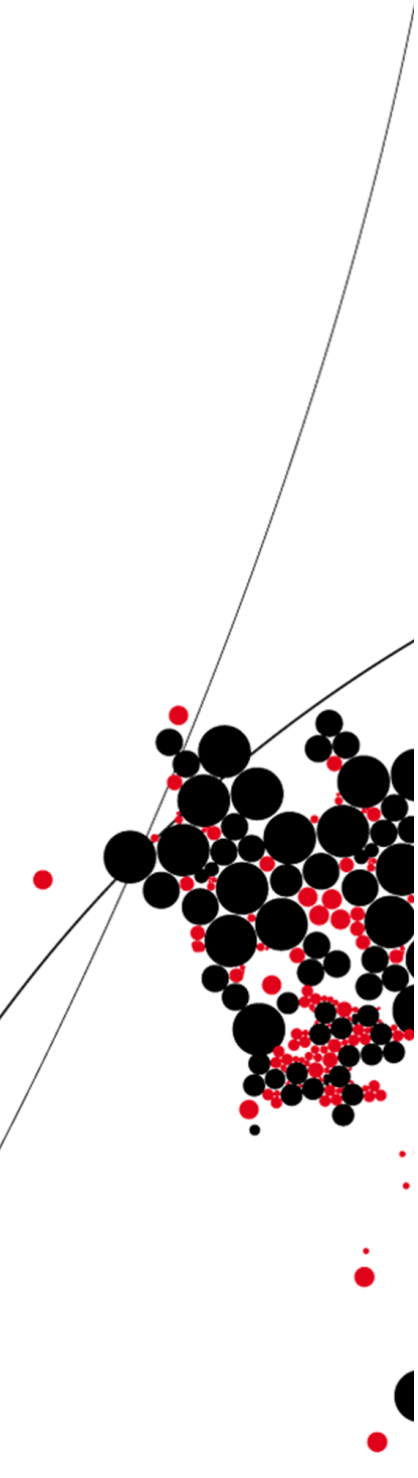
UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering,
Mathematics & Computer Science

Investigating DNS Information Flow In Corporate Networks

R.L.H. Fontein

M.Sc. Thesis
June 2023



Supervisors:
dr. A. Sperotto
dr. ir. R. Sommese

Design and Analysis of Communication Systems
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

The reverse DNS is a fascinating part of the DNS, as all sorts of hidden treasures might be encoded within. Next to the use in email systems to verify domain origin, it has been used to encode all sorts of information about a networked device, be it device properties or topological information.

In corporate networks however, we see different types of information, namely hostnames. In some networks, these hostnames carry identifying information, either about the device or the user of that device.

Previous work has measured the occurrence of these types of names on the global Internet [1]. However, it is unclear as to how this information ends up in the reverse DNS in the first place.

The work performed in the thesis aims to uncover and describe the information flow of personal and device information through a network to the (reverse) DNS. Our approach is to work from the destination to the source: We first investigate the reverse DNS records themselves and work our way through the network layers back to the origin of the information on the end user devices.

Our investigation starts with a case study of the reverse DNS of the University of Twente. We find out the the network is logically separated, and can identify network blocks that are used for a specific function, or are used for a specific location. Moreover, we find network blocks that show dynamic behaviour, that is to say a significant number of changes to the records that are presumed to be automatic. The structure of the names living in these dynamic network blocks hint that certain protocols are involved in providing information to the DNS, namely DHCP and Active Directory.

Subsequently, we perform a literature study into network protocols that might be able to leak user and device information to the network, and detail flows that include propagation to the reverse DNS. For DHCP we detail the specific options that might carry this information, and how a protocol such as DDNS can be used in conjunction with DHCP to set DNS records. We uncovered how AD enrolled devices construct their names, and how they are communicating to the network using DHCP. And finally, we outlined a flow that forwards LLDP information through SNMP by local network equipment.

Thirdly, we investigate networking management appliances that aim to integrate the core network services DNS, DHCP and IPAM. We uncover the role these appliances play in the flows we have seen in our literate study, by performing case studies on four different DDI appliances. From our case studies, we extract com-

mon flow patterns that push information into the DNS. The main culprit for pushing information into the DNS is the DHCP protocol, where several DHCP options are used to construct names. Moreover, we see that all DDI appliances support other information flows to gather identifying information about connected clients, however none of DDI appliances are able to forward information from these flows to the DNS out of the box.

Fourthly, since we identified DHCP as the main source of information for the names we see in the reverse DNS, we investigate what user devices actually leak in practise. We do this by gathering DHCP handshakes from various end user devices such as smart phones, laptops and desktops, and analysing the data encapsulated in their DHCP traffic. We show that nearly all devices in our data set leak information identifying the device or user. IPhones are the best in class, with minimal leakage. The opposite end of the spectrum sees laptops (including Macbooks and Windows devices) providing little privacy enhancing features to their users when it comes to DHCP.

Finally, we explain how these dynamic names in the DNS might be abused, and argue that this is problematic behaviour. We show that this behaviour is present on a significant number of DNS servers: 134,451 /24 prefixes on the Internet show dynamic behaviour, additionally 4% of globally reachable DNS servers return results for private address space that might contain these records. Lastly, we outline mitigation strategies that aim to prevent user and device information ending up in the reverse DNS: Network operators should disable dynamic DNS zones, end users should remove personal identifying information from their device's hostname, and device vendors should implement privacy best practices including MAC address randomisation and anonymity profiles across their products.

Contents

List of acronyms

ASN	Autonomous System Number
CID	Company Identifier
DDI	DNS, DHCP & IPAM
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
FQDN	Fully Qualified Domain Name
IANA	Internet Assigned Numbers Authority
ICANN	Internet Corporation for Assigned Names and Numbers
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPAM	IP Address Management
MAC	Media Access Control
NIC	Network Interface Controller
OUI	Orginazationally Unique Identifier
RFC	Request for Comments

Chapter 1

Introduction

Privacy is a fundamental human right that has been recognised and protected by various legal systems around the world [2]. It refers to the ability of individuals to control the collection, use, and disclosure of their personal information. In today's digital age, where data is constantly being generated, collected, and shared, protecting privacy has become more important than ever.

The leakage of privacy-sensitive information can have serious consequences for individuals, such as identity theft, financial fraud, and reputational damage. Moreover, it can have significant societal implications, such as discrimination, surveillance, and censorship. Therefore, it is essential to develop effective strategies and technologies to minimise the leakage of privacy-sensitive information and to protect individuals' privacy rights.

1.1 Privacy In Networks

The Internet was not designed with security in mind. This is also true for one of the core networking protocols in use today: the Domain Name System (DNS). The DNS is the network service that translates human readable names to machine understandable Internet Protocol (IP) addresses. On the Internet, it is the system that lets you visit your favourite website whenever you enter its domain name (also referred to as URL), in your browser's address bar. The system was designed with speed, reliability and scalability in mind. Notably missing among these goals is security. Over the years, researchers and network engineers have identified several privacy related issues [3].

Most studies into the privacy aspects of the DNS target the transfer of information in the DNS on the protocol level. DNS communicates in plain text over the network. That means that anyone monitoring or forwarding your DNS request can see and modify the messages exchanged [4]. Studies have shown that users can be fingerprinted by the DNS traffic they generate [5]–[8]. While another study demonstrated that DNS traffic is actively being intercepted and rerouted [9].

A less researched area is the privacy aspects of the information stored in the

DNS records. A recent study called 'Saving Brian's Privacy: the Perils of Privacy Exposure through Reverse DNS' highlighted a significant leakage of personal identifiable information through the DNS by tracking automated changes to the reverse DNS [1]. This is the part of the DNS that translates IP addresses to the human readable name identifying the physical machine. They demonstrated that certain parts of the reverse DNS carries personal names, device names, and other information, and that these sections of the DNS exhibit dynamic behaviour. That is to say the names change over time when querying for the same IP address.

The DNS servers of the University of Twente also exhibit this behaviour in parts of its reverse address space. We queried the reverse DNS for the IP addresses belonging to some of our personal devices, and it returned identifying information. Listing 1.1 highlights this leakage.

```
1 > dig -x 130.89.169.47
2 ;; ANSWER SECTION:
3 ... mobielc46e1fb396a9.roaming.utwente.nl.
4
5 > dig -x 130.89.132.165
6 ;; ANSWER SECTION:
7 ... ricks-nord-n10.
```

Listing 1.1: Reverse DNS lookup of a wired and wireless device connected to the network of the University of Twente. The wired device (top) needs to be registered using the MAC address of the network adapter in a portal. The MAC address is now visible in the DNS system. For the wireless device (bottom) we see personal information: An educated guess would lead us to believe the person connected on this address is called Rick, and the device is a mobile phone (OnePlus Nord N10.)

More specifically, this measurement study presented a methodology for identifying dynamic address blocks, a characterisation of the behaviour in these network blocks, as well as a method for fingerprinting and tracking persons through the reverse DNS. The authors hypothesise that the Dynamic Host Configuration Protocol (DHCP) might be one of the culprits that allows information to be exchanged between client devices and the DNS. However an extensive investigation into the role of DHCP or any other potential mechanism has not been conducted.

1.2 Managing Corporate Networks

The dynamic behaviour we observe on the network of the University of Twente points us in the direction of corporate networks. These networks distinguish themselves from other types of networks such as data centres and home networks by its scale and the types of devices that are attached to it. These types of networks may grow

to the scale of data centres in terms of number of devices attached, while at the same needing to support both static servers and dynamic client devices.

To support these devices, typically three core network services need to be provided:

- **IP Address Management (IPAM)** - IPAM is the discipline of managing the IP address space and subnets.
- **Dynamic Host Configuration Protocol (DHCP)** - A network protocol that automatically assigns IP addresses to devices connected to a network. DHCP eliminates the need for manual IP address assignment and ensures that all devices on the network have unique IP addresses. When a device connects to the network, it sends a request to the DHCP server, which assigns an available IP address to the device.
- **Domain Name System (DNS)** - The service that translates human-readable names to network addresses.

These three components are essential for the smooth operation of an organisation's network, and they are all closely interconnected. The industry term for solutions integrating these three services is DNS, DHCP & IPAM (DDI). They provide a centralised interface for managing and synchronising the three parts.

1.3 Goals And Thesis Structure

As hinted above, an investigation into the mechanisms that allow user identifiable to end up in the DNS has not been done. In this thesis we aim to investigate this phenomenon and uncover the root cause(s) of this data leakage. Formally, we aim to answer the following main research question:

- **Main RQ** - How does information flow through corporate networks to the reverse DNS?

Our investigation will lead us through the full flow of this data leakage. We take a reverse approach by starting at the end point of the data flow, and working our way to the source(s) of this information. Each chapter in this thesis will investigate a different part of the information flow.

Before we start looking into the matter at hand, we will provide relevant background information in chapter 2. We will focus on network technologies that are relevant to the later chapters.

Our main investigation is structured in 4 chapters. In chapter 3 we will take a closer look at the reverse DNS of the University of Twente and aim to answer the following research questions:

- **Sub RQ 1 -** What kind of information is present in the reverse DNS of the UT?
- **Sub RQ 2 -** How can we identify patterns in these names?

Using our understanding of a live reverse DNS system, we will aim to identify and investigate protocols that are capable of propagating information through the network in chapter 4. We want to answer the following questions:

- **Sub RQ 3 -** What protocols carry user identifying information?
- **Sub RQ 4 -** How do these protocols facilitate the transfer of this information?

In chapter 5 we will investigate the DDI glue layer that configures and ties the network services together. We investigate what kind of role these systems have in the information flow. We aim to answer these research questions:

- **Sub RQ 5 -** How does DDI software affect the flow of personal information through a corporate network?

In chapter 6 we will have a closer look at one of the protocols that we have identifies as a major source for information leakage, namely DHCP. We investigate how personal devices leak the information by diving into the details of the DHCP protocol. Specifically, we want to answer the following research questions:

- **Sub RQ 6 -** What kind of information do clients leak in their DHCP handshake?
- **Sub RQ 7 -** How are privacy enhancing strategies for DHCP implemented in practise?

Finally, in chapter 7 we will use the knowledge we gained in chapters 4, 5 and 6 to answer our main research question. At that point, the reader might go back to the case study of the reverse DNS of the University of Twente presented in chapter 3 to connect the mechanisms with actual reverse DNS data.

In the same chapter, we will discuss the implications of this privacy leakage, as well as the scale of this problem. Moreover we will discuss the purpose and merits of the reverse DNS, as well as mechanisms for mitigating or minimising the leakage of this data.

Chapter 2

Background

In this chapter we describe a selection of topics that are required to understand the work performed in this thesis. This sections is meant as a general introduction to these topics, and will focus only on the required background relevant to this thesis. For a deeper understanding of these topics, please refer to the references in the text.

In this section we will discuss the following, and the relevance for the following chapters:

- **Internet Protocol (IP) addresses** - Structure and allocation (chapter 3).
- **The Domain Name System (DNS)** - Its purpose, name structure, zones and records, the resolution process, and the reverse DNS. (chapters 3 & 5)
- **Media Access Control (MAC)** - Fundamentals (chapters 3 & 6) and randomisation (chapter 6).
- **Dynamic Host Configuration Protocol (DHCP)** - Its purpose and allocation strategies (chapters 4, 5 & 6), anonymity profiles (chapter 6).
- **IP Address Management (IPAM)** - Intention and usage within DDI (chapter 5).

2.1 The Internet Protocol

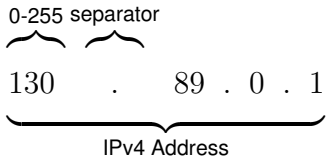
2.1.1 IP Addresses

The Internet Protocol (IP) is one of the most important protocols in computer networks. A key element of the protocol is the IP address. The classical analogy for an IP address is the telephone number. It is the number that identifies where a device is located. There are two version of this protocol commonly in use today, version 4 and version 6.

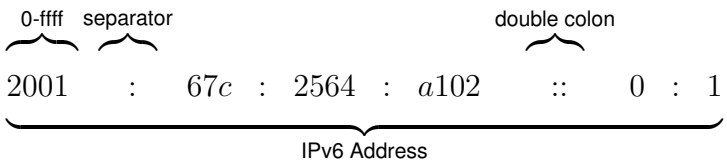
The Internet Protocol version 4 (IPv4) address is defined as a 32 bit number [10]. IPv4 allows us to support 4 billion ($\sim 4 \cdot 10^9$) devices. This number may seem large, however the Internet has seen explosive growth in the past decades. Statista estimates a growth from 3.9 billion Internet users in 2018 to 5.3 billion users in 2023 [11]. Even though this number alone does not paint the full picture of IP address usage, it means we have exhausted all our IPv4 addresses. The regional Internet registries for North America (ARIN), Europe, the Middle East, Russia (RIPE), and Latin America (LACNIC) announced assignment of their last available IPv4 Address blocks in 2015, 2019 and 2020 respectively [12]–[14]. For this reason, an updated version of the Internet Protocol, IPv6, was introduced. IPv6 uses 128 bits addresses instead, which allows for a practically non-depletable number of addresses ($\sim 10^{38}$).

IPv6 deployment on the Internet has been ongoing since the early 2000s, and co-exists with IPv4. The goal is to replace IPv4 fully with IPv6, but the transition to the new protocol has been slow due to measures which aim to extend the lifespan of IPv4, such as Network Address Translation (NAT). To illustrate the adoption of IPv6, Google reports that even in 2023, only 41% of users connect using the new protocol [15].

Numbers in the order of billions and bigger are hard to read when presented in the decimal system, so in practise these addresses are written in a specific way. For IPv4 addresses, the address is written as 4 groups of decimal numbers that range between 0 and 255, also referred to as octets. The octet groups are separated by dots (.), see example 2.1. IPv6 addresses are represented by 8 groups of 4 hexadecimal characters, separated by colons (:). A single hexadecimal character is also referred to as a nibble. To further improve readability, leading zeroes per group are omitted, though any group needs at least one nibble. Furthermore, the longest sequence of groups containing all zeroes are replaced with a double colon (::), see example 2.2.



Example 2.1: Representation of an IPv4 address, consisting of 4 groups of decimal characters in range 0-255.



Example 2.2: Representation of an IPv6 address, consisting of 8 groups of 4 hexadecimal characters. In this example, only 6 groups are visible, the double colon is used to shorten the longest consecutive group of all zeroes.

2.1.2 IP Address Allocation

For routing purposes, network addresses are not handed out in a random fashion. Instead, on the Internet we have seen two different approaches to handing out dividing the IP address space: Classful and classless networks.

Classful Networks

Classful networking is an obsolete addressing strategy historically used for IPv4. In a classful IP network, the address space is divided into five classes: A, B, C, D, and E. Each class has a predefined size and range of IP addresses that can be assigned to it. Classful addressing is based on the first octet of the IP address, which determines the class of the address. For example, an IP address starting with a number between 1 and 126 is a class A address, while an IP address starting with a number between 128 and 191 is a class B address [16].

In a classful IP network, the network portion of an IP address is determined by the class of the address. For example, in a class A network, the first octet is used to identify the network, while the remaining three octets are used to identify the host. This means that the number of available networks and hosts is fixed for each class, which can result in inefficient use of IP addresses.

Table 2.1 lists the classes used historically. One could imagine a significant inefficiency in IP address space when the limits of a smaller class are barely exceeded, and a bigger class is used instead.

Classless Networks

In a classless IP network, the address space is divided into smaller subnets, which can be of any size. This allows for more efficient use of IP addresses and more flexibility in network design. Classless addressing is based on the use of subnet masks, which define the boundary between the network and host portions of an IP address.

In a classless IP network, the network portion of an IP address is not determined by the class of the address, but by the subnet mask. This means that the number of

Class	**Network/Host bits**	**Range**
A	8/24	0.0.0.0 - 127.255.255.255
B	16/16	128.0.0.0 - 191.255.255.255
C	24/8	192.0.0.0 - 223.255.255.255
D (multicast)	undefined	224.0.0.0 - 239.255.255.255
E (reserved)	undefined	240.0.0.0 - 255.255.255.255
Class	**Number of networks**	**Addresses per network**
A	128	16,777,216
B	16,384	65,536
C	2,097,152	256
D (multicast)	undefined	undefined
E (reserved)	undefined	undefined

Table 2.1: Historical division of network classes.

available networks and hosts can vary depending on the subnet mask used, allowing for more efficient use of IP addresses.

For IP addresses, this is achieved by specifying both an IP address, and a number representing the network mask that indicates the number of most significant bits in that address that identify the network. This combination is then written down in Classless Inter-Domain Routing (CIDR) notation [17]. For example, the IPv4 address space represented by *192.168.0.0/24* includes all addresses between *192.168.0.0* and *192.168.0.255* inclusive.

The same addressing scheme works for IPv6. IPv6 has the added constrains that the minimum subnet size for a single host is a /64. This is among other reasons, due to IPv6 protocol features such as stateless address auto-configuration (SLAAC) [18].

2.2 The Domain Name System

The DNS is a naming system used for referring to resources on the Internet; It is the system that translates human readable names to numerical resource locators. It refers to both the database containing the referrals, as well as the protocol used for querying the database [4]. Almost every action on the Internet is preceded by an interaction with the DNS: You use it every time you visit your favourite website, e.g. *utwente.nl* in your web browser, open your favourite app, or email someone, e.g. *someone@utwente.nl*.

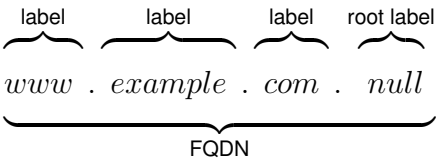
Domain Name Structure

The central entity in the DNS is the Fully Qualified Domain Name (FQDN), often shortened to simply domain name, or domain. This is the ASCII character string we are familiar with when browsing the web, e.g. *www.utwente.nl*. A domain name has two building blocks: The *labels* and the dot separators, and is constructed from right to left.

The *label* is the ASCII text between each dot in the FQDN. They are limited by the character sets of A-Z, a-z, 0-9 and hyphen (-). Even though capitalisation is allowed, the DNS itself is case-insensitive: *utwente.nl* and *UTWENTE.nl* are considered equivalent. Labels with characters that fall outside the allowed ASCII character set might show up in your browser bar, however this is merely a visual representation as the browser decodes a DNS compliant *punycode* representation [19].

