Detection of NAT64/DNS64 by SRV Records: Detection Using Global DNS Tree in the World Beyond Plain-Text DNS

Martin Hunek(✉) and Zdenek Pliva

Institute of Information Technology and Electronics, Technical University of Liberec, Studentska 1402/2, 46117 Liberec 1, Czech Republic
{martin.hunek,zdenek.pliva}@tul.cz
https://www.tul.cz

Abstract. Since it has been introduced the NAT64/DNS64 transition mechanism has reputation of method which simply works. This could change as currently used detection method, RFC7050 [16], for this transition mechanism doesn't work with third party/foreign DNS resolvers. These resolvers have been lately introduced by Mozilla Firefox [1] with implementation of DNS over HTTPS. This paper describes problems connected with default usage of third party DNS resolvers and provides a way how to solve issues of RFC7050 [16] with and without third party resolvers.

Keywords: NAT64/DNS64 · DNS · DNSSEC · DoH · RFC7050

1 Introduction

The Internet Protocol version 6 (IPv6) and a whole internet changed a lot during more than 20 years the first standard of IPv6 has been around. At a beginning every device used EUI-64 as its identifier, device autoconfiguration has been split into two services - neighbor discovery for routing and addressing in case of Stateless Address Autoconfiguration (SLAAC) and Dynamic Host Configuration Protocol version 6 (DHCPv6) for everything else. Global Content Distribution Networks (CDNs) didn't exist, cloud was still just a condensed water in the atmosphere and most of the internet services where self-hosted. Because of that the internet itself was truly decentralized network, made of smaller networks connected in several Internet Exchange Points (IXPs).

When going bit more to the history, to the beginning of internet itself, to days when Internet Protocol version 4 (IPv4) switchover happened. The internet was a network of mutually trusted networks with network administrators, who knew each other. And from this age some of the essential protocols have been established. These include telnet, Simple Mail Transfer Protocol (SMTP), Domain Name System (DNS) and many more. Some of them are not in use

P. Gaj et al. (Eds.): CN 2020, CCIS 1231, pp. 27–40, 2020.
https://doi.org/10.1007/978-3-030-50719-0_3

nowadays, some really shouldn't but still are (telnet) and some still performs its essential role in the internet - like DNS.

Unfortunately, we are not living in the same world and we are not using the same internet anymore. Internet has become a network of content providers making their own CDNs distributed around the globe, slowly taking more and more services concentrated into them. Network administrators no longer knows each other, they hardly know a few people from their IXPs. The Internet is no longer just a network of computers. There are tons of devices which are poorly developed/managed and shouldn't be really connected to public network, the internet has become. It is no longer safe trusted network of professionally maintained computers, it is a jungle for masses.

This change in way how the internet is perceived of course initiated changes in some of these protocols, but those changes also caused some collateral damage which would be described in this article.

2 Encrypted Domain Name System Protocols

As already mentioned, one of the most important protocols of the internet, which wasn't originally designed with any security concerns is a DNS. It was introduced as a replacement of distributed host file. However as network protocol it does not provide a same security level as locally managed file and in the same time a DNS itself does not provide any protection against spoofing, other then race condition.

DNS is plain-text protocol without cryptography signatures and without end to end encryption. At the beginning of the internet it really didn't caused an issues as it has been viewed as trusted network. But as the internet became more broadly adopted, it has been realized that this nature of DNS could be leveraged to perform Man in the Middle (MitM) attacks.

In this attack a client is given spoofed replies to its DNS queries. Attacker utilize either closer proximity to a client or simpler and faster DNS software, as it must provide a same reply on every query. As the only first reply received by a client is used and as there is not cryptography signature present in a reply, client has no means to validate received data, so it is inevitably redirected to attacker.

As a DNS doesn't provide any defense against these kind of attacks, a cryptographic signatures have been introduced into DNS tree by Domain Name System Security (DNSSEC). By establishing the chain of trust from IANA maintained root zone up to every single record in every signed zone, the MitM attack has been mitigated. When a zone for which a validating client is performing a query is signed, any manipulation with reply would be detected and rejected as forged. This is inherently safer then connecting to attacker's device. In fact for validating client, a MitM attack changes to Denial of Service (DoS) type of attack, as when a client receives forged reply first and legitimate second, it may ignore both forged and legitimate. The first one would be ignored due to failure in DNSSEC validation and the second one would get ignored as there was already reply

received for the same query. For this reason a service would not be accessible so it would mean successful DoS attack[1]. However forged reply must not been cached in client so it would mean when attacker would either stop transmitting packet to client or when legitimate reply would arrive first, then a service would become accessible again.

