OPSAWG Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: 22 September 2024

M. Richardson
Sandelman Software Works
W. Pan
Huawei Technologies

21 March 2024

# Operational Considerations for Use of DNS in IoT Devices
draft-ietf-opsawg-mud-iot-dns-considerations-13

## Abstract

This document details concerns about how Internet of Things (IoT) devices use IP addresses and DNS names. These concerns become acute as network operators begin deploying RFC 8520 Manufacturer Usage Description (MUD) definitions to control device access.

Alos, this document makes recommendations on when and how to use DNS names in MUD files.

## About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-opsawg-mud-iot-dns-considerations/.

Discussion of this document takes place on the opsawg Working Group mailing list (mailto:opsawg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/opsawg/.

Source for this draft and an issue tracker can be found at https://github.com/mcr/iot-mud-dns-considerations.

## Status of This Memo

Table of Contents

1. Introduction . . . . . 3
2. Terminology . . . . . 4
3. A model for MUD controller mapping of names to addresses . . 4
3.1. Non-Deterministic Mappings . . . . . 4
4. DNS and IP Anti-Patterns for IoT Device Manufacturers . . . . 6
4.1. Use of IP Address Literals in-protocol . . . . . 6
4.2. Use of Non-deterministic DNS Names in-protocol . . . . . 8
4.3. Use of a Too Generic DNS Name . . . . . 9
5. DNS Privacy and Outsourcing versus MUD Controllers . . . . . 9
6. Recommendations To IoT Device Manufacturers on MUD and DNS Usage . . . . . 10
6.1. Consistently use DNS . . . . . 10
6.2. Use Primary DNS Names Controlled By The Manufacturer . . 10
6.3. Use Content-Distribution Network with Stable Names . . . 10
6.4. Do Not Use Tailored Responses to answer DNS Names . . . . 11
6.5. Prefer DNS Servers Learnt From DHCP/Route Advertisements . . . . . 11
7. Privacy Considerations . . . . . 12
8. Security Considerations . . . . . 13
9. References . . . . . 13
9.1. Normative References . . . . . 13
9.2. Informative References . . . . . 14
Appendix A. A Failing Strategy --- Anti-Patterns . . . . . 16
A.1. Too Slow . . . . . 16
A.2. Reveals Patterns of Usage . . . . . 16

A.3. Mappings Are Often Incomplete . . . . . 17
A.4. Forward Names Can Have Wildcards . . . . . 17
Contributors . . . . . 18
Authors' Addresses . . . . . 18

## 1. Introduction

[RFC8520] provides a standardized way to describe how a specific purpose device makes use of Internet resources. Access Control Lists (ACLs) can be defined in an RFC 8520 Manufacturer Usage Description (MUD) file that permit a device to access Internet resources by their DNS names or IP addresses.

Use of a DNS name rather than an IP address in an ACL has many advantages: not only does the layer of indirection permit the mapping of a name to IP address(es) to be changed over time, it also generalizes automatically to IPv4 and IPv6 addresses, as well as permitting a variety of load balancing strategies, including multi-CDN deployments wherein load balancing can account for geography and load.

However, the use of DNS names has implications on how ACL are executed at the MUD policy enforcement point (typically, a firewall). Conceretely, the firewall has access only to the Layer 3 headers of the packet. This includes the source and destination IP address, and if not encrypted by IPsec, the destination UDP or TCP port number present in the transport header. The DNS name is not present!

So in order to implement these name based ACLs, there must be a mapping between the names in the ACLs and IP addresses.

In order for manufacturers to understand how to configure DNS associated with name based ACLs, a model of how the DNS resolution will be done by MUD controllers is necessary. Section 3 models some good strategies that are used.

This model is non-normative: but is included so that IoT device manufacturers can understand how the DNS will be used to resolve the names they use.

There are some ways of using DNS that will present problems for MUD controllers, which Section 4 explains.

Section 5 details how current trends in DNS resolution such as public DNS servers, DNS over TLS (DoT) [RFC7858], DNS over HTTPS (DoH) [RFC8484], or DNS over QUIC (DoQ) [RFC9250] can cause problems with the strategies employed.

