A Multicast-Avoiding Privacy Extension for the Avahi Zeroconf Daemon

Daniel Kaiser Andreas Rain Marcel Waldvogel Holger Strittmatter
University of Konstanz, Konstanz, Germany
<first>.<last>@uni-konstanz.de

***Abstract*—In today's local networks, a significant amount of traffic is caused by Multicast packets, such as Multicast DNS Service Discovery (mDNS-SD), a widespread technique used for configurationless service distribution and discovery. It suffers from two major problems inherent in multicast: privacy and network load. We present a privacy extension for the Avahi Zeroconf Daemon that tackles both problems while being very efficient.**

I. mDNS-SD Problems

Multicast DNS Service Discovery (mDNS-SD) is a prevalent technique used for configurationless service distribution and discovery. It uses the upper two layers of the Zeroconf stack, namely, DNS Service Discovery [1] built on Multicast DNS [2], and allows users to offer and use services like device synchronization, file sharing, and chat, when joining a local network without any manual configuration.

This is very convenient but as Zeroconf is built on multicast, two major problems arise: *privacy* and *network load*.

The *privacy problem* arises because every machine in the same network will automatically receive all the announcement traffic and thus obtain a lot of information about the users in the network without having to send a single packet itself. Using mDNS-SD, devices publish their hostnames – frequently containing the user's name – when entering a network, followed by information about offered and requested services (see Figure 1). We show highly revealing real-world examples of private information published by mDNS-SD and present a thread model in our previous work [5]. Many users are completely unaware of how chatty their devices are [7].

The *network load problem* arises because the multicast traffic caused by mDNS-SD may use a vast amount of bandwidth, which is especially bad for large Wi-Fi networks [3]. Therefore some institutional networks do not allow multicast at all or are subdivided in several multicast cells to reduce traffic, which renders mDNS-SD unusable.

II. Solution

Our privacy extension [6] allows to use mDNS-SD without publishing private information and tackles the traffic problem by significantly reducing the number of multicast packets sent. The Zeroconf advantage of not requiring any network configuration is fully maintained for public services; this extends to private services after an initial user pairing.

In addition, the implementation we are going to present can avoid multicast altogether, by using DNS in a very unconventional, innovative way, resulting in an enhanced privacy extension which is

```
daniel@Daniel's Notebook._presence._tcp.local:
type TXT,
vc=! ver=2.10.6 node=libpurple
port.p2pj=5298 txtvers=1
status=gaming
last=Kaiser
1st=Daniel
```

Fig. 1. DNS TXT record that contains several critical key value pairs like the first and last name of the user, the chat status and the version of the service. This TXT record is not constructed; it was published by the pidgin chat client on the author's notebook.

- privacy preserving
- usable without multicast
- completely transparent[1]
- very efficient[2]
- fully backwards compatible, and
- easy to deploy.

Besides solving the privacy and network load problem, it also allows to use zero-configuration service discovery in environments where standard mDNS-SD would not work.

III. Implementation

The implementation of our privacy extension is based on the open source *Avahi*[3] Zeroconf daemon. The new functionality was added with minimal changes to the existing source code. Together with loose coupling the extension is easy to maintain. Our implementation consists of three main logical components:

1) Privacy Socket Distribution: To be able to offer configurationless service discovery over unicast, we need means to distribute the network parameters of a special socket, called *privacy socket*, when entering a network. To this end our implementation uses a metaservice type, `_ppSOS`, whose service instance name allows finding friend's devices. This instance name is a random string that is exchanged once[4]

[1]Both client software and the network infrastructure are oblivious to our privacy extension.

[2]It demands imperceptible additional CPU time and significantly reduces the network load [6].

[3]http://avahi.org

[4]Even if a user has several devices, it is sufficient to pair one of his devices to one of each friends devices, because we allow pairing data synchronization among the devices of the same user via our *Privacy Data Sync Service*, `_pdss`.

Konstanzer Online-Publikations-System (KOPS)
URL: http://nbn-resolving.de/urn:nbn:de:bsz:352-0-299586

per pair of users during an initial pairing, which is realized via another metaservice but can also use out-of-band secure channels like Bluetooth, encrypted email or photographing a QR code.

The `_ppSOS` instance can be distributed in two ways. The first way of distribution is similar to the standard mDNS-SD service announcement: the service instance is announced and queried using multicast. To grant privacy, our implementation encrypts all the information in the corresponding resource records. This method is limited to a single multicast zone. The second way of distribution, which allows using mDNS-SD even if multicast is disabled, is based on Stateless DNS [4]. It stores the `SRV`, `TXT` and `A` resource records needed to resolve the `_ppSOS` service instance in the cache of the local DNS Server. This is done by sending a special "programming query" to our echo server, which uses only the information contained in this query to generate a DNS response the local cache will store. Because the response is built only from information contained in the "programming query", our echo server does not need any configuration files or state; it consists of a single Perl script using the Perl DNS library[5], making the redundant or local deployment very easy. It is not even necessary to have an echo server running within an institution. *Any* echo server instance can be used as long as it supports our query format. The lack of state also allows the use of anycast.