There are also legitimate reasons to modify DNS replies like Network Address Translation 6-to-4 (NAT64)/Domain Name System 6-to-4 (DNS64) transition mechanism (defined in RFC6146 [14] and further explained in [2]) which had to deal with presence of DNSSEC. This will be mentioned further in this article.

The MitM and DoS are not the only risks connected to plain-text DNS. Essentially when using unencrypted channel over insecure network there is always risk of interception and with that interconnected risk of leakage sensitive information, as well as targeted DoS attacks often also used for government censorship.

This privacy related concerns lead to two independent standards of encrypted DNS protocols. Both of them are using the same principle of encapsulation DNS traffic inside encrypted channel but differs in transport channel.

2.1 DNS-Over-TLS

The first method of encapsulation of DNS traffic is the DNS over TLS (DoT) (RFC7858 [9]). It is actually not more then its name says. It is the plain-text DNS encapsulated in either TLS tunnel in case of Transmission Control Protocol (TCP) transport or the DTLS in case of User Datagram Protocol (UDP) transport.

It uses a separate port number 853, and it is an alternative way of transport for the same servers as used for plain-text DNS on port 53. When a client supports this method of transport then it tries to establish connection on port 853. When server supports DoT, then connection is made and DoT is used for transporting DNS queries.

As a DoT uses the same resolvers as regular DNS, it does not require any detection method other then trying to establish a connection on DoT port. Also by using the same resolvers it does not introduce third parties into communication of a client.

The down side of this method is that it allows easy way how to block client access to DoT by setting a firewall forward chain anywhere between client and resolver blocking port 853 which is used solely by DoT. This would also make government censorship easier.

2.2 DNS-Over-HTTPS

The second method, defined by RFC8484 [8] uses, as the name implies, HTTPS as means of transport.

The main advantage of this method is that it is using well-known port of 443, this makes it quite hard to filter DNS traffic from regular HTTPS content.

[1] This behavior can be mitigated by stub resolver and depends on its implementation.

This way it would require performing deep packet inspection and SSL stripping to inspect and filter DNS traffic.

The main disadvantage is a missing detection method. As regular DNS uses IP addresses for accessing resolver, the IP have to use a whole URL to access DoH API. This would seems as a small and insignificant change, however CAs are not issuing TLS certificates for IP addresses and single IP address can host a multiple sites by leveraging SNI. Also URL suffix of DoH API could differ from typical location and in such case using just IP would not provide sufficient information to successful setup DoH.

By missing detection method a DoH clients had to depend either on list of DoH provider shipped with software, or on manual entry made by user. This is especially concerning in case of Mozilla Firefox. Mozilla has made a DoH as a default (currently US only) [1], with default provider being Cloudflare, but with possibility to change to other DoH providers or disabling DoH all together. It is also worth mentioning that one of the DoH providers is also Google (not listed in Mozilla Firefox) and the people working for Google were one which proposed DoH to IETF.

When looking into this from privacy point of view, this leads to less privacy then plain-text DNS. This is caused by introducing a third party into resolving process, in difference to regular DNS, which utilizes Internet Service Provider (ISP) resolver with cached records. By asking DoH provider instead, client is giving that provider every query, which could be directly connected back to client. And by utilizing cookies and other techniques for user tracking on web, it is not just identified as an IP (possibly shared with multiple users in case of IPv4), it is identified as single browser on client.

It is not surprising that people working for Google would propose standard, which would give a third party (DoH provider) all DNS queries, which it would not otherwise get and when it just happen that one of DoH providers is Google itself.

Privacy issues are not the only one connected with DoH. Introducing a third party into DNS also breaks policy based DNS and also split view in DNS. This then breaks current DNS based method for detecting transition mechanisms like NAT64/DNS64 and also situation when network uses private addresses for locally hosted services.