The core of this document, is Section 6, which makes a series of recommendations ("best current practices") for manufacturers on how to use DNS and IP addresses with MUD supporting IoT devices.

Section 7 discusses a set of privacy issues that encrypted DNS (DoT, DoH, for example) are frequently used to deal with. How these concerns apply to IoT devices located within a residence or enterprise is a key concern.

Section 8 also covers some of the negative outcomes should MUD/ firewall managers and IoT manufacturers choose not to cooperate.

## 2. Terminology

Although this document is not an IETF Standards Track publication, it adopts the conventions for normative language to provide clarity of instructions to the implementer. 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.

This document makes use of terms defined in [RFC8520] and [I-D.ietf-dnsop-rfc8499bis].

The term "anti-pattern" comes from agile software design literature, as per [antipatterns].

## 3. A model for MUD controller mapping of names to addresses

This section details a strategy that a MUD controller could take. Within the limits of DNS use detailed in Section 6, this process can work. The methods detailed in Appendix A just will not work.

The simplest successful strategy for translating names for a MUD controller to take is to do a DNS lookup on the name (a forward lookup), and then use the resulting IP addresses to populate the actual ACLs.

There a number of possible failures, and the goal of this section is to explain how some common DNS usages may fail.

### 3.1. Non-Deterministic Mappings

The most important one is that the mapping of the names to IP addresses may be non-deterministic.

[RFC1794] describes the very common mechanism that returns DNS A (or reasonably AAAA) records in a permuted order. This is known as Round Robin DNS, and it has been used for many decades. The historical intent is that the requestor will tend to use the first IP address that is returned. As each query results in addresses in a different ordering, the effect is to split the load among many servers.

This situation does not result in failures as long as all possible A/AAAA records are returned. The MUD controller and the device get a matching set, and the ACLs that are set up cover all possibilities.

There are a number of circumstances in which the list is not exhaustive. The simplest is when the round-robin does not return all addresses. This is routinely done by geographical DNS load balancing systems: only the addresses that the balancing system wishes to be used are returned.

It can also happen if there are more addresses than will conveniently fit into a DNS reply. The reply will be marked as truncated. (If DNSSEC resolution will be done, then the entire RR must be retrieved over TCP (or using a larger EDNS(0) size) before being validated)

However, in a geographical DNS load balancing system, different answers are given based upon the locality of the system asking. There may also be further layers of round-robin indirection.

Aside from the list of records being incomplete, the list may have changed between the time that the MUD controller did the lookup and the time that the IoT device did the lookup, and this change can result in a failure for the ACL to match. If the IoT device did not use the same recursive servers as the MUD controller, then geofencing and/or truncated round-robin results could return a different, and non-overlapping set of addresses.

In order to compensate for this, the MUD controller SHOULD regularly perform DNS lookups in order to never have stale data. These lookups must be rate limited to avoid excessive load on the DNS servers, and it may be necessary to avoid local recursive resolvers. The MUD controller SHOULD incorporate its own recursive caching DNS server. Properly designed recursive servers should cache data for at least some number of minutes, up to some number of days (respecting the TTL), while the underlying DNS data can change at a higher frequency, providing different answers to different queries!

A MUD controller that is aware of which recursive DNS server the IoT device will use can instead query that server on a periodic basis. Doing so provides three advantages:

1. Any geographic load balancing will base the decision on the geolocation of the recursive DNS server, and the recursive name server will provide the same answer to the MUD controller as to the IoT device.

2. The resulting name to IP address mapping in the recursive name server will be cached, and will remain the same for the entire advertised Time-To-Live reported in the DNS query return. This also allows the MUD controller to avoid doing unnecessary queries.

3. if any addresses have been omitted in a round-robin DNS process, the cache will have the same set of addresses that were returned.

