



TECHNISCHE UNIVERSITÄT MÜNCHEN
DEPARTMENT OF INFORMATICS

BACHELOR'S THESIS IN INFORMATICS: GAMES ENGINEERING

Privacy Implications of mDNS

Winfried Baumann



TECHNISCHE UNIVERSITÄT MÜNCHEN
DEPARTMENT OF INFORMATICS

BACHELOR'S THESIS IN INFORMATICS: GAMES ENGINEERING

Privacy Implications of mDNS

Auswirkungen von mDNS auf die Privatsphäre

Author Winfried Baumann
Supervisor Prof. Dr.-Ing. Georg Carle
Advisor Dr. Matthias Wachs
Date February 15, 2017



I confirm that this thesis is my own work and I have documented all sources and material used.

Garching b. München, February 15, 2017

Signature

Abstract

With Zeroconf computers automatically create a network without requiring the user to make any configuration. Service Discovery is one part of this idea which enables devices to announce and discover services like music or filesharing. These services are primarily used in home networks and have potential to publish sensitive information about the user providing the service. This becomes a problem with the growing number of mobile devices such as smartphones [1] and laptops, because they leave the trusted network and still announce their services. Thus the goal of this thesis is to analyze Multicast DNS with DNS Service Discovery, Apple's approach to Zeroconf, in real networks to assess the privacy and security impact for users whose devices announce services. We start by learning how Zeroconf works and have a special look at Multicast DNS Service Discovery. This knowledge is then used to develop a tool for collecting service announcements in real networks. After collecting service announcements we analyze them, to learn about the current state of mDNS-SD and the implications on privacy and security, and come to the conclusion that the biggest problem of mDNS-SD is the naming of the devices because they contain the first name of the user in 55% of all cases and 10% even contain the full name.

Zusammenfassung

Hinter Zeroconf steckt die Idee, dass man Rechnernetze ohne Konfiguration durch Menschen aufbaut. Ein Teil davon ist das automatische Auffinden von Diensten wie z.B. Musik- oder Dateisharing. Diese Dienste werden eigentlich in Heimnetzwerken benutzt, aber die steigende Anzahl an mobilen Geräten, wie Laptops und Smartphones [1], bringt diese Dienste auch in fremde Netze und gibt dort möglicherweise private Daten preis. Deshalb ist das Ziel dieser Arbeit die Analyse von Multicast DNS zusammen mit DNS Service Discovery, einer Variante von Zeroconf entwickelt von Apple, in echten Netzen und das Bestimmen der Auswirkungen auf die Privatsphäre und die Sicherheit der Dienstanbieter. Um die Auswirkungen bestimmen zu können eignen wir uns zuerst Hintergrundwissen zu Zeroconf im Allgemeinen und dann Multicast DNS Service Discovery (mDNS-SD) im Speziellen an. Mit diesem Wissen wie mDNS-SD funktioniert bauen wir ein Werkzeug, welches uns hilft die Informationen zu sammeln welche die Dienste im Netz veröffentlichen. Nachdem wir die Daten gesammelt und analysiert haben, um den aktuellen Stand von mDNS-SD und die Auswirkungen auf Sicherheit und Privatsphäre zu kennen, kommen wir zu dem Schluss, dass das größte Problem bei mDNS-SD die Namensgebung der Geräte ist und diese in vielen Fällen (65% aller Geräte) den Namen des Benutzers zumindest teilweise (55%) oder sogar ganz (10%) enthält.

Contents

1	Introduction	1
1.1	Goals of the Thesis	1
1.2	Outcome	2
1.3	Outline	2
2	Multicast DNS and Service Discovery	3
2.1	Zero-configuration Networking	3
2.2	Multicast DNS	4
2.3	DNS - Service Discovery	6
2.4	Examples	7
2.4.1	Querying for Instances	7
2.4.2	Getting Instance Information	8
2.5	Summary	8
3	mDNS-SD Privacy Analysis	9
4	mDNS-SD Analysis Toolchain	11
4.1	Objective	11
4.2	Requirements	11
4.3	Structure	12
4.3.1	Logger	12
4.3.2	Database Server	12
4.3.3	Analyzer	13
4.4	Implementation	13
4.4.1	Logger	13
4.4.2	Database Server	13
4.4.3	Analyzer	14
4.5	Summary	14
5	Ethical Discussion	17
5.1	Data Economy	17
5.2	Data Privacy	17

5.3	Disclosure	17
6	Real World mDNS-SD Analysis	19
6.1	Methodology	19
6.1.1	Categorization by the Accessibility of Networks	19
6.1.2	Approach to Collection of Announcements	19
6.1.3	Categorization by Data in Announcements	20
6.1.4	Analysis of Operating Systems	20
6.2	Data Collection	20
6.2.1	Network Analysis	21
6.3	Hostnames and Devices	21
6.4	Services	22
6.4.1	Apple Mobile Device Protocol (apple-mobdev2)	23
6.4.2	Apple File Sharing (afpovertcp) and Server Message Block (smb)	23
6.4.3	NVIDIA streaming service (nvstream)	23
6.4.4	iTunes Home Sharing (home-sharing)	24
6.5	Differences by Network Accessibility	24
6.5.1	Private Networks	24
6.5.2	Semi-Public Networks	24
6.5.3	Public Networks	25
6.6	Service Categories	25
6.6.1	Personal	25
6.6.2	Semi-Personal	25
6.6.3	Non-Personal	25
6.7	Analysis of Differences in Operating Systems	26
6.8	Analysis of Queries	26
7	Related Work	27
8	Conclusion	29
8.1	Discussion	30
8.2	Future Work	30
	Bibliography	31

List of Figures

2.1	Hostname resolution in a local network using mDNS	5
2.2	Querying for airplay instances in a local network using mDNS-SD . . .	7
4.1	The Service Instance Explorer of the Analyzer	15

List of Tables

6.1	Overview over the location, date and duration of service announcement collection	21
6.2	Service Instance Distribution	22
6.3	Instance Name Distribution	22
6.4	Service Distribution by Network Accessibility	24

Chapter 1

Introduction

Zeroconf Networking(Zeroconf) aims to create usable networks without configuration by enabling automatic network address assignment together with resolution and distribution of computer hostnames. *Service Discovery* is also a component of this technology: It allows devices to find and offer services like printing or filesharing. That behavior is handy for users of the network because they can simply utilize services announced by other clients without the trouble of configuration.

Due to the fact that the user has to be able to distinguish between announced instances of a service, each one of them has to share metadata like a name and other information depending on the type of service it offers. Sharing images from a smartphone to be able to have a look at them on a laptop with a bigger screen is not a real problem as long as you are in a network you can trust like your home network. The issue arises when these mobile devices connect to other networks in which unknown clients reside and still offer the service to everyone altogether with its metadata.