The Fig. 1 shows both traditional DNS (solid line) and DoH route how a DNS query is send and reply is received. It could be seen that when DoH is used every query is send directly to DoH provider and from its web server is processed via traditional DNS. When would a client ask for locally hosted service, then query would be also send via DoH to the DoH provider, which would then query required record from ISP DNS. But this query would have come from WAN facing interface (from public address not belonging to ISP) so outside view would be used. This could potentially lead to unreachable service as ISP could use RFC1918 [15] addresses for local access or requested service could be accessible for locally connected clients only. If so, then client querying record for such service could receive either public address of such service (which would not

be reachable by client) or client would receive empty reply as a service would be internal only and it would appear that client is connecting from outside of local network. This is why a DoH breaks split view zones.

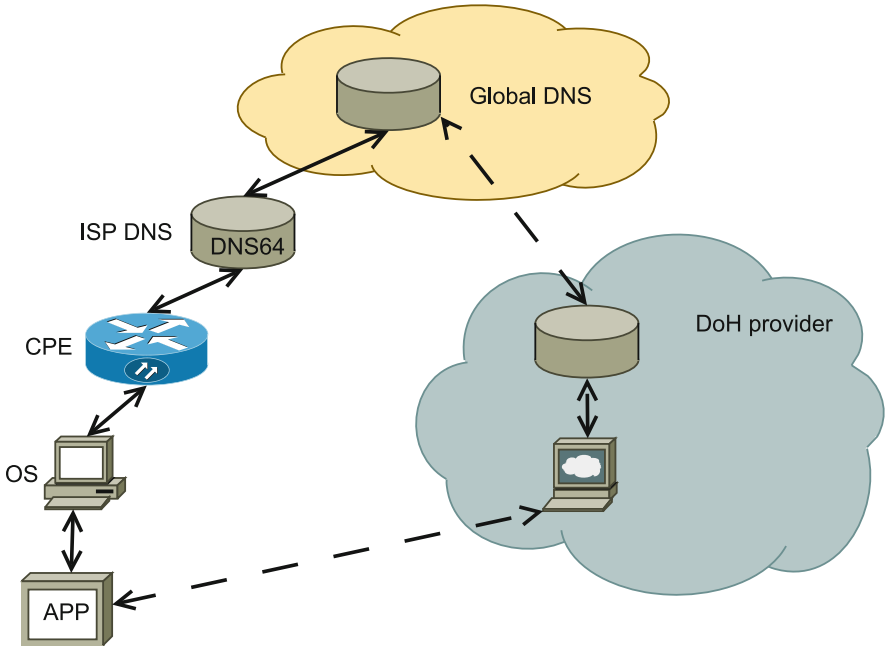

Fig. 1. Comparison of DoH and traditional DNS route of queries

It is fair to say, that both privacy and split view problem is not direct property of DoH. These issues are caused by its implementation and by lack of detection method. There are some studies which are advocating use of DoH like [3] or [13] but such studies are concentrated on performance, availability measurements and stub resolver to recursive resolver link security. From that point of view DoH seems fine, but they are missing problem of metadata leakage to the third party - DoH provider. Such problem is stated in [7].

3 NAT64/DNS64

The NAT64/DNS64 is one of transition mechanisms which utilizes two separate technologies. One is NAT64 which translates between IPv6 and IPv4 and vice versa (this is a difference between NAT44 - commonly referred as Network Address Translation (NAT) and NAT66 which translates addresses inside a same address family).

The second part of this mechanism is DNS64. It provides synthesized AAAA record for services only having A record. This way, as IPv6 has priority over

IPv4, client has possibility to connect to IPv4 only service via IPv6 network. As DNS64 is synthesizing records not originally present in a zone, it conflicts with DNSSEC. This conflict has to be resolved by NAT64/DNS64 detection method, otherwise DNS64 response would get discarded by client so that NAT64 would not be used and if client has no IPv4 connectivity it would not be able to reach IPv4 only service.

3.1 Current Detection Method - RFC7050

Detection method specified in RFC7050 [16] is DNS based solution. It uses a AAAA query for Well-Known IPv4-only Name (WKN) *ipv4only.arpa.* which has got only IPv4 address in the global DNS tree. This query is performed with DNS flag "CD" set to zero so that DNS64 could perform address synthesis. This query and reply is shown by Fig. 2.