The solution of using the same caching recursive resolver as the target device is very simple when the MUD controller is located in a residential CPE device. The device is usually also the policy enforcement point for the ACLs, and a caching resolver is typically located on the same device. In addition to convenience, there is a shared fate advantage: as all three components are running on the same device, if the device is rebooted, clearing the cache, then all three components will get restarted when the device is restarted.

Where the solution is more complex is when the MUD controller is located elsewhere in an Enterprise, or remotely in a cloud such as when a Software Defined Network (SDN) is used to manage the ACLs. The DNS servers for a particular device may not be known to the MUD controller, nor the MUD controller be even permitted to make recursive queries to that server if it is known. In this case, additional installation specific mechanisms are probably needed to get the right view of the DNS.

## 4. DNS and IP Anti-Patterns for IoT Device Manufacturers

In many design fields, there are good patterns that should be emulated, and often there are patterns that should not be emulated. The latter are called anti-patterns, as per [antipatterns].

This section describes a number of things which IoT manufacturers have been observed to do in the field, each of which presents difficulties for MUD enforcement points.

### 4.1. Use of IP Address Literals in-protocol

A common pattern for a number of devices is to look for firmware updates in a two-step process. An initial query is made (often over HTTPS, sometimes with a POST, but the method is immaterial) to a vendor system that knows whether an update is required.

The current firmware model of the device is sometimes provided and then the authoritative server provides a determination if a new version is required and, if so, what version. In simpler cases, an HTTPS endpoint is queried which provides the name and URL of the most recent firmware.

The authoritative upgrade server then responds with a URL of a firmware blob that the device should download and install. Best practice is that firmware is either signed internally ([RFC9019]) so that it can be verified, or a hash of the blob is provided.

An authoritative server might be tempted to provide an IP address literal inside the protocol: there are two arguments (anti-patterns) for doing this.

The first is that it eliminates problems with firmware updates that might be caused by lack of DNS, or incompatibilities with DNS. For instance a bug that causes interoperability issues with some recursive servers would become unpatchable for devices that were forced to use that recursive resolver type.

The second reason to avoid an IP address literal in the URL is when an inhouse content-distribution system is involved that involves on-demand instances being added (or removed) from a cloud computing architecture.

But, there are more problems with use of IP address literals for the location of the firmware.

The first is that the update service server must decide whether to provide an IPv4 or an IPv6 literal. A DNS name can contain both kinds of addresses, and can also contain many different IP addresses of each kind.

The second problem is that it forces the MUD file definition to contain the exact same IP address literals. It must also contain an ACL for each address literal. DNS provides a useful indirection method that naturally aggregates the addresses.

A third problem involves the use of HTTPS. IP address literals do not provide enough context for TLS ServerNameIndicator to be useful [RFC6066]. This limits the firmware repository to be a single tenant on that IP address, and for IPv4 (at least), this is no longer a sustainable use of IP addresses.

Finally, it is common in some Content Distribution Networks (CDNs) to use multiple layers of DNS CNAMEs in order to isolate the content-owner's naming system from changes in how the distribution network is organized.

A non-deterministic name or address that is returned within the update protocol, the MUD controller is unable to know what the name is. It is therefore unable to make sure that the communication to retrieve the new firmware is permitted by the MUD enforcement point.

### 4.2. Use of Non-deterministic DNS Names in-protocol

A second pattern is for a control protocol to connect to a known HTTP endpoint. This is easily described in MUD. References within that control protocol are made to additional content at other URLs. The values of those URLs do not fit any easily described pattern and may point at arbitrary names.

Those names are often within some third-party CDN system, or may be arbitrary names in a cloud-provider storage system (e.g., [AmazonS3], or [Akamai]). Some of the name components may be specified by the provider.

Such names may be unpredictably chosen by the content provider, and not the content owner, and so impossible to insert into a MUD file.

Even if the content provider chosen names are deterministic they may change at a rate much faster than MUD files can be updated.

This in particular may apply to the location where firmware updates may be retrieved.

A solution is to use a deterministic DNS name, within the control of the firmware vendor. This may be a problem if the content distribution network needs to reorganize which IP address is responsible for which content, or if there is a desire to provide content in geographically relevant ways.

