an IPv4 address trigger, the internal prefix length is the numeric value of its first label plus 96 (that is, 128-32). This adjustment allows IPv4 and IPv6 addresses to use the same internal representation, with 32-bit IPv4 addresses expanded to 128 bits by zero filling as described in section 2.5.5.1 of [RFC4291].

5.7. "IP Address Order" Precedence Rule

This precedence rule applies when more than one Response IP Address policy rule or more than one NSIP policy rule matches while computing a response, and they have the same internal prefix length. The rule triggers are compared.

When comparing Response IP Address or NSIP triggers with equal internal prefix lengths, choose the trigger which encodes the smaller IP address. The IP addresses from the triggers are compared as 128 bit numbers, with IPv4 addresses expanded to 128 bits by zero filling

as described in section 2.5.5.1 of [RFC4291].

Important note: if this precedence rule is reached, the matches being compared originate from different IP addresses, not from the same address matching multiple rules, as those conflicts would have been dispensed with by the "Prefix Length" Precedence Rule (Section 5.6). There is no a priori reason to prefer one IP address over another, so while this tie-breaker precedence rule is dispositive, it is also arbitrary. All RPZ implementations MUST make the same choices in these cases to ensure consistent, predictable results among installations.

In the following example, the internal prefix length of all three IP addresses is 121 as described in Section 5.6.

```
25.0.2.0.192.rpz-ip               CNAME   most.example.com.
25.128.2.0.192.rpz-ip             CNAME   middle.example.com.
121.280.c000.zz.db8.2001.rpz-ip   CNAME   least.example.com.
```

The three addresses above are compared as these 128-bit hexadecimal numbers, respectively:

```
0x00000000000000000000000c0000200,
0x00000000000000000000000c0000280, and
0x20010db80000000000000000c0000280.
```

Thus, the first IPv4 address is most preferred and the IPv6 address is least preferred.

6. Application of the Policy

To enable the use of RPZs, the recursive name server's control plane is connected to the DNS RPZ data plane by supplying an ordered list of RPZs in the name server's configuration.

By default, policies are applied only on DNS requests that ask for recursion (RD=1). Recursive DNS servers generally send their requests to authority servers without asking for recursion (RD=0), while stub resolvers ask for recursion (RD=1). Thus, the use of RPZ at a recursive server by default affects requests from stub resolvers only. An implementation SHOULD include a configuration option such as "recursive-only no" to relax this restriction.

Also by default, RPZ policies are applied only while responding to DNS requests that do not request DNSSEC metadata (DO=0) or for which no DNSSEC metadata exists. An implementation MAY include a configuration option such as "break-dnssec yes" to relax this restriction. See Section 12.2 about the implications of responding with modified DNS responses when the DNS client seems to be expecting DNSSEC signatures.

RPZ rules do not apply to synthetic data generated by using RPZ rules. For example, if RPZ supplies a CNAME pointing to a walled garden, RPZ policies will not be used while following that CNAME. If RPZ supplies local data giving a particular A record, RPZ policies will not apply to that response IP address.

If an illegal record is encountered while computing a response, for example an NS record with a CNAME target, and this illegal record is ignored by the resolver, then it is likewise not used for RPZ rule matches.

If a policy rule matches and results in a modified answer, then that modified answer will include in its additional section the SOA RR of the policy zone whose rule was used to generate the modified answer. This SOA RR includes the name of the DNS RPZ and the serial number of the policy data which was connected to the DNS control plane when the answer was modified.

6.1. Per-Zone Action Overrides

For each DNS RPZ configured for use by a recursive name server, it SHOULD be possible to specify a single, OPTIONAL overriding policy action to be used for all policy triggers found in that RPZ. These policy action overrides include the following:

NXDOMAIN
: SHOULD be implemented and is the same as the NXDOMAIN action defined in Section 3.1.

NODATA
: SHOULD be implemented and is the same as the NODATA action defined in Section 3.2.

PASSTHRU
SHOULD be implemented and is the same as the PASSTHRU action defined in Section 3.3.