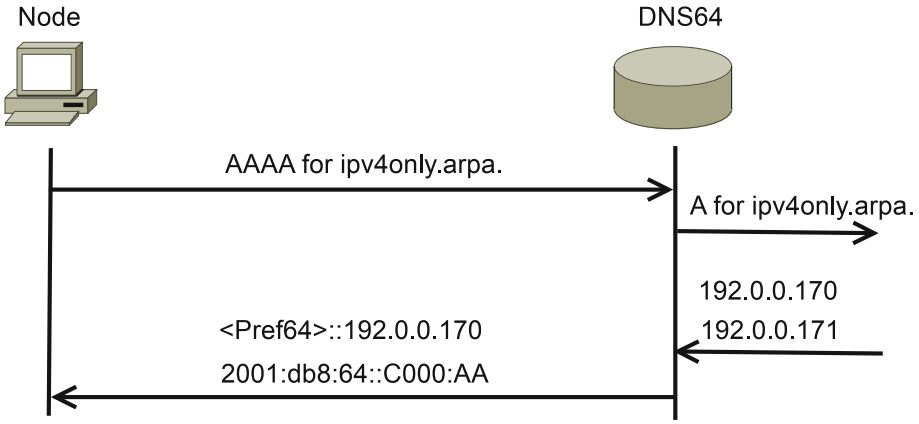

Fig. 2. Detection of NAT64 prefix according to RFC7050 [16] (Source: [10])

When client receives reply containing IPv6 addresses, then DNS64 is present in network and prefixes of these addresses are equal to prefixes used for NAT64. In the example shown in Fig. 2 the NAT64 prefix is *2001:db8:64::/96*.

When there is no DNS64 service provided, then client should receive NODATA[2] status code for AAAA *ipv4only.arpa.* query. The RFC7050 [16] also allows reply of NXDOMAIN for negative answer, however this is against specification of RFC1035 as there is an A record for *ipv4only.arpa.* so there is another record type for WKN while NXDOMAIN would indicate that WKN does not exist in DNS tree which is obviously not true. This means either configuration error, bug in DNS resolver implementation or the *arpa.* zone is not being correctly resolved.

[2] NODATA is not actually transmitted as a return code. It is a combination of NOERROR code and missing answer section.

If client is not performing validation of received NAT64 prefix it is allowed to finish detection process in the first step. When client is capable of validation it should proceed with sending PTR queries for every received address and then for every PTR reply it should query AAAA record. Reply of AAAA query then must match with reply for AAAA record of WKN.

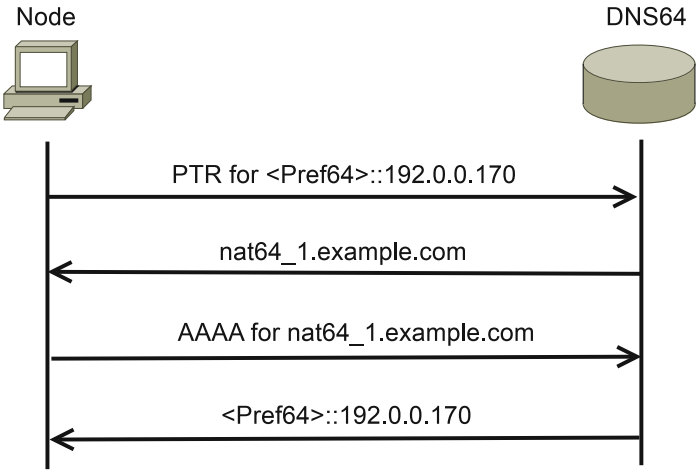

Fig. 3. Validation of NAT64 prefix according to RFC7050 [16] (Source: [10])

Figure 3 shows the whole process of NAT64 prefix validation. It uses generic *<Pref64>::192.0.0.170* and generic domain *example.com.* but it could be substituted by address from previous example in Fig. 2, in which case address in PTR query and AAAA reply would be *2001:db8:64::c000:aa*. Domain name in PTR reply and AAAA query could be any valid domain name, but it has to be under the ISP control and must lead back to address detected in the first step, otherwise validation would fail.

