Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 4, 2017

T. Lemon
Nominum, Inc.

October 31, 2016

Stateful Multi-Link DNS Service Discovery
draft-lemon-stateful-dnssd-00

Abstract

This document proposes a stateful model for automating Multi-Link DNS Service Discovery, as an extension to the existing solution, which relies entirely on multicast DNS for discovering services, and does not formally maintain DNS zone state. When fully deployed this will confer several advantages: the ability to do DNS zone transfers, integrating with existing DNS infrastructure; the elimination of the need for regular multicast queries; and the ability for services to securely register and defend their names, preventing malicious spoofing of services on the network.

Table of Contents

1. Introduction 2
2. Terminology 3
3. Overview 3
4. Service Behavior 4
4.1. Detecting Stateful ML-DNSSD 4
4.2. Publishing Services when Stateful ML-DNSSD is present . . 4
4.3. Maintenance 5
5. Discovery 6
6. DNS Service Infrastructure 6
7. Legacy Service Discovery 6
8. Security Considerations 6
9. Normative References 7
Author's Address 7

1. Introduction

This document describes a way of doing DNS service discovery using DNS updates [RFC2136] rather than Multicast DNS[RFC6762]. Update validation is done on the same basis as Multicast DNS validation: the assumption is that a device connected to a local link is permitted to advertise services. However, in contrast to mDNS, which provides no mechanism for defending claims made by services, we propose that services should publish keys when initially registering names, and use SIG(0) authentication [RFC2931] when issuing DNS updates, using the published key.

Advantages of this proposal over the Multicast DNS Hybrid proposal [I-D.ietf-dnssd-hybrid] are:

- Service advertisement does not require multicast.
- Names are stored in DNS zone databases, and therefore can be published using standard DNS protocol features such as zone transfers.
- Names can be defended by services that register them, so that it is difficult for an attacker to spoof an existing service.

There are, however, disadvantages to this approach. The first disadvantage is that this proposal does not actually eliminate multicast except in the case that all local services implement the

new update mechanism. Because this approach maintains state, and that state must include existing services that only support advertising via Multicast DNS, additional complexity is required to avoid retaining stale information; this complexity is not required for the stateless model proposed in the mDNS hybrid specification.

Another disadvantage of this approach is that it requires a stable naming infrastructure, and requires forwarders on each local link.

Some sites may find it preferable to rely on the stateless model for this reason. However, the stateful model provides sufficient advantages that it will make sense for some sites to implement it, even in the legacy mode that still supports service discovery using Multicast DNS.

2. Terminology

For the sake of brevity this document uses a number of abbreviations. These are expanded here:

mDNS Multicast DNS

ML-DNSSD Multi-Link DNS Service Discovery

3. Overview

Stateful Multi-Link DNS service discovery attempts to provide stateful service that is otherwise equivalent to Hybrid Unicast/ Multicast DNS-Based Service Discovery, except that where possible multicast is avoided, and DNS zones are maintained such that full interoperation with the DNS is possible.

In order to accomplish this, service providers detect whether the local network supports stateful operation. If not, they simply provide service using mDNS as before. If so, they advertise services solely using DNS updates.

The DNS infrastructure is prepared to take DNS updates from devices on served networks; each unique link has a DNS forwarder that can detect that a packet originated locally and was not forwarded; this serves as validation that the service can be advertised.

Legacy services are supported using the same query process used in the hybrid model. Unlike with the hybrid model, however, discovered services are added to DNS zones.

As with the Hybrid model, services are discovered using unicast DNS. Multicast DNS service discovery is not usable on networks offering stateful multi-link DNS service discovery.

4. Service Behavior

Hosts offering services using DNS service discovery must advertise these services. When a host offering services is connected to a network that does not offer stateful ML-DNSSD, it offers service discovery using Multicast DNS. When stateful ML-DNSSD is offered, the host does not offer service discovery using Multicast DNS.

4.1. Detecting Stateful ML-DNSSD

In order to detect the presence of stateful ML-DNSSD, the host first performs registration domain discovery as in section 11 of [RFC6763] to acquire the name of the recommended default domain for registering services. If this process fails, Stateful ML-DNSSD is not present. If the process succeeds, the host looks for a PTR record using the well-known name "_mldnssd.<domain>". If a PTR record is present, stateful ML-DNSSD is present.

Whenever a host detects a change to the link, a change to the IP addresses of the DNS resolvers provided on the link, or a change to the set of prefixes available on the link, the host re-tries the ML-DNSSD detection process.

4.2. Publishing Services when Stateful ML-DNSSD is present

When stateful ML-DNSSD is present, a host adds its own information into the DNS. This information is added into three separate domains, as described in [I-D.ietf-dnssd-hybrid], section 4. The subdomain for services and the subdomain for names are a single subdomain, the recommended default domain for registering services. The IPv4 and IPv6 reverse-mapping zones are discovered by querying the well-known names "_inaddr_zone.<domain>" and "_ipv6_zone.<domain>" for PTR records. Each PTR record points to a specific zone to which updates are sent for IPv4 and IPv6 PTR records.