LOCAL-DATA-OR-PASSTHRU
MAY be implemented. It modifies the Local Data action (Section 3.6) so that requests for RR types that would otherwise give a Local Data result of NODATA (ANCOUNT=0) are instead handled using the PASSTHRU action (Section 3.3). Note that computing a DNS response for QTYPE ANY can never cause a Local Data NODATA result. The LOCAL-DATA-OR-PASSTHRU override applies only to Local Data rules and has no effect on rules with other actions.

DROP

SHOULD be implemented and is the same as the DROP action defined in Section 3.4.

TCP-ONLY
SHOULD be implemented and is the same as the TCP-ONLY action defined in Section 3.5.

CNAME domain
SHOULD be implemented and is a special case of the Local Data action defined in Section 3.6. The overriding data is a CNAME RR with the given domain as its target.

GIVEN
MAY be implemented to explicitly affirm the default, which is to respect all policy actions found in this DNS RPZ.

DISABLED
SHOULD be implemented and causes any rule of the zone, when selected as a "best match", to have no effect, except to log what would otherwise have happened, provided sufficient logging is enabled. The DISABLED override thus causes the next highest precedence match (if any) to be used (see Section 5, Precedence Rules). This is useful for debugging or previewing a policy zone.

LOCAL-DATA-OR-DISABLED

MAY be implemented. It modifies the Local Data action (Section 3.6) so that requests for RR types that would otherwise give a Local Data result of NODATA (ANCOUNT=0) are instead handled similarly to the DISABLED override, but without logging. In this case, the next highest precedence match (if any) is used. Note that computing a DNS response for QTYPE ANY can never cause a Local Data NODATA result. The LOCAL-DATA-OR-DISABLED override applies only to Local Data rules and has no effect on rules with other actions.

7. Producer Behavior

A DNS RPZ producer SHOULD make every effort to ensure that incremental zone transfers (IXFR [RFC1995]) rather than full zone transfers (AXFR [RFC5936]) are used to move new policy data toward subscribers. DNS RPZ subscribers are "stealth slaves" as described in RFC 1996. The master MUST allow each such slave to perform the needed zone transfers, for example via its access control lists (ACLs). The master also SHOULD be configured as necessary to send NOTIFY messages to each slave. Because minimal data latency is critical both to the prevention of crime and abuse and to the withdrawal of erroneous or outdated policy, a DNS RPZ producer SHOULD

also make every effort to minimize data latency, including ensuring that NOTIFY messages are sent in a timely manner after each change of the DNS RPZ on the master server.

In a response policy zone domain, each addition, deletion, or expiration could be handled using DNS UPDATE [RFC2136] to trigger normal DNS NOTIFY and subsequent DNS IXFR activity which can keep the subscribing servers well synchronized to the master RPZ. Alternatively, on some primary name servers (such as ISC BIND9) it is possible to generate an entirely new primary RPZ file and have the server compute the differences between each new version and its predecessor. In ISC BIND9 this option is called "ixfr-from-differences" and is known to be performant even for million-rule DNS RPZ's with significant churn on a minute by minute basis.

It is good operational practice to include test records in each DNS RPZ to help that DNS RPZ's subscribers verify that response policy rewriting is working. For example, a DNS RPZ might include a QNAME

policy rule for "BAD.EXAMPLE.COM" as well as a Response IP Address policy rule for 192.0.2.1. A subscriber can verify the correctness of their installation by querying for "BAD.EXAMPLE.COM", which does not exist in real DNS. If an answer is received it will be from the DNS RPZ. That answer's additional data section will usually contain an SOA RR denoting the fully qualified name of the DNS RPZ itself.

8. Subscriber Behavior

RPZs MUST be primary or secondary zones at subscriber recursive resolvers. They can be searched using only a recursive server's own storage, because additional network transactions for DNS resolvers would be unworkable for this purpose.

Response policy zones are loaded in the usual way. For primary zones this may mean loading the contents of a local file into memory or connecting to a database. For secondary zones this means transferring the zone from the primary server using zone transfer such as IXFR [RFC1995] or AXFR [RFC5936]. All secondary zone transfer relationships SHOULD be protected with transaction signatures (DNS TSIG) and real time change notification SHOULD be enabled using the DNS NOTIFY protocol [RFC1996].