Nowadays, the most fundamental problem connected with this method would be its dependence on DNS server provided by ISP. Before introduction of DNS the usage of third party DNS servers was a rare setup, which had to be configured manually. Usually a client is provided with DNS resolver address is via autoconfiguration (DHCP or SLAAC) and such information would be also passed to downstream interface of any router, which could be located between a client and ISP autoconfiguration server.

However, after DoH has been introduced and actively used, the third party DNS resolvers have been automatically configured inside a client. Then both system resolver containing NAT64/DNS64 detection method and DNS resolver provided by ISP, which runs DNS64 service, are getting automatically bypassed (shown by Fig. 1). This effectively means no NAT64/DNS64 for clients using DoH and so no IPv4 service in IPv6-only networks which utilizes this transition mechanism.

There are also some security implications connected with this method. The first step the detection process could not be validated by DNSSEC. This is due to the fact that DNS64 provider (usually ISP), is not legitimate holder of *arpa.* zone. This means that AAAA record for WKN could not be signed and it has to be solved by setting a "CD" flag as mentioned previously. When this first query would be intercepted and replied with forged records it would allow to perform DoS, MitM and flooding attacks. Validation would not necessary protect from these situations because it could be successfully passed for every address which would have matching PTR and AAAA record pair. Furthermore, standard is using word "SHOULD" instead of "MUST" for requirement of signing NAT64 AAAA resource record and there is no requirement for signing PTR record. This allows to perform mentioned attacks even on host which doesn't have matching PTR and AAAA record pair.

Standard is trying to address this concerns by requirement of secure channel between client and DNS64 server and via secure domain list. The first requirement could be easily done on some type of network - fixed service utilizing star topology with encapsulation and strict filtering; but is hardly usable for others - utilizing shared segments, bus topology or some radio based networks.

Second mitigation tool is also not universally applicable. Networks forcing strict policies concerning connected devices, which would require provisioning of device prior connection to a network, would be able to populate trusted domain list as required by RFC7050 [16]. But for others, small ISPs using stock firmware for CPEs or on networks with BYOD policy, this requirement would not be possible to fulfill.

3.2 Alternative Means of Detection

Following methods have one thing in common, there are not widely used. It is either because they are using not widely adopted protocol or because of they haven't been standardized yet.

The most relevant of these method already standardized is RFC8115 [5]. It uses DHCPv6 option code 113 called "OPTION_V6_PREFIX64". This option includes two multicast prefixes and prefix lengths and single unicast NAT64 prefix. Problem of this method is that it uses DHCPv6 - protocol which is not mandatory and which is not implemented is some clients (Android) and because of that it is not widely deployed in residential networks. This causes so called "circulus vitiosus" as no support in clients means no deployment in a networks and no deployment means no need for implementation in clients. This religious battle of Android against DHCPv6 caught RFC8115 [5] in it, which is a shame as it could work quite well, maybe even with DoH.

Another possibly relevant way how to detect NAT64/DNS64 is via Port Control Protocol (PCP) - specified by RFC7225 [4]. The PCP is protocol for controlling behavior of NAT and firewall by a client requests. The RFC7225 [4] is extending PCP by option with code 129, which includes NAT64 prefix and its length as well as IPv4 prefixes used for translation (optionally). Even that PCP showed an interesting concept of client managing firewall and NAT of upstream

router, it has not been adopted inside either residential or enterprise networks. Also as non-essential protocol for autoconfiguration, even knowledge of its existence is pretty low among network administrators.

There is also one new method of NAT64/DNS64 detection [6], which uses ICMPv6 Router Advertisement extension. Same as previous method it includes NAT64 prefix, its encoded length and it adds encoded validity time. However it differs in protocol used, which in this case is essential for autoconfiguration of any IPv6 client and further more the ICMPv6 is essential for IPv6 itself. This is a huge advantage of this method and if it gets standardized it has a potential to solve current issues of NAT64/DNS64 detection. Only possible pitfall could be hidden within operating systems (network stack) as it has to provide a way how to distribute learned NAT64 prefix to application including DNS resolver (web browser in case of DoH). If there would not be such interface, then DoH enabled application would have to implement ICMPv6 Neighbor Discovery protocol by themselves in order to support this method.

