Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 31, 2018

C. Huitema
Private Octopus Inc.
June 29, 2018

DNS-SD Privacy Scaling Tradeoffs
draft-huitema-dnssd-privacyscaling-01

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 DNS-SD Working Group assumes peer-to-peer pairing between the service to be discovered and each of its clients. This has good security properties, but creates scaling issues, because each server needs to publish as many announcements as it has paired clients. This leads to large number of operations when servers are 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

Copyright Notice

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 users' 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 client. In this draft, we compare scaling and privacy properties of three different designs:

- o The individual pairing defined in [I-D.ietf-dnssd-privacy],
- o A single server discovery secret, shared by all authorized clients,
- o 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 each other 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 designs 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 scaling, 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. Group public keys

In contrast to pair-wise shared secrets, applications may associate public and private key pairs with groups of equally authorized clients. This is identical to the pairwise sharing case if each client is given a unique key pair. However, this option permits multiple users to belong to the same group associated with a public key, depending on the type of public key and cryptographic scheme used. For example, broadcast encryption is a scheme where many users, each with their own private key, can access content encrypted under a single broadcast key. The scaling properties of this variant depend not only on how private keys are managed, but also on the associated cryptographic algorithm(s) by which those keys are used.

2.3. Shared symmetric 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 secrets.

2.4. Shared public key

Instead of a discovery secret used in Section 2.3, 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 construct 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.

G: The average number of authorized groups 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.

Group public keys: O(G): One record per group.

Shared symmetric secret: O(1): One record for all (shared) clients.

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

There are other elements of scaling, linked to the mapping of the privacy discovery service to DNS-SD. DNS-SD 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.

Group public keys: O(P*G): There are O(P) servers present, and each publishes O(G) instances.

Shared symmetric secret: O(P): One record per server present.

Shared public secret: O(P): One record per server present.

The DNS-SD 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.

Group public keys: 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.

Shared symmetric secret: O(M): Same behavior as in the pairing secret case.

Shared public secret: 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.

Group public keys: Caching is possible for among members of a group.

Shared symmetric secret: Caching is possible, since there is just one server instance.

Shared public secret: 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 and parse announcements. 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.

Group public keys: With a valid group private key, the adversary can issue queries and parse announcements. It will be able to track the presence of all the servers with which the compromised group was authenticated. 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 or groups.

Shared symmetric secret: With a valid discovery secret, the adversary can issue queries and parse announcements. 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 server impersonation attack.

Shared public secret: With a valid discovery public key, the adversary can issue queries and parse announcements. 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. Revocation

Assume 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.

Group public key: The administrator must revoke the private key associated with the compromised group members and, depending on the cryptographic scheme in use, generate new private keys for each existing, non-compromised group member. The latter is necessary for public key encryption schemes wherein group access is permitted based on ownership (or not) to an included private key. Some public key encryption schemes permit revocation without rotating any non-compromised group member private keys.

Shared symmetric secret: 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.

Shared public secret: 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, or O(G) groups, 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 preceding 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
Group public key	Medium	Bad	Maybe
Shared symmetric secret	Good	Really bad	Poor
Shared public secret	Good	Bad	Maybe

Table 1: Comparison of secret sharing solutions

All four types of solutions provide reasonable privacy when the secrets are not compromised. They all have poor resistance to the compromise of a client, as explained in Section 4.1, but sharing a symmetric secret is much worse because it does not prevent server impersonation. The pairing secret solution scales worse than the discovery secret and discovery public key solutions. The group public key scales as the number of groups for the total set of clients; this depends on group assignment and will be intermediate between the pairing secret and shared secret solutions. The pairing secret solution can recover from a compromise with a smaller number of updates, but the public key solutions may benefit from a simple recovery solution using some form of "revocation list".

6. Security Considerations

This document does not specify a solution, but discusses 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]. The text on Group public keys is based on Chris Wood's contributions.

9. Informative References

[I-D.ietf-dnssd-pairing]
Huitema, C. and D. Kaiser, "Device Pairing Using Short Authentication Strings", draft-ietf-dnssd-pairing-04 (work in progress), April 2018.

[I-D.ietf-dnssd-privacy]
Huitema, C. and D. Kaiser, "Privacy Extensions for DNS-SD", draft-ietf-dnssd-privacy-04 (work in progress), April 2018.

[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>.

[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>.

[SIGMA] Krawczyk, H., "SIGMA: The 'SIGn-and-MAc'approach to authenticated Diffie-Hellman and its use in the IKE protocols", 2003, <http://link.springer.com/content/pdf/10.1007/978-3-540-45146-4_24.pdf>.

