Stream: Internet Engineering Task Force (IETF)
RFC: 9726
BCP: 241
Category: Best Current Practice
Published: March 2025
ISSN: 2070-1721
Authors: M. Richardson, *Sandelman Software Works*
W. Pan, *Huawei Technologies*

RFC 9726
Operational Considerations for Use of DNS in Internet of Things (IoT) Devices

Abstract

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

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

Status of This Memo

This memo documents an Internet Best Current Practice.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on BCPs is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9726.

Copyright Notice

Table of Contents

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 a Manufacturer Usage Description (MUD) file [RFC8520] that permits a device to access Internet resources by their DNS names or IP addresses.

The 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 addresses to be changed over time, but it also generalizes automatically to IPv4 and IPv6 addresses as well as permits 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 ACLs are executed at the MUD policy enforcement point (typically, a firewall). Concretely, the firewall has access only to the Layer 3 headers of the packet. This includes the source and destination IP addresses 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 could be 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 IoT devices described by MUD.

Section 8 discusses a set of privacy issues that encrypted DNS (for example, DoT and DoH) are frequently used to deal with. How these concerns apply to IoT devices located within a residence or enterprise is a key concern.

Section 9 also covers some of the negative outcomes should MUD/firewall managers and IoT manufacturers choose not to cooperate.

2. Terminology

This document makes use of terms defined in [RFC8520] and [RFC9499].

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

"CDNs" refers to Content Distribution Networks, such as those described in [RFC6707], Section 1.1.

3. A Model for MUD Controller Mapping of DNS Names to Addresses

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

The simplest successful strategy for a MUD controller to translate DNS names 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 are 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

Most importantly, the mapping of the DNS names to IP addresses could 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 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 being in a different order, 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 DNS 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.

Failure can also occur if there are more addresses than what 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 tailored DNS replies 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 performs regular 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. A MUD controller that incorporates its own recursive caching DNS client will be able to observe the TTL on the entries and cause them to expire appropriately. This cache will last for at least some number of minutes and 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 mapping (of name to IP address) in the recursive name server will be cached and will remain the same for the entire advertised TTL 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 Customer Premises Equipment (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 (which clears the cache), then all three components will get restarted when the device is restarted.

The solution is more complex and sometimes more fragile 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, and the MUD controller may not even be 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.

A critical failure can occur when the device makes a new DNS request and receives a new set of IP addresses, but the MUD controller's copy of the addresses has not yet reached their TTL. In that case, the MUD controller still has the old addresses implemented in the ACLs, but the IoT device has a new address not previously returned to the MUD controller. This can result in a connectivity failure.

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 [antipattern].

This section describes a number of things that 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

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 vendor's 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 either firmware is 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. An argument for doing this is that it eliminates problems with firmware updates that might be caused by a lack of DNS or by 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.

But, there are several problems with the 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, assuming that only one URL can be provided. A DNS name can contain both kinds of addresses and can also contain many different IP addresses of each kind. An update server might believe that if the connection were on IPv4, then an IPv4 literal would be acceptable. However, due to NAT64 [RFC6146], a device with only IPv6 connectivity will often be able to reach an IPv4 firmware update server by name (through DNS64 [RFC6147]) but not be able to reach an arbitrary IPv4 address.

A MUD file for this access would need to resolve to the set of IP addresses that might be returned by the update server. This can be done with IP address literals in the MUD file, but this may require continuing updates to the MUD file if the addresses change frequently. A DNS name in the MUD could resolve to the set of all possible IPv4 and IPv6 addresses that would be used, with DNS providing a level of indirection that obviates the need to update the MUD file itself.

A third problem involves the use of HTTPS. It is often more difficult to get TLS certificates for an IP address, and so it is less likely that the firmware download will be protected by TLS. An IP address literal in the TLS ServerNameIndicator [RFC6066] might not provide enough context for a web server to distinguish which of the (potentially many) tenants the client wishes to reach. This drives the use of an IP address per tenant, and for IPv4 (at least), this is no longer a sustainable use of IP addresses.

Finally, it is common in some 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.

When a name or address is returned within an update protocol for which a MUD rule cannot be written, then the MUD controller is unable to authorize the connection. In order for the connection to be authorized, the set of names returned within the update protocol needs to be known ahead of time and must be from a finite set of possibilities. Such a set of names or addresses can be placed into the MUD file as an ACL in advance, and the connections can be authorized.

4.2. Use of Non-Deterministic DNS Names in Protocols

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 to arbitrary DNS names.

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

Such DNS names may be unpredictably chosen by the CDN and not the device manufacturer and therefore impossible to insert into a MUD file. Implementation of the CDN system may also involve HTTP redirections to downstream CDN systems.