4 Proposed Detection Method

4.1 Reasoning

Due to the standardization of DoH and its introduction of third party DNS providers as a default settings the problems of RFC7050 [16] had to be addressed. But because failure of previous standards in real network adoption, we had to introduce some design goal for new method.

These design goals are:

Goal 1 No new protocol or alteration of existing one.
Goal 2 Utilize widely supported protocols.
Goal 3 Utilize information already provided by network.
Goal 4 Must work with foreign DNS.
Goal 5 Must not require DNS64 synthesis on a host.
Goal 6 Must not require prior provisioning.
Goal 7 Must provide secure detection over insecure channel.

Goal 1 is purely motivated by ease of standardization process - less changes into working and already deployed protocols means less testing and lower probability of introducing vulnerabilities. The second goal is motivated by situation of RFC7225 [4] and RFC8115 [5]. Both are using non-essential protocols, so any deployment of these standards requires configuration work done by network administrator. This is strongly connected with design goals 3 and 6 as if deployment of detection method is easier for network administrator it is more likely to be deployed.

The fourth design goal is motivated by the current problem of RFC7050 [16] with DoH. The fifth goal is reaction to method currently discussed on 6man work group which utilizes Router Advertisement messages. This method does not provide detection of DNS64, just NAT64 and client is then forced to perform DNS64

itself. It is certainly a way how to solve DNS64 problem but this solution adds requirements on client implementation which hasn't been there up to this point. This could potentially slow down deployment process or restrict its audience on devices with limited resources.

The final design goal is also reaction on RFC7050 [16] and its requirements of secure channel. By somehow expecting trustworthy network, method usage would be either limited or the worse scenario - method would introduce vulnerability to a network, which would deploy it regardless of its prerequisites.

4.2 Principle of Method

The newly proposed method [11] utilize DNS, similar way how it is done in RFC7050 [16], but instead of using split view to *arpa.* zone and well-known name *ipv4only.arpa.* it utilizes SRV records inside local domain. These records are globally resolvable and because of that, they can be used even in the situations when client is using third party DNS resolver.

Draft of this method [11] introduces two new services - SRV records: One for signaling NAT64 prefix (and optionally also outside IPv4 addresses) a the second for providing address of DNS64 resolver. Example of such records in operator zone is shown in Listing 1.1.

```
$ORIGIN example.net

% NAT64 records
_nat64._ipv6     IN SRV  5 10 9632 nat64-pool.example.net
nat64-pool       IN AAAA 2001:db8:64:ff9b::c000:aa
nat64-pool       IN A    192.0.2.64

% DNS64 records - stating priorities
_dns64._tcp      IN SRV  5  10 53  dns64.example.net
_dns64._udp      IN SRV  10 10 53  dns64.example.net
_dns64._tcp      IN SRV  20 10 443 dns64.example.net
dns64            IN AAAA 2001:db8::53
```

A zone of this example contains single NAT64 prefix (*2001:db8:64:ff9b::/96*) which is translated into single IPv4 address (*192.0.2.64*). Length of both prefixes are encoded into port number field of SRV record by prepending an IPv6 prefix length before an IPv4 prefix length. This field could be safely used as IP doesn't have port numbers (port number is used in layer 4 protocols - TCP, UDP etc.) and as 16b number it can handle theoretical maximum of /128 IPv6 prefix length and /32 of IPv4 which would make 12832 in decimal notation. This information is only optional as it is indicating size of IPv4 pool used for translation. As long as operator is using typical prefix length of /96 for embedding an IPv4 address it could just set port number field to 0. This would be signal to client that information about pool size is not available and that IPv4 address is just appended after a NAT64 prefix. Otherwise format of this record does not differ from standard SRV record.

The second proposed record represents DNS64 resolvers. In the example it shows a single DNS64 resolver accessible by plain-text DNS protocol over TCP (most preferred) and UDP and by DoH (least preferred). Implementation of this record is not strictly required for this method to work, but it adds an interesting possibilities for network configuration. Because of this record, operator is given a tool how to indicate client on which servers the DNS64 service is provided and which protocols are used and preferred. This also adds a possibility to run DNS64 service outside of main recursive DNS infrastructure and also provides an easy way how to provide client domain names of ac:DoH servers (which so far does not exits).