1.1 Goals of the Thesis

This Bachelor's thesis has a look at the effect of mobile devices announcing their services in foreign networks by answering the following question:

Which sensitive information can be extracted from local networks using Multicast DNS Service Discovery and what are implications in terms of privacy and security for the user?

This question can be split into several subquestions:

Q1: How does Multicast DNS Service Discovery realize service announcement and discovery?

Q2: Which data is published during announcement and discovery of services?

Q3: Which parts of the announcement and discovery may contain sensitive data?

Q4: Can we qualify and quantify privacy violations?

Q5: Can the leakage of data be limited or prevented?

To answer this question we start with an analysis of Zeroconf and its current state of the art technology. Afterwards we will have a look at the current deployment of Multicast DNS Services in real networks by implementing a tool that allows the collection of service announcements.

1.2 Outcome

Answering the questions above will result in a deeper understanding of how Multicast DNS-based Service Discovery works, how its current state in terms of deployment is and the impact on privacy and security. We will discuss which actions can be taken to preserve privacy and security. The tool we will implement for discovering and analyzing service instances will be made available to the public [2]. This enables others to redo the analysis and if necessary improve it in the future.

1.3 Outline

The thesis is structured as follows: We start with a description how Multicast DNS and DNS-based Service Discovery works (Chapter 2). After a privacy analysis of the Multicast DNS Service Discovery technology (Chapter 3) we will design a toolchain (Chapter 4) for collecting and analyzing service announcements to be able to evaluate the current deployment in real world networks. In the analysis chapter (Chapter 6) we will have a look at the collected information. Finally, the thesis is concluded with a summary of our findings (Chapter 8).

Chapter 2

Multicast DNS and Service Discovery

Classical computer networks require the presence of special configuration servers to be able to connect to other clients and use services like printing or filesharing. Examples for configuration servers are *Dynamic Host Configuration Protocol* (DHCP) [3] servers used to distribute IP addresses or *Domain Name System* (DNS) [4] servers for name resolution, which describes the process of obtaining an IP address for a given hostname. But even if everything is set-up correctly the servers need to be operated and can fail because their underlying hardware malfunctions or other events shut down the device. This creates the need for a decentralized solution for the problem.

2.1 Zero-configuration Networking

The idea of *Zero-configuration Networking* (Zeroconf) is that computers automatically configure themselves to create a usable computer network. Automatic assignment of network addresses, distribution and resolution of hostnames and automatic location of network services make up the core of this technology. This facilitates building the network which would otherwise require manual configuration as described above. Failure is impossible with this setup because every client manages itself by talking to the other clients in a predefined way. The problem was that there existed no standard for all three core features of Zeroconf so companies and organisations came up with different approaches:

- Microsoft proposed *NetBIOS* [5] for IPv4 and *Link-local Multicast Name Resolution* (LLMNR) [6] for IPv6, since NetBIOS is not available over IPv6, for name resolution. Service Discovery is provided using NetBIOS Service Discovery which allows devices to simply share their services or even share services of other hosts connected to them for example a printer connected over a parallel port. One major drawback of this choice is that two different technologies are used for the

single task of resolving names.

- *Web Services Dynamic Discovery* (WS-Discovery) [7] was shaped by Microsoft, Intel, Canon and other companies and is primarily used by printers. Communication is realized using SOAP over UDP in combination with other web service standards.
- *AllJoyn*, a collaborative open source software stack, was created to allow devices to communicate with each other. A part of the stack contains the Zeroconf idea where Multicast DNS is used in combination with HTTP over UDP.
- Apple's *Bonjour* [8] implementation uses Multicast DNS and DNS-based Service Discovery and is one of the most used Zeroconf implementations today, probably due to its simplicity and reuse of existing structures of DNS.

A lot of companies, including the big ones like Apple and Google, use Multicast DNS Service Discovery in their products. MacBooks, iPhones, Android-based smartphones, computers running Linux but also printers, Streaming-Devices and other IoT devices use it to connect to each other. It is possible to discover filesharing devices, game servers like Minecraft, voice communication servers, printers and a lot more.

The following sections contain the necessary parts of Multicast DNS and Service Discovery required as background for subsequent chapters.

2.2 Multicast DNS

Multicast DNS (mDNS) is specified in RFC 6762 [9] written by Cheshire et al.. With mDNS, clients request DNS-like resource records from authoritative clients in the local network using IP Multicast. Query and response messages are similar to classic DNS so the main and most noticeable difference is that there is no central DNS instance involved.

The specification contains several parts of interest for the upcoming chapters:

mDNS uses a local namespace, separated from global DNS, with the pseudo-top-level domain „local“ for hostnames of devices in the local network which can be resolved using mDNS. These hostnames are unique and have the form of *single-dns-label.local*. with a maximum length of 255 bytes. Each query must be sent to the multicast address *224.0.0.251* for IPv4 or *FF02::FB* for IPv6 and use port 5353. These queries will not be routed due to the fact that mDNS is link-local.

There are two ways of querying for information in an mDNS network:

1. One-Shot Querying:

Suitable for requests where only one answer is enough, like resolving a hostname to an IP address.

After sending the query only the first answer will be used.

2. Continuous Querying:

Suitable for all requests but especially for queries where multiple answers are possible, e.g. searching for one of multiple printers.

After sending the query all answers will be gathered until the program is interrupted. The time between successive queries must increase at least by a factor of two.

Using multicast has the following major advantages: Everyone in the network can see the responses to a query and if clients have queried the same question before they can silently update their caches and detect conflicts.

When new clients join the network they can tell the responder that unicast responses are allowed if the question has been answered not so long ago to reduce network load. Responding to queries is only allowed for authoritative sources for a given record and when the response is positive, non-null or the message that a particular record doesn't exist. Responses may not contain questions but multiple answers.

To reduce collisions queriers and responders send each message with a random delay.