9. Implementation Considerations

DNS resolvers often have limited notion or no notion of a DNS zone or zone file. They sometimes have special local zones, but generally have no implementations of IXFR, AXFR, or NOTIFY. Therefore, an

external module or service that maintains local copies of policy zones can be useful.

RPZ checks can add significant processing and network costs to the processing of recursive DNS requests. This is particularly true of rules with NSDNAME and NSIP triggers. However, NSDNAME and NSIP triggers can be more effective than QNAME and IP triggers, because miscreants can more easily change their direct domain names and IP addresses (which are detected by QNAME and IP triggers) than they can change their NS names and addresses.

To minimize the costs associated with RPZ processing, implementations

MAY use various optimizations, some of which are described below.

9.1. Avoid Waiting for QNAME Recursion

When data already in the resolver's cache combined with the precedence rules in Section 5 are sufficient to produce a final RPZ result, an implementation MAY choose not to finish computing the unmodified DNS response, thus avoiding unnecessary recursion. For example, a matching Client IP Address trigger (Section 4.1) in the first specified RPZ always provides the result regardless of what might be learned by asking (recursing to) the authority for the QNAME. Of course, if the data already in the cache are not dispositive, then the implementation MUST use recursion. For example a matching QNAME trigger (Section 4.2) in the second RPZ could potentially be overruled by a Response IP Address trigger (Section 4.3) in the first RPZ.

Implementations which include this optimization SHOULD provide a configuration switch (for example, "qname-wait-recurse") to turn it on and off. The default value of the switch MAY be on or off. Some security implications of avoiding unnecessary recursion are considered in Section 12.4.

Implementations MAY cause the LOCAL-DATA-OR-DISABLED override (described in Section 6.1) to imply not waiting for recursion, or to require that optimization.

9.2. Check Parents Domains versus Zone Cuts

In principle, the NSDNAME and NSIP processes of following the data path toward the root look for NS, A, and AAAA RRsets only in the apexes of zones. In practice, DNS resolvers often do not know which domain name parent-child relationships reflect zone delegations, and so RPZ implementations generally iteratively remove labels from the left end of the domain name to request NS RRsets from their caches or authoritative servers for each ancestor of the queried name. The

cost of looking for an NS RRset for a domain name that is not at a delegation point is usually only asking for the RRset and receiving only an SOA from the cache.

9.3. Abbreviate the Data Path for NSDNAME and NSIP Checks

The process of traversing the data path from the level nearest the queried record to the top (root domain) level to check NSIP or NSDNAME triggers can be expensive. There are many NS records for the top level domains and the root, but checking them is rarely worthwhile because the root and TLDs are rarely used as NSIP or NSDNAME triggers. Some RPZ implementations have a "min-ns-dots" parameter that stops NSDNAME and NSIP checking short of the root zone, by not checking the name servers for domains whose names contain fewer than the configured minimum number of dots.

9.4. Use Glue for NSDNAME and NSIP Checks

Some implementations of DNS RPZ attempt to exhaustively discover all ancestral zone cuts above the query name and learn the NS RRset at the apex of each delegated zone. Other implementations use only the information which has naturally come into their caches, which often includes only delegations and glue, that is name server names and addresses sent by servers for the parent zone and not by those for the zone itself. Both of these tactics have shortcomings, but the more accurate tactic of checking delegations and glue as well as authoritative NS, A, and AAAA RRsets is generally considered too expensive.

To limit the costs of checking NSDNAME and NSIP triggers, an implementation MAY have settings (for example "nsdname-wait-recurse no" or "nsip-wait-recurse no") which cause the resolver to use only the NS RRsets or only the A and AAAA RRsets already in the cache. With these settings, the resolver will not recurse if there are relevant cached entries (including negative entries) when processing NSDNAME or NSIP triggers, respectively.

Note that using only already-cached instead of current and complete delegation, glue, and authoritative data can cause inconsistent RPZ behavior. Most recursive DNS server implementations do not renew delegation and glue RRsets when their TTLs expire if authoritative data are available in the cache. Requests by the RPZ part of a recursive server for NS RRsets and associated A and AAAA RRsets will often initially receive only delegations and glue from the cache. As time passes and TTLs expire, those requests tend to be answered less with delegations and glue and more with authoritative RRsets. If and when sufficient RRsets expire from the cache, the cycle repeats with RPZ once again relying more on delegations and glue. Thus, if the

glue and authoritative RRsets differ, then RPZ installations that use whichever is available in the cache can produce inconsistent results.