A special 'empty' label exists, known as the *root label*. Much like a null terminated string, it is represented by a null byte (*0x0*) As it is considered a label in itself, the separator dot preceding the root label is present at the end of a domain name. In practise however, this dot is often omitted. The root label is of importance in the DNS protocol, as the name suggests it is the root of the domain name from which processing should begin.

Example 2.3 illustrates these concepts.



Example 2.3: Structure of a Fully Qualified Domain Name (FQDN). The root label and the preceding dot is usually omitted from user interfaces.

DNS Hierarchy

The DNS is designed as a hierarchical system. The root of the DNS is managed by the Internet Corporation for Assigned Names and Numbers (ICANN). Together

with the Internet Assigned Numbers Authority (IANA), they delegate the control over Top Level Domains (TLD) to *registries*. IANA for ccTLDs (see below), and ICANN for generic and sponsored TLDs. The registries are the entities that manage the domain space directly under their TLD. Registries are often run by countries or large Internet corporations. Depending on the policy and purpose of the TLD, third parties such as consumers and businesses can register their own names under their preferred TLD. These registrations are known as Second Level Domains (SLD). Some TLDs define a fixed set of SLDs known as public suffixes, and only allow registrations on a lower level. These lower levels can be abbreviated, e.g. Third Level Domain (3LD), though these abbreviations are rarely used.

There are four types of TLD:

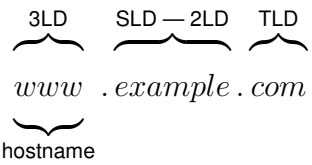
- **gTLD** - Generic TLD, as the name implies used for generic purposes. The core of these TLDs include *.com*, *.org*, *.net*, *.info*. In 2013, ICANN opened the gTLD space for new labels. These include TLDs such as *.top*, *.xyz*, *.shop* as well as city specific TLDs such as *.tokyo*, *.london*, and *.nyc*.

- **ccTLD** - Country Code TLD, reserved for countries, sovereign states or dependent territories. Examples include *.uk* (United Kingdom), *.nl* (the Netherlands), *.de* (Germany), which allow for direct SLD registrations. Other ccTLDs only allow third level registrations. A good example is the *.uk* space which pre 2014 only allowed registrations under *.ac.uk* (acedemia), *.co.uk* (companies), *.gov.uk* (governmental) among others.

- **sTLD** - Sponsored TLD, akin to the gTLD with the difference that the TLD has a sponsor representing a narrow community. Generally enforcing registration to members of the community. Examples are *.int* for international treaty organizations, *.aero* for members of the air transport industry, and *.xxx* for erotic content.

- **ARPA** - Address and Routing Parameter Area (*.arpa*), Infrastructure TLD. This domain was the first TLD, only meant for temporary usage when the Internet community shifted from the legacy ARPANET hostname system to DNS. As reverse DNS entries (see section 2.2) also makes use of this TLD, it is found to be impractical to completely retire it. Nowadays it is used for infrastructure purposes such as reverse DNS, telephone number mapping and a few other infrastructure related uses.

Owners of SLDs (or lower registrations) often create lower levels under their registrations, such as the label *www* for a web server. When a FQDN is used to refer to a specific machine, such as a machine hosting the web server, the leftmost label then corresponds to the local name of said machine. In other cases, the hostname might refer to the whole FQDN. When we use the term hostname in this document, we refer to the former. The concepts discussed in this section are visually represented in Example. 2.4.



Example 2.4: Hierarchy terminology used in FQDN names.

DNS Zones And Records

The main purpose of a domain name is to translate human readable text to computer understandable information. The owner of a domain name has control over the *zone* of their domain. These zones contain resource records to encode information. The DNS specification describes the format of these resource types, though was designed to be flexible enough so new record types can be added [4]. We list a few common resource records, and their purpose:

- **A**/**AAAA** - Used to translate domain names to ipv4/6 addresses.
- **CNAME** - Maps an alias to another name. One can for example, map *www.example.com* and *ftp.example.com* to *example.com*.
- **NS** - Name Server Record. Delegates control over a (lower) DNS zone to an authoritative name server.
- **MX** - Mail Exchange Record. Points to email servers.
- **TXT** - Originally used to store arbitrary human readable data. Nowadays used to store machine readable data such as cryptographic signatures.
- **PTR** - Pointer record. Pointer to a name, used to implement reverse DNS (see section 2.2) among others.

An extensive list of record types, their purpose and associated Request for Comments (RFC) can be found on Wikipedia [20].

DNS Resolution

DNS resolution is the automated process of translating and retrieving a FQDN and record type into its associated information. Since the DNS system is hierarchical and distributed in nature (reflected not only by the structure of a FQDN, but as well as how it is resolved), it is perhaps best understood using an example.

1. When your browser wants to visit the website *www.utwente.nl*, it uses a recursive resolver to find the Internet Protocol (IP) address encoded in an A type record that hosts the website ①.

2. The resolver will try the DNS root to resolve the full FQDN, but it is unlikely that the root will hold this information. It does this by sending an *A www.utwente.nl* query to the root server ②.

3. The root server will not know this information, but does hold information about the *.nl* zone, and will reply with the corresponding NS record for *.nl* ③. Note that NS records themselves might point to a FQDN, and thus the need exists to resolve this domain name as well. The first option is to simply perform an A lookup for this FQDN. However, the DNS protocol allows for an additional records field in the response, so the other option is that the A record of the NS FQDN gets supplied in the same response.

4. Now the name server for *.nl* is known, we ask it to resolve the A record for *www.utwente.nl* ④.

5. This server does not hold this information either, but does know where the name server for *utwente.nl* is located, and replies with the NS (and possibly the corresponding A) record ⑤.

6. This process repeats itself by asking the *utwente.nl* name server for the A record of *www.utwente.nl* ⑥.

7. This name server has the exact matching record, and responds with the requested A record ⑦.

8. The recursive resolver now has the answer the user was looking for. DNS processing stops, and the answer is supplied to the user ⑧.

See Fig 2.1 for a graphical representation. As we have seen in this example, the DNS enables us to resolve any name with only a single requirement: An address of a root DNS server.

Reverse DNS

The DNS is used to translate human readable names to computer understandable data. The reverse, finding the domain name for an IP address, is what we mean by reverse DNS. It requires searching domain name registry tables. It is used by tools such as *traceroute* to find domain names for the hosts in the trace. Although the Internet Engineering Task Force (IETF) recommends a reverse DNS entry for every host, this is not a requirement [21].

Reverse lookups for (ipv4) addresses use the special infrastructure domain *in-addr.arpa*. The reverse DNS entry will encode the common IPv4 dot-decimal notation (see 2.1 in reverse order. For example, the IPv4 address of *www.utwente.nl* is *130.89.3.249*. The reverse entry then becomes *249.3.89.130.in-addr.arpa*. The reason for the reverse notation is historical in nature: The Internet Assigned Numbers

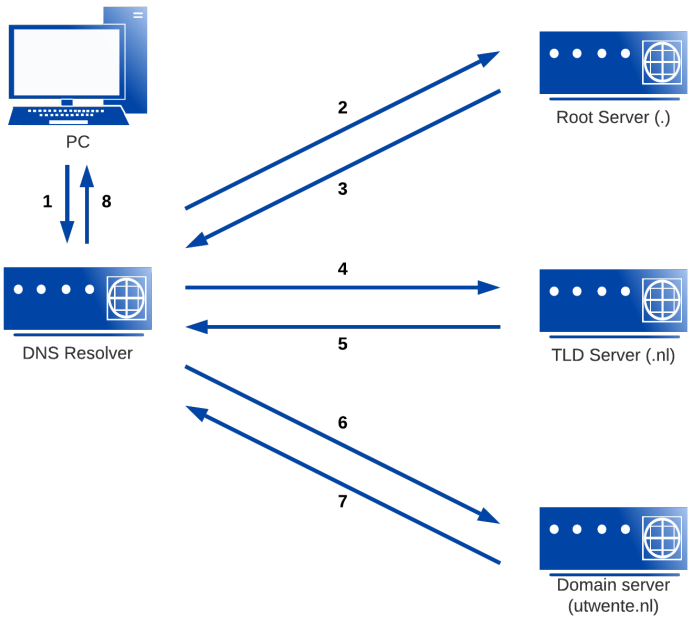


Figure 2.1: Recursive resolving process for *www.utwente.nl*. A computer will ask a DNS resolver to resolve a full domain name. The recursive resolver will then iteratively query the name servers starting from the root servers, till the full name is resolved.

Authority (IANA) assigned blocks divided on the octet boundary, allowing reverse entries for blocks, e.g. *89.130.in-addr.arpa* for the University of Twente owned IPv4 block *130.89.0.0/16*.

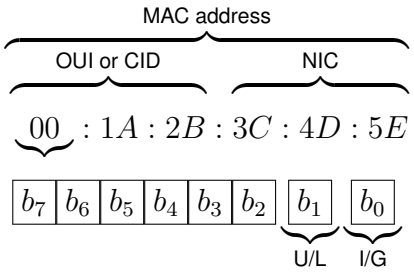
A similar scheme exists for IPv6 addresses using the domain *ip6.arpa* [22]. Recall that IPv6 uses 8 groups of 4 nibbles (hexadecimal characters), that are separated by colons. For the reverse however, we use the full IPv6 address representation (without the shortening strategies) in reverse order, separated by a dot between every nibble. The reverse record for *2001:67c:2564:a102::1:1*, the IPv6 address for *utwente.nl* is *1.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.2.0.1.a.4.6.5.2.c.7.6.0.1.0.0.2.ip6.arpa*.

2.3 Media Access Control Addresses

A Media Access Control address is a unique identifier that represents a network interface. These addresses operate on the data link layer (layer 2), and are used to uniquely identify network devices and ensure that data frames are sent to the correct destination on a local network. Example of layer 2 protocols that utilise MAC addresses include Ethernet and Wi-Fi (802.11) [23].

2.3.1 MAC Address Fundamentals

The MAC address used in IEEE 802 (Wi-Fi, Ethernet, amongst others) is a 48-bit (6-byte) identifier, expressed as a series of 12 hexadecimal digits, typically represented in pairs separated by colons (:) or hyphens (-). The first half of the MAC address (the first 24 bits) identifies the manufacturer of the Network Interface Controller (NIC), technically known as the Orginazationally Unique Identifier (OUI), while the second half (the last 24 bits) is unique to each NIC and identifies the device itself [24]. Example 2.5 details this structure.



Example 2.5: An IEEE 802 (Wi-Fi, Ethernet, amongst others) MAC address is a 48 bit identifier. They are recognisable as six groups of two hexadecimal digits, separated by hyphens, colons, or sometimes without a separator. The first three groups represent the OUI or CID and the remaining three groups represent the NIC. The first and second bit of the first byte of the OUI part have additional functional meaning. b_0 is the unicast/multicast (I/G) bit, and b_1 denotes a universal or local (U/L) address.

Additionally, the first two bits of the first MAC octet have an additional functional purpose [23], [25]:

- **Individual/Group (I/G) bit** - This is the first bit of the first octet of the MAC. If the I/G bit is set to 0, it means that the MAC address is an individual address. Individual addresses are assigned to NICs that are used by a single device, such as a computer or a mobile phone. If the I/G bit is set to 1, it means that the MAC address is a group address. Group addresses are used to communicate with multiple devices simultaneously, such as in multicast or broadcast

communication. Group addresses are not assigned to a specific NIC and are instead used by multiple devices.

- **Universal/Local (U/L) bit** - This is the second bit of the first octet of the MAC. If the U/L bit is set to 0, it means that the MAC address is universally administered. Universally administered addresses are assigned by the Institute of Electrical and Electronics Engineers (IEEE) and are unique to each NIC. These addresses are meant to be globally unique and ensure that there are no conflicts between MAC addresses on different networks. If the U/L bit is set to 1, it means that the MAC address is locally administered. Locally administered addresses are assigned by the network administrator and are not guaranteed to be globally unique. These addresses are often used in private networks where there is no need to ensure global uniqueness.

2.3.2 Randomisation

In 2014, the Canadian news organisation CBC published an article detailing an Wi-FI tracking operation by the Canadian secret services [26]. This has elicited a response both from the IETF community in the form of RFC 7844: *Anonymity Profiles for DHCP clients* [27]. As well as the industry: Apple announced that starting with iOS 8, a random MAC address would be used when scanning for networks [28]. Around the same time, other vendors followed this trend. In this section we aim to provide a bit more background to both Anonymity profiles for DHCP clients, as well as MAC address randomisation mechanisms.

After it became apparent that governments were using Wi-Fi to track devices, even when they are not associated with a Wi-Fi network, vendors started to take measures. One of these measures is MAC address randomisation. Generally, the randomisation strategies started for network probe requests (devices trying to discovery available Wi-Fi networks). However, it soon moved into the higher network layers as randomisation started being applied to associated Wi-Fi networks. Figure 2.2 shows a timeline of the implementation of this measure across different vendors [29]–[31].

2.3.3 Strategies

MAC address randomisation happens on two levels: First the randomisation of the address and the parts of the MAC address. And secondly, the strategy for using mac address over time and across networks.

Randomising the MAC address itself consists out of three parts. The functional bits of the MAC address. Randomization of either the OUI, or CID. And finally the randomisation of the NIC part of the MAC address.

- The functional bits need to be in a specific state for these type of addresses. The I/G bit needs to be set up for unicast traffic, and the U/L bit needs to be set

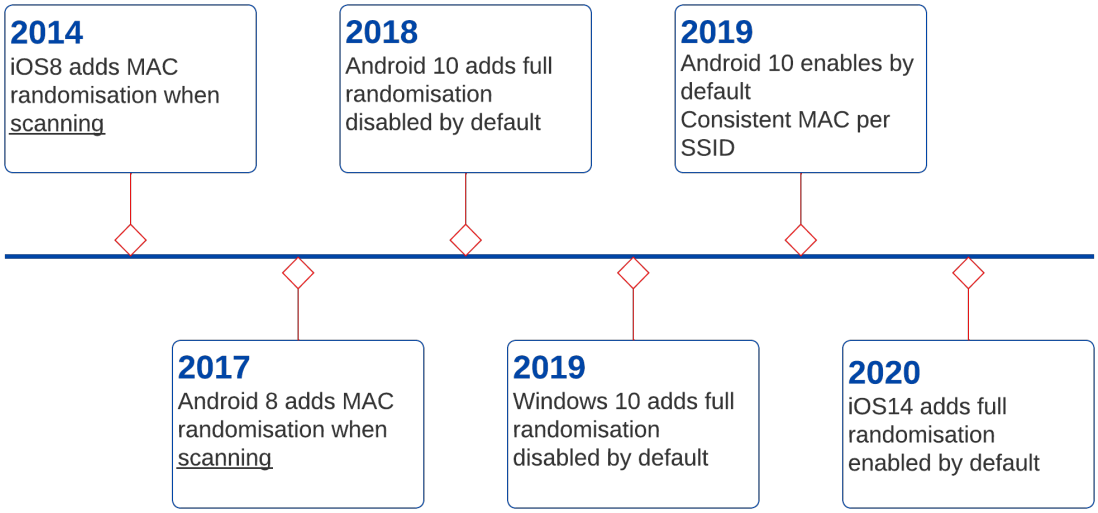


Figure 2.2: Timeline for various vendors implementing MAC address randomisation. We differentiate between two forms: scanning for randomisation when non associated with a network, and full randomisation for associated networks.

to local. The local bit is used to signal to the network that this MAC address might not be globally unique, and should treat it as such. This combination of functional bits has a consequence for the recognisability of randomised MAC addresses, as the consequence of this combination is that a randomised MAC address has the digits 2, 6, A, or E in the second position. Table 2.2 shows the relation between the functional bits and the resulting MAC address. (It should be noted that not all addresses starting with these digits are randomised addresses. They might also be locally administered by network operators.)

- Dependent on the state of the functional bits, we either need to randomise the OUI, or CID:
  - When the U/L bit is set to universal, the first three bytes are an OUI. The OUI is specific to the manufacturer of the NIC, and can not be randomised to random value as that value might belong to another OUI. However, since the OUI is a direct identifier of the manufacturer of the NIC, it is still desirable to apply some technique to minimise leakage. An example strategy is to use the OUI registered to the operating system. Android is a prime example, as in Google's android versions randomised MAC addresses use a Google OUI [32]. It must be noted that this is a trade-off between leaking the manufacturer of the NIC, with leaking the operating system.
  - When the U/L bit is set to local, the first three bytes are a CID. This address space has no further restrictions when it comes to randomisation

other than it is the network administrator's responsibility to guarantee local uniqueness.

- The NIC has no other constraints for randomisation other than ensuring uniqueness within the scope set by the U/L bit.

	Universal (U)	**Local (L)**
Unicast (I)	x**0**:xx:xx:xx x**4**:xx:xx:xx x**8**:xx:xx:xx x**C**:xx:xx:xx	x**2**:xx:xx:xx x**6**:xx:xx:xx x**A**:xx:xx:xx x**E**:xx:xx:xx
Multicast (G)	x**1**:xx:xx:xx x**5**:xx:xx:xx x**9**:xx:xx:xx x**D**:xx:xx:xx	x**3**:xx:xx:xx x**7**:xx:xx:xx x**B**:xx:xx:xx x**F**:xx:xx:xx

Table 2.2: MAC addresses can be sorted in 4 categories depending on the U/L and I/G bits. The category of an address can be immediately recognised based on the second hexadecimal digit. Shaded in blue is the category where randomised addresses reside.

The second consideration is how MAC addresses are randomised over time and networks. There have been prototypes with three options [27]:

- **Per connection** - MAC address is randomised on each connection. On Wi-Fi, this has the major drawback that on MAC address change, the existing Wi-Fi connection needs to be dropped, and re-established. This propagates to higher network levels as well, and as such TCP sessions need to be re-established as well. An observer might be to deduce a client changed MAC address as it sees a host communicating to a set of IP addresses before leaving the network, before seeing another host joining the network communicating with the same set of IP addresses.
- **Per time interval** - Randomisation on a pre-determined time interval. Depending on the time interval chosen, this might encounter the same issues as a per connection based strategy.
- **Per network** - A random MAC address is generated for each separate network. This has the advantage that is becomes harder to identify the same device on different networks.

2.3.4 Randomisation Support Across OS Vendors

We have compiled a list of common end user operating systems, and the implementation status and strategy for randomisation [33]–[35]. Table 2.3 provides an overview of the status quo.

OS	Randomisation support	Enabled by default	Strategy
Android (10)	✓	✓	Network (profile) Time*
Apple iOS (14)	✓(”private Wi-Fi”)	✓	Network (SSID)
Apple MacOS	✗(As of writing)	N/A	N/A
Windows (10 1903)	✓	✗	Network (SSID)

Table 2.3: Support for MAC address randomisation for common end-user operating systems. *Different strategy applied to open networks.

Apple devices running iOS generate a random MAC address per SSID. The address is re-randomised if the device's network settings are reset, on network forget, and when the device has not connected to the network in 6 weeks [33].

Android OS makes a distinction between persistent and non-persistent randomisation. For protected networks, it uses the persistent strategy: The random MAC address is derived from the parameters of the network profile including SSID and security type. This strategy will not re-randomise on network forget or device reset, as it is derived from the network profile. For open networks however, it uses the non-persistent randomisation. This strategy will randomise the MAC address every 24 hours, or 4 hours after DHCP lease expiration. Network 'forget' will cause the device to release any leases associated with the network and cause re-randomisation on reconnect [34].

It must be noted that these are the strategies for the operating system. Vendors that tightly integrate their hardware and software, such as Apple, will have a consistent strategy across their product lines. As we will later see, this is not the case for Android, where different hardware vendors might have altered version of the operating system, or have hardware constraints that disallows MAC address randomisation.