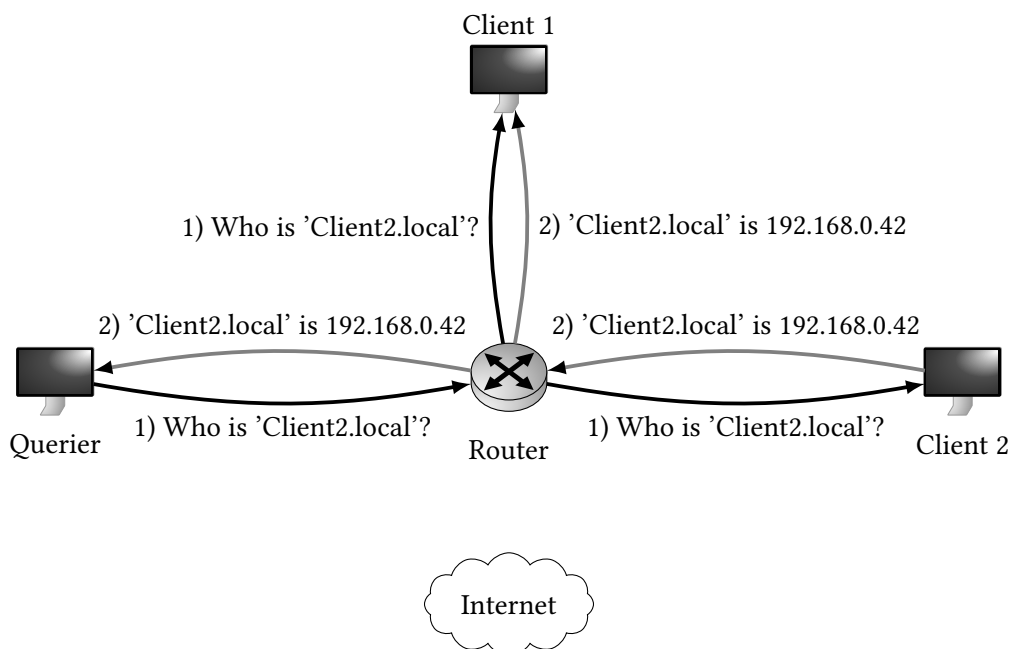


Figure 2.1: Hostname resolution in a local network using mDNS

2.3 DNS - Service Discovery

The DNS-SD (RFC 6763 [10]) mechanism allows discovering named instances of services using standard DNS queries.

Services are identified by the service type and protocol they use. IANA provides a list of currently registered services [11] and offers the option to register own ones. Examples for services are filesharing (e.g. *afpovertcp*), printing (e.g. *ipp*) and music streaming (e.g. *tivo-music*). Each instance of a service announces a DNS SRV and DNS TXT record. The SRV record with a name of the form *Instance.Service.Domain* contains the host and port where this instance can be found. The TXT record contains further information in a key/value based structure such as the version of the service this instance is running. The content of the TXT record depends on the application, so even if both applications implement the same service their payload can differ due to the fact that some values are optional.

Acquiring a list of available instances of a service is done by querying for PTR records using a name with the pattern *Service.Domain*. The result of this query is a list of PTR records giving instance names of the queried service which describe SRV/TXT record pairs.

Instance names should be preconfigured and changeable by the user to be in line with the idea of Zeroconf. There are some limitations which must be considered when picking one automatically. They are limited to 63 octets in length (worst case: 15 Unicode characters) and should be readable by the user which means that names containing only spaces or similar characters should be avoided. These names must be using Net-Unicode [12], which enforces the UTF-8 encoding and limits the amount of control characters to the ones really needed like spaces. The name should not contain any parts of the MAC address, serial number or anything else to make it globally unique, because it only has to be unique in the current local network and easy for humans to map the name to a real device.

The paragraph below will explain the remaining parts of the SRV resource record name: Two DNS labels make up the *Service* part. The first one starts with an underscore followed by the service name, which describes what the service does and has a maximum length of 15 characters. And the second one is „*_tcp*“ when the services run over TCP or „*_udp*“ for UDP or every other connection protocol like SCTP or DCCP. The last part in a service instance name is the *Domain*. This can be anything ranging from standard unicast DNS domain names like „*example.com*“ to domain names with rich-text service subdomains like „*Apartment 3, 1st Floor.example.com*“. When using mDNS the domain is always „*local*.“ as described in the previous section.

„*_services._dns-sd._udp.Domain*“ is a special address which returns when queried all

available service types in the queried domain.

The lifecycle of a service instance consists primarily of the following events:

1. Announcing Services:

When service instances start they cannot send responses containing the new instance because there is no query they can respond to. Some implementations of mDNS-SD (e.g. Avahi [13]) use the trick of querying their own instance on start to announce themselves to the network. In general announcing the service simply comes down to responding to queries if a record exists which matches the query.

2. Responding to queries:

If a record contains the service type queried for then the DNS server or the client, in case mDNS is used, offering the service responds with the instance name of the service. To reduce network load the response may also contain the SRV and TXT resource record of the instance because the querier is very likely to issue a request for these records in the not so distant future.

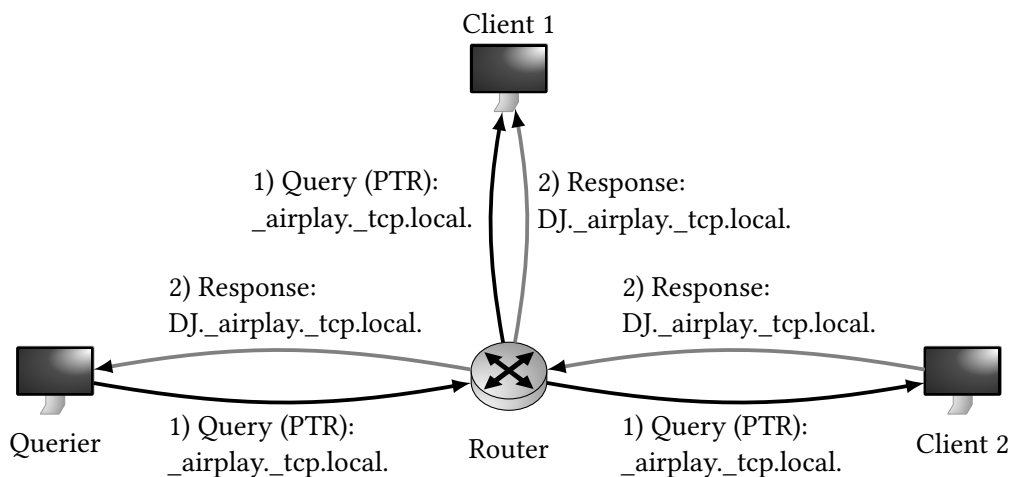


Figure 2.2: Querying for airplay instances in a local network using mDNS-SD

2.4 Examples

2.4.1 Querying for Instances

This example requests PTR records of all instances offering the presence service. The presence service is used for local chats in applications like iChat/Messages or Pidgin.

```

| _Query: presence._tcp.local type PTR
|   _presence._tcp.local
|     name = michael@Michael's iPhone._presence._tcp.local
|   _presence._tcp.local
|     name = michael@Michael's Laptop._presence._tcp.local

```

There are two devices currently running the presence service.