The firmware vendor is therefore likely to be asked to point a CNAME to the CDN, to a name that might look like "g7.a.example", with the expectation that the CDN vendors DNS will do all the appropriate work to geolocate the transfer. This can be fine for a MUD file, as the MUD controller, if located in the same geography as the IoT device, can follow the CNAME, and can collect the set of resulting IP addresses, along with the TTL for each. The MUD controller can then take charge of refreshing that mapping at intervals driven by the TTL.

In some cases, a complete set of geographically distributed servers is known ahead of time, and the firmware vendor can list all those addresses in the DNS for the the name that it lists in the MUD file. As long as the active set of addresses used by the CDN is a strict subset of that list, then the geolocated name can be used for the firmware download itself. This use of two addresses is ripe for confusion, however.

### 4.3. Use of a Too Generic DNS Name

Some CDNs make all customer content available at a single URL (such as "s3.example.com"). This seems to be ideal from a MUD point of view: a completely predictable URL.

The problem is that a compromised device could then connect to the contents of any bucket, potentially attacking the data from other customers.

Exactly what the risk is depends upon what the other customers are doing: it could be limited to simply causing a distributed denial-of-service attack resulting in high costs to those customers, or such an attack could potentially include writing content.

Amazon has recognized the problems associated with this practice, and aims to change it to a virtual hosting model, as per [awss3virtualhosting].

The MUD ACLs provide only for permitting end points (hostnames and ports), but do not filter URLs (nor could filtering be enforced within HTTPS).

## 5. DNS Privacy and Outsourcing versus MUD Controllers

[RFC7858] and [RFC8094] provide for DoT and DoH. [I-D.ietf-dnsop-rfc8499bis] details the terms. But, even with the unencrypted DNS (a.k.a. Do53), it is possible to outsource DNS queries to other public services, such as those operated by Google, CloudFlare, Verisign, etc.

For some users and classes of devices, revealing the DNS queries to those outside entities may constitute a privacy concern. For other users the use of an insecure local resolver may constitute a privacy concern.

As described above in Section 3 the MUD controller needs to have access to the same resolver(s) as the IoT device.

## 6. Recommendations To IoT Device Manufacturers on MUD and DNS Usage

Inclusion of a MUD file with IoT devices is operationally quite simple. It requires only a few small changes to the DHCP client code to express the MUD URL. It can even be done without code changes via the use of a QR code affixed to the packaging (see [RFC9238])

The difficult part is determining what to put into the MUD file itself. There are currently tools that help with the definition and analysis of MUD files, see [mudmaker]. The remaining difficulty is now the actual list of expected connections to put in the MUD file. An IoT manufacturer must now spend some time reviewing the network communications by their device.

This document discusses a number of challenges that occur relating to how DNS requests are made and resolved, and the goal of this section is to make recommendations on how to modify IoT systems to work well with MUD.

### 6.1. Consistently use DNS

For the reasons explained in Section 4.1, the most important recommendation is to avoid using IP address literals in any protocol. Names should always be used.

### 6.2. Use Primary DNS Names Controlled By The Manufacturer

The second recommendation is to allocate and use names within zones controlled by the manufacturer. These names can be populated with an alias (see [I-D.ietf-dnsop-rfc8499bis] section 2) that points to the production system. Ideally, a different name is used for each logical function, allowing for different rules in the MUD file to be enabled and disabled.

While it used to be costly to have a large number of aliases in a web server certificate, this is no longer the case. Wildcard certificates are also commonly available which allow for an infinite number of possible names.

### 6.3. Use Content-Distribution Network with Stable Names

When aliases point to a CDN, prefer stable names that point to appropriately load balanced targets. CDNs that employ very low time-to-live (TTL) values for DNS make it harder for the MUD controller to get the same answer as the IoT Device. A CDN that always returns the same set of A and AAAA records, but permutes them to provide the best one first provides a more reliable answer.