2.4 Dynamic Host Configuration Protocol

The Dynamic Host Configuration Protocol (DHCP) is a network protocol for automated assignment of IP addresses to network clients [36]. In the early days of IP networking, there were usually only a limited number of clients on any given network. Connecting a client to the network would require a simple but manual 'set and forget' of the IP address. In networks that only contain static clients, that is, clients that never move, this was often sufficient. Current networks might still employ this strategy to assign addresses for static clients such as infrastructure and servers. However, this strategy is impractical for various reasons. Users need the knowledge on how to retrieve the MAC address, and how to configure the client and server. Moreover, when you want to support mobile clients that move between networks, e.g. laptops and smartphones. Static assignments would waste a lot of address space, especially when a lot of these clients connect and disconnect over time. The address space can be used more efficiently if the addresses can be allocated and de-allocated dynamically, allowing for re-use across multiple devices. A process that is also more difficult in a static setting.

2.4.1 Information Flow In DHCP

DHCP makes use of a client-server architecture. The DHCP server will store a list of network devices, their assigned IP addresses and other device specific configuration information. An entry in this list is known as a *DHCP lease*. The DHCP protocol supports the exchange of this information and the creation/alteration of these leases.

The basic DHCP flow works as follows: When a client joins a network, it sends out a *DHCPDISCOVER* message to the network. This message can be unicasted to the DHCP server when its address is known, alternatively it can be broadcast to the LAN address, this is the special network address 255.255.255.255. The DHCP server has a set of reserved IP addresses, often called a (DHCP) address pool, or simply pool. When a broadcast is received, the DHCP server temporarily reserves the client an address from the pool. The *DHCPOFFER* is usually unicasted using the requesting client's physical address. Not all clients are able to receive the unicast offer without an IP address, so to avoid deadlock a broadcast might be used instead. If the client accepts the proposed IP address from the offer, it requests the offered address from the server by sending a *DHCPREQUEST*. The server will commit the reserved IP address, and store it alongside with the other configuration information in the lease table. At this point the IP address is only available to this client for the specified lease time. It will notify the client of the acceptance and the duration of its lease by responding with a *DHCPACK* packet. This time graphed flow is represented in 2.3.

On large IP networks, localised relay agents (with optional localised address pools) may be installed, as to optimise the efficiency of the network [37].

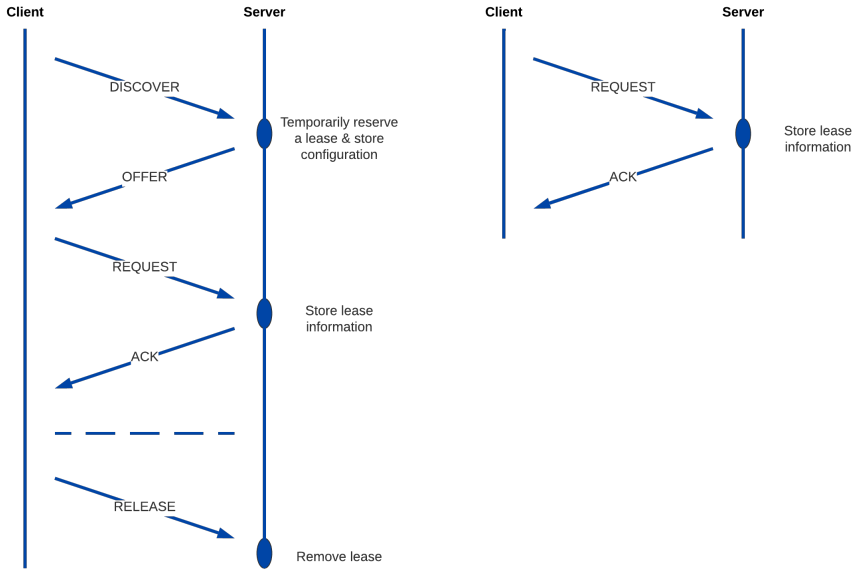


Figure 2.3: The flow of DHCP requests. The left time graph represents requests to unknown networks, whereas the right graph depicts a shortened version for networks that the device has connected to before.

Depending on the implementation of the DHCP server, it might employ various strategies for allocating IP addresses:

- **Dynamic Allocation** - The network administrator reserves a pool of reusable IP addresses. The joining client requests an IP address using the DHCP protocol. The DHCP server will respond with a *lease* for a specific IP address from the pool. The lease is time limited; the DHCP server will reserve that specific IP address for that specific physical address for a specific time period. This way the IP address can be reused for other clients when the lease expires.
- **Automatic Allocation** - Similar to dynamic allocation, with the difference that the goal is to give a requesting client a permanent IP address. This is achieved by setting the lease time to infinite, or the maximum allowable value.
- **Static Allocation** - Also known as DHCP reservation. A network administrator fixes an IP address for a MAC address in the DHCP lease table. The DHCP server will then always provide the same IP address, even before the client has communicated with the DHCP server.

Large organisations with limited IP address space might opt for a dynamic strategy, as to get most of the limited IP space. In a smaller setting where there is plenty IP address space available, such as a private network in a home, it might be desirable that clients can join without prior configuration while retaining their assigned address forever. In such a situation, an automatic strategy might be a better fit.

2.4.2 Anonymity Profiles For DHCP Clients

In section 2.3.2 we have seen MAC address randomisation, a privacy enhancing measure before devices associate with a network [32]. However, the process of connecting to a network often involves a DHCP transaction. The IETF acknowledged that MAC address randomisation in itself is not sufficient when connecting to a network. RFC 7844 explains that even if a DHCP request is not carrying directly identifying information, they might be structured in such a way that they are fingerprintable [27]. In order to improve the privacy of a client, RFC 7844 outlines a set of guidelines to minimise information disclosure and aims to make fingerprinting more difficult. DHCP profiles that implement these guidelines are known as *anonymity profiles*.

Below we present the main areas of concern discussed relevant for this thesis: MAC address randomisation; strategies and impact of randomisation, and Operating System (OS) fingerprinting through DHCP options. The RFC details potential issues regarding radio fingerprinting, however we will omit this as it is related to physical characteristics of the radio transceiver which we consider out of scope.

MAC Address Randomisation In The Context Of DHCP

The first consideration is how MAC addresses correlate to the *client-identifier*. We recall from chapter 4 that the DHCP server uses the client-identifier as a key in the lease table in order to keep track of DHCP clients. When aiming to improve the privacy of DHCP, the obvious consequence is that the the client-identifier, MAC address, and assigned IP address are highly related, and should be randomised in lockstep fashion. Ignoring this relation would present problems: If for example the client-identifier were to remain the same while the MAC address was randomised, then it would be trivial to correlate the randomised MAC address by comparing the client-identifiers. Conversely, if the client-identifier is randomised whilst the MAC address is static, then we can correlate the client-identifier by listening to layer 2 traffic.

The second consideration is how MAC address, and client-identifier randomisation impacts the server side of DHCP, and DHCP management. In chapter 5 we investigated IPAM systems, and have seen that MAC addresses are used for keeping track of (mobile) clients. Randomisation of MAC addresses will break this operational procedure, which is a valid concern for enterprise networks. Moreover, depending on the type of randomisation strategy it could impact the availability of the network: Strategies that dictate that the randomisation interval is shorter than the lease time provided by the DHCP server will result in reserved, but unused IP address space. Which in turn could cause IP address exhaustion.

The guidelines for anonymity profiles intent to solve this problem by deriving the client-identifier (and other DHCP identifiers) from the MAC address.

Operating System Fingerprinting

The DHCP standard has a diverse set of (optional) options that may be communicated in a request. Some options might carry directly finger-printable information. A good example is the *vendor-class* option. As we will later see, most Android phones will put both the operating system name, as well as the version of said operating system in this field. Moreover, since most DHCP options are optional, the selection and the ordering of options can be used to fingerprint the operating system or device [38].

An anonymity profile attempts to solve these issues by minimising the number of DHCP options and the choice of values for said options.

Anonymity Profile

The anonymity profiles give guidelines on two facets of the protocol: First, the options communicated per DHCP message. And secondly, specific guidelines for options that might be problematic.

The options communicated during a DHCP session depends on the specific DHCP message being sent to the server. This is to guarantee DHCP operation whilst attempting to minimise the leakage of private information. The DHCP discover and request messages are the most susceptible to fingerprinting, as according to the DHCP specifications are allowed to communicate a wide range of options [36].

For the discover message, it is suggested that it may only contain the client-identifier and parameter request list. It should not use any other option.

The request message has the same requirements, with the added requirement that it includes the server-identifier for the DHCP server it is responding to (recall that clients that do not hold an IP address will communicate on the broadcast address of the network), and the requested IP address offered by the server. These two additional options are mandatory for this message type. Much like the discover message, it should not use any other option.

Additionally, there are guidelines for these specific options:

- **Requested IP Address Option** - This option allows the client to request a specific IP address. Mandatory in the request message, clients should refrain from using this option in discover messages.
- **Client Hardware Address Field (chaddr)** - This is a mandatory header *field* in the DHCP protocol that contains the MAC address, this field should be synchronised with the randomisation strategy for the MAC and client-identifier.
- **Client Identifier option** - The purpose of the client identifier is to identify the client in a manner that is separate from using the MAC address [39]. This would allow clients to continue using the same IP address when for example the physical adaptor on the same client is changed.

- **Hostname and FQDN extension options** - Extensively discussed in chapter 4. Should obviously never be used.
- **Vendor class options** - There are several vendor identifying options defined for DHCP: Vendor-Specific Information option, Vendor Class option, V-I Vendor class option, and the V-I Vendor Specific Information option [40], [41]. Clients should refrain from using these options as they typically reveal identifying information.

Finally, to avoid fingerprints based on the ordering of the options, the DHCP client should randomise the ordering in each request. If the random ordering cannot be implemented, then the client should aim to order the options by their associated option code number, lowest to highest.

2.5 IP Address Management And DDI

IP address management (IPAM) is the practice of managing and monitoring high-performing IP infrastructure. It is characterised by low latency, high bandwidth capacity, robust security measures, and seamless scalability [42]. Any network with more network infrastructure than the average household, such as multiple routers, switches, and access points can quickly become tedious to manage. The challenge lies in managing the individual devices and network equipment without causing conflicting configurations.

IPAM aims to solve these problems. It involves a range of tasks including IP address assignment, DNS and DHCP management, subnet management, IP address allocation tracking, and IP address inventory management. It allows network administrators to automate and centralise the management of IP addresses, reducing the likelihood of conflicts or overlapping IP addresses, which can cause network outages or other problems.

IPAM tools often provide features such as IP address discovery, IP address request and approval workflows, IP address reservations, and reporting capabilities. The goal of IPAM is to ensure that IP addresses are used efficiently and effectively, and that network resources are managed in a way that is both secure and scalable [42].

2.5.1 IPAM And DDI Terminology

IPAM is tightly coupled to management of the DNS and DHCP. The industry uses another term to refer to essentially the same concept: DDI, which stands for DNS, DHCP and IPAM. As we will later see, vendors of DDI appliances use these terms interchangeably. For better accuracy, in the rest of this document we will use the term IPAM to only refer to the IP address management discipline. Whereas we will use the term DDI to refer to the integration of DNS and DHCP with IPAM, as well as to refer to the soft and hardware appliances to manage these three components.

2.5.2 DDI Appliances

DDI appliances often only supply one core function itself, the IPAM part. DNS and DHCP might be integrated into the DDI appliances, however typically it does not provide this functionality itself. Instead, its role is to aggregate and manage information for these components from a centralised place. Example vendors of these software packages are SolarWinds[1], BlueCat[2], Infoblox[3], and EfficientIP[4].

[1] https://www.solarwinds.com/ip-address-manager
[2] https://bluecatnetworks.com/adaptive-dns/ddi/
[3] https://www.infoblox.com/products/ddi/
[4] https://www.efficientip.com/

In the remainder of this section we highlight the purpose and key challenges of the three DDI components.

IP Addresses

An IPAM must include the planning and tracking of the IP address space under its control. This includes several facets:

- Tracking of subnets and its function within the network. Networks might be split for various reasons. For example, you might have different subnets for workstations, servers or specific applications such as VOIP phones. Or to provide differing levels of security, providing a more secure (restricted) subnet with access to sensitive services.
- Tracking of IP address usage within the subnets. This is especially important for subnet with dynamic clients. Shortage of available IP addresses will essentially deny clients network access.
- Define a cohesive allocation plan that promotes route summarisation (maintaining a limited number of announced routes to minimise stress on the routing infrastructure).

DHCP

DHCP is a core component in executing the IPAM strategy. The address allocation plan is configured in the IPAM section of the DDI appliance. The appliance's function is to automatically translate this strategy to the DHCP servers and relays in the network. Moreover, it needs to report back on the usage information of the subnets under its control.

DNS

The names of devices and policies of the DNS might be dependent on the IPAM strategy. For example, your naming and exposure of DNS names might differ between publicly routable parts of the network versus parts of the network only intended for internal use. The DDI appliance can enforce policies and structure to these names from a centralised place. Additionally, the DNS provides naming information within the IPAM environment, allowing a network administrator to more easily identify what and where devices live in a subnet.

Chapter 3

A Case Study Of The UT Reverse DNS

The reverse DNS is a fascinating part of the DNS, as all sorts of hidden treasures might be encoded within. In the introduction we have seen interesting names popping up in the reverse DNS space of the University of Twente. At first glance, these names seem to have a different structure, and seem to serve a different function when compared to reverse names we see elsewhere, for example in (backbone) routing infrastructure.

The names in the introduction are gathered from our own devices, and seem to hold information that is specific to our person, or devices. This is an interesting phenomenon, as it is not immediately clear how the DNS has gotten hold of this information. This leads us to wonder whether the DNS also holds information for other hosts and persons. Moreover, we question what other types of names might be present in these types of networks.

In order to gain a better understanding of what kind of information the reverse DNS of a corporate network might hold, we dive deeper into the names and patterns present in the reverse DNS space of the UT.

The aim of this chapter is to create a starting point for further research by answering the following research questions:

- **Sub RQ 1 -** What kind of information is present in the reverse DNS of the UT?
- **Sub RQ 2 -** How can we identify patterns in these names?

In order to answer these questions, we take a two step approach: First we will look at the reverse DNS without limiting ourselves to the enterprise environment. We will see that significant work has been done in extracting and using all sorts of information in the reverse DNS. These techniques might provide useful insight in the names we might encounter. Secondly, we will perform measurements on the reverse DNS of the UT, and analyse the results in more depth.

3.1 Reverse DNS Outside Corporate Networks

In chapter 2.2 we briefly mentioned the usage of the reverse DNS to provide names to the IP addresses in a *traceroute*. Example 3.1 shows a *traceroute* from a host in the University of Twente in the Netherlands, to a famous paper selling company in Scranton Pennsylvania, USA. Most will recognize information like (abbreviated) city or country names. Individuals more familiar with the networking space might spot more: IP addresses, ISP names, or be able to identify other networking properties.

```
                    My traceroute [v0.95]

me (130.89.15.42) → dundermifflin.com
 Hop  Host                              Ping Avg
   1. cr-ewi.routing.utwente.nl           0.4
   2. 130.89.254.201                      0.4
   3. e0-0-3-0.es001b-jnx-01.surf.net     0.6
   4. ae20.zl001a-jnx-01.surf.net         3.0
   5. (waiting for reply)
   6. surfnet-gw.ip4.gtt.net              3.0
   7. ae21.cr6-ams1.ip4.gtt.net           3.0
   8. ae17.cr6-lax2.ip4.gtt.net         132.6
   9. ip4.gtt.net                       133.7
  10. (waiting for reply)
```

Example 3.1: A traceroute to the Dunder Mifflin paper company using the tool *mtr*. The names under host are pulled from the reverse DNS. We can recognize city/airport names, e.g. 'ams' for Amsterdam, or 'lax' for Los Angeles. The 'ip4' label seems to indicate that (part of) this connection is making use of IPv4. We also spot ISP names (Surf, GTT), and can identify a likely peering/edge router in hop 6.

This has inspired a body of work that leverages the reverse DNS to extract all sorts of interesting information. They have been used to create maps of the Internet, both from a topological [43], Autonomous System Number (ASN) interconnections [44], [45] and geolocational perspectives [46]–[48]. As well as physical device properties [49], or residential subscriber line types [50]. Or to simply extract a device's interface names [44], or hostnames [51]. The cited work often focuses on a specific type of network(s). One notable network type not found in literature are Enterprise networks, which we will discuss in more detail below.

3.2 Reverse DNS Treasures In Corporate Networks

We shift our focus to enterprise networks. We define these networks to include corporate networks, university networks and the like. It most notably excludes (backbone) routing, ISPs (residential), and data centre infrastructure. We expect to see different types of names in the reverse DNS here, as most hosts in these types of networks will likely be user terminals (personal computers, laptops, tablets, phones, etc.) and other corporate devices like printers and network attached storage.

To gain some inspiration into what we might find, we go treasure hunting on the University of Twente's reverse DNS space. Listing 3.2 implements a simple script that gathers all reverse DNS records for all IP addresses in the UT's *130.89.0.0/16* block, using the *dig*[1] command line utility and parsing tools.

We inspect the results obtained from running the script, to see whether we can find some (hints to) structure in the reverse DNS of a single organization.

Table 3.1 lists some high level statistics for our measurement. Upon further inspection of the reverses we can extrapolate more network properties.

Subnet	Subnet size	Single reverse records	Multiple reverse records
130.89.0.0/16	65,536*	29,226 (44.6%)	69 (0.1%)

Table 3.1: High level statistics for our measurement of the University of Twente reverse address space. *The actual number of hosts is lower due the subnet being split in smaller subnets, and hence removing available addresses for functional purposes such as network and broadcast addresses.

3.2.1 Subnetting

We see some structured reverse names ending in *routing.utwente.nl*, these often appear at the start of logical network blocks, e.g. at the start of a /24 or /25 block. We notice a pattern in the labels preceding *routing.utwente.nl*: We see both labels that might indicate a *location*, or a *function*.

Physical Location

To investigate the location labels further, we need to find some more information about the actual network blocks announced by the routers. This will allow us to correlate the *routing.utwente.nl* names we see at IP block boundaries. We can get this information from a machine inside the network. By inspecting the networking configuration of this machine, we will not only get its assigned IP address, but also the

[1] Dig is part of the ISC BIND DNS suite, see: `https://www.isc.org/bind/`

```
1  #!/bin/bash --posix
2
3  #Iterate over 3th address octet
4  for i in {0..255}
5  do
6    #Iterate over 4th address octet
7    for j in {0..255}
8    do
9      #The IPv4 address for this lookup
10     IPV4=`echo 130.89.$i.$j`
11
12     #Fetch the reverse DNS of IPV4 from UT DNS
13     PTR=`dig @130.89.1.2 -x $IPV4 +noall +answer +multiline
14                  | awk 'NF{ print $NF }'`
15     DATE=`date --iso-8601=seconds`
16
17     #IP might hold multiple reverses, iterate over them
18     COUNT=0
19     IFS=$'\n'
20     while IFS= read -r line; do
21       #Write to stdout with format
22       echo -e "$DATE,$IPV4,$COUNT,$line"
23       ((COUNT+=1))
24     done <<<  "$PTR"
25
26     #Introduce delay to not overload the DNS server
27     sleep 0.1
28   done
29 done
```

Listing 3.1: Bash script that iterates over the University of Twente IPv4 address space (130.89.0.0/16), requests reverse DNS records associated, and writes to output.

network block the IP address is a part of. To inspect the IP address information of a machine connected via ethernet we use the command line utility *ip addr*. The output of the command tells us this machine lives on the *130.89.12.0/22* subnet. When we inspect the reverse DNS records in this block, we see the familiar *routing.utwente.nl* names, with a preceding faculty name label at the start of the */22* network block. The remaining addresses in this block contains reverses that hint at them belonging to the faculty or organisation being housed in that building. Table 3.2 shows a sample of the reverse records in this subnet. Similar patterns can be spotted in different parts of the UT IP address space. This hints at the fact that a part of the network is logically separated following the housing policy of the organisation.

IP address	Reverse Name
130.89.12.1	if-ewi.routing.utwente.nl.
130.89.12.2	hp-zi.routing.utwente.nl.
130.89.12.4	cr-ewi.routing.utwente.nl.
130.89.12.5	br-ewi.routing.utwente.nl.
130.89.12.112	printer235.ewi.utwente.nl.
130.89.13.50	utwks10207.ewi.utwente.nl.
”	utwks10207.ad.utwente.nl.
130.89.14.90	utwks10872.ewi.utwente.nl.
130.89.15.15	mobield89d67c89d7c.roaming.utwente.nl.