4.3 Client Behavior

Client, when connected to a network, will typically receive some autoconfiguration information. This in IPv6 capable network consist of Router Advertisement message which can optionally include *DNSSL* option (domain search list - RFC8106 [12]), and/or DHCPv6 which can include option codes 24, 39, 57, 74, 118 and others which can provide local domain as well. Detection of local domain is important for this proposed method to work, as it uses DNS queries for getting SRV records in this local domain.

When this information is not provided passively as part of autoconfiguration process, client can perform PTR query for its own IPv6 address. This should produce dynamically generated record pointing to resource holder domain (typically either ISP or large company). Such domain can be then used to perform detection process. The last usable source of an information about local domain could be client's Fully Qualified Domain Name (FQDN), which should also include local domain.

After local domain is established, client can start detection method. This consist of querying *_nat64._ipv6* subdomain for every local domain candidate, and if it is not capable of DNS64 synthesis, it should also query for DNS64 service record (*_dns64*) under both TCP and UDP. All replies for this queries must be signed by DNSSEC and their signatures must be valid. If client is capable of validating DNSSEC signatures, it must discard any record with invalid or missing signature. This requirement solves design goal number 7 as when network utilizes countermeasures against spoofing RA or DHCPv6 (RA-guard and DHCPv6 snooping), it provides sufficient proof that provided prefixes and DNS64 servers are legitimate (or at least their addresses). This also conform design goal 6 as DNSSEC key of root zone is included in DNS resolver code and no other key has to be distributed.

If client is not capable of performing DNS64 address synthesis, it can use DNS64 servers provided by SRV record. Default behavior of client in this case should be to use these servers only when it receives NODATA reply for AAAA query. This would most likely indicate presence of an A record, so that it could be IPv4-only service and NAT64/DNS64 must be used to access it. Client could potentially use DNS64 servers from SRV record as default, but it could

potentially override its user's wishes and lead for unintended leak of sensitive information so it should be uses only when there is no reply on AAAA queries.

If a client is capable of performing DNS64 address synthesis, then it can do that with received NAT64 prefix or it can use DNS64 servers. This should be configurable but either option would work.

4.4 Fulfillment of Design Goals

Proposed method fulfills the goal 1 as it is not introducing any new protocols, and it fulfills goal 2 as it is utilizing DNS as source of configuration information and DNS is essential network protocol. Further more, if there will any future version of DNS transport, this method should work with it as it is independent of transport method.

Design goal 3 is at least partially fulfilled. As this method uses *DNSSL* as source of information about local domain (this would be usually present in the network) or DHCPv6 options (which might not present already). In case of simple enterprise network without routers managed by end user (CPE), network administrator can just utilize *DNSSL* already configured and just add corresponding NAT64 SRV and AAAA records. This can be done in single place - forward zone file. When there are CPEs in the network then network administrator must also deploy DHCPv6 option preferably code 57 or dynamic creation of PTR record for client addresses. That is why this goal is considered fulfilled partially - it can work in some networks out of the box, but in other networks additional steps must be taken.

Goal 4 is fulfilled as method uses global DNS tree, so it would work with any recursive resolver and with any transport method. Goal 5 is fulfilled with providing DNS64 SRV records. Goals 6 and 7 are fulfilled with strict requirement of DNSSEC. Only security measures required for network is basic security of autoconfiguration methods, but in absence of these security measures it would not add any other issues to those, which would be already present in such network. Without secure channel it is still possible to perform some attacks on this method like DoS or to intercept DNS queries between client and resolver, but with DNSSEC it is not possible to inject client with rogue prefix as long as client is validating responses.

5 Conclusion

The NAT64/DNS64 is really easy to use and reliable transition mechanism, so it would be a shame to loose such possibility to widely deploy it to phase out the IPv4 in access networks. But with current detection methods this could become a reality, as the most deployed detection method (RFC7050 [16]) is not compatible with DoH resolver in Mozilla Firefox [1]. There are also other methods, how to provide detection of NAT64 prefix, but these either depends on non-essential protocols (not necessary deployed in access networks) or they are not yet standardized.

Detection method described in this document proposes one possible way how to solve issues related to RFC7050 [16] and how to move an information about both NAT64 and DNS64 from local view of *arpa.* zone to operator's global zone. This also allows to secure all related records by DNSSEC, without need to disable validation in any point so that detection can be done over insecure channel.

