Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 14, 2018

C. Huitema
Private Octopus Inc.

February 10, 2018

Privacy Extensions for DNS-SD
draft-huitema-dnssd-privacyscaling-00

Abstract

DNS-SD (DNS Service Discovery) normally discloses information about both the devices offering services and the devices requesting services. This information includes host names, network parameters, and possibly a further description of the corresponding service instance. Especially when mobile devices engage in DNS Service Discovery over Multicast DNS at a public hotspot, a serious privacy problem arises.

The draft currently progressing in the DNSSD Working Group assumes peer-to-peer pairing between the service to be discovered and each of its client. This has good security properties, but create scaling issues. Each server needs to publish as many announcements as it has paired clients. Each client needs to process all announcements from all servers present in the network. This leads to large number of operations when each server is paired with many clients.

Different designs are possible. For example, if there was only one server "discovery key" known by each authorized client, each server would only have to announce a single record, and clients would only have to process one response for each server that is present on the network. Yet, these designs will present different privacy profiles, and pose different management challenges. This draft analyses the tradeoffs between privacy and scaling in a set of different designs, using either shared secrets or public keys.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 14, 2018.

Table of Contents

1. Introduction

DNS-SD [RFC6763] over mDNS [RFC6762] enables configurationless service discovery in local networks. It is very convenient for users, but it requires the public exposure of the offering and requesting identities along with information about the offered and requested services. Parts of the published information can seriously breach the user's privacy. These privacy issues and potential solutions are discussed in [KW14a] and [KW14b].

A recent draft [I-D.ietf-dnssd-privacy] proposes to solve this problem by relying on device pairing. Only clients that have paired with a device would be able to discover that device, and the discovery would not be observable by third parties. This design has a number of good privacy and security properties, but it has a cost, because each server must provide separate annoucements for each clients. In this draft, we compare scaling and privacy properties of three different designs:

- The individual pairing defined in [I-D.ietf-dnssd-privacy],
- A single server discovery secret, shared by all authorized clients,
- A single server discovery public key, known by all authorized clients.

After presenting briefly these three solutions, the draft presents the scaling and privacy properties of each of them.

2. Privacy and Secrets

Private discovery tries to ensure that clients and servers can discover eachother in a potentially hostile network context, while maintaining privacy. Unauthorized third parties must not be able to discover that a specific server or device is currently present on the network, and they must not be able to discover that a particular client is trying to discover a particular service. This cannot be achieved without some kind of shared secret between client and servers. We review here three particular design for sharing these secrets.

2.1. Pairing secrets

The solution proposed in [I-D.ietf-dnssd-privacy] relies on pairing secrets. Each client obtains a pairing secret from each server that they are authorized to use. The servers publish announcements of the form "nonce|proof", in which the proof is the hash of the nonce and the pairing secret. The proof is of course different for each client, because the secrets are different. For better scalling, the nonce is common to all clients, and defined as a coarse function of time, such as the current 30 minutes interval.

Clients discover the required server by issuing queries containing the current nonce and proof. Servers respond to these queries if the nonce matches the current time interval, and if the proof matches the hash of the nonce with one of the pairing key of an authorized client.

2.2. Discovery secret

Instead of using a different secret for each client as in Section 2.1, another design is to have a single secret per server, shared by all authorized clients of that server. As in the previous solution, the servers publish announcements of the form "nonce|proof", but this time they only need to publish a single announcement per server, because each server maintains a single discovery secret. Again, the nonce can be common to all clients, and defined as a coarse function of time.

Clients discover the required server by issuing queries containing the current nonce and proof. Servers respond to these queries if the nonce matches the current time interval, and if the proof matches the hash of the nonce with one of the discovery secret.

2.3. Discovery public key

Instead of a discovery secret used in Section 2.2, clients could obtain the public keys of the servers that they are authorized to use.