Table 3.2: Sample reverse DNS records from the University of Twente *130.89.12.0/22* subnet. The labels *ewi* and *zi* are abbreviations used for the faculty name and building respectively. Note that one IP address can have multiple reverses.

Function

We do the same for our wireless interface, and find that our devices lives on the *130.89.128.0/21* subnet. This time we see a *routing.utwente.nl* name containing a *wlan* (Wireless LAN) label. The remaining reverses in this subnet are single label names, and do not seem to extract some higher level structure such as locality. Table 3.3 shows a sample of reverses in this block.

3.2.2 Low Level Structure

When inspecting the left most label of the reverse names, we can extract some more structure on device level. We make a distinction between single label names where leftmost equals the single label, and the leftmost labels that are present in multi-label names.

IP address	Reverse Name
130.89.128.1	if-wlan.routing.utwente.nl.
130.89.128.4	cr-wlan.routing.utwente.nl.
130.89.128.5	br-wlan.routing.utwente.nl.
130.89.128.44	ut171973.
130.89.129.66	air-von-alice.
130.89.134.180	s21-ultra-van-bob.

Table 3.3: Sample reverse DNS records from the University of Twente *130.89.128.0/21* subnet. The *wlan* label indicates wireless clients. We see single label hostnames in this subnet. These names are likely device host names. We can extract some personal identifiable information from these names: Device make and model (Apple Macbook air/Samsung s21 ultra), device owners (Alice, Bob) and language (German: von, Dutch: van).

Single Label

We mostly find the single label reverses in the *wlan* subnet. Table 3.3 shows a sample of the reverses in this subnet. Based on experience with our own devices, we can conclude that (most of) these names are device hostnames. We will substantiate this claim in section 4.1.4. Although these names are unlikely to carry network information, it does seem to carry information that might identify persons or device properties, such as device manufacturer and model, name of the owner, as well as device language.

Multi Label

In multi label names, we find different types of names in the leftmost label. We see labels that either encode system names and/or hostnames, and names that encode network properties.

Wired desktop machines in the UT carry stickers with a number on it. These stickers are in the form of *utXXXXX* and *utwksXXXXX*, where the X's represent decimal numbers. We see these names reflected as leftmost label in some reverses, see table 3.2. A reasonable assumption to make would be that this might simply be the actual hostname of the device. However, some IP addresses that carry these sticker names have multiple reverses. One reverse carrying faculty information, while the second one carries the label *'ad'*. This is likely an abbreviation for Active Directory, a popular directory service used to manage corporate devices. As we will later see in section 4.2, we believe that Active Directory is supplying these names to the DNS.

The University of Twente also allows users to connect their own equipment to the network using a wired ethernet connection. Before the equipment is allowed network access, the MAC address of the adapter needs to be registered in a portal. These

MAC addresses are reflected in the reverse DNS by encoding the MAC address in the hostname label, see table 3.2.

3.2.3 Multiple Reverses

In our measurement we have seen hosts with two reverses, see table 3.2. From a technical perspective this is a strange occurrence. Although multiple PTR records are legal within the specification [4], it is highly discouraged. One major reason is that the order of returned values when querying the DNS is undefined [21], [52]. Services and protocols that rely on accurate DNS information to function properly (e.g. mail and spam filtering) might malfunction as not all reverses might be used by the software.

Even though this is likely a configuration error on the network administrators part, it does help us understand where these names originate. Upon inspection of the reverses, as seen in 3.2, we see that all IP addresses with multiple reverses follow the same pattern. They are multi-label reverses with the same labels, except the third label (from root). In one reverse the third label hints at a department name, whereas the other is the fixed string *ad*. This string hints that the source of these reverse duplicates originate from Microsoft Active Directory, as *ad* is a commonly used abbreviation.

3.2.4 Blanket Reverses

We mentioned earlier that not all IP addresses have a reverse name associated with it. For example, the network blocks in tables 3.2 and 3.3 are sparsely populated with reverse names. However, a part of the reverse DNS space stands out due to 'dense' blocks of uniform looking names. More specifically, we see blocks where every address has a reverse record, and all reverse records follow the exact same structure. Table 3.4 lists a few of these blocks.

The 'blanket' reverses in these blocks all follow a similar structure: They are multi label names where labels indicate a function for this network block. Additionally, the leftmost label also encodes the last octet(s) of the IP address the reverse belongs to. For example, the reverse *wlan**176086**.mobiel.utwente.nl* belongs to the IP address *130.89.**176.86***.

The dense nature of these names, and the fact that the names only house static information (network block function and associated IP address) leads us to believe that these names were statically set, and will not change unless manually changed by a network administrator.

IP address	Reverse Name
130.89.96.171	dvpn096171.vpn.utwente.nl.
130.89.96.172	dvpn096172.vpn.utwente.nl.
130.89.96.173	dvpn096173.vpn.utwente.nl.
130.89.176.86	wlan176086.mobiel.utwente.nl.
130.89.176.87	wlan176087.mobiel.utwente.nl.
130.89.176.88	wlan176088.mobiel.utwente.nl.
130.89.196.207	ce207.ewi.utwente.nl.
130.89.196.208	ce208.ewi.utwente.nl.
130.89.196.209	ce209.ewi.utwente.nl.

Table 3.4: Sample reverse DNS records from various Univesity of Twente subnets. The labels seem to indicate that the addresses are reserved for a purpose, e.g. VPN and wireless clients. Moreover, the reverses carry the last octets (including leading zeroes) of the IP address they belong to.

3.2.5 Dynamic clients

In this rough analysis, we have seen some names that might indicate that a fair share of the network is being used for dynamic clients, for example clients that disappear from the network outside working hours, or move between access points. We draw this educated guess from keywords we have seen in structured names such as *mobiel* (English: mobile), *roaming*, and *wlan* as well as names that are likely to be mobile device hostnames.

For this category of clients we aim to better illustrate that these names indeed represent dynamic clients. We therefor perform a second measurement roughly a week later and look for differences in reverse records between the two measurements. Although chosen arbitrarily, this time frame is a fair estimate to show automatic changes to the reverse DNS, while manual changes made by network administrators will likely be in the minority (or even absent).

The second measurement shows that of 65.536 IP addresses, 2.559 (3.9%) have a different reverse record. 2.474 unique reverse records were involved in these changes. Of the 2.474 unique reverses, 1.866 records only had a single label as reverse record, and 585 records belonged to a naming structure indicating mobile (mobiel, roaming) clients. It is also interesting to note that 442 reverse names were seen before, but have moved to a different IP address.

3.3 What We Learned So Far

In this chapter we have extracted various network aspects from the reverse DNS space of the University of Twente. We see that the reverse DNS holds a treasure trove of information. Much like previous academic work, we see topological informa-

tion of the network. However we also see reverse names that directly identify device make and model, as well as personal names.

Moreover, our two measurements show that the reverse address space displays dynamic behaviour when it comes to the names that carry device and user information. This dynamic behaviour are likely automated changes performed by other protocols.

Our investigation into the structure of the reverse names gives us hints as to what protocols are responsible for leaking information. For example, the plain hostnames we see point to DHCP as a source, while the the IP addresses that carry multiple reverse records are likely the result of Active Directory enrolled devices.

The existence of dynamic name records constitutes a significant leak of privacy sensitive information. It warrants further research into how this information is leaked into the DNS. In the next chapter we will investigate these protocols in more detail.

Chapter 4

Information Flow Through Corporate Networks

In the previous chapter we have speculated about sources of reverse names in the University of Twente network space. These included manually set 'static' blanket names, as well as seemingly dynamic names flowing from the end user devices, or other networked services. In this chapter we aim to investigate the protocols that might facilitate the flow of information.

Specifically we aim to answer the following research questions:

- **Sub RQ 3 -** - What protocols carry user identifying information?
- **Sub RQ 4 -** - How do these protocols facilitate the transfer of this information?

To answer these questions, we draw from previous chapters and have a deeper dive into DHCP and Active Directory. Additionally, we will have a loot at networking protocols we suspect might also communicate this type of information such as the Simple Network Management Protocol (SNMP), and Link Layer Discovery Protocol (LLDP).

4.1 Dynamic Host Configuration Protocol

In table 3.3 we have seen device's hostnames in the reverse DNS. This sort of information could be communicated by a network client in an automated fashion. Section 2.4 gives an introduction and background information to DHCP. In this section we will investigate the information exchanged by the DHCP protocol.

There are several fields in the protocol that would allow a client to communicate personal, and device information to the server. We will have a look at three of them: the Client-Identifier option, the Hostname option, and the FQDN extension options.

4.1.1 Client-Identifier Option

As mentioned in section 2.4, the role of the DHCP server is to maintain a list of leases. The model used for the persistent storage in DHCP is a key/value store, where the key is some unique value used to identify a device, and the value is used to store the configuration for the device. This configuration includes the assigned IP address and the lease time at minimum, but often includes more information.

The DHCP protocol defines two possibilities for the key. The first straightforward option is the tuple (IP subnet, hardware-address). Alternatively, the key might be the pair (IP subnet, hostname). The latter option allows the DHCP server to identify the client and assign the same IP address when the client for example moves between subnets, or changes the interface (and with it the MAC-address). The protocol defines (IP subnet, hardware-address) as the default. The alternative (IP subnet, hostname) is only used when the client requests it by providing the 'client-identifier' option when communicating with the server. It should be noted that even though the DHCP RFC (RFC 2131) uses the terminology (IP Subnet, hostname), the RFC containing the specification for DHCP Options (RFC 1533) imposes no limitations on legal values and thus can be set to anything [36], [40].

4.1.2 Hostname Option

The DHCP protocol also allows for explicitly passing the client's hostname in the aptly named 'host name' option. It must be noted that this value is not necessarily the same as the value defined in the Client-Identifier option.

4.1.3 FQDN Extensions

In section 2.2 we outlined the definition of the Fully Qualified Domain Name (FQDN). In the context of DHCP, FQDN extensions refer to the ability of a DHCP client to provide its FQDN to the DHCP server as part of the lease negotiation process.

When a DHCP client requests an IP address lease from a DHCP server, it can also send its FQDN to the server as part of the DHCP request. The DHCP server can then use this information to update its DNS (Domain Name System) records, associating the IP address with the FQDN.

According to the RFC outlining the specification for this option, it is intended to operate in the following two cases [53]:

1. DHCP client updates the forward record, DHCP server updates the reverse record.

2. DHCP server updates both the forward and the reverse records.

The way this is achieved in the protocol is providing a single bit in which the client can request whether the server should update the forward record. The responsibility

for the reverse record is always deemed to be with the DHCP server, and a reverse should be set when a forward is requested. However, there is another bit that is responsible for requesting the server to perform no DNS updates at all, both forward and reverse. Along with these bits, the client can send the names for both the forward and the reverse along with the request.

From these options we can conclude a possible third case: The client might request that no DNS updates should be made. However, the RFC assumes that this is an unlikely scenario. The RFC assumes that by including the FQDN option, the client requests that DNS records will be set. Requesting that the server will make no changes to the DNS implies that the client will. In most scenarios, this is deemed problematic as the authority over the IP address lies with the DHCP server, and is thus the only logical entity to update the PTR record [53]. The RFC specifies that the server 'should not' update the DNS. This terminology in RFC documents means that there may exist valid reasons in particular circumstances when the particular behaviour is acceptable or even useful [54]. And thus a DHCP server might ignore the request to perform no DNS updates. We argue that this scenario might be used to explicitly inform a DHCP server to never set a DNS record, while at the same time having no intention of setting records itself.

4.1.4 Demonstrating Information Flow From DHCP To DNS

When we lookup the reverse of one of our device, we see its hostname in the DNS. We assume that this hostname is pulled from the DHCP request when joining the network. A network capture of DHCP packets does indeed confirm that our hostname is communicated to the DHCP server using the DHCP hostname option. Listing 4.1.4 lists the DHCP configuration file for this machine. Line 4 indicates that indeed the hostname is pulled from the operating system, and sent along with the DHCP request.

```
1  # Configuration file for /sbin/dhclient.
2
3  ...
4  send host-name = gethostname();
5  request subnet-mask, broadcast-address, time-offset, routers,
6    domain-name, domain-name-servers, domain-search,
7    host-name,dhcp6.name-servers, dhcp6.domain-search,
8    dhcp6.fqdn, dhcp6.sntp-servers, netbios-name-servers,
9    netbios-scope, interface-mtu,
10   rfc3442-classless-static-routes, ntp-servers;
11 ...
```

Listing 4.1: Standard DHCP configuration file on a Ubuntu 22.04 machine. Line 4 shows that the host name is pulled from the OS, and sent along in the

DHCP request.

We now try to add the DHCP options discussed above, and set them to easily recognisable custom values. We try all different permutations of enables and disabled. By manually forcing a DHCP release, and a subsequent new request, we can confirm that the name in the reverse DNS is indeed pulled from the host-name option. Listing 4.1.4 shows the new configuration.

```
1  # Configuration file for /sbin/dhclient.
2
3  ...
4  send host-name "my-custom-hostname";
5  send dhcp-client-identifier "my-custom-cid";
6  request subnet-mask, broadcast-address, time-offset, routers,
7    domain-name, domain-name-servers, domain-search,
8    host-name,dhcp6.name-servers, dhcp6.domain-search,
9    dhcp6.fqdn, dhcp6.sntp-servers, netbios-name-servers,
10   netbios-scope, interface-mtu,
11   rfc3442-classless-static-routes, ntp-servers;
12 ...
```

Listing 4.2: Edited DHCP configuration file. Lines 4 & 5 allow us to set specific DHCP options.

When we force a DHCP release, the reverse name is not immediately removed from the DNS, but has a slight delay of anywhere between 10 to 20 seconds when directly querying the authoritative DNS server. It is not immediately clear where and how this delay is introduced. We can however conclude two particularities that are visible from the client perspective:

1. The TTL (time-to-live) for the reverse record is not the cause for the removal. The procedure where the record is removed can be completed within the TTL (600 seconds) of the record pushed into the DNS on DHCP request.

2. We also note that the DNS zone transfer mechanism does not play a role here. This is a mechanism to share the DNS zone file from the primary DNS server to the secondary DNS servers. A DNS zone transfer is triggered when the serial number in the primary's SOA record increases. We see no such behaviour between DHCP request and release.

4.2 Active Directory

Microsoft Windows Active Directory (AD) is a directory service that stores information about objects in networks. Objects is used in the broadest sense of the word: it includes tangible objects, such as devices. But also abstract concepts defined

in software, such as users, groups, domains, and more. It is a software platform for organizations to manage (shared) access to Windows devices on the basis of users accounts. A prime example in the university setting is the desktop computers connected to the projectors in lecture halls: These machines are managed by the organization, yet anyone with an authenticated university account can log in and use the machine.

In the previous chapter we have seen the string *ad* in a reverse name, see table 3.2. To confirm that this reverse indeed is connected to the AD infrastructure, we investigated a machine that is known to be part of the University's AD environment.

4.2.1 The Active Directory Domain Tree

An Active Directory service typically mimics the hierarchical structure of the organization. To best understand how an organization is translated to Active Directory, we look at an example: We take the University of Twente as an organization. The university has multiple faculties with housing its own staff and devices. In Active Directory, the university would be represented by a *Domain*, e.g. *utwente.nl*. A faculty can be represented as an *Organizational Unit (OU)* in the domain. This is simply an object inside the domain, that holds other objects. To provide further structure, the faculty OU holds nested OUs named *employees* and *computers*, which in turn hold the actual user and computer objects.

4.2.2 Data flow from AD managed devices

For Windows based devices, when creating, updating or deleting records in the DNS, Active Directory is typically not directly communicating with the DNS server. Instead, the DNS names are communicated from the device through DHCP in a similar manner as described in the section detailing DHCP.

From the way the DNS is updated on these hosts, we can conclude that all required information must be present on the device. The structure of the domain name as we see it in the DNS is constructed from two parts. First, the host name of the device, or in Windows terminology *computer name* is configured on the device. And secondly, the AD domain of the device needs to be set on the device locally. This is a manual one-time process. The Windows DHCP client service uses these two pieces of information to construct a DHCP request including FQDN extensions for this device in the form of: $\{$*hostname*$\}.\{$*AD domain*$\}$ [55]. The time graph for this flow in depicted in figure 4.1.

4.3 Simple Network Management Protocol

SNMP (Simple Network Management Protocol) is a protocol used for network management and monitoring. It is an application layer protocol used to manage and

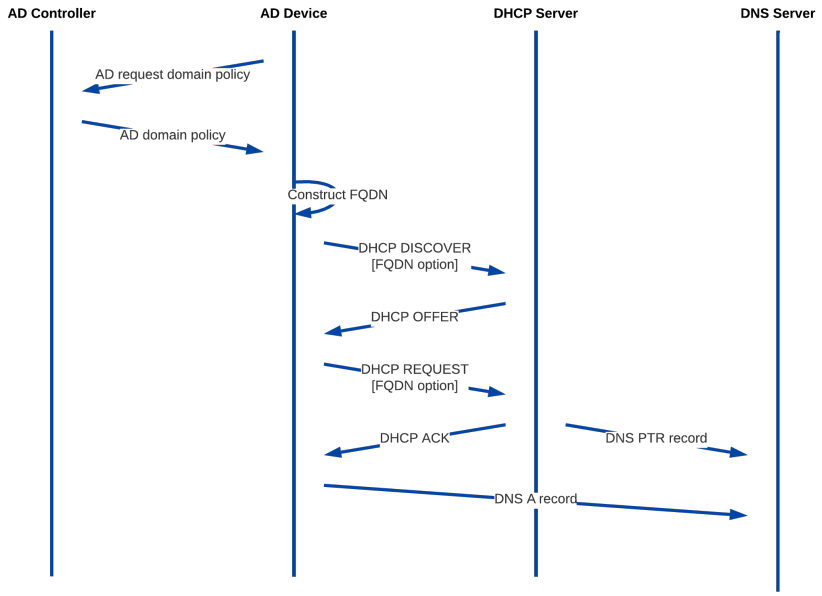


Figure 4.1: The flow of information from the AD server to the DNS.

monitor network devices, such as routers, switches, servers, printers, and other devices [56].

SNMP enables administrators to monitor network performance, detect network faults, and manage network devices. It provides a standardized way for network devices to communicate with network management systems, allowing administrators to access information such as device configuration, performance metrics, and status information.

SNMP is based on a client-server model, where network devices act as servers that respond to requests from SNMP managers. SNMP managers send requests to network devices to retrieve information, set configurations, or perform other tasks. SNMP uses a set of standardized messages, called protocol data units (PDUs), to communicate between SNMP managers and devices.

4.3.1 Management Information Bases

A Management Information Bases (MIB) is a hierarchical database that contains information about the objects managed by SNMP. It defines the structure and content of the data that can be accessed via SNMP from a network device.

Each object in the MIB has a unique identifier, called an Object Identifier (OID), that is used to identify and access the object. The MIB contains a list of OIDs and their associated information, such as data type, access permissions, and description.

The MIB is organized in a tree-like structure, similar to a file system, with each node in the tree representing an object. The root of the tree is represented by the

OID '.iso.org.dod.internet', which is followed by a series of sub-nodes that represent different organizations and standards bodies. Each organization or standards body has its own subtree within the MIB, which can be further divided into sub-nodes representing different objects.

4.3.2 Device Information

We are interested in the device hostname or other MIB fields that might carry information that might end up in the DNS. A common OID location for this information is defined in the 'sysName' (system name) field of the SNMP MIB [57]. This field is intended to be equal to the device's FQDN by convention.

It must be noted that there could be more MIBs specifying this type of information. Especially since MIBs can be extended by parties to carry new information. In practise we see vendors such as Cisco defining their own MIBs to manage their network appliances.

4.4 Link Layer Discovery Protocol

LLDP (Link Layer Discovery Protocol), is a vendor-neutral layer 2 protocol used by networking devices to advertise and discover information about other directly connected devices (neighbours) on a network. It is formally defined in 802.1AB, named 'Station and Media Access Control Connectivity Discovery' by the IEEE [58]. It enables devices to discover each other's capabilities, such as device type, port number, VLAN ID, MAC address, and system name [59]. Historically, network vendors have defined version of LLDP for their own equipment, such as the Cisco Discovery Protocol (CDP [60]). However, LLDP was intended to operate independently of any particular vendor or protocol, making it inter-operable with different network devices from various vendors.