This is not an argument for NSDNAME or NSIP triggers to not use glue, because as discussed in Section 12.5, delegations and glue NS, A, and AAAA RRsets for miscreant domains are often more revealing and effective than their authoritative RRsets.

9.5. Reduce Zone Size using Implied Rules

In 2016, an RPZ zone of QNAME rules for many miscreant domain names contained 200 MBytes of data representing almost 8 million rules. In such an RPZ, many of the listed domain names are as undesirable as DNS servers as they are as mail senders, and so might reasonably have NSDNAME rules in addition to their QNAME rules. However, instead of paying the costs of doubling that RPZ zone to 16 million rules and 400 MBytes, an RPZ implementations MAY offer a per-zone setting (for example "qname-as-ns yes") which causes the implementation to act as if for every rule with the trigger "EXAMPLE.COM", there is another rule in the RPZ with the trigger "EXAMPLE.COM.RPZ-NSDNAME" and the same policy action.

An implementation MAY also offer a setting (for example "ip-as-ns yes") which similarly generates NSIP rules based on Response IP Address rules. Given an RPZ rule with a trigger of "24.0.2.0.192.rpz-ip", act as if the RPZ also contains another rule with the trigger "24.0.2.0.192.rpz-nsip" and the same policy action.

10. History and Evolution

An early description of RPZ can be found in a technical note from Internet Systems Consortium [ISC-RPZ]. RPZ was also described in 2010 in chapter 6 of the "BIND 9 Administrator Reference Manual" [ISC-ARM].

RPZ was designed by Paul Vixie and Vernon Schryver in 2009. The initial implementation and first patch adding it to BIND were written by Vernon Schryver in late 2009. Patches for various versions of BIND9 including 9.4, 9.6, and 9.7 were distributed from FTP servers at redbarn.org and rhyolite.com starting in 2010.

If all RPZ triggers and actions had been foreseen at the start in 2009, they would probably have been encoded differently. Instead RPZ grew incrementally, and upward compatibility required support of the original encodings. The initial specification or Format 1 contained only QNAME triggers. Changes for Format 2 included adding triggers based on response contents (Response IP Address), the targets of NS records (NSDNAME), and contents of A and AAAA records that resolve NS

records (NSIP). Format 3 included "rpz-passthru" for the PASSTHRU action and wildcards in Local Data CNAME targets to synthesize responses dynamically based on the query domain.

Today, with a number of commercial RPZ providers with many users and no functional problems with the encodings, any lack of aesthetic appeal is balanced by the ever increasing weight of the installed base. For example, it is impossible to replace the original QNAME trigger encoding NXDOMAIN and NODATA policy action encodings with encodings that involve rpz-* pseudo-TLDs at RPZ providers without breaking the many existing RPZ subscriber installations. The original, deprecated PASSTHRU encoding of a CNAME pointing to the trigger QNAME might still be in use in local, private policy zones, and so it is still recognized by RPZ subscriber implementations as of 2016.

The initial idea for RPZ was only to deny the existence of objectionable domain names, and so there existed only QNAME triggers and NXDOMAIN actions. Given that single kind of trigger, encoding it as the owner name of a policy record was clearly best. A CNAME pointing to the root domain (.) is a legal and valid but not generally useful record, and so that became the encoding for the NXDOMAIN action. The encoding of the NODATA action as "CNAME *." followed similar reasoning. The addition of more triggers and more actions required a more general scheme, and so newer actions are encoded as CNAMEs with targets in bogus TLDs whose names start with "rpz-". Newer triggers are owner names containing DNS labels that start with "rpz-".

It would have been better if all actions had been encoded as subdomains of the invalid "*" top level domain, for example "drop.rpz.*" and "passthru.rpz.*" and even "nxdomain.rpz.*". "rpz" would always be the second level domain and the action would be specified by the third level domain. That would have polluted only a single bogus TLD. Perhaps future versions of RPZ that define new actions will define those new actions using that scheme, and will use that scheme to redefine existing actions while deprecating but retaining the old definitions.