[Wu16] Wu, D., Taly, A., Shankar, A., and D. Boneh, "Privacy, discovery, and authentication for the internet of things", 2016, <https://arxiv.org/pdf/1604.06959.pdf%22>.

Appendix A. Survey of Implementations

This section surveys several private service discovery designs in the context of the threat model detailed above.

A.1. DNS-SD Privacy Extensions

Huitema and Kaiser [I-D.ietf-dnssd-privacy] decompose private service discovery into two stages: (1) identify specific peers offering private services, and (2) issue unicast DNS-SD queries to those hosts after connecting over TLS using a previously agreed upon pre-shared key (PSK), or pairing key. Any out-of-band pairing mechanism will suffice for PSK establishment, though the authors specifically mention [I-D.ietf-dnssd-pairing] as the pairing mechanism. Step (1) is done by broadcasting "private instance names" to local peers, using service-specific pairing keys. A private instance name N' for some service with name N is composed of a unique nonce r and commitment to r using N_k. Commitments are constructed by hashing

N_k with the nonce. Only owners of N_k may verify its correctness and, upon doing so, answer as needed. The draft recommends randomizing hostnames in SRV responses along with other identifiers, such as MAC addresses, to minimize likability to specific hosts. Note that this alone does not prevent fingerprinting and tracking using that hostname. However, when done in conjunction with steps (1) and (2) above, this mitigates fingerprinting and tracking since different hostnames are used across venues and real discovered services remain hidden behind private instance names.

After discovering its peers, a node will directly connect to each device using TLS, authenticated with a PSK derived from each associated pairing key, and issue DNS-SD queries per usual. DNS messages are formulated as per [RFC7858].

As an optimization, the authors recommend that each nonce be deterministically derived based on time so that commitment proofs may be precomputed asynchronously. This avoids O(N*M) computation, where N is the number of nodes in a local network and M is the number of per-node pairings.

This system has the following properties:

1. Symmetric work load: clients and servers can pre-compute private instance names as a function of their pairing secret and predictable nonce.

2. Mutual identity privacy: Both client and server identities are hidden from active and passive attackers that do not subvert the pairing process.

3. No client set size hiding: The number of private instance names reveals the number of unique pairings a server has with its clients. (Servers may pad the list of records with random instance names, though this introduces more work for clients.)

4. Unlinkability: Private service names are unlinkable to post-discovery TLS connections. (Note that if deterministic nonces repeat, servers risk linkability across private service names.)

5. No fingerprinting: Assuming servers use fresh nonces per private instance name, advertisements change regularly.

A.2. Private IoT

Boneh et al. [Wu16] developed an approach for private service discovery that reduces to private mutual authentication. Moreover, it should be infeasible for any adversary to forge advertisements or impersonate anyone else on the network. Specifically, service discoverers only wish to reveal their identity to services they trust, and vice versa. Existing protocols such as TLS, IKE, and SIGMA [SIGMA] require that one side reveal its identity first. Their approach first allocates, via some policy manager, key pairs associated with human-readable policy names. For example, user Alice might have a key pair associated with the names /Alice, /Alice/Family, and /Alice/Device. Her key is bound to each of these names. Authentication policies (and trust models) are then expressed as policy prefix patterns, e.g., /Alice/*. Broadcast messages are encrypted to policies. For example, Alice might encrypt a message m to the policy /Bob/*. Only Bob, who owns a private key bound to, e.g., /Bob/Devices, can decrypt m. (This procedure uses a form of identity-based encryption called prefix-based encryption. Readers are referred to [Wu16] for a thorough description.)

Using prefix- and policy-based encryption, service discovery is decomposed into two steps: (1) service announcement and (2) key exchange, similar to [I-D.ietf-dnssd-privacy]. Announcements carry service identities, ephemeral key shares, and a signature, all encrypted under the service's desired policy prefix, e.g., /Alice/Family/*. Upon receipt of an announcement, clients with matching policy private keys can decrypt the announcement and use the ephemeral key share to perform an Authenticated Diffie Hellman key exchange with the service. Upon completion, the derived shared secret may be used for any further communication, e.g., DNS-SD queries, if needed.

This system has the following properties:

1. Asymmetric work load: computation for clients is on the order of advertisements.

2. Mutual identity privacy: Both client and server identities are

hidden from active and passive attackers.

3. Client set size hiding: Policy-based encryption advertisements hides the number of clients with matching policy keys.

4. Unlinkability: Client initiated connections are unlinkable to service advertisements (modulo network-layer connection information, such as advertisement origin and connection destination).

Author's Address

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

Email: huitema@huitema.net