Workgroup: IPv6 Operations
Internet-Draft: draft-hunek-v6ops-nat64-srv
Published: 14 June 2023
Intended Status: Standards Track
Expires: 16 December 2023
Authors: M. Hunek
Technical University of Liberec

NAT64/DNS64 detection via SRV Records

Abstract

This document specifies how to discover the NAT64 pools and DNS servers providing DNS64 service to the local Nodes. The discovery made via SRV records allows the assignment of priorities to the NAT64 pools and DNS64 servers. It also allows Nodes to have different DNS providers than NAT64 providers while providing a secure way via DNSSEC validation of provided SRV records. This way, it allows providing the NAT64/DNS64 services regardless of DNS operator and DNS transport protocol.

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 https://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 16 December 2023.

Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

The slower-than-expected adoption of IPv6 resulted in the need for reliable transition mechanisms to shut down legacy protocols in the early adopters' network without waiting for latecomers. Transition mechanisms like NAT64/DNS64 or 464XLAT [RFC6877] are essential in the transition between dual-stack networks and IPv6-only networks while not sacrificing the accessibility of the IPv4-only services. It is essential for these transition mechanisms to reliably and securely detect prefixes used for translation. Failing to do so, the IPv4-only services would not be accessible, or the traffic for these services could be kidnapped.

There are multiple solutions for detecting NAT64 prefixes, but none of those are without problems and can fit different applications' needs. This document describes a new DNS-based method that could replace the method standardized by [RFC7050] and lately updated by [RFC8880], as this detection method is incompatible with DNSSEC and does have unrealistic prerequisites.

This document is not proposing where the DNS64 synthesis should take place. It only provides a secure way to transmit information about Pref64::/n and inform a Node about the DNS recursive resolvers providing the DNS64 service. A Node can use such information to

perform DNS64 synthesis locally, set up the CLAT portion of the 464XLAT [RFC6877], or redirect DNS queries for non-existing AAAA records to the resolver providing the DNS64 service.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

CLAT: Customer-side translator as defined in [RFC6877].

Node: Either physical device or an application capable of performing DNS queries.

NAT64 FQDN: a fully qualified domain name (FQDN) for a NAT64 protocol translator.

Pref64::/n: a IPv6 prefix used for IPv6 address synthesis [RFC6146].

Pref64::WKA: an IPv6 address consisting of Pref64::/n and WKA at any of the locations allowed by RFC 6052 [RFC6052].

Secure Channel: a communication channel a Node has between itself and a DNS64 server protecting DNS protocol-related messages from interception and tampering. The Channel can be, for example, IPsec-based virtual private network (VPN) tunnel or a link-layer utilizing data encryption technologies.

Well-Known IPv4 Address (WKA): an IPv4 address that is well-known and present in an A record for the well-known name as defined in [RFC7050].

3. Problems with Current Solutions

For means of comparison, current solutions are split into two groups. The first one is the DNS-based solutions and the second one are solutions based on other protocols.

3.1. DNS-based method

The DNS based method is represented by the current method of [RFC7050] updated by the [RFC8880]. This method uses the Well-Known Name 'ipv4only.arpa.' with only an A record to detect DNS64 service. As this method depends on a specific DNS64 capable resolver with a specific Pref64::/n, a Node has to use the resolver provided by the

NAT64 service provider - at least for the Pref64::/n detection. Furthermore, information about the Pref64::/n in use is distributed only locally, so the third-party resolvers have no information about it, so they cannot provide DNS64 service for them.

With the introduction of the DNS-over-HTTPS (DoH) [RFC8484], the introduction of the third-party resolver made the [RFC7050] unusable for Nodes using DoH. There is a quick fix provided by the [RFC8880] that the Well-Known name should be treated differently - resolved by autoconfigured resolver on a specific outbound interface only. However, this would mean that the application/system stub resolver has to keep track of the source of configured DNS resolvers, which may be an unrealistic expectation.

Another design property of the [RFC7050] is its incompatibility with the DNSSEC. In order for [RFC7050] to work, the DNSSEC has to be turned off, and even the detection phase of this method could not use it to verify the provided information. As the network operator does not own the 'arpa.' domain, it cannot properly sign the AAAA record for the Well-Known Name. Because of it, the first step of the Pref64::/n detection is always insecure.