### 6.4. Do Not Use Tailored Responses to answer DNS Names

[RFC7871] defines the edns-client-subnet (ECS) EDNS0 option, and explains how authoritative servers sometimes answer queries differently based upon the IP address of the end system making the request. Ultimately, the decision is based upon some topological notion of closeness. This is often used to provide tailored responses to clients, providing them with a geographically advantageous answer.

When the MUD controller makes it's DNS query, it is critical that it receive an answer which is based upon the same topological decision as when the IoT device makes its query.

There are probably ways in which the MUD controller could use the edns-client-subnet option to make a query that would get the same treatment as when the IoT device makes its query. If this worked then it would receive the same answer as the IoT device.

In practice it could be quite difficult if the IoT device uses a different Internet connection, a different firewall, or a different recursive DNS server. The edns-client-server might be ignored or overridden by any of the DNS infrastructure.

Some tailored responses might only re-order the replies so that the most preferred address is first. Such a system would be acceptable if the MUD controller had a way to know that the list was complete.

But, due to the above problems, a strong recommendation is to avoid using tailored responses as part of the names in the MUD file.

### 6.5. Prefer DNS Servers Learnt From DHCP/Route Advertisements

IoT Devices SHOULD prefer doing DNS with the DHCP provided DNS servers.

The ADD WG has written [RFC9463] and [RFC9462] to provide information to end devices on how to find locally provisioned secure/private DNS servers.

Use of public resolvers instead of the provided DNS resolver, whether Do53, DoQ, DoT or DoH is discouraged. Should the network provide such a resolver for use, then there is no reason not to use it, as the network operator has clearly thought about this.

Some manufacturers would like to have a fallback to using a public resolver to mitigate against local misconfiguration. There are a number of reasons to avoid this, or at least do this very carefully.

It is recommended that use of non-local resolvers is only done when the locally provided resolvers provide no answers to any queries at all, and do so repeatedly. The use of the operator provided resolvers SHOULD be retried on a periodic basis, and once they answer, there SHOULD be no further attempts to contact public resolvers.

Finally, the list of public resolvers that might be contacted MUST be listed in the MUD file as destinations that are to be permitted! This should include the port numbers (i.e., 53, 853 for DoT, 443 for DoH) that will be used as well.

## 7. Privacy Considerations

The use of non-local DNS servers exposes the list of names resolved to a third party, including passive eavesdroppers.

The use of DoT and DoH eliminates the threat from passive eavesdropping, but still exposes the list to the operator of the DoT or DoH server. There are additional methods to help preserve privacy, such as described by [RFC9230].

The use of unencrypted (Do53) requests to a local DNS server exposes the list to any internal passive eavesdroppers, and for some situations that may be significant, particularly if unencrypted Wi-Fi is used. Use of Encrypted DNS connection to a local DNS recursive resolver is the preferred choice.

IoT devices that reach out to the manufacturer at regular intervals to check for firmware updates are informing passive eavesdroppers of the existence of a specific manufacturer’s device being present at the origin location.

Identifying the IoT device type empowers the attacker to launch targeted attacks to the IoT device (e.g., Attacker can take advantage of any known vulnerability on the device).

While possession of a Large (Kitchen) Appliance at a residence may be uninteresting to most, possession of intimate personal devices (e.g., "sex toys") may be a cause for embarrassment.

IoT device manufacturers are encouraged to find ways to anonymize their update queries. For instance, contracting out the update notification service to a third party that deals with a large variety of devices would provide a level of defense against passive eavesdropping. Other update mechanisms should be investigated, including use of DNSSEC signed TXT records with current version information. This would permit DoT or DoH to convey the update

notification in a private fashion. This is particularly powerful if a local recursive DoT server is used, which then communicates using DoT over the Internet.

The more complex case of section Section 4.1 postulates that the version number needs to be provided to an intelligent agent that can decide the correct route to do upgrades. [RFC9019] provides a wide variety of ways to accomplish the same thing without having to divulge the current version number.