Even if the CDN provider's chosen DNS names are deterministic, they may change at a rate much faster than MUD files can be updated.

This situation applies to firmware updates but also applies to many other kinds of content: video content, in-game content, etc.

A solution may be to use a deterministic DNS name within the control of the device manufacturer. The device manufacturer is asked to point a CNAME to the CDN, to a name that might look like "g7.a.example", with the expectation that the CDN provider's 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 collect the set of resulting IP addresses along with the TTL for each. Then, the MUD controller can take charge of refreshing that mapping at intervals driven by the TTL.

In some cases, a complete set of geographically distributed servers may be known ahead of time (or that it changes very slowly), and the device manufacturer can list all those IP addresses in the DNS for 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 content download itself.

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 endpoints (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. [RFC9499] 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 in Section 3, the MUD controller needs to have access to the same resolver or resolvers as the IoT device. If the IoT device does not use the DNS servers provided to it via DHCP or Router Advertisements, then the MUD controller will need to be told which servers will in fact be used. As yet, there is no protocol to do this, but future work could provide this as an extension to MUD.

Until such time as such a protocol exists, the best practice is for the IoT device to always use the DNS servers provided by DHCP or Router Advertisements.

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 the actual list of expected connections to put in the MUD file. An IoT manufacturer must 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. DNS names should always be used.

6.2. Use Primary DNS Names Controlled by the Manufacturer

The second recommendation is to allocate and use DNS names within zones controlled by the manufacturer. These DNS names can be populated with an alias (see [RFC9499], Section 2) that points to the production system. Ideally, a different name is used for each logical function, allowing 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; they allow for an infinite number of possible DNS names.

6.3. Use a Content Distribution Network with Stable DNS Names

When aliases point to a CDN, give preference to stable DNS names that point to appropriately load-balanced targets. CDNs that employ very low 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 its DNS query, it is critical that it receives an answer that 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 reorder 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 DNS names in the MUD file.

6.5. Prefer DNS Servers Learned from DHCP/Router Advertisements

The best practice is for IoT devices to do DNS with the DHCP-provided DNS servers or with DNS servers learned from Router Advertisements [RFC8106].

The Adaptive DNS Discovery (ADD) Working Group has written [RFC9462] and [RFC9463] to provide information to end devices on how to find locally provisioned secure/private DNS servers.

Use of public resolvers instead of the locally provided DNS resolver, whether Do53, DoQ, DoT, or DoH, is discouraged.

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, detailed in Section 6.4. The public resolver might not return the same tailored names that the MUD controller would get.

It is recommended that non-local resolvers are only used when the locally provided resolvers provide no answers to any queries at all and do so repeatedly. The status of the operator-provided resolvers needs to be re-evaluated on a periodic basis.

Finally, if a device will ever attempt to use non-local resolvers, then the addresses of those resolvers need to be listed in the MUD file as destinations that are to be permitted. This needs to include the port numbers (i.e., 53, 853 for DoT, 443 for DoH) that will be used as well.

7. Interactions with mDNS and DNS-SD

Unicast DNS requests are not the only way to map names to IP addresses. IoT devices might also use Multicast DNS (mDNS) [RFC6762], both to be discovered by other devices and also to discover other devices.

mDNS replies include A and AAAA records, and it is conceivable that these replies contain addresses that are not local to the link on which they are made. This could be the result of another device that contains malware. An unsuspecting IoT device could be led to contact some external host as a result. Protecting against such things is one of the benefits of MUD.

In the unlikely case that the external host has been listed as a legitimate destination in a MUD file, communication will continue as expected. As an example, an IoT device might look for a name like "update.local" in order to find a source of firmware updates. It could be led to connect to some external host that was listed as "update.example" in the MUD file. This should work fine if the name "update.example" does not require any kind of tailored reply.

In residential networks, there has typically not been more than one network (although this is changing through work like [AUTO-STUB-NETWORKS]), but on campus or enterprise networks, having more than one network is not unusual. In such networks, mDNS is being replaced with DNS-based Service Discovery (DNS-SD) [RFC8882], and in such a situation, connections could be initiated to other parts of the network. Such connections might traverse the MUD policy enforcement point (an intra-department firewall) and could very well be rejected because the MUD controller did not know about that interaction.

[RFC8250] includes a number of provisions for controlling internal communications, including complex communications like same manufacturer ACLs. To date, this aspect of MUD has been difficult to describe. This document does not consider internal communications to be in scope.

8. Privacy Considerations

The use of non-local DNS servers exposes the list of DNS 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 that described by [RFC9230].

The use of unencrypted (Do53) requests to a local DNS server exposes the list to any internal passive eavesdroppers. For some situations, that may be significant, particularly if unencrypted WiFi is used.

Use of an 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., the 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 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.