A Node can still do the DNS64 synthesis locally and verify the A records by the DNSSEC, but it is performing the synthesis based on information obtained via the insecure method ([RFC7050]/[RFC8880]) - the Pref64::/n is not and cannot be secured by the DNSSEC.

In order for [RFC7050] method to be secure, this method requires these prerequisites:

- *DNSSEC signed NAT64 FQDN
- *Corresponding PTR
- *Secure Channel between Node and resolver
- *Trusted domain list
- *No user input

The [RFC8880] adds another set of prerequisites:

- *Stub resolver must distinguish between configuration sources of recursive DNS
- *Only autoconfiguration sources can provide recursive DNS to resolve Well-Known Name
- *Recursive DNS resolver is interface specific

Some of these listed prerequisites cannot be achieved in certain networks without prior provisioning of the Node. This includes recommended secure channel between a Node and DNS64 recursive resolver on shared network segments and the Trusted Domain List mandatory requirement that implicitly cannot be entered by a user. As not every Node is provisioned by a network operator, especially in smaller networks, and the generation process of the Trusted Domain List is not defined. The implementations of the [RFC7050] seems to ignore this requirement. However, without it, these implementations are insecure. This is due to the fact that the path between a Node and a resolver cannot be secured by the DNSSEC, and without Trusted Domain List, any arbitrary data can be injected into a Node configuration.

Requirements of the [RFC8880] are also hard to implement strictly according to the standard. When the user-space application vendor that implements its own stub resolver would like to implement [RFC8880], it would require access to the information about network configuration to keep track of which recursive DNS server has been received from which interface and from which protocol. Presenting such information to the user-space is not typical and requires system-level changes. Furthermore, when strictly following the [RFC8880], the network cannot use static configuration to have NAT64 functionality autoconfigured from the DNS.

3.2. Methods based on other protocols

There are other solutions for detecting Pref64::/n based on various protocols. Namely [RFC7225], [RFC8115] and [RFC8781]. Regardless of the protocol used, these solutions have some common properties that limit their user-space use. If an application vendor would like to implement any of these methods, it would need to include a client implementation of an underlying protocol, or the system would need to provide an interface to obtain Pref64::/n detected by these methods to the user-space application. This is much harder than making a DNS query inside an application.

Another common property of these methods is an expectation of local DNS64 synthesis or CLAT presence on a Node. This may be the case for some Nodes, but others may depend on this functionality from the DNS64 resolver. For such Nodes, the DNS-based detection mechanism could be the preferred solution.

It is fair to say that these methods are viable solutions for system-level Pref64::/n detection - implementations of a system DNS stub resolver or CLAT daemon. These methods are not so easy to implement for user-space applications and daemons that are not so tightly integrated into an operating system.

4. Local domain detection

The Node should perform detection of the domain used by the network operator. A Node MAY use any source of such information, but a Node implementing the method described in this document MUST be able to use the PTR record for Node's unicast address as one of such source.

A network operator that uses the method described in this document to distribute NAT64 configuration to Nodes connected to its network MUST provide a PTR record for every IPv6 address handed to the Node. The PTR record MUST have a valid DNSSEC signature and MUST point to a securely delegated zone with DNSSEC signed NAT64 SRV record. The SRV record itself MUST point to DNSSEC signed AAAA record, and MAY also point to DNSSEC signed A record.

Both Node and a network operator MAY use other sources of information about a local domain. If they decide to do so, the channel to provide such information MUST be secured against undetected data manipulation, as these sources may not provide the same level of security as the DNSSEC signed PTR record.

Other sources of information about local domain MAY include (but are not limited to):

- *Node's FQDN
- *Router Advertisement DNSSL option [RFC8106]
- *DHCPv6 options: 57, 24, 39, 74 or 118

At least when detection of the local domain is done by the PTR record, a Node MUST consider not only its FQDN as the detected local domain. When there is no SRV record associated with the detected domain name, a Node MUST disregard the lowest level domain (part of the domain name until the first dot) and repeat the detection process. This MUST be repeated until there is an SRV record associated with the domain name (even the empty "." one) or until the top-level domain for the respective FQDN is reached. This might be the second-level domain, but it can also be the third-level. Implementers MAY use tools like the Mozilla public suffix list ([PubSuffix]) to achieve that. The same process MAY be deployed for other domain detection sources as well.