Many public key systems assume that the public key of the server is, well, not secret. But if adversaries know the public key of a server, they can use that public key as a unique identifier to track the server. Moreover, they could use variations of the padding oracle to observe discovery protocol messages and attribute them to a specific public key, thus breaking server privacy. For these reasons, we assume here that the discovery public key is kept secret, only known to authorized clients.

As in the previous solution, the servers publish announcements of the form "nonce|proof", but this time they only need to publish a single announcement per server, because each server maintains a single discovery secret. The proof is obtained by either hashing the nonce with the public key, or using the public key to encrypt the nonce -- the point being that both clients and server can contruct the proof. Again, the nonce can be common to all clients, and defined as a coarse function of time.

The advantage of public key based solutions is that the clients can easily verify the identity of the server, for example if the service is accessed over TLS. On the other hand, just using standard TLS would disclose the certificate of the server to any client that attempts a connection, not just to authorized clients. The server should thus only accept connections from clients that demonstrate knowledge of its public key.

3. Scaling properties of different solutions

To analyze scaling issues we will use the following variables:

N: The average number of authorized clients per server.

M: The average number of servers per client.

P: The average total number of servers present during discovery.

The big difference between the three proposals is the number of records that need to be published by a server when using DNS-SD in server mode, or the number of broadcast messages that needs to be announced per server in MDNS mode:

Pairing secrets: O(N). One record per client.

Discovery secrets: O(1). One record for all clients.

Discovery public key: O(1). One record for all clients.

There are other elements of scaling, linked to the mapping of the privacy discovery service to DNSSD. DNSSD identifies services by a combination of a service type and an instance name. In classic mapping behavior, clients send a query for a service type, and will receive responses from each server instance supporting that type:

Pairing secrets: O(P*N). There are O(P) servers present, and each publishes O(N) instances.

Discovery secrets: O(P). One record per server present.

Discovery public key: O(P). One record per server present.

The DNSSD Privacy draft suggests an optimization that considerably reduces the considerations about scaling of responses -- see section 4.6 of [I-D.ietf-dnssd-privacy]. In that case, clients compose the list of instance names that they are looking for, and specifically query for these instance names:

Pairing secrets: O(M). The client will compose O(M) queries to discover all the servers that it is interested in. There will be at most O(M) responses.

Discovery secrets: O(M). Same behavior as in the pairing secret case.

Discovery public key: O(M). Same behavior as in the pairing secret case.

Finally, another element of scaling is cacheability. Responses to DNS queries can be cached by DNS resolvers, and MDNS responses can be cached by MDNS resolvers. If several clients send the same queries, and if previous responses could be cached, the client can be served immediately. There are of course differences between the solutions:

Pairing secrets: No caching possible, since there are separate server instances for separate clients.

Discovery secrets: Caching is possible, since there is just one server instance.

Discovery public key: Caching is possible, since there is just one server instance.

4. Comparing privacy posture of different solutions

The analysis of scaling issues in Section 3 shows that the solutions base on a common discovery secret or discovery public key scale much better than the solutions based on pairing secret. All these solutions protect against tracking of clients or servers by third parties, as long as the secret on which they rely are kept secret. There are however significant differences in privacy properties, which become visible when one of the clients becomes compromised.

4.1. Effects of compromized client

If a client is compromised, an adversary will take possession of the secrets owned by that client. The effects will be the following:

Pairing secrets: With a valid pairing key, the adversary can issue queries or parse annoucements. It will be able to track the presence of all the servers to which the compromised client was paired. It may be able to track other clients of these servers if it can infer that multiple independent instances are tied to the same server, for example by assessing the IP address associated with a specific instance. It will not be able to impersonate the servers for other clients.

Discovery secrets: With a valid discovery secret, the adversary can issue queries or parse annoucements. It will be able to track the presence of all the servers that the compromised client could discover. It will also be able to detect the clients that try to use one of these servers. This will not reveal the identity of the client, but it can provide clues for network analysis. The

adversary will also be able to spoof the server's announcements, which could be the first step in a serve impersonation attack.