2.4.2 Getting Instance Information

After finding an interesting instance the next step would be to get information on how to connect to the instance.

```

| Query: michael@Michael's Laptop._presence._tcp.local type SRV
|   service = Michael's Laptop.local 5298
|
| Query: michael@Michael's Laptop._presence._tcp.local type TXT
|   text = "textvers=1"
|         "client=Pidgin"
|         "clientvers=4.11"
|         "first=Michael"
|         "last=Maier"
|         "status=Away"

```

Querying for the SRV and TXT records returns information about „michael@Michael's Laptop“ required to be able to connect to the instance together with some additional information like his status.

2.5 Summary

Multicast DNS with DNS-based Service Discovery is the current standard to create a network where clients can connect to each other and use their services that requires zero configuration and is thus easy to set up. mDNS allows clients to communicate with each other by sending and listening to a specific multicast address which removes the need for a central DNS server. Discovery and announcement of services like filesharing and printing is achieved with DNS Service Discovery to simplify their use.

Every service can announce sensitive information in the instance name or the TXT resource record depending on the application implementing the service, thus even though two users announce the same service one might leak more sensitive information than the other.

Chapter 3

mDNS-SD Privacy Analysis

With the knowledge of how mDNS-SD works, this chapter is about finding the critical spots for the users privacy.

Announcing services shows the services and hosts a client is interested in. This shows a potential attacker which services he can offer to the victim if the original instance is not available. Offering fake versions of services a client is looking for can lead to privacy and security issues.

Discovering services is a three step process, where each of the steps shows potential to leak information:

1. Announcing services:

This has two problems: The first one being the fact that everyone can enumerate services and options for private services doesn't exist which shows information to all clients in the whole link-local domain.

The second one relates to the naming of instances because they are often configured to contain the username and/or the hostname. When looking at the examples in the previous chapter all users find michael@Michael's Laptop in their Pidgin IM-Application even without having each other sent a buddy request. Even though this seems to be constructed it is common practice to use the username in combination with the hostname to make the users of a service unique in the network.

2. Getting SRV and TXT resource records:

After finding the instance, the querier tries to connect to the service using the SRV and TXT records. These two records face the same problems of containing instance/service name and the service type like the PTR record in the enumeration does but there are even more problems.

The SRV record contains the port number which often refers to a particular service type, e.g. port 22 for ssh. This is not only a privacy problem but an even bigger one in terms of security, because it enables passive port scanning without rendering the attacker suspicious like a port scan would do. The list of key and value pairs in the TXT record has an even greater probability to provide others with sensitive information. In the example above the users full name and his status is announced to the anyone on the network. Security hazards arise when the record contains the version of the service which can be used to test whether it is vulnerable to an attack or not.

3. Resolving the hostname:

When getting the IP from the A or AAAA record found in the SRV resource the hostname, which is used as DNS name to create the mapping, is once again publicly accessible in the network. The creation of a mapping from hostname to IP might allow gaining even more insights about the client.

The elaboration shows that the announcement of services has a high potential of creating a privacy threat if services use personal data to announce themselves. Due to the fact that the services have the capability of publishing sensitive information, real world service need to be looked at to be able to gauge the current privacy implications.

Chapter 4

mDNS-SD Analysis Toolchain

4.1 Objective

It is not possible to talk about privacy impacts without knowing the data found when using mDNS-SD, because there is no complete list of all services, no list of applications announcing a service and registering services at IANA [11] is voluntary. Due to the fact that the specification doesn't define what information services announce there will be differences in the amount of data a service publishes. There might even be differences between implementations of programs which use the same service type, e.g. chat programs which only publish the username in contrast to ones that also show the real name and status. The goal is to build a set of tools which collect and analyze the data advertised by mDNS-SD services to gain insights which information is published and the privacy hazards this creates for the user. These tools can then be run continuously as a service.

4.2 Requirements

The first task of the service is to find all mDNS-SD instances currently running in the link-local domain. Then it has to query each instance for their SRV and TXT record to get detailed information about the service instance and store it continuously in a database. With all the data in the database, the service is able to generate statistics which can help the user to reveal privacy risks and understand the information transmitted by the services.

Out of it the following requirements for the service can be derived:

1. Querying all service-instances in the local domain

2. Ability to query in intervals (e.g. every 15 seconds)
3. Support for parallel collection of data in different networks
4. Persistent storage of information about each instance
5. Simple interface for storing and accessing data in the database
6. Continuous monitoring
7. Visualization of the data collected

4.3 Structure

The requirements can be separated into three modules:

The first three requirements are part of the logging module because they describe how data is collected. The next two describe a storage module and the last two depict an analysis module to help the user analyzing the services.

4.3.1 Logger

The logger is responsible for querying service instances in the link-local domain in configurable intervals. Each service instance found will be sent with additional data to the database.

The information stored consists of the following:

- the domain name
- the service name
- the service protocol
- the instance name
- content of the TXT resource record
- time when service was discovered
- descriptive name of the location where the logger is used

4.3.2 Database Server

The primary task of this application is to permanently store the logs received from the logger. To be able to support multiple logging modules in parallel the database needs to be reachable from several locations at once. With this in mind and the requirement of

having a simple interface for querying and adding logs to the database it is reasonable to use a central database server for persistent storage.

4.3.3 Analyzer

Analysis of the collected data happens in the analyzer. It uses the database servers interface to query for data and displays it in different formats for the user. These formats include a table containing all the instances found at a certain location and the ability to filter the results to display only one service type. Another table is used to count the number of available service instances of a service type which allows to quickly see the most announced service. This helps the user to quickly see the information services publish and to find sensitive announcements of services.

4.4 Implementation

4.4.1 Logger

Querying all available service instances is the primary task of the logger. Due to the fact that most Linux systems have Python preinstalled and libraries for the required tasks exist, it seems to be a good choice for collecting the announcements and sending them to the database server. The implementation of this module makes heavy use of two libraries:

With the *zeroconf* [14] library searching for all available instances is done by searching for all service types currently available first and afterwards continuously querying for all instances using one of the services for a few seconds (15) which is enough for running instances to reply. The result of this querying process is a list of instances which is sent to the database server for persistent storage using the *requests* [15] library to be able to evaluate them later on.

4.4.2 Database Server

The database server receives service instances from the logger and needs to be able to send them to the analyzer.

Using REST is the current state of the art for servers which want to offer a simple and clean interface for their clients. Due to the fact that REST makes use of the HTTP protocol and every major programming language has support for web queries, it is reasonable to use it as the technology of choice for the server.