If a Node has more than one global IPv6 address, it MUST run PTR resolution for every address with a stable suffix. If a Node uses temporary address suffixes, a Node SHOULD perform just one PTR resolution for every network prefix. If a Node is using both stable suffixes and temporary suffixes in a single network prefix, only the stable ones MUST be used for PTR resolution.

5. NAT64 service SRV record

This document specifies two new well-known SRV records. The Node MUST implement the one for Pref64::/n:

_nat64._ipv6.Name TTL Class SRV Priority Weight Port Target

The TTL, Class, Priority, and Weight follow the same scheme as defined in [RFC2782] and have their standard meaning. The service name follows the naming convention defined in [RFC6763].

Port: IPv6, as the L3 protocol, does not use port numbers. Because of that, this field SHOULD be either set to zero or MAY be used to indicate the network prefix length in both IPv6 and IPv4 used for NAT64. In such a case, the port 16b integer MUST be constructed by directly appending the IPv4 pool prefix length after the IPv6 prefix length decimally. Usually, this would mean 9632, stating that the IPv6 prefix with a length of /96 is translated into a single IPv4 address (/32).

Target: MUST point to AAAA record formed from Pref64::/n prefix and WKA same way as in [RFC7050] (Pref64::WKA). The target MAY also points to A record, where it SHOULD point to the IPv4 address used for NAT64 (or base address of the NAT64 IPv4 prefix). A network operator MAY indicate to Node that NAT64 service is not provided by putting root domain target (".") into the SRV record. The Port field value MUST be set to zero for such a record, and Node MUST stop further Pref64::/n detection for a given domain.

Note: The target MAY also point to AAAA record of Any-Source Multicast prefix or Source-Specific Multicast prefix, similarly to [RFC8115] this MAY be used to indicate a Node prefix used for multicast translation. For this reason, a Node MUST check address type before its use. One SRV record MUST NOT combine unicast and multicast targets, and in the case of a multicast target, the Port field value MUST be set to a value of 9600, and A record target MUST be ignored by a Node.

6. DNS64 service SRV record

The second SRV record is for the discovery of the DNS64 service. Support of this record is OPTIONAL, but Node SHOULD implement it.

_dns64.Protocol.Name TTL Class SRV Priority Weight Port Target

Record informs about location of DNS64 service. This record might be used if the network operator does not want to deploy DNS64 in their main DNS infrastructure. A DNS64 SRV record follows the rules specified by [RFC2782] and does not modify the meaning of any field.

Server provided by this record SHOULD only be used for domain names which have returned NODATA for AAAA record and for A record queries when a Node is not performing DNS64 function and is not using CLAT.

7. Node Behavior

In the initial stage of the Node connected to the network - after the Node is configured with an IP address; the Node MUST get local domains used in the network. The method of obtaining such information is described in the section Local domain detection. When no local domain can be discovered, the Node SHOULD continue NAT64/DNS64 detection by other means.

After the list of local domains has been established, the Node MUST query for a NAT64 SRV record for every domain in the list. The result of such queries SHOULD be ordered by following the rules of [RFC2782]. When multiple records have equal values of both priority and weight, the records SHOULD maintain the same order as its domain in the discovered domain list.

If a Node is not configured to perform DNS64 address synthesis and is not using CLAT, it SHOULD perform a query for DNS64 SRV record for every discovered domain with NAT64 SRV record. If such a record is obtained, the Node SHOULD use preferred target of DNS64 SRV record to query for FQDNs without AAAA record - when Node received NODATA response for its query. Similarly, such Node SHOULD prefer target of DNS64 SRV record for any A record query (like caused by Happy Eye-Balls).

If the Node can validate DNS records via DNSSEC, the Node MUST perform validation of NAT64/DNS64 SRV record. The default behavior of Node SHOULD be to ignore any NAT64/DNS64 SRV records which cannot be validated or did not pass the validation.

Any information received from DNS MUST respect TTL of received records. The Node MUST perform a new detection before currently used information expires. This also applies to information received from other sources that include expiration.