Both ways of *privacy socket* distribution can be used simultaneously and do not interfere.

2) Privacy Aware Unicast Service Discovery: The queries and responses of all three stages of service discovery, namely service browsing, service resolution, and hostname resolution, are sent via unicast, encrypted using the symmetric key exchanged during pairing, to the *privacy sockets* of offering and requesting devices (Figure 2). A hashtable containing online friends and their *privacy sockets* is maintained with the help of the `_ppSOS` service described above.

Our implementation receives DNS queries that were generated by client software via DBUS and instead of multicasting them to the mDNS-SD multicast address, it demultiplexes them to the *privacy sockets* of online friends' devices. Despite the fact that there are many special cases to consider, our implementation is very efficient and causes imperceptible additional CPU usage. The privacy extension is fully transparent to all client software, completely obviating any need for change.

3) User Control: Our *Enhanced Service Browser* allows selecting which services are offered to whom. It is further used to control the pairing process described above and informs the user when a (newly installed) application wants to offer or request a service which is not configured yet, instead of publishing the information without the user's awareness. To ease configuration the *Enhanced Service Browser* allows setting sensible defaults. The tool also gives the functionality of a traditional service browser, namely listing available service instances; in addition it shows friends and other devices of the user that are currently online.

[5]http://www.net-dns.org

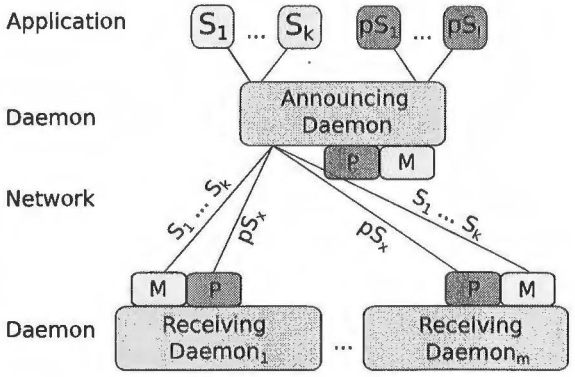

Fig. 2. While our solution still allows traditional announcement of services $S_1, \ldots, S_k$, it is possible to announce services $pS_1, \ldots, pS_l$ in a privacy-preserving way. This is done by sending chosen services pS_x to chosen devices' *privacy sockets* P using unicast.

IV. Demonstration

In our demonstration we will show the privacy problems of standard mDNS-SD, present our privacy extension and demonstrate the Enhanced Service Browser; thus the demonstration will be divided in three parts:

1) Privacy Problem: We will demonstrate the leakage of private data when using the standard daemon and show which information Eve, an example adversary, can get when using a simple service browser like `avahi-browse`. Using Wireshark, we will also show the packets transmitted.

2) Solution: We will present our privacy extension publishing and requesting the same services and demonstrate that none of the private data shown before is accessible by Eve. We will further show that our privacy extension is backwards compatible and works with existing applications (e.g. Pidgin).

3) User Control: We will further demonstrate how to use the Enhanced Service Browser to browse for services in the network, pair to a new device, and tune the privacy settings for an example service.

References

[1] S. Cheshire and M. Krochmal. *DNS-Based Service Discovery*. Number 6763 in Request for Comments. Internet Engineering Task Force (IETF), 2013. I

[2] S. Cheshire and M. Krochmal. *Multicast DNS*. Number 6762 in Request for Comments. Internet Engineering Task Force (IETF), 2013. I

[3] S.G. Hong, S. Srinivasan, and H. Schulzrinne. Measurements of multicast service discovery in a campus wireless network. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1–6. IEEE, 2009. I

[4] Daniel Kaiser, Matthias Fratz, Marcel Waldvogel, and Valentin Dietrich. Stateless DNS. Technical Report KN-2014-DiSy-004, University of Konstanz, Dec 2014. III-1

[5] Daniel Kaiser and Marcel Waldvogel. Adding privacy to multicast DNS service discovery. In *Proceedings of IEEE TrustCom 2014 (IEEE EFINS 2014 Workshop)*, 2014. I

[6] Daniel Kaiser and Marcel Waldvogel. Efficient privacy preserving multicast DNS service discovery. In *Workshop on Privacy-Preserving Cyberspace Safety and Security (IEEE CSS 2014)*, 2014. II, 2

[7] Bastian Konings, Christoph Bachmaier, Florian Schaub, and Michael Weber. Device names in the wild: Investigating privacy risks of zero configuration networking. In *IEEE MDM*, 2013. I