9. Security Considerations

This document deals with conflicting security requirements:

- devices that an operator wants to manage using [RFC8520]
- 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 be used by MUD controllers and IoT devices.

When an IoT device with an inaccurate MUD file is deployed into a network that uses MUD, there is a significant possibility that the device will cause a spurious security exception to be raised. There is significant evidence that such spurious exceptions can cause significant

overhead to personnel. In particular, repeated spurious exceptions are likely to cause the entire exception process to be turned off. When MUD alerts are turned off, then even legitimate exceptions are ignored. This is very much a Boy Who Calls Wolf [boywhocriedwolf] situation.

In order to avoid this situation, and for MUD alerts to be given appropriate attention, it is key that IoT device manufacturers create accurate MUD files. This may require some significant thought and even rework of key systems so that all network access required by the IoT device can be described by a MUD file. This level of informed cooperation within the IoT device vendor and with MUD controller manufacturers is key to getting significant return on investment from MUD.

Manufacturers are encouraged to write MUD files that are good enough rather than perfect. If in doubt, they should write MUD files that are somewhat more permissive if the files result in no spurious alerts.

10. References

10.1. Normative References

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

[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>.

[RFC8250] Elkins, N., Hamilton, R., and M. Ackermann, "IPv6 Performance and Diagnostic Metrics (PDM) Destination Option", RFC 8250, DOI 10.17487/RFC8250, September 2017, <https://www.rfc-editor.org/info/rfc8250>.

[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>.

[RFC9499] Hoffman, P. and K. Fujiwara, "DNS Terminology", BCP 219, RFC 9499, DOI 10.17487/RFC9499, March 2024, <https://www.rfc-editor.org/info/rfc9499>.

10.2. Informative References

[Akamai] Wikipedia, "Akamai Technologies", 26 February 2025, <https://en.wikipedia.org/w/index.php?title=Akamai_Technologies&oldid=1277665363>.

[AmazonS3] Wikipedia, "Amazon S3", 14 March 2025, <https://en.wikipedia.org/w/index.php?title=Amazon_S3&oldid=1280379498>.

[antipattern] Agile Alliance, "AntiPattern", <https://www.agilealliance.org/glossary/antipattern>.

[AUTO-STUB-NETWORKS] Lemon, T. and J. Hui, "Automatically Connecting Stub Networks to Unmanaged Infrastructure", Work in Progress, Internet-Draft, draft-ietf-snac-simple-06, 4 November 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-snac-simple-06>.

[awss3virtualhosting] Tech Monitor, "Down to the Wire: AWS Delays 'Path-Style' S3 Deprecation at Last Minute", 24 September 2020, <https://techmonitor.ai/techonology/cloud/aws-s3-path-deprecation>.

[boywhocriedwolf] Wikipedia, "The Boy Who Cried Wolf", 6 February 2025, <https://en.wikipedia.org/w/index.php?title=The_Boy_Who_Cried_Wolf&oldid=1274257821>.

[mudmaker] "MUD Maker", <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>.

[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>.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, DOI 10.17487/RFC6147, April 2011, <https://www.rfc-editor.org/info/rfc6147>.

[RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", RFC 6707, DOI 10.17487/RFC6707, September 2012, <https://www.rfc-editor.org/info/rfc6707>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[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>.

[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>.

[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>.

[RFC8882] Huitema, C. and D. Kaiser, "DNS-Based Service Discovery (DNS-SD) Privacy and Security Requirements", RFC 8882, DOI 10.17487/RFC8882, September 2020, <https://www.rfc-editor.org/info/rfc8882>.

[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 DNS names in real time often fail 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 for the reasons explained below.

A.1. Too Slow

Mappings of IP addresses to DNS names require 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 a significant amount of 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 binding (of IP address to name) 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

An IoT manufacturer with a cloud 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 DNS mappings: They can often be incomplete or incorrect for months or even years without a visible effect on operations.

IoT manufacturer cloud service providers often find it difficult to update reverse DNS 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 reused 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 DNS mapping 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 the churn of DNS PTR records and reduce failures of the MUD ACLs, operators would want to add all possible DNS names for each reverse DNS mapping, whether or not the DNS load-balancing in the forward DNS space lists that endpoint at that moment.

A.4. Forward DNS 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 hosting content, including the Editors' copy of Internet-Drafts stored on GitHub, does not actually publish any DNS names. Instead, a wildcard exists to answer all potential DNS names: Requests are routed appropriately 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 thousands of entries are not possible, nor is it possible to deal with the unlimited (infinite) number of possibilities that a wildcard supports.

Therefore, it would be impossible for the PTR reverse lookup to ever work with these wildcard DNS names.

Contributors

Tirumaleswar Reddy.K
Nokia

Authors' Addresses

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

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