Discovery public key: With a valid discovery public key, the adversary can issue queries or parse annoucements. It will be able to track the presence of all the servers that the compromised client could discover. It will also be able to detect the clients that try to use one of these servers. This will not reveal the identity of the client, but it can provide clues for network analysis. The adversary will not be able to spoof the server's announcements, or to impersonate the server.

4.2. Remediation of compromized client

Let's assume that an administrator discovers that a client has been compromised. As seen in Section 4.1, compromising a client entails a loss of privacy for all the servers that the client was authorized to use, and also to all other users of these servers. The worse situation happens in the solutions based on "discovery secrets", but no solution provides a great defense. The administrator will have to remedy the problem, which means different actions based on the different solutions:

Pairing secrets: The administrator will need to revoke the pairing keys used by the compromised client. This implies contacting the O(M) servers to which the client was paired.

Discovery secrets: The administrator will need to revoke the discovery secrets used by the compromised client. This implies contacting the O(M) servers that the client was authorized to discover, and then the O(N) clients of each of these servers. This will require a total of O(N*M) management operations.

Discovery public key: The administrator will need to revoke the discovery public keys used by the compromised client. This implies contacting the O(M) servers that the client was authorized to discover, and then the O(N) clients of each of these servers. Just as in the case of discovery secrets, this will require O(N*M) management operations.

The revocation of public keys might benefit from some kind of centralized revocation list, and thus may actually be easier to organize than simple scaling considerations would dictate.

4.3. Effect of compromized server

If a server is compromised, an adversary will take possession of the secrets owned by that server. The effects are pretty much the same in all configurations. With a set of valid credentials, the adversary can impersonate the server. It can track all of the server's clients. There are no differences between the various solutions.

As remedy, once the compromise is discovered, the administrator will have to revoke the credentials of O(N) clients connected to that server. In all cases, this could be done by notifying all potential clients to not trust this particular server anymore.

5. Summary of tradeoffs

In the preceeding sections, we have reviewed the scaling and privacy properties of three possible secret sharing solutions for privacy discovery. The comparison can be summed up as follow:

Solution	Scaling	Resistance	Remediation
Pairing secret	Poor	Bad	Good
Discovery secret	Good	Really bad	Poor
Discovery public key	Good	Bad	Maybe

Table 1: Comparison of secret sharing solutions

All three types of solutions provide reasonable privacy when the secrets are not compromized. They all have poor resistance to the compromise of one a client, as explained in Section 4.1, but pairing secret and public key solution have the advantage of preventing server impersonation. The pairing secret solution scales worse than the discovery secret and discovery public key solutions. The pairing secret solution can recover from a compromise with a smaller number of updates, but the public key solution may benefit from a simple recovery solution using some form of "revocation list".

6. Security Considerations

This document does not specify a solution, but inform future choices when providing privacy for discovery protocols.

7. IANA Considerations

This draft does not require any IANA action.

8. Acknowledgments

This draft results from initial feedback in the DNS SD working group on [I-D.ietf-dnssd-privacy].

9. Informative References

[I-D.ietf-dnssd-privacy]
Huitema, C. and D. Kaiser, "Privacy Extensions for DNS-SD", draft-ietf-dnssd-privacy-03 (work in progress), September 2017.

[KW14a] Kaiser, D. and M. Waldvogel, "Adding Privacy to Multicast DNS Service Discovery", DOI 10.1109/TrustCom.2014.107, 2014, <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7011331>.

[KW14b] Kaiser, D. and M. Waldvogel, "Efficient Privacy Preserving Multicast DNS Service Discovery", DOI 10.1109/HPCC.2014.141, 2014, <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7056899>.

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

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

Author's Address

Christian Huitema
Private Octopus Inc.
Friday Harbor, WA 98250
U.S.A.

Email: huitema@huitema.net