7.1. Interaction with other methods

Proposed method does not aim to replace all other Pref64::/n detection methods. In fact, it should be the network operator who should decide which detection method should be used in the network and which should have a preference. One advantage of using SRV records for NAT64 detection is their Priority and Weight fields that allows to communicate such preference to a Node.

In accordance with the latest detection method the [RFC8781], other detection method should be treated equally to SRV method with following Priority and Weight fields:

Method	Priority	Weight
RFC8115	100	0
RFC7225	150	0
RFC8781	200	0
RFC7050	250	0

Table 1: Default priorities of other methods

If a network operator prefers another method than the SRV method and wants to provide the SRV method as a fallback, it should set the priority field of the NAT64 SRV record to a higher value than specified for a method that should be used as a primary. For example, when a network operator uses Router Advertisement to distribute Pref64::/n information to its network and also wants to use the SRV method as a fallback; it should set Priority field to number higher than 200.

It is RECOMMENDED that network operators SHOULD NOT use values higher than 249 in the Priority field of the NAT64 SRV record unless they want to use [RFC7050] as a primary source of Pref64::/n configuration, and they have all Nodes connected in their network properly configured for this. In order for this configuration to be safe, the Nodes MUST follow all the mandatory and optional requirements of both [RFC7050] and [RFC8880]. Otherwise, the [RFC7050] SHOULD NOT be used as a primary configuration source.

Default priority values on Node SHOULD be user-configurable.

A Node MAY start the NAT64 detection process by performing the SRV method. If it is successful and the SRV record Priority field value is lower than configured values for other methods, the Node MUST NOT use other detection methods (or utilize information received by them). If the Priority value is higher than configured Priority value of any other methods, the Node SHOULD also perform detection methods with the lower priority values. Detection SHOULD be done starting from the lowest configured Priority value to the highest. The successful completion of any detection method MUST stop further detection.

Similarly, the DNS64 function of the recursive resolver in use SHOULD be treated equally to the DNS64 SRV record with the Priority field value of 250. If the Node supports the DNS64 SRV record, Node is not performing DNS64 function, it is not using CLAT and the DNS64 SRV record has a lower Priority field value; the A record queries

MUST be sent to the target of such SRV record instead of Node's default recursive resolver.

If the network configuration time for NAT64 is more important than prefix stability, a Node MAY perform other detection methods simultaneously with this SRV method. When a Node receives Pref64::/n by method with a higher priority (lower Priority field value), a Node MUST respect the method Priority field, and it MUST stop using configuration information received by a method with a lower priority (a higher Priority field value). However, by doing so, it can resolve several NAT64 prefix configuration changes as methods with higher priorities would override those with lower ones. This may result in several prefixes being used in a short time. For this reason, it is NOT RECOMMENDED to act upon any detection method that can be overwritten by a method with a higher priority unless the configuration of the Pref64::/n is time-critical.

Regardless of the detection method used for DNS64 discovery, the Node MUST NOT accept any DNS64 synthesized AAAA record outside detected NAT64 prefixes.

8. Example

The Node is a home router connected to the ISP network in which the NAT64/DNS64 is used, and the ISP has the following SRV records in their zones:

```
*_nat64._ipv6.example.com. IN SRV 5 10 9632 nat64-
 pool-1.example.com.

*nat64-pool-1.example.com. IN AAAA 2001:db8:64:ff9b:1::c000:aa

*nat64-pool-1.example.com. IN A 192.0.2.64

*_nat64._ipv6.example.com. IN SRV 10 10 9632 nat64-
 pool-2.example.com.

*nat64-pool-2.example.com. IN AAAA 2001:db8:64:ff9b:2::c000:aa

*nat64-pool-2.example.com. IN A 192.0.2.164

*_nat64._ipv6.example.net. IN SRV 10 10 9624 nat64-
 pool.example.net.

*nat64-pool.example.net. IN AAAA 2001:db8:64:ff9b:abc::c000:aa

*nat64-pool.example.net. IN A 198.51.100.0

*_nat64._ipv6.example.invalid. IN SRV 10 10 9624 nat64-
 pool.example.org.
```

*nat64-pool.example.org. IN AAAA 2001:db8:64:ff9b:def::c000:aa

*nat64-pool.example.org. IN A 203.0.113.0

In addition, the zones "example.net" and "example.invalid" has got DNS64 SRV records:

*_dns64._tcp.example.net. IN SRV 5 10 53 dns64.example.net.