When a host that offers service first starts, it generates a key that is used to authenticate its DNS updates. This key is included whenever updating the service’s name.

There are four domain names that are updated when a service advertises itself: the human-readable name, the machine-readable name, the service entry or entries, and the resverse-mapping pointers. The update proceeds by first adding or updating the machine-readable name, then adding a PTR record from the human-

readable name to the machine-readable name, then adding the reverse-mapping pointers, then adding the service. All updates are signed using SIG(0), authenticated with the private half of the host's key.

To add the machine-readable name, the host creates a DNS update that adds its name. The update is predicated on the nonexistence of the name. The update includes A and AAAA records for all of the hosts IP addresses except its link-local addresses. If this update succeeds, the machine-readable name has been added.

The update can fail for one of two reasons: either the signature was invalid, or the name already exists. In the former case, there is a bug, and the host should revert to providing service using mDNS.

Otherwise, if the update fails, the name already exists. The host creates a new update that deletes any A and AAAA RRsets and adds A and AAAA as before. There is no predicate for this update because the server should reject it if the name belongs to some other host (that is, has a different key). If this update fails, the host chooses a new machine-readable name and restarts the process.

The host then creates a PTR record under "Human Readable Name.<domain>" pointing to the machine-readable name. If this fails, the host must choose a different name and attempt to add it, until successful.

The host now creates updates for the reverse-mapping name of every IPv4 address it has that is not a link-local address, and adds a PTR record for each, pointing back at the machine-readable name. These adds should not fail. The process is repeated for every IPv6 address that is not link-local.

Finally, the host updates the well-known name for its service or services, adding an entry for each one. These names may already have SRV RRtypes, so this update must add records.

TODO: consider whether this is really the right way to do this--it's really complicated, and might be better done as a single HTTP request.

4.3. Maintenance

Whenever the host adds its service to the DNS, it queries the machine-readable name to see what the TTL is. When 80% of that TTL has expired, the host refreshes all of its records. This prevents the records from being cleaned up by the DNS server as stale.

If the host is being shut down cleanly, it may remove all names and SRV records that it has added, or may remove all SRV records, leaving everything else intact in order to reserve the name. In most cases, it is better to leave the name.

5. Discovery

Service Discovery is done as per RFC 6763. Service discovery defaults to '.local', which is resolved using mDNS. If ML-DNSSD is present in any form, hosts doing service discovery should successfully discover this following the method described in RFC 6763. The service appears to the host doing service discovery the same way whether the hybrid model or the stateful model is being used. Hosts do not do mDNS if ML-DNSSD is present.

In order to support progressive queries in situations where legacy service discovery is in operation, hosts should use DNS push [I-D.ietf-dnssd-push].

6. DNS Service Infrastructure

Updates are sent to a forwarder on the local link. The forwarder uses neighbor discovery or ARP to validate each of the IP addresses presented in an A or AAAA record. Updates that do not come from local hosts are silently discarded. Other updates are forwarded to the primary name server without changes.

The primary server validates all updates by using the key stored on the machine-readable name to which the update points. If the update is an update of the machine-readable name, the update is validated based on the key stored at that name, if any, or else using the key contained in the update.

Any number of secondaries may be configured. Secondaries may also serve as forwarders if appropriate.

7. Legacy Service Discovery

Service discovery done as in mdns-hybrid, except that state is retained. State is periodically probed; stale state is discarded. Discovery service listens for initial service announcements.

8. Security Considerations

Any host on a network on which service discovery is supported can advertise services, which might be spoofed so as to capture private information. One solution to this is to only accept updates from designated infrastructure networks, so that networks to which regular

users connect are not permitted to advertise services. This will, however, limit the usefulness of various services which may be present on user devices.

It may be possible to only allow anonymous pairing [I-D.ietf-dnssd-pairing] on public-facing networks, so that infrastructure services cannot be advertised, but users can still rendezvous.

9. Normative References

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

[RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <http://www.rfc-editor.org/info/rfc2931>.

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

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

[I-D.ietf-dnssd-pairing] Huitema, C. and D. Kaiser, "Device Pairing Using Short Authentication Strings", draft-ietf-dnssd-pairing-00 (work in progress), October 2016.

[I-D.ietf-dnssd-hybrid] Cheshire, S., "Hybrid Unicast/Multicast DNS-Based Service Discovery", draft-ietf-dnssd-hybrid-03 (work in progress), February 2016.

[I-D.ietf-dnssd-push] Pusateri, T. and S. Cheshire, "DNS Push Notifications", draft-ietf-dnssd-push-08 (work in progress), July 2016.

Author’s Address

Ted Lemon
Nominum, Inc.
800 Bridge Parkway
Redwood City, California 94065
United States of America

Phone: +1 650 381 6000
Email: ted.lemon@nominum.com