4.4.1 Procotol Flow

A device with LLDP enabled will send out Ethernet frames containing LLDP Data Units (LLDPDU) on all connected interfaces periodically, typically every 30 seconds. These Ethernet frames are addressed at a special multicast MAC address that switches will not forward. These LLDPDU contain some mandatory information, such as port ID, Time To Live, and Chassis ID. This Chassis ID is a value that uniquely identifies a device and is often equal to the MAC address of the device. LLDPU's can also contain optional data. These might include description for the device or system. Listing system capabilities, as well as what is interesting for us: the system name [58].

4.4.2 Management Information Base

Devices receiving LLDPDU frames will parse and use this information to update their own knowledge of the directly connected network topology. Moreover, the IETF has defined a MIB for LLDP data [59]. This MIB makes it possible to request a device's directly connected links through SNMP.

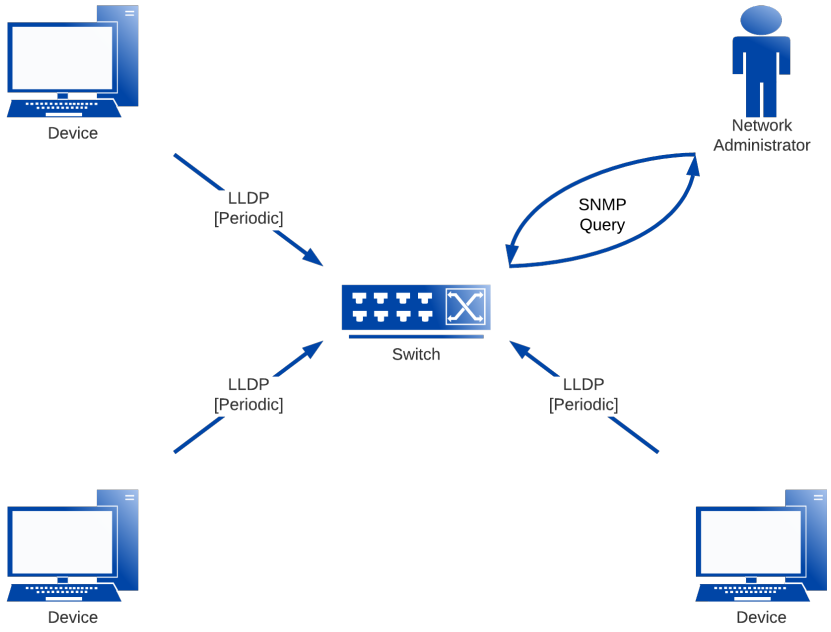


Figure 4.2: Devices send out LLDP information periodically. Centralized network devices such as switcher store and aggregate this information, and make it available through SNMP

4.4.3 LLDP And SNMP Interplay

The added benefit of using LLDP over SNMP is that device information from directly connected neighbours can be discovered, without having an SNMP agent running on all devices. Hypothetically, the router, switch or access point could collect and store all LLDP data in a localized SNMP agent. Since these network devices are often in control of the network administrators, they can be used to collect a complete overview of all connected devices. This workflow is graphically depicted in figure 4.2 This seems to be especially relevant in enterprise networks, as SNMP agents are often not enabled by default on end user devices. However, Microsoft Windows has LLDP enabled by default starting windows 8, though it only shares the MAC address of the connected interface.

4.5 Manually Managed Static Reverses

In table 3.4 we have seen blanket reverses, that seem to be statically configured. There are several ways to configure this. The most straightforward way would be to directly set these in the zone file of the DNS server. As we will see next chapter, it also possible to let these records be managed in the same location where IP address management takes place; the DDI appliance.

4.6 Other Sources

The list of sources discussed here is not complete. We have used the University of Twente network as a hook to dive deeper into a specific set of sources. In this section we discussed Active Directory as a source of information as this is used to manage windows devices at the university. However, other directory services that might exhibit similar behaviour exists, but are not discussed here because we need to define a limited scope.

We do acknowledge the existence of other LDAP implementation that might serve as information source, for potential future research. Options include: OpenLDAP[1], 389 Directory Server (previously Fedora Directory Server)[2] Apache Directory[3] and Apple Open Directory[4].

4.7 What We Learned So Far

In this section we have investigated a set of protocols that are able to communicate user and device data to the network. For the DHCP protocol, we have identified three options that are capable of directly carrying user identifiying information, namely the client-identifier, hostname and FQDN options. We demonstrated using the UT network that in this particular network, the information in the hostname field is propagated to the DNS. We have seen how Active Directory communicates names to the AD enrolled device. This information is then used locally to construct a name which is communicated to the DNS by means of DHCP. When running LLDP clients, the local network routing equipment is able to retrieve device information, which can be propagated by for example SNMP further to the network. And finally, we touched upon static names that are set by network management appliances: DDI.

[1] `https://www.openldap.org/`
[2] `https://www.port389.org/`
[3] `https://directory.apache.org/`
[4] `https://developer.apple.com/documentation/opendirectory`

Chapter 5

The Role Of DDI Appliances

In this chapter we will investigate the effects of DDI software on the information flow from source to the DNS. In section 2.5 we have seen the purpose of these systems. In the context of this thesis we note that DDI software nestles itself in a central position in the network. i.e. it has the ability to be connected to all core network services: DNS, DHCP, and Active Directory, as well as monitor connected devices.

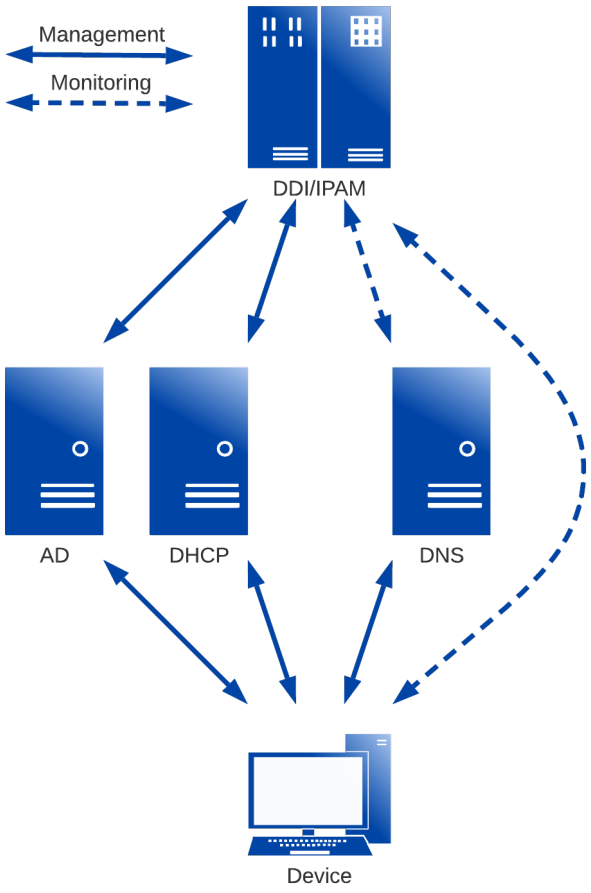


Figure 5.1: DDI applicances are positioned centrally in the network, connecting to all core network services.

The purpose of DDI software is to manage DNS, DHCP and other information carrying network resources. Given this functionality, we pose the main research question we aim to answer in this chapter:

- **Sub RQ 5 -** How does DDI software affect the flow of personal information through a corporate network?

Our investigation is laid out as follows. We first make an inventory of providers, and try to gain access to software licenses and knowledge banks. We then make a selection of providers to perform a case study on, and using the soft/hardware requirements for these appliances we design a test bed to conduct our case studies.

The case study on the DDI providers will investigate how these systems connect to the core network services, and if they have any additional role in the data flow to the (reverse) DNS. More specifically, we will check if the assumptions made in the previous chapter (how data flows from various sources) holds true.

5.1 Sourcing DDI Software

Our process for sourcing DDI software suppliers is a twofold. We look for publicly available market research into the IPAM/DDI space, as well as manual searches using a search engine using terms such as DDI and IPAM. We include IPAM in our efforts as the terminology is a bit muddy in marketing material.

We have found one source providing semi public market research. In 2019, Gartner performed a solution comparison for enterprise DDI solutions, focussing on deployment models of DDI solutions [61]. This research is not freely available, however in the summary it lists the six DDI providers included in the comparison. More providers were found using a search engine. Table 5.1 lists the set of providers we have found. We perform a preliminary investigation of the providers by checking functionality, and whether we are able to gain access to the appliance software. We list functionality as it seems that some providers only seem to offer pure IPAM services, that is, no (full) integration into DNS and/or DHCP.

We limit our provider selection based on two criteria: First we look at the providers that have full DDI support and focus on the enterprise environment. As the boundary between what is marketed as DDI and IPAM is a bit muddy, we focus on those providers that focus on integrating IPAM as a discipline, with DNS and DHCP. Secondly, we note that this is mostly enterprise grade software, and as such we need to go through the effort of obtaining the software. Because this is enterprise grade software, this often means going through a sales department. Table 5.1 lists whether we managed to obtain a (trial) license for testing purposes. For those that required to go through sales, we openly stated that this was for research related purposes.

This process leaves us with 4 DDI providers that we will investigate further, namely: Infoblox, Men and Mice, SolarWinds and Windows Server.

IPAM/DDI Provider	Source	Functionality	Availability
Infoblox	Gartner	DDI	Obtained trial license
Men and Mice	Gartner	DDI	Obtained trial license
Solar Winds	Search	DDI	Obtained trial license
TCPWave	Gartner	DDI	Denied by sales
Windows Server	Gartner	DDI	Obtained license
Bluecat	Gartner	DDI	Denied by sales
EfficientIP	Gartner	DDI	Denied by sales
LightMesh	Search	IPAM Only	No trial requested
ManageEngine	Search	IPAM Only	No trial requested
GestioIP	Search	IPAM Only	Open source
netBox	Search	IPAM Only	Open source
NIPAP	Search	IPAM Only	Open source
phpIPAM	Search	IPAM Only	Open source

Table 5.1: Overview of IPAM/DDI providers using both Gartner market research and manual search.

5.2 Creating A Test Bed

In order to design a test bed, we first figure out the system requirements for each of the platforms we plan to investigate. We then aim to design a test bed that will be suitable for all applications in order to analyze the flow of data on a similar setup. For the four remaining DDI providers, we have listed their additional requirements in table 5.2.

IPAM/DDI Provider	Requirements
Infoblox	- Virtual Machine with embedded DNS and DHCP - Hypervisor: VMWare ESX, Hyper-V or KVM - SQL server
Men and Mice	- Windows and Linux based executables - Postgres (SQL) server
Solar Winds	- Windows based executable - VMWare vOrchestrate - SQL server
Windows Server	- No additional requirements

Table 5.2: The requirements for the four selected DDI providers.

5.2.1 Server Node

The various DDI providers have slightly differing requirements. Some of the providers are flexible with regards to the platform it runs on, whilst others are strictly available

on a single platform. Nevertheless, for three of our four DDI providers selected, we can create a single server platform to host the DDI appliances, as well as the other network services required, such as Active Directory, DNS, DHCP and storage back-ends. We settled on a Windows 10 Enterprise based host with Hyper-V support. This will be the server node used for Men and Mice, SolarWinds, and Windows Server.

Infoblox

The remaining provider, Infoblox, provides virtual appliances that can be hosted on a Hyper-V capable machine. In our test bed we will substitute a machine running this image for the server node described above.

5.2.2 Test Network And Client Nodes

Our goal is to investigate data flow in a network, and thus we need some client devices to initiate and/or receive these flows. We add two 'user' nodes: An Android phone running Android 11, and a Windows 10 device capable of enrolling in Active Directory. The three nodes will be connected via routing equipment. This setup allows us to attach two measurement probes on the node running the DDI services: One probe on the interface to monitor client to server traffic. And a second probe on the loopback device of the server to monitor traffic between the network services. This setup is graphically depicted in figure 5.2.

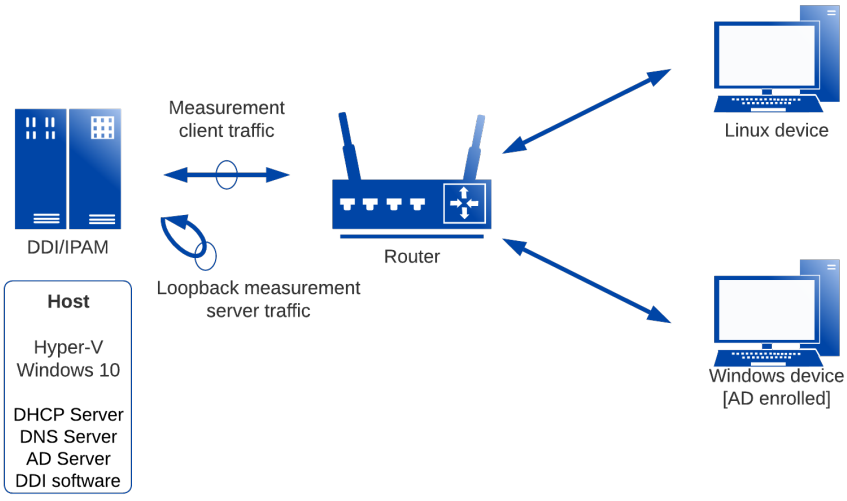


Figure 5.2: Network diagram for the test bed. Lists the services running on the network nodes, as well as network measurement probe locations.

5.2.3 Methodology

Our methodology for the case studies will use the physical setup outlined above. We will re-use part of the setup between DDI providers: The end user nodes and routing equipment will remain the same in each case study. The network services on the server node will also remain the same between DDI providers as we setup our environment with the same DNS, DHCP and AD server.

The distinguishing factor between the case studies are the DDI providers. In the next section we will describe in more detail how the connection between the network services and DDI appliance is set up.

Our interest is the network flow from end user device to the DNS. To measure this, we attach the network probes as described in the previous section. Our measurement procedure is based on the protocols study in the previous chapter. To measure the protocols in action we take the following aspects into account:

1. For DHCP, we need to initiate a DHCP session. Most devices are configured to run DHCP automatically when connecting to a network. Simply connecting and disconnecting from a network should initiate DHCP traffic. However, for a bit more fine grained control, we have configured one of our clients to not perform automatic DHCP. Instead, we manually initiate and terminate DHCP sessions. This has the added benefit of being able to configure the passed information in the DHCP request, and see which DHCP options the DDI and DNS respond to.

2. For AD, we need to enroll one of our devices. According to our literature study in the previous chapter, we found that AD information stored on the device is communicated to the network over DHCP. Again, we will need to initiate a DHCP session to measure this traffic.

3. For SNMP, we need to wait for the SNMP requests to be generated. In our case study, we expect these requests to be performed by the DDI appliance for network discovery purposes.

4. For LLDP, we do not directly measure this traffic. This has to do with where our measurement probes are located within the test bench. LLDP is a layer 2 protocol, and is thus only used to communicate information between immediate neighbours. In the previous chapter we have described a possible mechanism where LLDP information is communicated to the DDI appliance, namely through SNMP of centralized network components such as switches and routers.

These requirements leave us with a procedure for testing the IPAM provider: First we install and connect the DDI appliance to the other network services. Next, we will configure the DDI appliance for automatic changes to the DNS. This might differ

between providers, but will likely involve changing DHCP settings, and enabling network discovery if available. We will then enrol one of the devices in AD, if applicable. And initiate DHCP requests and releases. Moreover, we will allow the some time for the test bench to idle, to make sure our capture will capture an network discovery scan.

5.3 Connecting DDI To Our Test Bed

In this section, we install and connect DDI providers into our test bed. We detail for every provider what the DDI's role is, and how it is connected to the core network services.

5.3.1 DDI: Men And Mice

Men and Mice consists of a central piece of management software called Men and Mice Central, and is connected to a database and exposes and API. Various UI applications, Graphical, terminal based, or the API is available to configure the software. Fig 5.3 displays this centralized role graphically.

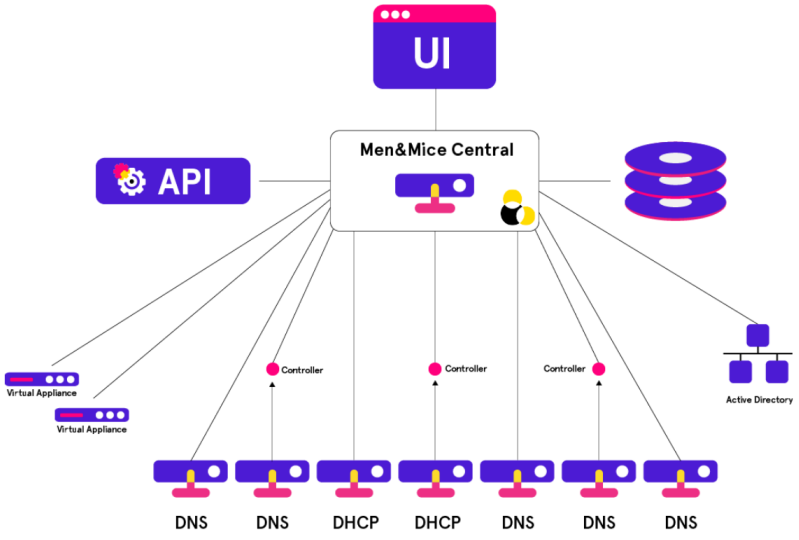


Figure 5.3: Men and Mice Architecture. Note the controller agents sitting between the central DDI software and the DNS and DHCP services. Source: Men and Mice

DHCP And DNS Agents

Men and Mice typically does not directly connect to the DHCP and DNS services in the network. Instead, it requires the administrator to install *agent* software next

to the network service. The agent software talks a proprietary protocol to Men and Mice Central. The connection to the DHCP and DNS server is dependent on the specific server installed. For e.g. ISC BIND (DNS), the agent makes use of the CLI management interface.

DHCP And AD As Source

As seen in the previous chapter, both DHCP and AD expose client information through the DHCP protocol to the network. From Men and Mice central, the DHCP server can be configured to use the Dynamic DNS (DDNS) protocol to facilitate automatic changes to the DNS based on DHCP leases. There are three options available:

- Disable DDNS updates completely.
- Update DDNS based on DHCP options in lease negotiation.
- Force DDNS updates by DHCP server on behalf of the client.

We note that there are no separate forward or reverse configuration options, although this is possible by manual configuration of the DHCP server, the DDI (and DHCP service) assume both a dynamic forward and reverse policy is intended when selecting force updates.

In our measurements, we confirm the direct flow from DHCP to DNS using DDNS. We conclude that the agents are only involved in this flow during configuration. During normal operation, the DHCP server will directly communicate AD and DHCP client configuration to the DNS using DDNS.

Network Scanning

In the previous chapter we have seen SNMP as one of the sources of local host information. Men and Mice Central provides a host discovery module that makes use of SNMP to query routers in the network for host information. Our measurements confirm that this information is reflected in the IPAM interface, however there is default configuration allowing this information to propagate to the DNS. We do note that the DDI appliance provides an API and automation hooks, making such a flow from SNMP to DNS possible.

5.3.2 DDI: Windows Server

Windows server bundles a built in DNS, DHCP and IPAM module.

DHCP

When configuring a DHCP zone in Windows Server, the setup wizard will offer the option to enable automatic DNS updates on behalf of this DHCP server. The default settings for this takes the default Windows/Active Directory configuration in consideration: As we have seen in chapter 4, Windows based devices enrolled in active directory perform DHCP requests with FQDN extensions enabled. More specifically, they request the DHCP server to update the reverse record, while the client intents to update the forward itself.

However, the wizard also provides the option to ignore the clients FQDN request, and perform the forward and reverse update itself.

Network Discovery

Windows Server has a built in SNMP client that can be configured to retrieve LLDP information from routers. However, this information is not reflected in the IPAM interface. In the same vein, there is no out of the box solution for propagating this to DNS.

Windows also bundles a network discovery module. This module is based on the Universal Plug and Play (UPnP) protocol [62]. We do not investigate this further as this protocol is unsuitable for the corporate network environment due to chatty protocol design and security considerations [63].