11. IANA Considerations

No actions are required from IANA as result of the publication of

this document.

12. Security Considerations

12.1. DNS Data Security

RPZ is a mechanism for providing "untruthful" DNS results to consenting stub resolvers from recursive servers. Nevertheless, RPZ does not exacerbate the existing vulnerability of DNS servers to falsified DNS data; it merely formalizes and facilitates modifying DNS data on its way from DNS authority servers to clients. Note that a non-consenting stub resolver can simply opt out by using a different recursive name server.

12.2. DNSSEC

The use of DNSSEC (see [RFC4033] and [RFC4034]) prevents the acceptance by clients of such RPZ-induced changes to DNS data. Therefore, by default, DNS resolvers using RPZ avoid modifying DNS results when DNSSEC signatures are available and are requested by the DNS client. However, when the common "break-dnssec" configuration setting is used, RPZ-using resolvers rewrite responses even in that case. They omit DNSSEC RRsets, because the modified responses cannot be verified by DNSSEC signatures. This renders the results invalid according to DNSSEC. In such a case, a querying client which checks DNSSEC reacts as though it has been subjected to a data poisoning attack and rejects the data. That reaction could entail expensive bypass operations.

12.3. TSIG

The policy zones might be considered sensitive, because they contain information about miscreants. Like other DNS zones in most situations, RPZs are transferred from publishers to subscribers as cleartext vulnerable to observation. DNS zones are vulnerable to forgery or modification if TSIG transaction signatures [RFC2845] are not used, and so TSIG SHOULD be used to authenticate RPZ contents and

protect them from modification.

12.4. Counterintelligence

Recursive servers using RPZ can be optimized to avoid completing recursion if a policy rule provides a rewritten answer without needing this recursion (Section 9.1). However, the use of such an optimization allows miscreants to infer whether RPZ is in use and whether their RRs are listed, simply by observing requests sent to their own authority servers. Therefore, implementations that provide this optimization SHOULD allow it to be disabled.

12.5. NSIP, NSDNAME, Glue, and Authoritative RRsets

Apex ("below the cut") or authoritative NS, A, and AAAA RRsets containing the name server names and addresses for a zone often do not exactly match glue from the parent zone. Delegation and glue records or "above the cut" records from the parent zone are often incomplete or out of date compared to the NS, A, and AAAA RRs from authoritative servers for the zone itself. Regardless of any lack of completeness or accuracy, the delegation and glue records for a functioning domain are sufficently accurate for the domain to be resolvable: the authoritative servers for a zone cannot be found without a delegation and (if necessary) some glue. On the other hand, a domain with entirely broken authoritative name server data can largely work.

The authoritative NS data for miscreant domains is often fanciful or even unavailable. For example, an authoritative NS RRset consisting of "NS ." is popular, because while it is wrong, it betrays nothing in particular about the miscreant who owns the domain.

Thus delegations and glue are generally best for detecting and rewriting miscreant DNS data, if resource constraints prohibit checking both above-delegation and below-delegation NS and glue.

13. Acknowledgements

The authors gratefully acknowledge the substantial contributed material and editorial scrutiny of Anne Bennett. She directed the

reorganization and clarification of the entire document.

Eric Ziegast, Jeroen Massar, Ben April, Ray Bellis and Mukund Sivaraman provided improvements to the document and caught errors.

14. References

14.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, DOI 10.17487/RFC1995, August 1996, <http://www.rfc-editor.org/info/rfc1995>.

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <http://www.rfc-editor.org/info/rfc1996>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <http://www.rfc-editor.org/info/rfc2136>.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <http://www.rfc-editor.org/info/rfc2845>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements",

RFC 4033, DOI 10.17487/RFC4033, March 2005, <http://www.rfc-editor.org/info/rfc4033>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <http://www.rfc-editor.org/info/rfc4034>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[RFC5936] Lewis, E. and A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, DOI 10.17487/RFC5936, June 2010, <http://www.rfc-editor.org/info/rfc5936>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <http://www.rfc-editor.org/info/rfc5952>.