We use the Spring-Boot [16] framework which consists of a REST implementation written in Java and offers database integration by adding the database url, username and password to a properties file (`application.properties`) and annotating the class which

we want to save. Spring-Boot creates database schemes automatically by analyzing the annotated Java classes and uses the Java Persistence API [17] when storing or accessing data from the database. The REST-API we designed consist of four actions which transmit their data in the JSON format and only use strings:

- POST /log/add
Adds a single service instance with additional data found to the database. The data which needs to be sent was defined in Section 4.3.1.
- GET /log/get
Returns all logged instances from the database.
- GET /log/getOrigins
Returns all origins which is the name the logger used to describe the location of the network.
- GET /log/getByOrigin
Returns all logged instances for the requested origin.

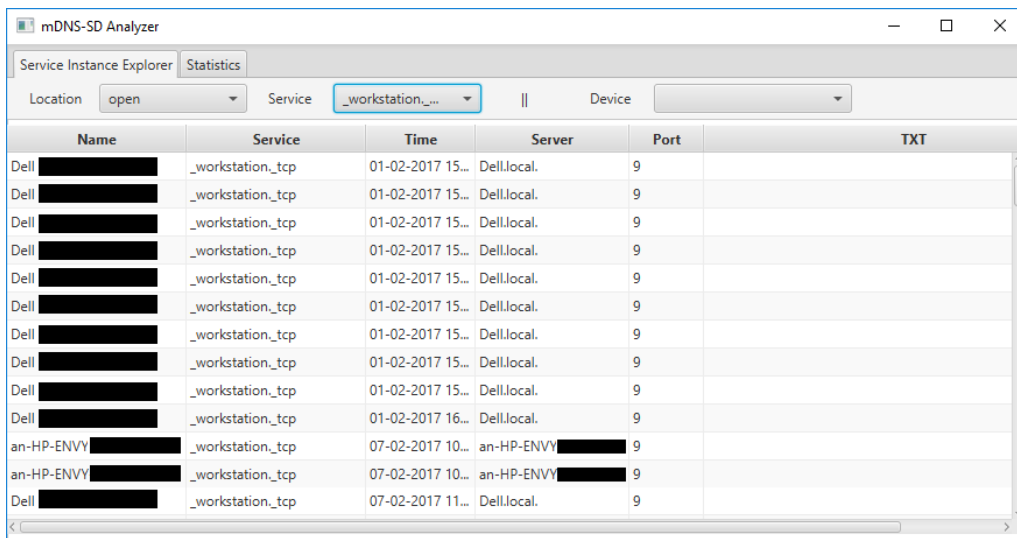
4.4.3 Analyzer

The analyzer module requests found service instances from the database server by querying the REST API and offers a user interface which displays the instances to see the information they published.

The querying is done with the help of Unirest [18], a library which simplifies the generation of REST requests in Java. JavaFX [19], Javas latest user interface toolkit, is used to build a desktop application which contains different views on the data that help the user when making an analysis. One example for such a view is a table that displays the number of instances for each service. This helps the user to quickly identify important services which might be worth to look at.

4.5 Summary

Collecting the required information for the analysis is split into three modules: A logger to collect the announced information. The database on a server for persistent storage and an analyzer for continuous monitoring of the data collected.



The screenshot shows the 'mDNS-SD Analyzer' window with the 'Service Instance Explorer' tab active. The 'Service' dropdown is set to '_workstation_tcp'. The table below displays the following data:

Name	Service	Time	Server	Port	TXT
Dell [REDACTED]	_workstation_tcp	01-02-2017 15...	Dell.local.	9	
Dell [REDACTED]	_workstation_tcp	01-02-2017 15...	Dell.local.	9	
Dell [REDACTED]	_workstation_tcp	01-02-2017 15...	Dell.local.	9	
Dell [REDACTED]	_workstation_tcp	01-02-2017 15...	Dell.local.	9	
Dell [REDACTED]	_workstation_tcp	01-02-2017 15...	Dell.local.	9	
Dell [REDACTED]	_workstation_tcp	01-02-2017 15...	Dell.local.	9	
Dell [REDACTED]	_workstation_tcp	01-02-2017 15...	Dell.local.	9	
Dell [REDACTED]	_workstation_tcp	01-02-2017 15...	Dell.local.	9	
Dell [REDACTED]	_workstation_tcp	01-02-2017 16...	Dell.local.	9	
an-HP-ENVY [REDACTED]	_workstation_tcp	07-02-2017 10...	an-HP-ENVY [REDACTED]	9	
an-HP-ENVY [REDACTED]	_workstation_tcp	07-02-2017 10...	an-HP-ENVY [REDACTED]	9	
Dell [REDACTED]	_workstation_tcp	07-02-2017 11...	Dell.local.	9	

Figure 4.1: The Service Instance Explorer of the Analyzer

Chapter 5

Ethical Discussion

Considering the amount and kind of data which can be gathered with the tool and the possible privacy hazards this creates, an ethical consideration has to be made. The two points which will be taken into account are data economy and data privacy.

5.1 Data Economy

The tool for collecting the data published by the services was designed in a way that only relevant parts for this work are collected and stored to be analyzed later on. In this case relevant data is data which will be used in the privacy analysis.

5.2 Data Privacy

It is not enough to limit the collection to the required minimum because in this particular instance the information collected is expected to be sensitive. Hence all pieces of information which will serve as examples in this paper are changed to prevent the mapping to a real person.

5.3 Disclosure

The database with all the aggregated resource records along with other metadata like when and where it was collected is not anonymized because it would go beyond the scope of this project and was thus permanently deleted after the study.

The two elements of reducing the data to the minimum and anonymizing the data when published create a solid foundation for using real world data as examples while preserving the owners privacy.

Chapter 6

Real World mDNS-SD Analysis

This chapter is about the findings in the data collected with the service described in Chapter 4.

6.1 Methodology

6.1.1 Categorization by the Accessibility of Networks

We categorize networks by their accessibility:

Private or home networks are networks which are not accessible for everyone because they require a password. Semi-Public networks can be found in cafés or inns which require a password that is obtainable from the staff or corporate networks where all employees have access. Public networks form the last category and describe networks that everyone can access without any requirements.

6.1.2 Approach to Collection of Announcements

Public and semi-public networks were searched at central places like train stations, shopping malls or large squares. Private networks were difficult to choose due to their nature of being password protected so we collected announcements in networks of friends with their permission. This is not a perfect solution but the idea is to gain a basic understanding of which services can be expected in home networks so we can make basic comparisons between them. We collected the data by going to the place where the network is and connecting to the network. To test whether a network allows mDNS traffic, we used a laptop to generate test announcements and if the announcement is displayed in the ZeroConf Browser [20], an Android application for discovery of mDNS Services, the network allows mDNS. During data collection, we followed our ethical considerations made in Chapter 5.