*_dns64._udp.example.net. IN SRV 10 10 53 dns64.example.net.

*dns64.example.net. IN AAAA 2001:db8::53

*_dns64._udp.example.invalid. IN SRV 10 10 53 dns64.example.org.

*dns64.example.org. IN AAAA 2001:db8:123::53

The Node has detected the following list of domains:

1. example.net

2. example.invalid

3. example.com

4. example.org

The Node would fetch all available SRV records and their A and AAAA counterparts and sort them in the following order:

Pool	DNSSEC	Priority	Reason
nat64-pool-1.example.com.	yes	5	lowest priority field
nat64-pool.example.net.	yes	10	discovered first
nat64-pool-2.example.net.	yes	10	higher priority field
nat64-pool.example.org.	no	10	no valid DNSSEC chain

Table 2: Detectected Prefixes

After sorting, the DNSSEC validating Node SHOULD graylist any record which cannot be validated by the DNSSEC. This example would be "nat64-pool.example.org." because it has been obtained from insecure domain "example.invalid". Such pool SHOULD NOT be used if it is not confirmed by other DNSSEC secured record.

If the Node can act as a recursive or caching DNS server and it is configured to provide the DNS64 service, it MUST provide this service using a sorted list of NAT64 pools. For such Node, the process of the NAT64/DNS64 ends here.

However, when the Node is not capable of performing AAAA record synthesis or it is not configured to provide DNS64 service, and it is not using CLAT, it MUST perform detection of DNS64.

When the Node supports the DNS64 SRV record, it MUST make a sorted list of DNS64 servers from the DNS64 SRV records. If the Priority field of the corresponding DNS64 record is higher than 250, and when the Node does not support the DNS64 SRV record; the Node MUST perform DNS64 detection for specified NAT64 pool by the [RFC7050] method.

The detection, according to [RFC7050], should be done by querying for "ipv4only.arpa". If the reply contains a pool listed in the NAT64 pool list, the corresponding entry is marked as having DNS64 provided by recursive DNS.

The DNS64 sorted list would look like this:

Server	Proto	DNSSEC	Priority	Reason
dns64.example.net.	tcp	yes	5	lowest priority field
dns64.example.net.	udp	yes	10	higher priority field
dns64.example.org.	udp	no	10	no valid DNSSEC chain

Table 3: Detectected DNS64 Servers

Sorting is done in the same fashion as any other SRV record with the same exception of graylisting records without a valid DNSSEC chain. Those SHOULD NOT be used when not confirmed by DNSSEC validated record and SHOULD be kept at the end of the list.

For example, when ISP is providing DNS64 service in their main DNS infrastructure only for pools in the domains "example.com" and "example.org" and the pool "nat64-pool.example.net" is used only with corresponding DNS64 server. The final sorted list of NAT64 prefixes used by the Node in the ISP network would be:

Pool	State	Priority	Reason
nat64-pool-1.example.com.	active	5	lowest priority field
nat64-pool-2.example.net.	backup	10	higher priority field
nat64-pool.example.net.	active*	10	only DNS64 SRV capable Node
nat64-pool.example.org.	inactive	10	no valid DNSSEC chain

Table 4: Used Prefixes

As the pool "nat64-pool.example.net" is used only with the server "dns64.example.net", this would effectively make it usable only for

Nodes supporting DNS64 SRV and not running DNS64 locally or not using CLAT. For such Nodes, this pool would have priority over others because lower Priority field value of the DNS64 SRV record.

Now, the Node has successfully identified NAT64 pools and the DNS64 servers in the ISP infrastructure. The discovered prefixes SHOULD be considered safe, and DNSSEC validation of answers in these prefixes, when a remote recursive resolver does the DNS64 synthesis, it MUST be either disabled or performed by validating only the suffix.