14.2. Informative References

[ISC-ARM] Internet Systems Consortium, "BIND 9 Administrator Reference Manual, https://ftp.isc.org/isc/bind9/cur/9.10/doc/arm/Bv9ARM.ch06.html#rpz", 2016.

[ISC-RPZ] Vixie, P. and V. Schryver, "DNS Response Policy Zones (DNS RPZ, Format 3), https://ftp.isc.org/isc/dnsrpz/isc-tn-2010-1.txt", 2010.

[ISC-RRL] Vixie, P. and V. Schryver, "DNS Response Rate Limiting (DNS RRL), https://ftp.isc.org/isc/pubs/tn/isc-tn-2012-1.txt", 2012.

Appendix A. Examples

An existing data feed capable of producing an RHSBL can be trivially used to generate a DNS RPZ. If the desired policy is to alias targeted domains to a local walled garden, then for each domain name, generate the following records, one for the name itself and perhaps

also one for its subdomains:

```
    bad.example.com             CNAME  walled-garden.example.net.
    *.bad.example.com           CNAME  walled-garden.example.net.
```

If it is desirable to return NXDOMAIN for each domain (and its subdomains in this example), try this:

```
    bad.example.com             CNAME   .
    *.bad.example.com           CNAME   .
```

Try this if there are walled gardens for mail versus everything else:

```
    bad.example.com             MX      0 wgmail.example.net.
    bad.example.com             A       192.0.2.66
    *.bad.example.com           MX      0 wgmail.example.net.
    *.bad.example.com           A       192.0.2.66
```

An extended example follows:


```
    $ORIGIN rpz.example.net.
    $TTL 1H
    @                   SOA LOCALHOST. named-mgr.example.net. (
    1 1h 15m 30d 2h) NS LOCALHOST.

    ; QNAME policy records.
    ; There are no periods (.) after the relative owner names.
```

```
nxdomain.example.com        CNAME   .       ; NXDOMAIN policy
nodata.example.com          CNAME   *.      ; NODATA policy

; Redirect to walled garden
bad.example.com             A       10.0.0.1
                            AAAA    2001:db8::1

; Rewrite all names inside "AZONE.EXAMPLE.COM"
; except "OK.AZONE.EXAMPLE.COM"
*.azone.example.com         CNAME   garden.example.net.
ok.azone.example.com        CNAME   rpz-passthru.

; Redirect "BZONE.EXAMPLE.COM" and "X.BZONE.EXAMPLE.COM"
; to "BZONE.EXAMPLE.COM.GARDEN.EXAMPLE.NET" and
; "X.BZONE.EXAMPLE.COM.GARDEN.EXAMPLE.NET", respectively.
bzone.example.com           CNAME   *.garden.example.net.
*.bzone.example.com         CNAME   *.garden.example.net.

; Rewrite all answers containing addresses in 192.0.2.0/24,
; except 192.0.2.1
24.0.2.0.192.rpz-ip         CNAME   .
32.1.2.0.192.rpz-ip         CNAME   rpz-passthru.

; Rewrite to NXDOMAIN all responses for domains for which
; "NS.EXAMPLE.COM" is an authoritative DNS server for that domain
; or any of its ancestors, or that have an authoritative server
; in 2001:db8::/32
ns.example.com.rpz-nsdname  CNAME   .
32.zz.db8.2001.rpz-nsip     CNAME   .

; Local Data can include many record types
25.128.2.0.192.rpz-ip       A       172.16.0.1
25.128.2.0.192.rpz-ip       A       172.16.0.2
25.128.2.0.192.rpz-ip       A       172.16.0.3
25.128.2.0.192.rpz-ip       MX      10 mx1.example.com
25.128.2.0.192.rpz-ip       MX      20 mx2.example.com
25.128.2.0.192.rpz-ip       TXT     "Contact Central Services"
25.128.2.0.192.rpz-ip       TXT     "Your system is infected."
```

Authors' Addresses

Paul Vixie
Farsight Security, Inc.

Email: paul@redbarn.org

Vernon Schryver
Rhyolite Software, LLC

Email: vjs@rhyolite.com