6.1.3 Categorization by Data in Announcements

Services can be assigned one of three groups depending on the amount of private data they announce: Personal services share information like the name of the user which is information that can be directly linked to a person. Semi-personal services leave a clue about the user for example which device or hardware he or she uses. Non-personal services provide no data about the user.

6.1.4 Analysis of Operating Systems

Apple's preferred Zeroconf technology is mDNS and DNS-SD which has integrated platform support in all their devices which means the underlying platform can be directly used to publish and discover service announcements without the requirement of an external library.

Windows does not ship with platform support for mDNS but Apple's mDNSResponder [21] has interfaces for C and Java which can be included in an application to announce services. With the ongoing development of native mDNS-SD support for Windows 10 will be able to announce and discover services in the future.

Linux devices have Avahi [13] at their disposal which allows them to announce and discover services.

6.2 Data Collection

We analyzed sixteen public networks near train stations and central squares, four semi-public networks and five home networks. Service announcements were collected between January 25th and February 8th as shown in Table 6.1. After analyzing the first two private networks it became clear that analyzing this type of network for a longer period does not yield additional information because no instances joined or left the networks. Thus the following collection times in this network category were halved. Longer collection times help at public and semi-public networks because a lot of people were joining and leaving the network during the collection of data. Half an hour was enough to see which services were the most dominant in the network and everything after that made the measurement more precise.

Altogether data of 189 different service instances, 19 different services and 167 unique devices was collected during the research.

Category	Location	Date	Duration
Private	Private1	February 5, 2017	30 min
	Private2	February 5, 2017	30 min
	Private3	February 6, 2017	15 min
	Private4	February 6, 2017	15 min
	Private5	February 7, 2017	15 min
	5 locations		1.75 hours
Semi-Open	Café1	January 25, 2017	30 min
	Café2	February 1, 2017	30 min
	Inn	February 1, 2017	30 min
	University	February 1, 2017	1 hour
	University	February 7, 2017	1.5 hours
	4 locations		4 hours
Open	Train station	February 4, 2017	45 min
	Shop1	February 8, 2017	15 min
	Shop2	February 3, 2017	30 min
	3 locations		1.5 hours

Table 6.1: Overview over the location, date and duration of service announcement collection

6.2.1 Network Analysis

In only three of the sixteen public networks mDNS services can be announced and discovered which shows that a lot of public network providers are aware of mDNS. All four semi-public networks allowed mDNS traffic with the exception being the eduroam network, a world-wide study and research network, but only the access points in the computer science building at TU Munich. A similar result was found in home networks where every one of them allowed mDNS messages to roam freely.

Some of the public networks required a form of authentication (e.g. username and password or a special code) before accessing the Internet is possible but allowed communication via mDNS before login. These networks are not part of the sixteen tested networks and not included in the measurement.

6.3 Hostnames and Devices

55% of the hostnames contained the first name of the user and 10% used the full name. Only one instance was announced with the last name. The other 35% can be split into 14% nicknames, 11% model names and 10% miscellaneous/random names (e.g. LAPTOP-fHfani13Fa). Having the users name in the hostname is problematic because the hostname is part of every service announcement which makes it easy to track specific persons by collecting their announcements.

It is not surprising to see that the majority of hostnames (74%) can be linked to Apple products (iPad, iPhone, iMac and MacBook) because they introduced mDNS and DNS-based SD. A lot of the other devices (7%) can be categorized into laptops or desktop computers because they offer the nvstream service which will be looked at in Section 6.4.3. When looking at the data it is interesting to see that there is not a single Android-based smartphone announcing mDNS-SD services. This could be because the Android world is not that interwoven with other devices like the Apple world is, for example when looking at music or file sharing.

6.4 Services

Instances of Service	Distribution
apple-mobdev2	65%
homekit	8%
nvstream	6%
smb	3%
afpovertcp	3%
ipp	2%
other services	13%

Table 6.2: Service Instance Distribution

Looking at Table 6.2 most announced (65%) are instances announcing the apple-mobdev2 service. The homekit service, used to control smart home devices, follows by a distance with only 8% and thereafter comes the nvstream service with 6%. The first and last service mentioned will be analyzed in Section 6.4.1 and Section 6.4.3. Smart home devices are currently known to be hackable [22] because producers lack incentives to enhance security [23]. Research about the homekit service showed that currently no security flaws were found but similar services might have flaws and such announcements help attackers to find vulnerable services.

Service Instance Names	Distribution
Hostname	48%
Hostname and Text	21%
MAC address	5%
MAC address and local IPv6 address	5%
Hostname and MAC Address	5%
Random	16%

Table 6.3: Instance Name Distribution

Table 6.3 shows that 74% of all services use the hostname in the instance name and 26% use either the MAC address (10%), or the local IPv6 address (5%), or a random

combination of letters and digits (16%). To be able to identify an instance it has to have a meaningful name to be able to distinguish it from the others. Thus using the instance name is a good choice but has the same implications as described in Section 6.3. Publishing the local IP or MAC address is not a problem because both can be acquired by listening to the network traffic, too.

75% of the services found did not use the TXT record to provide additional information. Only 25% published information about their service like which file formats are allowed for printing.

The further content of this section is about the most frequent, interesting or chatty services found during the collection of announcements.

6.4.1 Apple Mobile Device Protocol (apple-mobdev2)

This service [24] is by far the one with the most announced instances (65% of all instances) and allows system administrators to manage iOS devices by sending messages to them. Authorized IT administrators can modify or install profiles, remove passcodes or even erase the device. This functionality is primarily used by companies which need to manage a lot of devices. Private individuals make use of this service to synchronize their iTunes library [25] at home with the one on their mobile device.

Instances announce themselves with their MAC address followed by their local IPv6 address separated by the @ symbol which is no security problem as explained above but it is difficult for the user to distinguish between three different instances.

6.4.2 Apple File Sharing (afpovertcp) and Server Message Block (smb)

Both services are primarily used for filesharing and all devices offering one of the services also announced the other. The instances use, in contrast to to the apple-mobdev2 service, the hostname to announce the service. Devices are identifiable as MacBooks and all of them only contain the first name.

The default configuration for these services doesn't allow access to the files without entering the username and password of the device which protects files from intruders.

6.4.3 NVIDIA streaming service (nvstream)

The nvstream service [26] allows owners of a NVIDIA graphics cards to stream games from laptops or desktop computers to the NVIDIA Shield, a handheld game console. Instances of the service use the hostname as the name to announce themselves but no

	Private Networks	Semi-Open Networks	Open Networks
apple-mobdev2	13%	63%	83%
nvstream	38%	6%	0%
afpovertcp	0%	4%	0%
ipp	25%	0%	5%

Table 6.4: Service Distribution by Network Accessibility

instance found contained any information about the user of the device because the host-names were using the name of the model name of the device, the device manufacturer or the word „Laptop“ or „Desktop“ followed by a seemingly random combination of numbers and letters as a name. This was the most used service (32%) of non-Apple devices.