The use of a publicly specified firmware update protocol would also enhance privacy of IoT devices. In such a system, the IoT device would never contact the manufacturer for version information or for firmware itself. Instead, details of how to query and where to get the firmware would be provided as a MUD extension, and an Enterprise-wide mechanism would retrieve firmware, and then distribute it internally. Aside from the bandwidth savings of downloading the firmware only once, this also makes the number of devices active confidential, and provides some evidence about which devices have been upgraded and which ones might still be vulnerable. (The unpatched devices might be lurking, powered off, lost in a closet)

While a vendor proprietary scheme to distribute firmware updates would satisfy some of these criteria, operators/Enterprises are less likely to install one of these for every single device class. Home (residential) users are unlikely to install any system that did not provide service to all their devices (and came pre-installed on a home router or other home network management system, such as a home Network Attached Storage device), so only a system that was non-proprietary is likely to be present.

# 8. Security Considerations

This document deals with conflicting Security requirements:

1. devices which an operator wants to manage using [RFC8520]

2. requirements for the devices to get access to network resources that may be critical to their continued safe operation.

This document takes the view that the two requirements do not need to be in conflict, but resolving the conflict requires careful planning on how the DNS can be safely and effectively used by MUD controllers and IoT devices.

# 9. References

## 9.1. Normative References

[I-D.ietf-dnsop-rfc8499bis]
Hoffman, P. E. and K. Fujiwara, "DNS Terminology", Work in Progress, Internet-Draft, draft-ietf-dnsop-rfc8499bis-10, 25 September 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-rfc8499bis-10>.

[RFC1794] Brisco, T., "DNS Support for Load Balancing", RFC 1794, DOI 10.17487/RFC1794, April 1995, <https://www.rfc-editor.org/info/rfc1794>.

[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>.

[RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <https://www.rfc-editor.org/info/rfc8094>.

[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>.

[RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <https://www.rfc-editor.org/info/rfc8520>.

[RFC9019] Moran, B., Tschofenig, H., Brown, D., and M. Meriac, "A Firmware Update Architecture for Internet of Things", RFC 9019, DOI 10.17487/RFC9019, April 2021, <https://www.rfc-editor.org/info/rfc9019>.

## 9.2. Informative References

[Akamai] "Akamai", 2019, <https://en.wikipedia.org/wiki/Akamai_Technologies>.

[AmazonS3] "Amazon S3", 2019, <https://en.wikipedia.org/wiki/Amazon_S3>.

[antipatterns] "AntiPattern", 12 July 2021, <https://www.agilealliance.org/glossary/antipattern>.

[awss3virtualhosting]
"Down to the Wire: AWS Delays 'Path-Style' S3 Deprecation at Last Minute", 12 July 2021, <https://techmonitor.ai/techonology/cloud/aws-s3-path-deprecation>.

[mudmaker] "Mud Maker", 2019, <https://mudmaker.org>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://www.rfc-editor.org/info/rfc6066>.

[RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <https://www.rfc-editor.org/info/rfc7858>.

[RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <https://www.rfc-editor.org/info/rfc7871>.

[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>.

[RFC9230] Kinnear, E., McManus, P., Pauly, T., Verma, T., and C.A. Wood, "Oblivious DNS over HTTPS", RFC 9230, DOI 10.17487/RFC9230, June 2022, <https://www.rfc-editor.org/info/rfc9230>.

[RFC9238] Richardson, M., Latour, J., and H. Habibi Gharakheili, "Loading Manufacturer Usage Description (MUD) URLs from QR Codes", RFC 9238, DOI 10.17487/RFC9238, May 2022, <https://www.rfc-editor.org/info/rfc9238>.

[RFC9250] Huitema, C., Dickinson, S., and A. Mankin, "DNS over Dedicated QUIC Connections", RFC 9250, DOI 10.17487/RFC9250, May 2022, <https://www.rfc-editor.org/info/rfc9250>.

[RFC9462] Pauly, T., Kinnear, E., Wood, C. A., McManus, P., and T. Jensen, "Discovery of Designated Resolvers", RFC 9462, DOI 10.17487/RFC9462, November 2023, <https://www.rfc-editor.org/info/rfc9462>.