5.3.3 DDI: Solarwinds

SolarWinds provides integration with the DHCP and DNS servers. It provides bi-directional synchronization between the DDI appliance and the DNS and DHCP server. Changes in the DNS server will be reflected in the DDI software, and vice-versa. However it does not naively provide the ability to enable DDNS for managed IP address space

Linking DHCP And DNS

The DDI interface only allows for manual changes made by system administrator to be propagated to the DNS. Network administrators need to create a manual DDNS connection on the DHCP and DNS software to reflect automatic changes of DHCP leases in the DNS.

Network Discovery

Like other DDI sections discussed in this chapter, SolarWinds provides a network disovery module based on SNMP.

5.3.4 DDI: Infoblox

Infoblox does not run on the same unified server node as the other three DDI providers listed here. Instead, the appliance is offered as a virtual machine with a bundled DHCP and DNS server. According to the documentation, the DNS and DHCP server are ISC BIND (DNS) and ISC KEA (DHCP) derivatives.

DHCP

The DHCP server is connected to the DNS service via DDNS. The DHCP server bundled is an ISC KEA (DHCP) derivative with some extra functionality. It offers two notable functionalities:

- Generating names for devices not supplying a name in their DHCP request. The default is to generate a name using the template: *dhcp-IPADDRESS.example.tld*.
- Rewrite engine for hostnames. The default is to only allow alphanumeric characters: *A-Za-z0-9*. Names including dots (.),including the above mentioned generated names, will see these characters replaced by a hyphen (-), e.g. *dhcp-192-168-0-2.example.tld*.

In our measurements, we see that these processes happen in the DHCP server. These changes are then communicated to the client over DHCP, and to the DNS server using DDNS.

Network Discovery

A network discovery tool based on SNMP is bundled, however there is no default way of having this information propagate to DNS.

5.4 Extracting Common Patterns

Our testing with the various DDI systems allows us to extract common patterns between the providers. We can roughly categorise these in two categories:

- Flows through or preceded by DHCP using DDNS.
- Periodic network scanning/host discovery with propagation to the DNS.

Not all DDI providers support every type of flow, and subtle differences exists between DDI providers. Tables 5.3 and 5.4 aims to provide an overview of these categories for the tested DDI providers.

DDI Provider	DHCP to DDI	Propagation to DNS	FQDN Extensions Support
Windows Server	✓	✓	✓
Men and Mice	✓	✓	✓
Infoblox	✓	✓	✓
Solarwinds	✓	✓	✓

Table 5.3: Support for information gathering by DHCP, including propagating flow to the DNS. Moreover we list the support for FQDN extensions by the providers.

DDI Provider	Host discovery to DDI	Propagation to DNS
Windows Server	✗	✗
Men and Mice	✓	✗
Infoblox	✓	✗
Solarwinds	✓	✗

Table 5.4: Table listing the inclusion of host discovery, and the flow of this information to the DDI, and propagation to the DNS.

5.4.1 DHCP Flows

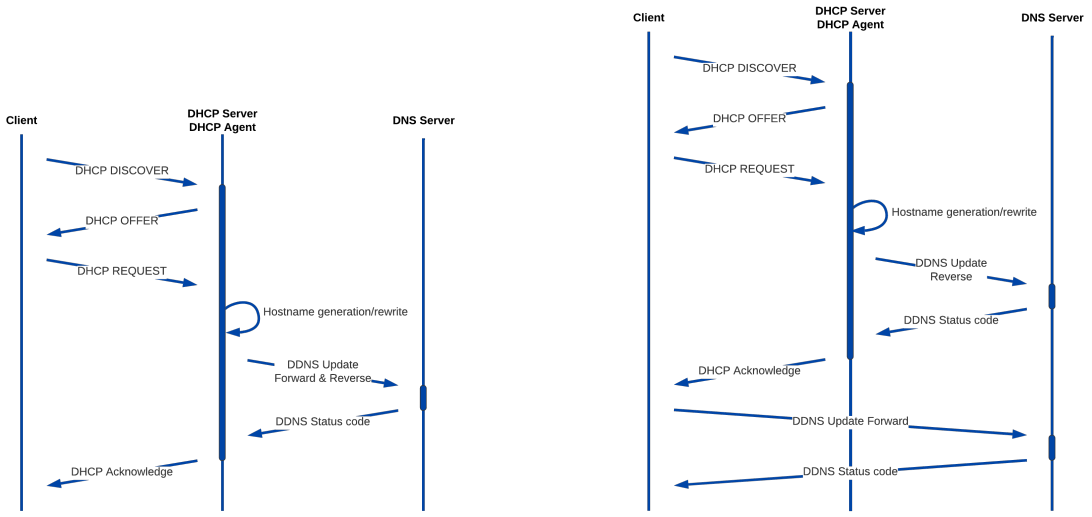
We notice that in our testing, all information that flows to the DNS at some point also communicates these names through DHCP. This is not to say that the DHCP server is always the entity to set the names in the DNS. Instead, a DHCP lease might be used by a client to confirm (temporary) ownership of an IP address before setting the DNS records itself. It must be noted that the flow for setting the forward versus reverse name might take different paths for the same DHCP lease.

We see the following two common patterns:

1. After DHCP lease acknowledgement, DHCP server sets both the forward and the reverse. Most often this achieved by a DDNS connection between the DHCP and the DNS server. Variations on this pattern exist where there is no direct connection, but instead a local agent provided by the DDI sets up this connection.

2. After DHCP lease acknowledgement, client uses DDNS to set forward, whilst server uses DDNS to set reverse. This is achieved using the FQDN client extensions in the DHCP protocol. We note that this pattern is only found in trusted networks where the clients are allowed modification access to the DNS server.

These two flow variations are depicted graphically in Fig 5.4.



(a) DHCP server updates forward

(b) Client updates forward

Figure 5.4: Information flows from DHCP to the DNS. Depicted are two flows, key difference being which entity updates the forward record. Variation on these flow patterns are common, as the connection between the DHCP and DNS server might be set up by an intermediary agent provided by the DDI appliance.

Hostname And FQDN Extensions

During our protocol study in chapter 4, we have seen that DHCP has several options that might carry information. Two of these are likely to be used to craft names that propagate to the DNS: The hostname option, and the FQDN extensions. To

investigate how the DDI appliances react to these parameters, we use specially crafted DHCP requests with permutations of these options:

- Request excluding hostname, and FQDN extensions.
- Request including hostname, excluding FQDN extensions.
- Request excluding hostname, including FQDN extensions:
  - FQDN Extension supplying a specific name
  - FQDN Extension requesting DHCP server to **not** set DNS records.
- Request iincluding both hostname and FQDN set, however these values are different for the same request.
  - FQDN Extension supplying a specific name
  - FQDN Extension requesting DHCP server to **not** set DNS records.

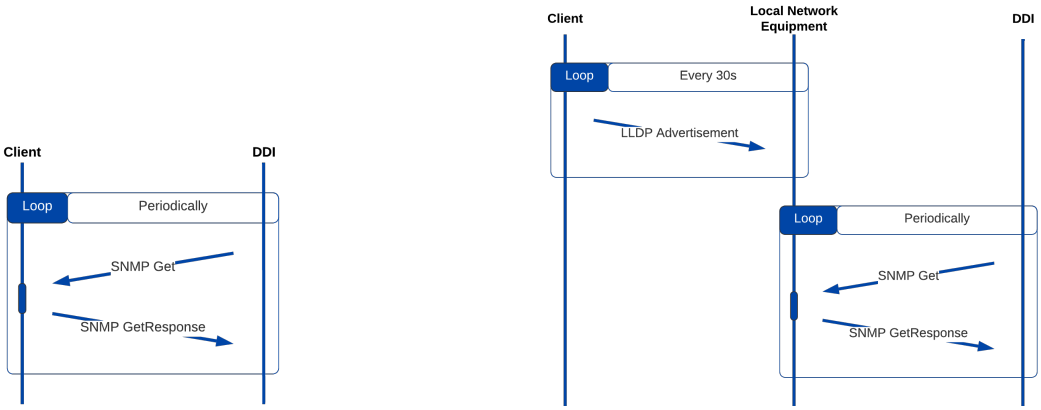
we see a pattern of precedence in the DHCP options supplied in a request. The DDI appliance generally use the value requested in the FQDN extensions over the value supplied by the hostname parameter. Generally we see the DDI appliance honouring the FQDN extensions request to not update the DNS. However, there is an exception here in the case of Infoblox. When Infoblox is configured to generate a DNS name for clients not supplying a hostname, the DHCP server will propagate a generated name to the DNS.

5.4.2 Network Discovery

DHCP will not always use the hostname option, or the FQDN client extenstions. A DDI appliance might opt to use other means to gather names for the DDI appliance. All of our tested appliances implement a form of network scanning, however none seem to be able to propagate this information to the DNS out of the box.

SNMP

One of the protocols used for network discovery is SNMP. The DDI appliance might query end user devices. It requires an SNMP agent to be installed and enabled on the device. This is not always the case: Windows has an SNMP agent, however it is disabled by default. Yet SNMP remains a powerful tool in network discovery. A network administrator might enable SNMP on centralized network devices such as routing equipment. The routing equipment can leverage the LLDP protocol to gather information about its direct neighbours, and store this information in a MIB. These flows are depicted in figure 5.5.



(a) Direct SNMP Query

(b) Periodic LLDP advertisement being propagated by SNMP

Figure 5.5: Information flows from devices to the DDI system using LLDP and SNMP.

5.5 What We Learned So Far

In this chapter we investigated a set of DDI providers, and how they handle information flow carrying personal information. We honed in on two strategies as to how these appliances get hold of this information: Network information carried by DHCP sessions, as well as network discovery using LLDP and SNMP.

All the DDI providers in our case studies support the creation of dynamic reverse DNS zones, where the names are constructed using device information originating from DHCP handshakes. We see that this information is communicated in two ways: Either through DDNS after a valid DHCP lease is created. This can both be performed by the client in networks where this is allowed, as well as the server setting the records on behalf of the client. Alternatively, the records will be set by custom DDI agent software that runs alongside the DHCP and DNS servers.

The DDI appliances are able to use information from several DHCP options. FQDN options take precedence when creating names over other options such as the DHCP hostname option. In some cases the DDI provider will generate a dynamic record when no hostname of FQDN extension is supplied. Moreover, some DDI appliances can be configured to ignore a client's FQDN extension request to not set DNS records.

All of our tested DDI providers implement some form of network scanning to gather more information about the connected hosts. Out of the box, none of the tested appliances allow this information to be further propagated to for example the DNS. It is only used inside the DDI, either for the convenience of the network administrator, or to apply network policies to the device.

Chapter 6

Devices

In the previous chapter we have identified DHCP as one of the main protocols through which information flows from client devices, to the network and the DNS. We investigated how the data originating from the DHCP protocol flows through the network from the server side.

In chapter 3 we have investigated the reverse address space of the University of Twente. We found out that this particular reverse address space carries hostnames, and in chapter 4 we found out that this information is supplied by DHCP. We noticed that the network blocks that carry these hostnames are non-contiguous. That is to say, not all addresses have a reverse record, even when we know that a device is actively using that IP address.

In order to better understand these behaviours, we are going to investigate DHCP from the client's perspective. Our aim is to uncover what kind of information (if at all) is communicated by end user devices. And how recent implementations of privacy enhancing technologies affect DHCP.

We formulate the following research questions:

- **Sub RQ 6 -** What kind of information do clients leak in their DHCP handshake?
- **Sub RQ 7 -** How are privacy enhancing strategies for DHCP implemented in practise?

To answer these questions, we will perform a measurement study by gathering and analysing DHCP handshakes from user devices we might typically find in corporate networks. This chapter requires background knowledge (chapter 2) on MAC address randomisation (section 2.3.2), DHCP (section 2.4), and anonymity profiles for DHCP (section 2.4.2).

6.1 Methodology

In this chapter we aim to investigate which clients leak what kind of information. To that end we intend to gather and analyse DHCP handshakes from devices that we might commonly see in corporate networks. These range from company issued

devices, as well as several types of personal devices. Specifically we will be looking at desktop computers, laptops, tablets and mobile phones.

6.1.1 Measurement Setup

To collect the DHCP handshakes, we need to provide a network on which we are able to attach a network probe at the DHCP server. This will allow us to capture all DHCP traffic on the network. For our measurement setup we use a GL-Inet AR-150[1] router with built-in DHCP server. The reason we use a mini router is that it provides the functionality we need: A DHCP server, routing capabilities, two Ethernet ports for wired clients, and a built-in wireless access point for wireless clients.

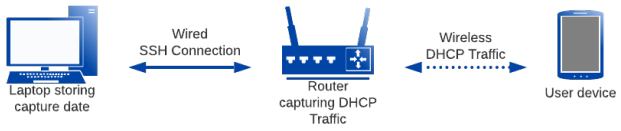


Figure 6.1: Test bed for collecting DHCP handshakes.

The router runs *OpenWRT*[2], a custom firmware image that allows us to install additional software packages. We connect a laptop to one of the Ethernet ports, and connect to the router over SSH. To capture the DHCP traffic, we install the command-line packet analysis tool *tcpdump*[3].

Since the router is memory limited, it is undesirable to store the capture file on the router itself. Instead when we start *tcpdump* over SSH, and we ask it to print to standard out. Next, we will pipe the capture output from the SSH connection into *Wireshark*[4], another network capture and analysis tool, running on the laptop. Listing 6.1.1 details the command we use for this process.

```
1 ssh root@192.168.8.1 \ # Connect over SSH
2   tcpdump -i br-lan  \ # Start tcpdump and filter on interface
3   port 67 or port 68 \ # Tcpdump filter
4   -U -w - |          \ # Write to stdout unbuffered
5   wireshark -k -i -    # Pipe stdout into wireshark
```

Listing 6.1: Bash command that connects to the router over ssh and start the network capture using tcpdump. The result is then piped into wireshark on the local machine.

[1] https://www.gl-inet.com/products/gl-ar150/
[2] https://openwrt.org/
[3] https://www.tcpdump.org/
[4] https://www.wireshark.org/

6.1.2 Procedure

Using the measurement setup described in the previous section, we follow the following procedure to capture our handshakes.

1. To start a new capture, we issue the command as described in listing 6.1.1.
2. We ask the participant to connect their phone to the wireless network.
3. We can use the DHCP traffic displayed in wireshark on the capturing laptop to confirm the connection was successful.
4. We ask the participant to switch off and on their WiFi connectivity. This will cause a new DHCP session for a 'known' network on the participant's device.
5. We ask the participant to 'forget' the network, and subsequently to connect to the network again.

This procedure will give us three DHCP handshakes per device: Two separate traces of connecting to an 'unknown' network. i.e. a network that is not saved on the client's device. And one trace reconnecting to a known network.

6.1.3 Ethics and Privacy

For this measurement, we are collecting traffic from personal devices. This means we need to take the privacy of the device owners into account. We first explain to the potential participants the aim of the study, and ask them to connect their devices to our network themselves. This provides the participant with the option not to provide data, and thus not participate. Moreover, we explain that they can retract their participation any time in the future if they so wish. Upon such a request, the handshake data will be deleted.

We also take some technical measures to limit our data collection. As a device connects to the router, the various applications installed on these devices might start communicating privacy sensitive information after a connection is established. Since we are only interested in the DHCP handshake data, we aim to limit the amount of data captured in the following two manners:

- The network we ask participants to connect to is not connected to the Internet.
- We start our capture with a filter enabled: We will only capture traffic on ports 67 and 68. These are the ports DHCP will open on the server and client respectively [36].

Lastly, we consider the sensitive data in the DHCP handshakes: The MAC address, hostname and FQDN options might contain information identifying specific individuals and/or device make and model. Specific person names are not of interest, and hence we will be omitted from the analysis.

6.1.4 Analysing Capture Data

To answer our research questions, we are interested in the following properties of the gathered DHCP handshake data:

- The supplied options in the various DHCP messages.
- The usage of MAC address randomisation throughout the full network frame.
- The application of Anonymity Profiles for DHCP.

For the analysis of the raw captured DHCP handshakes, we use Python in combination with the packet manipulation library Scapy[5]. Scapy is capable of parsing packets and splitting them into the individual protocols. The parser for DHCP is not complete. More specifically, it cannot parse the DHCP options field completely. To that extend, we have written an additional parsing script for DHCP options. This script can be found in appendix A.

6.2 Results

For our analysis, we have measures a total of 26 devices: 15 were Android devices, 7 were Apple devices and 4 were Windows based devices. Figure 6.2 has a breakdown of the devices including OS version. Appendix B has a full list of devices, including condensed data for the analysis of the results.

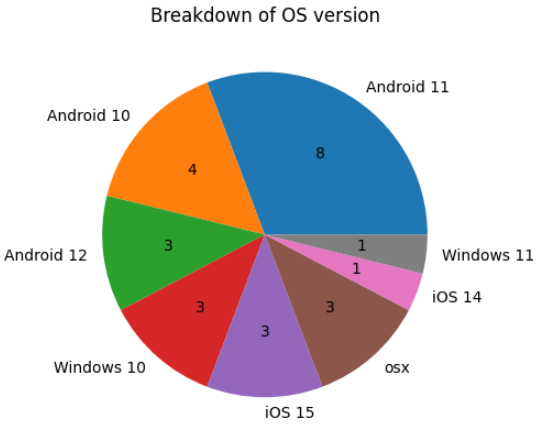


Figure 6.2: Breakdown of the OS and OS version of the devices in our data set.

[5] https://scapy.net/

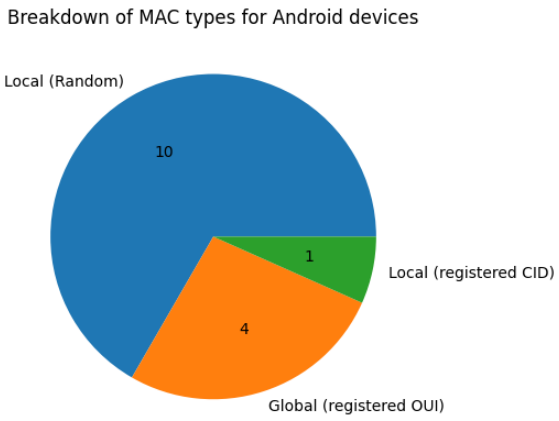


Figure 6.3: Types of MAC addresses for the Android devices in our data set. We distinguish between the two types of local addresses: full randomisation of the CID/OUI part, and those devices using a registered CID.

6.2.1 DHCP Discover vs. Request

For all DHCP handshakes collected we note that the options supplied are identical for both the discover and request message. This includes the parameters and the ordering of the requested parameter list option. There are two exceptions: both of technical nature. The request message has the additional option for the requested IP address. This is required according to the DHCP specification, and as we have explained in the previous chapter not a violation of an anonymity profile.

Additionally, for Android devices we have seen some clients send the DHCP option 80 *rapid commit* along in some (but not all) of their discovery messages. Rapid commit is protocol extension that allows implementing server and client to shorten the DHCP handshake procedure to two messages [64]. Instead of using the full discover-offer-request-acknowledge handshake, the server will upon request of a rapid commit discover message, immediately reserve an IP address and reply with an acknowledge message. This behaviour might be advantageous for devices that need an IP Address quickly with minimal network overhead.

Option 80 could be seen as a violation of the anonymity profile specification, as it indicates that the OS has rapid commit enabled. We note again that we have only seen this option in traces generated by Android devices, and as such it might constitute finger-printable behaviour.

6.2.2 MAC Address Randomisation

For the Windows devices in our data set, we see no MAC address randomisation. This is in line with the literature.

For Apple devices, we see no randomisation for the Macbooks in our data set. The devices use a global address with Apple's OUI. The iPhones in our measurement do randomise their MAC address. Moreover, the MAC address changes on network forget and reconnect. This is also in line with the literature.

For Android devices, we see mixed results, some of which are not in line with the literature. A breakdown of the different MAC addresses can be found in Figure 6.3. Of the 15 Android devices in our data set, 10 devices randomise including a random CID. 1 device randomises with a registered CID. This happens to be a Google made phone using Google's CID. 4 phones did not apply any randomisation at all, even though all devices in our data set are Android 10 or above, and according to literature should have randomisation turned on by default.

6.2.3 Client-identifier

In all our collected traces, the client-identifier is set to the (randomised) MAC address. This is the same MAC address as is used on the lower (802.2 Ethernet and 802.11 WLAN) layers.