Acknowledgement. This work was supported by the Student Grant Scheme at the Technical University of Liberec through project nr. SGS-2019-3017.

References

1. Firefox DNS-over-HTTPS (2019). https://support.mozilla.org/en-US/kb/firefox-dns-over-https
2. Bagnulo, M., Garcia-Martinez, A., Beijnum, I.V.: The NAT64/DNS64 tool suite for IPV6 transition. IEEE Commun. Mag. **50**(7), 177–183 (2012). https://doi.org/10.1109/MCOM.2012.6231295
3. Boettger, T., et al.: An empirical study of the cost of DNS-over-HTTPS. In: ACM Internet Measurement Conference (IMC) (2019)
4. Boucadair, M.: Discovering NAT64 IPv6 prefixes using the Port Control Protocol (PCP). RFC 7225, May 2014. https://doi.org/10.17487/RFC7225. https://rfc-editor.org/rfc/rfc7225.txt
5. Boucadair, M., Qin, J., Tsou, T., Deng, X.: DHCPv6 option for IPv4-embedded multicast and unicast IPv6 prefixes. RFC 8115, March 2017. https://doi.org/10.17487/RFC8115. https://rfc-editor.org/rfc/rfc8115.txt
6. Colitti, L., Linkova, J.: Discovering PREF64 in router advertisements. Internet-Draft draft-ietf-6man-ra-PREF64-09, Internet engineering task force, December 2019. https://datatracker.ietf.org/doc/html/draft-ietf-6man-ra-pref64-09. (Work in progress)
7. Hoang, N.P., Lin, I., Ghavamnia, S., Polychronakis, M.: K-resolver: towards decentralizing encrypted DNS resolution. In: The NDSS Workshop on Measurements, Attacks, and Defenses for the Web 2020 (MADWeb 2020), pp. 1–7, February 2020. https://doi.org/10.14722/madweb.2020.23009
8. Hoffman, P.E., McManus, P.: DNS queries over HTTPS (DoH). RFC 8484, October 2018. https://doi.org/10.17487/RFC8484. https://rfc-editor.org/rfc/rfc8484.txt
9. Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., Hoffman, P.E.: Specification for DNS over Transport Layer Security (TLS). RFC 7858, May 2016. https://doi.org/10.17487/RFC7858. https://rfc-editor.org/rfc/rfc7858.txt
10. Hunek, M., Pliva, Z.: DNSSEC in the networks with a NAT64/DNS64. In: 2018 International Conference on Applied Electronics (AE), pp. 1–4, September 2018. https://doi.org/10.23919/AE.2018.8501446
11. Hunek, M.: NAT64/DNS64 detection via SRV records. Internet-Draft draft-ietf-v6ops-nat64-srv-00, Internet Engineering Task Force, March 2019. https://datatracker.ietf.org/doc/html/draft-ietf-v6ops-nat64-srv-00. Work in Progress
12. Jeong, J.P., Park, S.D., Beloeil, L., Madanapalli, S.: IPv6 router advertisement options for DNS configuration. RFC 8106, March 2017. https://doi.org/10.17487/RFC8106. https://rfc-editor.org/rfc/rfc8106.txt
13. Lu, C., et al.: An end-to-end, large-scale measurement of DNS-over-encryption: how far have we come? In: Proceedings of the Internet Measurement Conference IMC 2019, pp. 22–35. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3355369.3355580

14. Matthews, P., van Beijnum, I., Bagnulo, M.: Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. RFC 6146, April 2011. https://doi.org/10.17487/RFC6146. https://rfc-editor.org/rfc/rfc6146.txt
15. Moskowitz, R., Karrenberg, D., Rekhter, Y., Lear, E., de Groot, G.J.: Address allocation for private Internets. RFC 1918, February 1996. https://doi.org/10.17487/RFC1918. https://rfc-editor.org/rfc/rfc1918.txt
16. Savolainen, T., Korhonen, J., Wing, D.: Discovery of the IPv6 prefix used for IPv6 address synthesis. RFC 7050, November 2013. https://doi.org/10.17487/RFC7050. https://rfc-editor.org/rfc/rfc7050.txt