6.4.4 iTunes Home Sharing (home-sharing)

As the name suggests this service [27] can be used to share the iTunes library to other devices in the home network. Even though only one instance of this service was found in all the different networks, the service is listed here because it utilized the full name of the user in the instance name and the TXT content as well.

After setting up an own iTunes home-sharing service to test the naming convention, it came to light that once again the service only uses the name of the device which was the full name in case mentioned above.

6.5 Differences by Network Accessibility

6.5.1 Private Networks

Private networks feature static devices like printers or desktop computers, as well as mobile devices like smartphones or laptops. This can be seen when looking at the services announced: Printers announcing instances of the *Internet Printing Protocol* (IPP) [28] and *Printer Page Description Language Data Stream* (pdl-datastream). Laptops and desktop computers with NVIDIA graphics cards announce the NVIDIA streaming service and iPhones can be found by looking at apple-mobdev2 service instances.

6.5.2 Semi-Public Networks

Networks that are more accessible like WiFi networks in cafés, protected by a password handed out to customers, or university networks contain less static devices and more mobile ones like laptops and smartphones. Filesharing services become more frequent

because the data is moved between the mobile devices and not stored on the central desktop computer.

6.5.3 Public Networks

The results of this network type are not that different to semi-public ones: The relation between smartphone and laptop shifts even more towards smartphones which can be seen by looking at the apple-mobdev2 service. This could be due to the fact that the WiFi networks which had mDNS enabled were at locations where using a smartphone was more comfortable than using a laptop.

6.6 Service Categories

Services fall into one of three categories in terms of privacy.

6.6.1 Personal

There are very sensitive services which often announce the full name of the user, parts of it or additional data like the status of the user.

A lot of services we found use the hostname as instance name to announce their service (see Table 6.3). The problem is that 65% of hostnames contain the full or parts of the users name which shows that the service itself is not the problem but the hostname is.

6.6.2 Semi-Personal

These services don't publish directly usable data like names of a user but due to the fact that the service is available information about the user can be inferred. We found the nvstream service to match this category because it shows which hardware, in this case a NVIDIA graphics card, someone uses. Even though this case might not be problematic other services also share information about a device which can be used by an attacker to get to know the system before he starts his attack. In general, tracking an instance over time can reveal the users habits by looking at the times when the instance is available and when it is not.

6.6.3 Non-Personal

This category of services provide data in their announcement which cannot be linked to a user. Printing services like ipp and pdl-datastream match this category as well as

the scanner service. These services do not have any information about the user and are thus not able to announce sensitive information.

6.7 Analysis of Differences in Operating Systems

Android and Linux devices have mDNS-SD support [29] but not a single device was found which it can be mapped to Android or Linux.

Apple OS devices are currently by far the most devices which use mDNS-SD (74%). This share could decrease after Microsoft finishes the integration of mDNS-SD into Windows 10 because with this feature implemented it is likely to see more Windows devices in the networks announcing their services.

In contrast to iOS and Linux, Windows has the feature of selecting the network type when connecting to a network the first time. This could be used to block mDNS traffic in public networks automatically but currently has no influence on neither sending nor receiving mDNS packages. Further analysis should be made when mDNS is implemented because this would be a simple way of disabling such traffic by network location awareness.

6.8 Analysis of Queries

Even though we did not capture mDNS-SD queries, we had a look at them while testing whether mDNS is enabled in a network.

There are queries for all instances of a certain service but we were also able to find ones for specific instance names which were not answered because they did not exist. These queries show service instances a client usually connects to (e.g. when in the home network). This creates substantial privacy and security risks because not only the service instance names of the devices which leave trusted networks are published but also the ones from other networks and attackers can find entry points into a system by collecting all queries and analyzing the services for weaknesses.

Unfortunately we cannot provide any deeper real world analysis on this due to time constraints and the lack of support for query collection in the zeroconf library.

Chapter 7

Related Work

There are a lot of papers which analyze mDNS-SD and make proposals to improve privacy.

Adding Privacy to Multicast DNS Service Discovery [30] suggests to add pairing of devices to be able to see the service announcement. Unpaired services see encrypted `_privacy` services which can only be decrypted when paired. We found no instance of a „privacy“ service in the data we collected which could be due to the short timespan in which we collected.

Chattering Laptops [31] suggests that the software should remember in which networks they have been configured to run. This can be used to prevent service announcements and discovery queries for example in open networks from the client side.

Device Names in the Wild [32], an analysis of the state of mDNS almost four years ago, had a bigger focus on the naming of the devices. They found out that 59% of the devices announced their first, last or full name which is similar to the result we found (65%). The number of devices announcing the full name of their user decreased from 17.6% in their research to the 10% we found.

Chapter 8

Conclusion

In this thesis we analyzed the current state of Multicast DNS Service Discovery (mDNS-SD) and the implications on privacy and security in different real world networks. To achieve this we had a look at the specifications of mDNS and SD which revealed that the naming of service instances, the hostname and additional data about the service show potential to publish sensitive information about the user. Based on this knowledge we developed a toolchain that allowed us to log service announcements, store them on a centralized database server and monitor the data in a desktop application. This toolchain is publicly available on Github [2] and was built to make it easy to replace or extend parts for in future researches. We collected data from networks of different categories: Private or home networks for networks which only few people can access. Semi-public networks which have a lot of users but are not open to the public (e.g corporate networks) and public networks that can be accessed by everyone.

The first discovery we made was that 81% of the public networks we tested blocked mDNS traffic which shows a lot of network providers know mDNS. The reasons behind the block might be different: Some might do it for privacy reasons while others simply want to reduce the traffic which caused by multicast. After analyzing all instances we found out that 65% of the hostnames contain either the full name (10%) or the first name (55%).

With regard to an older study [32], the number of users sharing their full name dropped by 7%, which could be due to the fact that we focused more on different networks than on longer collection times or a change of the default naming scheme is currently ongoing. Highly personal services like local chat services which share more than the name of the user were not found.

8.1 Discussion

Service Discovery was build to facilitate finding and using services in the current network. These services need to publish some data so the user is able to differentiate between them and select the right one. Thus it makes sense for the services to use the hostname because it is very likely that the user can map it to a real device.

The problem is that most of the devices use the real name of the user or parts of it. This could be tackled by letting the user choose a name during the first setup with the information that this name will be publicly visible. Implementing name selection would solve two problems at once: It creates user awareness for a part of the data a device shares and it can be expected to further reduce the number of user names in the device because most of the people prefer custom device names instead of any parts of their name or the model name [32].