6.2.4 Hostname Leakage

All Windows devices in our data set leak the hostname through this field.

For Apple devices, this is only the case for the Macbooks. The iPhones do not leak. For all Apple devices in our data set, there seems to be a relation between leaking the hostname, and whether the MAC address is randomised: We turned the 'private Wi-Fi' functionality on one of the iPhones, and performed another measurement. The MAC address changed to the globally unique address of the device, and the DHCP messages now contained the hostname option.

The Android devices do not seem to display such a relation. Our data set has traces that contain all four combinations of whether the hostname option is present versus whether the MAC address is randomised. Figure 6.4 displays the breakdown for these devices.

6.2.5 FQDN Leakage

No hosts in our data set use the FQDN parameters. None of the tested devices were enrolled in Active Directory. We have only seen FQDN parameters in synthetic clients enrolled in Active Directory in the previous chapter.

6.2.6 Vendor Leakage

All Windows devices the distinctive identifier *MSFT 5.0* through the vendor-class option. This identifier is the same for different OS versions.

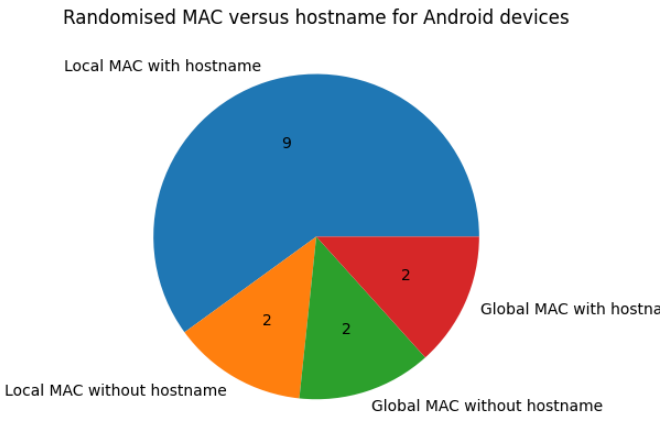


Figure 6.4: Breakdown of Android devices in our data set that use a local (Random and registered CID) or global MAC versus whether the hostname is sent along.

Apple devices do not leak any vendor related information in its DHCP requests. This is true for both the Macbooks and the iPhones in our data set.

All Android devices in our data set use the vendor-class option. Moreover, the leaked value is specific to the OS version, e.g. for a device running Android 11, the identifier *android-dhcp-11* is leaked.

6.2.7 Parameter List

In our measurement set we see similar parameter request lists (PRL) for devices that run the same operating system. Table 6.1 shows what a PRL looks like in our data set, this includes the ordering of parameters as seen in our traces.

For Windows, and both iPhones and Macbooks, we see the exact same PRL for every device. For Android we see 3 of our 15 devices with a slight alteration to this list, all adding a single extra option.

6.3 Discussion

From the results presented above, we can conclude that none of tested devices fully adhere to the guidelines set for anonymity profiles. In general, there are a few things that all devices seem to get right:

Firstly, none of the devices in our traces supply the FQDN option. Though we should mention that none of the devices in our data set were configured to do so, by for example, enrolling in Active Directory, or enabling DDNS on the client device.

Android Length: 12 Devices: 12 out of 15	**iPhone** Length: 9 Devices: 4 out of 4
(1) Subnet Mask (3) Router (6) Domain Name Server (15) Domain Name (26) Interface MTU (28) Broadcast Address (51) IP Address Lease Time (58) Renewal Time Value (59) Rebinding Time Value (43) Vendor-Specific Information (114) DHCP Captive-Portal (108) IPv6 Only Preferred	(1) Subnet Mask (121) Classless Static Route (3) Router (6) Domain Name Server (15) Domain Name (108) IPv6 Only Preferred (114) DHCP Captive-Portal (119) Domain Search (252) Private/Proxy autodiscovery
Windows Length: 14 Devices: 4 out of 4	**Macbook** Length: 12 Devices: 3 out of 3
(1) Subnet Mask (3) Router (6) Domain Name Server (15) Domain Name (31) Perform Router Discover (33) Static Route (43) Vendor-Specific Information (44) NetBIOS TCP/IP Name Server (46) NetBIOS TCP/IP Node Type (47) NetBIOS TCP/IP Scope (119) Domain Search (121) Classless Static Route *(249) Private static Route (Microsoft)* (252) Private/Proxy autodiscovery	(1) Subnet Mask (121) Classless Static Route (3) Router (6) Domain Name Server (15) Domain Name (108) IPv6 Only Preferred (114) DHCP Captive-Portal (119) Domain Search (252) Private/Proxy autodiscovery (95) LDAP (44) NetBIOS TCP/IP Name Server (46) NetBIOS TCP/IP Node Type

Table 6.1: Common Request Parameter List in DHCP requests across devices/operating systems. Note that none of the lists adhere to the ordering guidelines set by anonymity profiles. Windows devices even leak directly identifying options.

Secondly, for the devices that randomise MAC, this address is being used consistently on lower network layers, and in the DHCP messages themselves.

Thirdly, the client-identifier in all of our requests is set to the MAC address, for devices using a randomised MAC address, this field carries the correct (randomised) address.

And fourthly, the ordering is based on the option numbers. Although the guidelines for anonymity profiles suggest randomised ordering as the preferred option, ordering by option number is suggested as alternative. We argue that ordering by option number would make more sense anyway, as our data suggests that this behaviour more closely resembles the status quo than a fully randomised list would. If a device class or operating system would start randomising the ordering while other vendors might not, it could be used as finger-printable behaviour.

However, our data also suggest some violations of the guidelines for anonymity profiles:

6.3.1 Randomisation and Hostnames on Phones

Strangely, some Android devices that randomise their MAC address, also send the hostname in their DHCP traffic. This is strange as the hostname is often a direct identifier of the device. We hypothesise that this is due to Android only implementing MAC randomisation as a measure against fingerprinting before a device associates with a network. For some of the Android devices we gathered traces from, the option is simply called MAC address randomisation. IPhone on the other hand calls its functionality that includes randomisation 'private Wi-Fi'. We gathered a trace from a single iPhone device where private Wi-Fi was manually turned off. This resulted in the device switching to its globally unique MAC address, and caused the DHCP request to include the hostname option.

6.3.2 Macbooks and Windows Devices

For the laptops and desktop type devices in our data set (Windows devices and Macbooks), we see that the requests violate most of the guidelines for anonymity profiles.

First off, none of the devices in our data set perform MAC address randomisation. This means that these devices are finger-printable across time and networks, even before looking at DHCP data.

Secondly, the hostname is sent along in each DHCP request. As we have seen in earlier chapters, there is often structure to these names leaking PII and/or device name. Windows additionally provides the vendor-class option, directly identifying the device as being Windows based.

And finally, the values included in the parameter request option list seem to distinctly identify these devices as belonging to the class of laptops/desktops. Both operating systems include several NetBIOS related options. Windows additionally uses options in the reserved/private option space, namely option 249 which Windows uses for *private static routes*.

We hypothesise that the reason why these type of devices do not get the same treatment when it comes to randomisation and DHCP anonymity profiles is because of the difference in device class: MAC address randomisation was in part a response to tracking devices before they are associated with a network. There is an argument to be made that this is more important for mobile devices that are constantly on and searching for networks, such as mobile phones. Laptops on the other hand might only scan for networks when they are being actively used, and as such the need for randomisation might be seen as less prudent.

However, it is not unthinkable that these types of devices are used in a hostile environment where tracking might occur, say on public transport (planes and trains), coffee shops and airport lounges. As such, it would make sense to not only implement randomisation, but provide privacy measures for DHCP as well.

6.3.3 Parameter Request Lists

We see in our results that the Parameter Request List option seems to have distinctive features related to the device and/or operating system. Even though we have not attempted to extract a fingerprints in this work, previous academic work has shown that the usage of DHCP traffic is possible, indicating that the PRL is one of the most distinctive factors [65]. We also add that this is not just an academic exercise, as large DDI providers (EfficientIP, Infoblox) are amongst the top search results when searching the web for DHCP fingerprinting [66], [67]. These systems allow the network operators to automatically apply policies based on the device's DHCP fingerprint.

The guidelines for anonymity profiles state the the client needs to minimise the requested options, and the ordering strategy should either be fully randomised or in order of option number. None of the devices in our data set fully adhere to this guideline. Even the device class that uses a minimal number of requests options, and requests no OS specific options, fails to adhere to the ordering guideline.

Ordering of options should be easily solvable across vendors. The guideline to minimise the number options however, we deem too vague. A minimised PRL might still lead to differences, albeit smaller than an un-minimised list, that can be fingerprinted. A better approach would be to set guidelines to exactly which options to include and not to include for devices wishing to use an anonymity profile.

6.4 Conclusion

All the devices in our data set violate the guidelines for DHCP anonymity profiles. This ranges from very minor violations as seen by iPhones not adhering to ordering guidelines of the PRL, to devices not randomising MAC addresses and leaking direct identifiers such as hostnames and vendor-classes. This is problematic as DHCP traffic can be used to fingerprint and track devices across networks and over time.

Of all the devices in our data set, we see that the iPhones provide the best privacy implementation when it comes to DHCP traffic. The iPhones in our test set all randomise their MAC address, and leak no direct identifiers of personal data, or device info in any options. It is strange that this implementation is not extended to other mobile Apple devices such as Macbooks. These devices are also commonly used in hostile environments where tracking might occur, such as coffee shops and airport lounges.

For Android devices, there seems to be more concern for privacy before a device is associated with a network. Even though we see randomisation for the majority of Android devices, all devices leak the vendor-class that directly reveal the OS including version, and some even leak more information through the hostname. As we have seen in previous chapters, particularly the use of the hostname is problematic behaviour, as it we have seen this information flow through the network to the reverse DNS.

Chapter 7

Conclusion

In this thesis we investigated how (personal) information flows through a network to the DNS. We did this by formulating a set of sub-research questions, investigating the individual parts of the flows.

7.1 A Reverse DNS Case Study

We started in chapter 3 with a case study of the reverse DNS space of an actual network, namely the University of Twente. Our aim in this chapter was to investigate what kind of names are present (Sub RQ 1), and whether there are patterns to these names (Sub RQ 2). The key takeaway for chapter 3 is that the reverse DNS holds different kinds of structured information. On the higher level we found evidence of logical separation of the network. The higher labels in multi-label names provide us location, routing and sometimes function information of the network. We also found structure in the lower labels of the names, providing hints as to the source of the information, or information about the device properties in the case of plain hostnames. We deemed it likely that the origin of these names is not just from a single source, but instead coming from multiple automated and manual sources.

7.2 Leaky Protocols In Corporate Networks

In chapter 4 we performed a literature study into networking protocols that are capable of providing (Sub RQ 3) and/or propagating (Sub RQ 4) the names we have seen in chapter 3. We identified DHCP, AD, and LLDP/SNMP to be capable of providing information to the network.

- For DHCP we identified three fields that are capable of leaking device and user information: client-identifier, hostname, and FQDN extensions. Using the UT network, we demonstrated that we can set a custom hostname using specially crafted DHCP packets, and saw this hostname propagate to the reverse DNS.

- For Active Directory, we explained that the user device retrieves its name from the AD controller, after which it assembles a FQDN locally and supplies this to the network using DHCP FQDN extensions.
- For LLDP, we detailed a flow that shows periodic LLDP information being propagated to the network using SNMP.

7.3 The Role Of DDI Appliances

Afterwards, we performed four case studies on DDI systems in chapter 5. We aimed to find out how these network management systems affect the information flow from source to DNS (Sub RQ 5). During this research, we identified two common flow patterns that allow information to travel through the network: Periodic network scanning using for example LLDP and SNMP, and information gathered from DHCP traffic.

The DDI appliances in our case studies only support propagation of the DHCP flow to the DNS out of the box. The propagation mechanisms differs between DDI providers, some allow automatic updates through DDNS, whereas others implement custom software agents to be run alongside the DNS and DHCP servers. This allows more fine-grained control for name propagation by allowing for example name rewrites, or blocklists of certain names.

The DDI appliances are able to extract information from several DHCP options: In our case studies, FQDN options takes precedence when creating names over other options such as the DHCP hostname option. In some cases the DDI provider will generate a dynamic record when no hostname of FQDN extension is supplied. Moreover, some DDI appliances can be configured to ignore a client's FQDN extension request to not set DNS records.

All of our tested DDI providers implement some form of network scanning to gather more information about the connected hosts. Out of the box, none of the tested appliances allow this information to be further propagated to for example the DNS. It is only used inside the DDI, either for the convenience of the network administrator, or to apply network policies to the device.

The key take away for this chapter is that even though DDI appliances use a host of protocols to collect information about clients, DHCP (and DDNS) is the main culprit in forwarding names to the DNS.

7.4 DHCP Handshakes

And finally, because we identified DHCP to be the main culprit, we took a deeper look into DHCP data in chapter 6. In this chapter we investigated a set of end-user devices one might typically find inside corporate networks. We gathered DHCP handshake data from 26 devices, in order to look at what kind of information these

devices leak (Sub RQ 6). Moreover, we looked at what kind of privacy enhancing strategies these devices use (Sub RQ 7), and how these devices can improve.

For the devices in our data set, we looked at MAC address randomisation, and whether the devices adhere to a set of best privacy practises guidelines called 'Anonymity profiles for DHCP'. We found that the iPhones were the best in class, implementing MAC address randomisation by default, and only displaying a minimal violation of the anonymity profile. For the Android devices however, we found that the majority does implement MAC address randomisation, however all of the devices can be directly identified as Android devices by leaking the vendor-class. Moreover, a majority of the Android devices also leaked the hostname option, which often contains device and/or personal names. The laptops (including Macbooks) and desktops in our data set were the worst in class, providing neither MAC address randomisation, nor implementing any of the Anonymity profile guidelines.

7.5 The Full Picture

Over the chapters 3-6 we have seen what kind of information is leaked from devices through DHCP, and what this leakage looks like. The origins of this information is touched upon, as it turns out that this is either the device hostname set by the device owner, or pulled from a directory service like Active Directory. We see that the flow from the device passes through the DHCP server, is then either forwarded directly to the DNS using DDNS, or alternatively using an DDI layer that sits between the DHCP and DNS server providing additional policy services like rewrites and blocklists. Additional flows from devices to the DDI appliances also exists, often making use of network scanning techniques involving the LLDP and SNMP protocols.

7.6 Impact

This thesis presents a root cause analysis of what is a privacy and security issue at heart. We want to reiterate the impact of dynamic personal information in the reverse DNS:

- The reverse DNS holds a treasure trove of information. Not only does it contain information about the network's infrastructure, it also contains personal information, and the make and model of devices on the network.
- The dynamic nature of specific reverse address space allows for tracking user, and allows an attacker to effectively determine a time to stage a heist, as demonstrated by Toorn et al [1].
- The presence of device information in reverse DNS records would allow an attacker to use device-specific exploits for an attack.

- The reverse DNS allows for network reconnaissance without having to actively scan the entire network, we can simply iterate the IP address space much like we have showed in chapter 3.
- For networks that expose their DNS servers to the Internet, an attacker might gain internal network insight without being on that network.

7.7 Scale Of The Problem

To substantiate the last two points made in the previous section, we highlight the scale of the problem by referring to previous academic work:

Toorn et al. identified 134,451 /24 prefixes [1] on the Internet as displaying dynamic behaviour in the reverse DNS. This is already a staggering number of network addresses that expose hostnames.

This is however far from the full story as not all organisations own (large) IP address space that is globally rout-able. Common practise is to use reserved private address space within an organisation [68]. In these types of networks, the information leakage is minimised, as the leakage described in this thesis *should* remain localised to the private network.

In practise however, this is not always the case. Even if an IP address is not directly reachable from the global Internet, as long as the DNS server(s) on these networks are globally reachable, they can be queried for private address space. It makes little sense for a DNS server meant for internal name resolution to return this information to outside clients. However, Tatang et al. showed that up to 574,000 (4%) of globally reachable DNS servers return results for private address space [69].

We know that the University of Twente leaks information through the DNS for its public IP space. We also tested private address space, but found no results. However our sister institution, the Technical University Eindhoven, with whom we share DNS infrastructure[2], does leak private address space. We informed their IT department of this leakage. Their response was that this was 'intended behaviour' and not a security risk. However, they did inform us they were in the process of moving their DNS infrastructure, and this leakage would likely be resolved in the process.

7.8 Mechanisms For Mitigation

The question that remains now, is how do we improve the current situation? To that end, we will discuss some mitigation strategies. We will focus on DHCP and DDNS, as these are the main culprits in leaking data. We will split these in two categories:

[1] In this context, a /24 might hold up to 256 usable addresses [1]. This is due to the fact that these blocks might in practise not be actual /24 prefixes, but part of larger network blocks.

[2] ns3.utwente.nl is hosted at the Technical University Eindhoven, and vice versa.

- Mitigation inside the network and at the DNS server: the measures to be taken by network operators.
- Measures to be taken on the client side: Device settings, and hardware/software implementations.

7.8.1 Network Operators

The clear point to start is the DNS server itself. A network administrator should carefully weigh the need for dynamic (reverse) DNS records, versus the privacy implications. There are only a few technical reasons why a reverse DNS record is needed, namely:

- **Email servers** - In practise, Email servers require a reverse record to make sure emails sent by a server does not end up in the spam folder.
- **DNS Service Discovery (DNS-SD)**, A protocol used for service discovery in zero configuration networks [70].

Besides the aforementioned technical reasons for the reverse records, it is mostly used as a convenience for network administrators. In chapter 3 for example, we have seen the reverse DNS being used to store infrastructure related information, such as location, link type and more.

Reverse records for personal devices have no technical function as far as we are aware. As such, we recommend disabling the dynamic PTR zones that pull information from DHCP. This means either turning off DDNS inside the DHCP server directly, or configuring the DDI appliance to not forward this information to the DNS.

7.8.2 Client Side

On the client side, there are a few things end users can do:

- For devices that support it, make sure that features like MAC address randomisation, or 'Private Wi-Fi' are turned on. For most iPhone and Android phones this should already be the case. However Windows devices have MAC address randomisation turned off by default.
- Change the hostname of your device such that it excludes any device or personal identifying information.

The bad news is that the ball is mostly in the court of the device and OS vendors. In chapter 6 we have seen that there is significant work to be done to improve the situation:

- For laptops and desktops, MAC address randomisation is not enabled, or not possible in case of Macbooks. We argue that these devices see a degree of mobility (using it in an airport lounge) that warrants implementation of these features.
- Only the iPhone's 'Private Wi-Fi' feature takes DHCP anonymity profiles into account. Android, and laptop/desktop vendors need to implement these guidelines.

7.9 Closing Thoughts

In this thesis we have investigated how information ends up in the reverse DNS, and how information flows through the network. We found out that DHCP is the main culprit for information leakage to the DHCP and that there are few device vendors that fully consider the privacy implications of the DHCP traffic they generate. The good news is that this thesis has outlined the specific pain points, and where and how we can intervene to improve the situation.

Bibliography

[1] O. van der Toorn, R. van Rijswijk-Deij, R. Sommese, A. Sperotto, and M. Jonker, "Saving brian's privacy: The perils of privacy exposure through reverse dns," in *Proceedings of the 22nd ACM Internet Measurement Conference*, ser. IMC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1–13. [Online]. Available: https://doi.org/10.1145/3517745.3561424

[2] European Court of Human Rights, "Guide on article 8 of the european convention on human rights," aug 2022, accessed: 2023-05-02. [Online]. Available: https://www.echr.coe.int/documents/guide_art_8_eng.pdf

[3] G. Schmid, "Thirty years of dns insecurity: Current issues and perspectives," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2429–2459, 2021.

[4] "Domain names - concepts and facilities," RFC 1034, Nov. 1987. [Online]. Available: https://rfc-editor.org/rfc/rfc1034.txt

[5] D. Chang and Q. Zhang, "Study on os fingerprinting and nat / tethering based on dns log analysis," in *Proceedings of the 2015 IRTF ISOC Workshop on Research and Applications in Internet Measurements (RAIM)*, 2015, pp. 23–34.