[RFC9463] Boucadair, M., Ed., Reddy.K, T., Ed., Wing, D., Cook, N., and T. Jensen, "DHCP and Router Advertisement Options for the Discovery of Network-designated Resolvers (DNR)", RFC 9463, DOI 10.17487/RFC9463, November 2023, <https://www.rfc-editor.org/info/rfc9463>.

## Appendix A. A Failing Strategy --- Anti-Patterns

Attempts to map IP addresses to names in real time fails for a number of reasons:

1. it can not be done fast enough,

2. it reveals usage patterns of the devices,

3. the mappings are often incomplete,

4. Even if the mapping is present, due to virtual hosting, it may not map back to the name used in the ACL.

This is not a successful strategy, it MUST NOT be used for the reasons explained below.

### A.1. Too Slow

Mappings of IP addresses to names requires a DNS lookup in the in-addr.arpa or ip6.arpa space. For a cold DNS cache, this will typically require 2 to 3 NS record lookups to locate the DNS server that holds the information required. At 20 to 100 ms per round trip, this easily adds up to significant time before the packet that caused the lookup can be released.

While subsequent connections to the same site (and subsequent packets in the same flow) will not be affected if the results are cached, the effects will be felt. The ACL results can be cached for a period of time given by the TTL of the DNS results, but the DNS lookup must be repeated, e.g, in a few hours or days,when the cached IP address to name binding expires.

### A.2. Reveals Patterns of Usage

By doing the DNS lookups when the traffic occurs, then a passive attacker can see when the device is active, and may be able to derive usage patterns. They could determine when a home was occupied or not. This does not require access to all on-path data, just to the DNS requests to the bottom level of the DNS tree.

### A.3. Mappings Are Often Incomplete

A service provider that fails to include an A or AAAA record as part of their forward name publication will find that the new server is simply not used. The operational feedback for that mistake is immediate. The same is not true for reverse names: they can often be incomplete or incorrect for months or even years without visible effect on operations.

Service providers often find it difficult to update reverse maps in a timely fashion, assuming that they can do it at all. Many cloud based solutions dynamically assign IP addresses to services, often as the service grows and shrinks, reassigning those IP addresses to other services quickly. The use of HTTP 1.1 Virtual Hosting may allow addresses and entire front-end systems to be re-used dynamically without even reassigning the IP addresses.

In some cases there are multiple layers of CNAME between the original name and the target service name. This is often due to a load balancing layer in the DNS, followed by a load balancing layer at the HTTP level.

The reverse name for the IP address of the load balancer usually does not change. If hundreds of web services are funneled through the load balancer, it would require hundreds of PTR records to be deployed. This would easily exceed the UDP/DNS and EDNS0 limits, and require all queries to use TCP, which would further slow down loading of the records.

The enumeration of all services/sites that have been at that load balancer might also constitute a security concern. To limit churn of DNS PTR records, and reduce failures of the MUD ACLs, operators would want to add all possible names for each reverse name, whether or not the DNS load balancing in the forward DNS space lists that end-point at that moment.

### A.4. Forward Names Can Have Wildcards

In some large hosting providers content is hosted through a domain name that is published as a DNS wildcard (and uses a wildcard certificate). For instance, github.io, which is used for hosted content, including the Editors' copy of internet drafts stored on github, does not actually publish any names. Instead, a wildcard exists to answer all potential names: requests are routed appropriate once they are received.

This kind of system works well for self-managed hosted content. However, while it is possible to insert up to a few dozen PTR records, many thousand entries are not possible, nor is it possible to deal with the unlimited (infinite) number of possibilities that a wildcard supports.

It would be therefore impossible for the PTR reverse lookup to ever work with these wildcard names.

## Contributors

Tirumaleswar Reddy
Nokia

## Authors' Addresses

Michael Richardson
Sandelman Software Works
Email: mcr+ietf@sandelman.ca

Wei Pan
Huawei Technologies
Email: william.panwei@huawei.com