8.2 Future Work

During the time of measurement mDNS-SD was dominated by Apple devices but Microsoft is currently developing mDNS-SD for Windows 10. Some time after the mDNS-SD feature is released this analysis should be repeated to learn about the privacy implications the new service instances running on Windows hosts bring along.

When collecting the service instances in the future it would be interesting to see whether even less devices use the full name of their owner.

Analysis of service instances over a longer timespan (e.g. one month) could be performed to analyze whether they can reveal the habits of their users.

As pointed out in the last section of the real world analysis, we were not able to have a deeper look at the queries due to technical and time constraints but found out that they also show potential to have implications on privacy and security. When tackling the topic the analyzer module from the mDNS-SD collection toolchain can be used and extended to support this new feature.

Bibliography

- [1] eMarketer, “Number of smartphone users worldwide from 2014 to 2020 (in billions),” Jun. 2016. [Online]. Available: <http://www.emarketer.com/Article/Slowing-Growth-Ahead-Worldwide-Internet-Audience/1014045>
- [2] “mDNS-SD-Privacy-Analysis-Tools on Github.” [Online]. Available: <https://github.com/mDNS-SD-Privacy-Analysis-Tools>
- [3] R. Droms, “Dynamic Host Configuration Protocol,” RFC 2131 (Draft Standard), Internet Engineering Task Force, Mar. 1997, updated by RFCs 3396, 4361, 5494. [Online]. Available: <http://www.ietf.org/rfc/rfc2131.txt>
- [4] P. Mockapetris, “Domain names - implementation and specification,” RFC 1035 (Standard), Internet Engineering Task Force, Nov. 1987, updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966. [Online]. Available: <http://www.ietf.org/rfc/rfc1035.txt>
- [5] N. W. G. in the Defense Advanced Research Projects Agency, I. A. Board, and E. to End Services Task Force, “Protocol standard for a NetBIOS service on a TCP/UDP transport: Detailed specifications,” RFC 1002 (Standard), Internet Engineering Task Force, Mar. 1987. [Online]. Available: <http://www.ietf.org/rfc/rfc1002.txt>
- [6] B. Aboba, D. Thaler, and L. Esibov, “Link-local Multicast Name Resolution (LLMNR),” RFC 4795 (Informational), Internet Engineering Task Force, Jan. 2007. [Online]. Available: <http://www.ietf.org/rfc/rfc4795.txt>
- [7] V. Modi and D. Kemp, “Web Services Dynamic Discovery,” Jul. 2009. [Online]. Available: <http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1-1-spec-os.html>
- [8] Apple Inc., “Bonjour,” Aug. 2002. [Online]. Available: <https://developer.apple.com/bonjour/>

- [9] S. Cheshire and M. Krochmal, "Multicast DNS," RFC 6762 (Proposed Standard), Internet Engineering Task Force, Feb. 2013. [Online]. Available: <http://www.ietf.org/rfc/rfc6762.txt>
- [10] S. Cheshire and M. Krochmal, "DNS-Based Service Discovery," RFC 6763 (Proposed Standard), Internet Engineering Task Force, Feb. 2013. [Online]. Available: <http://www.ietf.org/rfc/rfc6763.txt>
- [11] "Service Name and Transport Protocol Port Number Registry." [Online]. Available: <http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml>
- [12] J. Klensin and M. Padlipsky, "Unicode Format for Network Interchange," RFC 5198 (Proposed Standard), Internet Engineering Task Force, Mar. 2008. [Online]. Available: <http://www.ietf.org/rfc/rfc5198.txt>
- [13] L. Poettering and T. Lloyd, "Avahi." [Online]. Available: <http://avahi.org/>
- [14] W. McBrine, "zeroconf library." [Online]. Available: <https://pypi.python.org/pypi/zeroconf>
- [15] K. Reitz, "requests library." [Online]. Available: <http://docs.python-requests.org/>
- [16] Pivotal Software, "Spring Boot." [Online]. Available: <https://projects.spring.io/spring-boot/>
- [17] "Java Persistence API." [Online]. Available: <http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html>
- [18] "Unirest." [Online]. Available: <http://unirest.io/java.html>
- [19] Oracle, "JavaFX." [Online]. Available: <http://docs.oracle.com/javafx/>
- [20] "ZeroConf Browser." [Online]. Available: <https://play.google.com/store/apps/details?id=com.melloware.zeroconf>
- [21] "mDNSResponder." [Online]. Available: <https://opensource.apple.com/tarballs/mDNSResponder/>
- [22] E. Fernandes, J. Jung, and A. Prakash, "Security Analysis of Emerging Smart Home Applications," in *Proceedings of the 37th IEEE Symposium on Security and Privacy*, May 2016.
- [23] C. Lévy-Bencheton, E. Darra, G. Tétu, G. Dufay, and M. Alattar, "Security and Resilience of Smart Home Environments," Dec. 2015. [Online]. Available: <https://www.enisa.europa.eu/publications/security-resilience-good-practices/>
- [24] Apple Inc., "Mobile Device Management protocol." [Online]. Available: <https://developer.apple.com/bonjour/>

- [25] WiFi Nigel, “mobdev iTunes synchronization.” [Online]. Available: <http://wifinigel.blogspot.de/2013/01/apple-itunes-services.html>
- [26] NVIDIA, “NVIDIA Shield streaming.” [Online]. Available: <http://shield.nvidia.com/game-stream>
- [27] Apple Inc., “iTunes library home sharing.” [Online]. Available: <https://support.apple.com/en-us/HT202190>
- [28] R. Herriot, S. Butler, P. Moore, and R. Turner, “Internet Printing Protocol/1.0: Encoding and Transport,” RFC 2565 (Experimental), Internet Engineering Task Force, Apr. 1999, obsoleted by RFC 2910. [Online]. Available: <http://www.ietf.org/rfc/rfc2565.txt>
- [29] “Android Documentation - mDNS-SD.” [Online]. Available: <https://developer.android.com/training/connect-devices-wirelessly/nsd.html>
- [30] D. Kaiser and M. Waldvogel, “Adding privacy to multicast dns service discovery,” in *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, Sept 2014, pp. 809–816.
- [31] T. Aura, J. Lindqvist, M. Roe, and A. Mohammed, *Chattering Laptops*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 167–186. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-70630-4_11
- [32] B. Könings, C. Bachmaier, F. Schaub, and M. Weber, “Device names in the wild: Investigating privacy risks of zero configuration networking,” in *2013 IEEE 14th International Conference on Mobile Data Management*, vol. 2, June 2013, pp. 51–56.