[6] D. W. Kim and J. Zhang, "You are how you query: Deriving behavioral fingerprints from dns traffic," in *Security and Privacy in Communication Networks: 11th EAI International Conference, SecureComm 2015, Dallas, TX, USA, October 26-29, 2015, Proceedings 11*. Springer, 2015, pp. 348–366.

[7] M. Kirchler, D. Herrmann, J. Lindemann, and M. Kloft, "Tracked without a trace: linking sessions of users by unsupervised learning of patterns in their dns traffic," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, 2016, pp. 23–34.

[8] T. Matsunaka, A. Yamada, and A. Kubota, "Passive os fingerprinting by dns traffic analysis," in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, 2013, pp. 243–250.

[9] B. Liu, C. Lu, H. Duan, Y. Liu, Z. Li, S. Hao, and M. Yang, "Who is answering my queries: Understanding and characterizing interception of the dns resolution path," in *27th USENIX Security Symposium*, 2018, pp. 1113–1128.

[10] “Internet Protocol,” RFC 791, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc791

[11] P. Taylor, “Internet users worldwide 2023,” Jan 2023, accessed: 2023-05-01. [Online]. Available: https://www.statista.com/statistics/1190263/internet-users-worldwide/

[12] “Arin ipv4 free pool reaches zero,” Sep 2015, accessed: 2023-05-22. [Online]. Available: https://www.arin.net/vault/announcements/2015/20150924.html

[13] “The ripe ncc has run out of ipv4 addresses,” Nov 2019, accessed: 2023-05-22. [Online]. Available: https://www.ripe.net/publications/news/about-ripe-ncc-and-ripe/the-ripe-ncc-has-run-out-of-ipv4-addresses

[14] “Ipv4 exhaustion: Lacnic has assigned the last remaining address block,” Aug 2020, accessed: 2023-05-22. [Online]. Available: https://www.lacnic.net/4848/2/lacnic/ipv4-exhaustion:-lacnic-has-assigned-the-last-remaining-address-block

[15] “Ipv6 usage among google users,” accessed: 2023-05-22. [Online]. Available: https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption

[16] “Internet numbers,” RFC 1117, Aug. 1989. [Online]. Available: https://www.rfc-editor.org/info/rfc1117

[17] H. Eidnes, P. A. Vixie, and G. J. de Groot, “Classless IN-ADDR.ARPA delegation,” RFC 2317, Mar. 1998. [Online]. Available: https://www.rfc-editor.org/info/rfc2317

[18] D. T. Narten, T. Jinmei, and D. S. Thomson, “IPv6 Stateless Address Autoconfiguration,” RFC 4862, Sep. 2007. [Online]. Available: https://www.rfc-editor.org/info/rfc4862

[19] A. M. Costello, “Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA),” RFC 3492, Mar. 2003. [Online]. Available: https://rfc-editor.org/rfc/rfc3492.txt

[20] Wikipedia contributors. List of dns record types — Wikipedia, the free encyclopedia. Accessed: 2023-05-01. [Online]. Available: https://en.wikipedia.org/wiki/List_of_DNS_record_types

[21] D. Barr, “Common DNS Operational and Configuration Errors,” RFC 1912, Feb. 1996. [Online]. Available: https://rfc-editor.org/rfc/rfc1912.txt

[22] V. Ksinant, C. Huitema, D. S. Thomson, and M. Souissi, “DNS Extensions to Support IP Version 6,” RFC 3596, Oct. 2003. [Online]. Available: https://rfc-editor.org/rfc/rfc3596.txt

[23] IEEE Standards Association and others, “Standard group mac addresses: A tutorial guide,” 2011. [Online]. Available: http://standards.ieee.org/develop/regauth/tut/macgrp.pdf

[24] ——, “Guidelines for use of extended unique identifier (eui), organizationally unique identifier (oui), and company id (cid),” *IEEE: Piscataway, NJ, USA*, 2018.

[25] D. E. E. 3rd and J. Abley, “IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters,” RFC 7042, Oct. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc7042

[26] G. Weston, “Csec used airport wi-fi to track canadian travellers: Snowden documents — cbc news,” Jan 2014, accessed: 2023-05-01. [Online]. Available: https://www.cbc.ca/news/politics/csec-used-airport-wi-fi-to-track-canadian-travellers-edward-snowden-documents-1.2517881

[27] C. Huitema, T. Mrugalski, and S. Krishnan, “Anonymity Profiles for DHCP Clients,” RFC 7844, May 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7844

[28] A. Mamiit, “Apple implements random mac address on ios 8. goodbye, marketers,” Jun 2014, accessed: 2023-05-01. [Online]. Available: https://www.techtimes.com/articles/8233/20140612/apple-implements-random-mac-address-on-ios-8-goodbye-marketers.htm

[29] “Analysis of mac randomization schemes in wi-fi clients,” Sep 2020, accessed: 2023-05-01. [Online]. Available: https://wifihelp.arista.com/post/analysis-of-mac-randomization-schemes-in-wifi-clients

[30] L. Hattery, “How mac address randomization can affect the wi-fi experience,” Jul 2020, accessed: 2023-05-01. [Online]. Available: https://blog.elevensoftware.com/how-mac-address-randomization-can-affect-the-wifi-experience

[31] W. Purvis and S. Dementyev, “Get to know mac address randomization in 2020,” Oct 2020, accessed: 2023-05-01. [Online]. Available: https://www.mist.com/get-to-know-mac-address-randomization-in-2020/

[32] E. Fenske, D. Brown, J. Martin, T. Mayberry, P. Ryan, and E. Rye, “Three years later: A study of mac address randomization in mobile devices and when it succeeds,” *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 3, pp. 164–181, 2021.

[33] “Use private wi-fi addresses on iphone, ipad, ipod touch, and apple watch,” Mar 2023, accessed: 2023-05-01. [Online]. Available: https://support.apple.com/en-us/HT211227

[34] “Mac randomization behavior,” Oct 2022, accessed: 2023-05-01. [Online]. Available: https://source.android.com/docs/core/connect/wifi-mac-randomization-behavior

[35] “How to use random hardware addresses in windows,” Oct 2022, accessed: 2023-05-01. [Online]. Available: https://support.microsoft.com/en-us/windows/how-to-use-random-hardware-addresses-in-windows-ac58de34-35fc-31ff-c650-823fc48eb1bc

[36] R. Droms, “Dynamic Host Configuration Protocol,” RFC 2131, Mar. 1997. [Online]. Available: https://rfc-editor.org/rfc/rfc2131.txt

[37] M. W. Patrick, “DHCP Relay Agent Information Option,” RFC 3046, Jan. 2001. [Online]. Available: https://rfc-editor.org/rfc/rfc3046.txt

[38] I. Papapanagiotou, E. M. Nahum, and V. Pappas, “Configuring dhcp leases in the smartphone era,” in *Proceedings of the 2012 Internet Measurement Conference*, 2012, pp. 365–370.

[39] T. Lemon and B. E. Sommerfeld, “Node-specific Client Identifiers for Dynamic Host Configuration Protocol Version Four (DHCPv4),” RFC 4361, Feb. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4361

[40] R. Droms and S. Alexander, “DHCP Options and BOOTP Vendor Extensions,” RFC 2132, Mar. 1997. [Online]. Available: https://www.rfc-editor.org/info/rfc2132

[41] J. B. Littlefield, “Vendor-Identifying Vendor Options for Dynamic Host Configuration Protocol version 4 (DHCPv4),” RFC 3925, Oct. 2004. [Online]. Available: https://www.rfc-editor.org/info/rfc3925

[42] T. Rooney, *Introduction to IP address management*. IEEE, 2010.

[43] N. Spring, R. Mahajan, and D. Wetherall, “Measuring isp topologies with rocketfuel,” in *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM ’02. New York, NY, USA: Association for Computing Machinery, 2002, p. 133–145. [Online]. Available: https://doi.org/10.1145/633025.633039

[44] B. Huffaker, A. Dhamdhere, M. Fomenkov *et al.*, “Toward topology dualism: improving the accuracy of as annotations for routers,” in *International Conference on Passive and Active Network Measurement*. Springer, 2010, pp. 101–110.

[45] M. Luckie, A. Marder, M. Fletcher, B. Huffaker, and K. Claffy, “Learning to extract and use asns in hostnames,” in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 386–392. [Online]. Available: https://doi.org/10.1145/3419394.3423639

[46] B. Huffaker, M. Fomenkov, and K. Claffy, “Drop: Dns-based router positioning,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 5–13, 2014.

[47] M. Gharaibeh, A. Shah, B. Huffaker, H. Zhang, R. Ensafi, and C. Papadopoulos, “A look at router geolocation in public and commercial databases,” in *Proceedings of the 2017 Internet Measurement Conference*, 2017, pp. 463–469.

[48] Q. Scheitle, O. Gasser, P. Sattler, and G. Carle, “Hloc: Hints-based geolocation leveraging multiple measurement frameworks,” in *2017 Network Traffic Measurement and Analysis Conference (TMA)*, 2017, pp. 1–9.

[49] J. Chabarek and P. Barford, “What’s in a name? decoding router interface names,” in *Proceedings of the 5th ACM Workshop on HotPlanet*, ser. HotPlanet ’13. New York, NY, USA: Association for Computing Machinery, 2013, p. 3–8. [Online]. Available: https://doi.org/10.1145/2491159.2491163

[50] Y. Lee and N. Spring, “Identifying and analyzing broadband internet reverse dns names,” in *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 35–40. [Online]. Available: https://doi.org/10.1145/3143361.3143392

[51] M. Luckie, B. Huffaker, and k. claffy, “Learning regexes to extract router names from hostnames,” in *Proceedings of the Internet Measurement Conference*, 2019, pp. 337–350.

[52] “Domain Administrators Operations Guide,” RFC 1033, Nov. 1987. [Online]. Available: https://www.rfc-editor.org/info/rfc1033

[53] B. Volz, “The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN) Option,” RFC 4704, Oct. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4704

[54] S. O. Bradner, “Key words for use in RFCs to Indicate Requirement Levels,” RFC 2119, Mar. 1997. [Online]. Available: https://www.rfc-editor.org/info/rfc2119

[55] Deland-Han, “How to configure dns dynamic updates in windows server - windows server,” accessed: 2023-05-01. [Online]. Available: https://learn.microsoft.com/en-us/troubleshoot/windows-server/networking/configure-dns-dynamic-updates-windows-server-2003

[56] M. Fedor, M. L. Schoffstall, J. R. Davin, and D. J. D. Case, “Simple Network Management Protocol (SNMP),” RFC 1157, May 1990. [Online]. Available: https://www.rfc-editor.org/info/rfc1157

[57] R. Presuhn, “Management Information Base (MIB) for the Simple Network Management Protocol (SNMP),” RFC 3418, Dec. 2002. [Online]. Available: https://www.rfc-editor.org/info/rfc3418

[58] LAN/MAN Standards Committee and others, “Ieee standard for local and metropolitan area networks - station and media access control connectivity discovery,” *IEEE Std 802.1AB-2016 (Revision of IEEE Std 802.1AB-2009)*, pp. 1–146, 2016.

[59] A. Bierman and K. S. Jones, “Physical Topology MIB,” RFC 2922, Sep. 2000. [Online]. Available: https://www.rfc-editor.org/info/rfc2922

[60] “Lldp-med and cisco discovery protocol,” Jun 2006, accessed: 2023-05-01. [Online]. Available: https://www.cisco.com/en/US/technologies/tk652/tk701/technologies_white_paper0900aecd804cd46d.html

[61] G. Siegfried, “Solution comparison for enterprise dns, dhcp and ip address management (ddi) solutions,” accessed: 2023-05-01. [Online]. Available: https://www.gartner.com/en/documents/3905998

[62] Deland-Han, “Can’t turn on network discovery - windows client,” accessed: 2023-05-01. [Online]. Available: https://learn.microsoft.com/en-us/troubleshoot/windows-client/networking/cannot-turn-on-network-discovery

[63] F. Reynolds, “The ubiquitous web, upnp and smart homes,” *ONLINE] http://www. w3. org/2006/02/reynolds-paper. pdf*, 2006, accessed: 2023-05-01.

[64] P. Kim, B. Volz, and S. D. Park, “Rapid Commit Option for the Dynamic Host Configuration Protocol version 4 (DHCPv4),” RFC 4039, Mar. 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc4039

[65] T. Matsunaka, A. Yamada, and A. Kubota, “Passive os fingerprinting by dns traffic analysis,” in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, 2013, pp. 243–250.

[66] “What is dhcp fingerprinting?” accessed: 2023-05-14. [Online]. Available: https://efficientip.com/glossary/dhcp-fingerprinting/

[67] “Dhcp fingerprint detection,” accessed: 2023-05-14. [Online]. Available: https://docs.infoblox.com/space/nios84/44605748

[68] R. Moskowitz, D. Karrenberg, Y. Rekhter, E. Lear, and G. J. de Groot, “Address Allocation for Private Internets,” RFC 1918, Feb. 1996. [Online]. Available: https://www.rfc-editor.org/info/rfc1918

[69] D. Tatang, C. Schneider, and T. Holz, “Large-scale analysis of infrastructure-leaking dns servers,” in *Detection of Intrusions and Malware, and Vulnerability Assessment*, R. Perdisci, C. Maurice, G. Giacinto, and M. Almgren, Eds. Cham: Springer International Publishing, 2019, pp. 353–373.

[70] S. Cheshire and M. Krochmal, “DNS-Based Service Discovery,” RFC 6763, Feb. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc6763.txt

Appendix A

Scapy DHCP Option parser

This appendix contains part of the code used in the analysis of DHCP handshakes. The function contained in the listing below is able to parse the DHCP options as delivered by scapy to a structured format. Most notably it is able to parse the FQDN options, which scapy is not able to by itself, instead returning a byte array.

```
1  from scapy.all import *
2  """
3  Parses a DHCP scapy packet
4  Returns the DHCP options in dictionary format
5  """
6  def parse_dhcp_pkt(dhcp_pkt):
7    #Scapy delivers DHCP Options in a mixed type list.
8    #We iterate over this list, parse if neccesary,
9    #and finally convert to a dictionary.
10   parsed = {}
11   for option in pkt[DHCP].fields["options"]:
12
13     #Skip padding and end options
14     if option in [DHCPOptions[0], DHCPOptions[255]]:
15       continue
16
17     #Convert the non key-value options
18     if not isinstance(option, tuple):
19       parsed[option] = None
20       continue
21
22     #Parse the key-values pairs if neccesary
23     match option[0]:
24       case "message-type":
25         #Convert message type to textual
26         parsed[option[0]] = DHCPTypes[option[1]]
27
```

```
28          case "hostname":
29            #Convert bytes to string
30            parsed[option[0]] = option[1].decode()
31
32          case "client_FQDN":
33            #Parse FQDN options information to a dict
34            fqdn_dict = {}
35            fqdn = option[1]
36
37            #Flags always present
38            flags = fqdn[0]
39
40            #Server SHOULD NOT perform any DNS (N)
41            n_bit = (flags >> 3) & 1
42            fqdn_dict['n'] = n_bit
43
44            #Encoding: 1: wire, 0: ASCII (E)
45            e_bit = (flags >> 2) & 1
46            fqdn_dict['e'] = e_bit
47
48            #Server override (O)
49            o_bit = (flags >> 1) & 1
50            fqdn_dict['o'] = o_bit
51
52            #Server sets forward A (S)
53            s_bit = (flags >> 0) & 1
54            fqdn_dict['s'] = s_bit
55
56            #Rcodes always present - deprecated in spec
57            #Always 0 for client, 255 for server
58            rcode1 = fqdn[1]
59            fqdn_dict['rcode1'] = rcode1
60
61            rcode2 = fqdn[2]
62            fqdn_dict['rcode2'] = rcode2
63
64            #Domains not always present
65            #Might contain multiple, or single partially
66            #qualified without terminating 0x00
67            if len(fqdn) >= 3:
68              #Use the remaining bytes
69              #Split on 0x00 as FQDN seperator
70              domains_raw = fqdn[3:].split(b'0x00')
```

```
71            domains = [d.decode() for d in domains_raw]
72            fqdn_dict['domains'] = domains
73
74          parsed[option[0]] = fqdn_dict
75
76        case default:
77          #Do not parse, simply append to parsed options
78          parsed[option[0]] = option[1]
79    return parsed
```

Appendix B

Devices used for DHCP handshakes

This appendix contains a complete overview of the data used in chapter 6. This is not the raw data, but an intermediary version during processing fit for publishing in this thesis.

	Model	OS	MAC on initial connect	MAC change on reconnect	MAC change on disassociate	Client-identifier
1	Huawei p20 pro	Android 10	Global	✗	✗	mac
2	Nexus 6p	Android 10	Random (CID)	✗	✗	mac
3	Samsung s10	Android 10	Random	✗	✗	mac
4	Xiaomi pocophone f1	Android 10	Global	✗	✗	mac
5	Oneplus Nord n10	Android 11	Random	✗	✗	mac
6	Oneplus 8T	Android 11	Global	✗	✗	mac
7	Oneplus 9	Android 11	Random	✗	✗	mac
8	Samsung a50	Android 11	Random	✗	✗	mac
9	Samsung a52	Android 11	Random	✗	✗	mac
10	Samsung a72	Android 11	Random	✗	✗	mac
11	Samsung s21 5g	Android 11	Random	✗	✗	mac
12	Redmi note 11	Android 11	Global	✗	✗	mac
13	Samsung tab a7	Android 12	Random	✗	✗	mac
14	Samsung s20 fe	Android 12	Random	✗	✗	mac
15	Samsung s22 fe	Android 12	Random	✗	✗	mac
16	Surface go	Windows 10	Global	✗	✗	mac
17	HP Elitebook	Windows 10	Global	✗	✗	mac
18	HP elitedesk	Windows 10	Global	✗	✗	mac
19	HP Elitebook	Windows 11	Global	✗	✗	mac
20	iPhone SE	iOS 14	Random	✗	✓	mac
21	iPhone SE	iOS 15	Random	✗	✓	mac
22	iPhone 8	iOS 15	Random	✗	✓	mac
23	iPhone 13	iOS 15	Random	✗	✓	mac
24	Macbook pro	osx	Global	✗	✗	mac
25	Macbook pro	osx	Global	✗	✗	mac
26	Macbook air	osx	Global	✗	✗	mac

	Hostname	Vendor	FQDN	Parameter Request List	Other options
1	✓	android-dhcp-10	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	
2	✓	android-dhcp-10	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	80
3	✓	android-dhcp-10	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	
4	✗	android-dhcp-10	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	
5	✗	android-dhcp-11	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	
6	✗	android-dhcp-11	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	80
7	✓	android-dhcp-11	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	
8	✓	android-dhcp-11	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	80
9	✗	android-dhcp-11	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	
10	✓	android-dhcp-11	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	
11	✓	android-dhcp-11	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	
12	✓	android-dhcp-11	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	
13	✓	android-dhcp-12	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	
14	✓	android-dhcp-11	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	80
15	✓	android-dhcp-12	✗	1, 3, 6, 15, 26, 28, 51, 58, 59, 43, 114, 108	80
16	✓	MSFT 5.0	✗	1, 3, 6, 15, 31, 33, 43, 44, 46, 47, 19, 121, 249, 252	
17	✓	MSFT 5.0	✗	1, 3, 6, 15, 31, 33, 43, 44, 46, 47, 19, 121, 249, 252	
18	✓	MSFT 5.0	✗	1, 3, 6, 15, 31, 33, 43, 44, 46, 47, 19, 121, 249, 252	
19	✓	MSFT 5.0	✗	1, 3, 6, 15, 31, 33, 43, 44, 46, 47, 19, 121, 249, 252	
20	✗	✗	✗	1, 121, 3, 6, 15, 108, 114, 119, 252	
21	✗	✗	✗	1, 121, 3, 6, 15, 108, 114, 119, 252	
22	✗	✗	✗	1, 121, 3, 6, 15, 108, 114, 119, 252	
23	✗	✗	✗	1, 121, 3, 6, 15, 108, 114, 119, 252	
24	✓	✗	✗	1, 121, 3, 6, 15, 108, 114, 119, 252, 95, 44, 46	
25	✓	✗	✗	1, 121, 3, 6, 15, 108, 114, 119, 252, 95, 44, 46	
26	✓	✗	✗	1, 121, 3, 6, 15, 108, 114, 119, 252, 95, 44, 46	