8.1. Example of negative records

The proposed method allows specifying negative records for the Pref64::/n. This can be used to specify that a single Node should not use the Pref64::/n, that the whole subdomain is not allowed to use NAT64, that the whole domain/operator is not using NAT64, or as a last resort indication that non-specified Nodes/subdomains are not allowed to use NAT64. The meaning of a negative record is given by its placement in the zone and other positive replies (pointing to a valid AAAA record).

```
*_nat64._ipv6.example.com. IN SRV 5 10 0 .

*_nat64._ipv6.clients.example.com. IN SRV 5 10 9632 nat64-
 pool-1.example.com.

*_nat64._ipv6.bad-host1.clients.example.com. IN SRV 5 10 0 .

*_nat64._ipv6.bad-host2.clients.example.com. IN SRV 255 10 0 .
```

The list shows a possible use of negative records. Every Node in the "clients" subdomain is given a Pref64::/n provided by the "nat64-pool-1" AAAA record, except for the "bad-host1" and "bad-host2" Nodes that are given negative records. This indicates to those Nodes that there is no Pref64::/n, so no NAT64 service is provided to them. This way, an operator is able to disable the NAT64 service to individual Nodes.

The difference between "bad-host1" and "bad-host2" is in the Priority field. Because of that, the "bad-host1" Node MUST NOT use other detection methods for NAT64 detection, while "bad-host2" MAY utilize any other method.

By placing the negative answer to the root of the operator's domain, the operator specifies that only listed Nodes or subdomains are allowed to use NAT64. Similarly, if the operator specified a positive record, non-listed Nodes would default to using such prefix. Basically, this allows to form policies like allowlists and blocklists and combine them.

9. IANA Considerations

This document proposes two services, "_nat64" and "_dns64" in Service field of SRV RR ([RFC2782]).

10. Security Considerations

The method proposed by this document relies on security principles based on DNSSEC and secure discovery of local domain. In order to be secure, the network operator MUST deploy DNSSEC on at least one domain (advertised to the Node), establish a secure channel to this advertisement, or provide every IPv6 address given to a Node with DNSSEC secured PTR record.

11. Normative References

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

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <https://www.rfc-editor.org/info/rfc2782>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <https://www.rfc-editor.org/info/rfc6146>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/info/rfc6763>.

[RFC7050] Savolainen, T., Korhonen, J., and D. Wing, "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis", RFC 7050, DOI 10.17487/RFC7050, November 2013, <https://www.rfc-editor.org/info/rfc7050>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12. Informative References

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, DOI 10.17487/RFC6052, October 2010, <https://www.rfc-editor.org/info/rfc6052>.

[RFC6877] Mawatari, M., Kawashima, M., and C. Byrne, "464XLAT: Combination of Stateful and Stateless Translation", RFC 6877, DOI 10.17487/RFC6877, April 2013, <https://www.rfc-editor.org/info/rfc6877>.

[RFC7225] Boucadair, M., "Discovering NAT64 IPv6 Prefixes Using the Port Control Protocol (PCP)", RFC 7225, DOI 10.17487/RFC7225, May 2014, <https://www.rfc-editor.org/info/rfc7225>.

[RFC8106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 8106, DOI 10.17487/RFC8106, March 2017, <https://www.rfc-editor.org/info/rfc8106>.

[RFC8115] Boucadair, M., Qin, J., Tsou, T., and X. Deng, "DHCPv6 Option for IPv4-Embedded Multicast and Unicast IPv6 Prefixes", RFC 8115, DOI 10.17487/RFC8115, March 2017, <https://www.rfc-editor.org/info/rfc8115>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/info/rfc8484>.

[RFC8781] Colitti, L. and J. Linkova, "Discovering PREF64 in Router Advertisements", RFC 8781, DOI 10.17487/RFC8781, April 2020, <https://www.rfc-editor.org/info/rfc8781>.

[RFC8880] Cheshire, S. and D. Schinazi, "Special Use Domain Name 'ipv4only.arpa'", RFC 8880, DOI 10.17487/RFC8880, August 2020, <https://www.rfc-editor.org/info/rfc8880>.

[PubSuffix] Mozilla, "Public Suffix List", April 2022, <https://publicsuffix.org/>.

Acknowledgements

The author of this document would like to thank Lee Howard, Gert Doering, Fred Baker, Philip Homburg, Mikael Abrahamsson, Jordi Palet Martinez, Gabor Lencse, Dan Wing, Ralf Weber, Ted Lemon, David Schinazi for their valuable comments.

Author's Address

Martin Hunek
Technical University of Liberec
Studentska 1402/2
46017 Liberec
Czechia

Email: martin.hunek@tul.cz