An Attack-in-Depth Analysis of Multicast DNS and DNS Service Discovery

by Antonios Atlasis

aatlasis@secfu.net, @AntoniosAtlasis

April 2017

Abstract

Multicast DNS and DNS Service Discovery are two protocols used for Zero Configuration Networking purposes from several devices and various vendors. Due to their objective of assisting Zero Configuration Networking, these protocols, which assume a “cooperating participants” environment, have some inherent weaknesses, like the “generous” broadcasting of a lot of information, and the use of easily “spoofable” messages. While these problems have been identified and related research has been published, a complete and in-depth threat analysis of all the potential attacking vectors has not been presented yet. This paper aims at filling this gap by providing a thorough study of the attack surface of these two protocols. By following closely the RFC specifications, potential attack vectors and specific testing scenarios are identified, which are examined using real life implementations. Specifically, these attacks are tested against popular devices, implementations and Operating Systems by using a tool specifically developed for this purpose, both for IPv4 and IPv6 environments. As it is shown, if this “cooperating participants” environment cannot be guaranteed, the usage of such protocols should highly be reconsidered. Finally, specific countermeasures suitable for mitigating the identified threats are also proposed.

Keywords: DNS Service Discovery, multicast DNS, Zero Configuration Networking, Internet of Things.

An Attack-in-Depth Analysis of multicast DNS and DNS Service Discovery

by Antonios Atlasis

aatlasis@secfu.net, @AntoniosAtlasis

April 2015

1 Introduction

Multicast DNS (*mDNS*) and *DNS Service Discovery* (*DNS-SD*) are two protocols used as part of the *Zero Configuration Networking* process. *Zeroconf* (*Zero Configuration Networking*) refers to the ability to create an IP network with no manual configuration and no servers. The three primary technologies that comprise Zeroconf are the automatic resolution and distribution of computer host names, the numeric network address assignment to networked devices (link-local address auto-configuration), and the automatic location of network services.

mDNS provides the ability to perform DNS-like operations on the local link in the absence of any conventional unicast DNS server [RFC 6762]. *DNS-Based Service Discovery*, defined in [RFC 6763], is a service which allows clients to discover a list of named instances of a desired service in a specific domain using standard DNS queries. DNS-SD can be used with both unicast DNS and mDNS. When used with mDNS, DNS-SD can assist ZeroConf operation and replace the *AppleTalk* [RFC 6760].

mDNS and DNS-SD are used by several Apple products like Apple TV, many network printers, Linux distributions (via the Avahi daemon), devices like NAS (Network-Attached Storage),Google Chromecast, as well as a number of third party products for various Operating Systems (OS). For example, many OS-X network applications written by Apple use DNS-SD to locate nearby servers and peer-to-peer clients. DNS-SD is even one of the tools needed from services that provide remote access to home or work networks, like the "*Back to My MAC*" (BTMM) service [RFC 6760]. Windows 10 also include support for DNS-SD for applications. Moreover, Apple offers for downloading several Windows applications that use Bonjour (Apple's version of the Zeroconf) [Apple, 2016]. ZeroConf is being spread even further with the proliferation of the *Internet of Things* (*IoT*).

The extensive usage of these protocols in combination with some inherent weaknesses render them a suitable target for attackers at the local link. However, the local link is often not a trusted one, since concepts like the BYOD (*Bring Your Own Device*) encourage the usage of our personal devices to actually untrustworthy environments. Moreover, under specific circumstances, their weaknesses can also be exploited remotely mainly due to bad implementation.

Related research regarding mDNS and DNS-SD security issues has already been presented in the

literature. However, this is typically focused on the known, inherent weaknesses of the protocols, which mainly stem from the fact that the employed messages can easily be spoofed, or related with the wealth of information provided from these services. On the contrary, an extended and in depth threat analysis of the two protocols based on a thorough study of the related specifications has not yet been presented. This paper aims at filling this gap not only by providing such an analysis, but also by presenting results of tests against real life popular implementations of the two protocols. For the purpose of this study, a tool tailored to its needs was developed and and it is released as an open-source one to the public community.

This paper is organised as follows: In section 2, the basic background of mDNS and DNS-SD is briefly discussed, while in section 3 related work is summarised. A thorough threat analysis of the two protocols is presented in section 4 along with related experimental results, whilst potential mitigation techniques are discussed in section 5. Finally, in section 6 we draw our conclusions.

2 mDNS and DNS-SD Background

For reasons of completeness of this paper, some background information regarding mDNS and DNS-SD important for understanding the related threats are given in this section. For further and more detailed information, the reader should refer to [RFC 6762] and [RFC 6763] respectively.

2.1 mDNS Fundamentals

The mDNS requests and responses are sent via UDP from source port 5353 to destination port 5353. If this is not the case regarding the source port, mDNS packets MUST silently be ignored. Moreover, all mDNS responses SHOULD[1] be sent with IP TTL set to 255, but received mDNS packets with IP TTL lower than 255 do not have to be discarded.

Typically, mDNS messages may be up to the Maximun Transmission unit (MTU) of the physical interface. Placing multiple questions in a single query is allowed. Similarly, multiple responses can be sent in one multicast packet.

mDNS answers contain a Time-to-Live (TTL)[2] value that indicates for how many seconds this answer is valid. An DNS TTL=0 indicates that the corresponding record should be deleted.

Finally, [RFC 6762] foresees the suppression of mDNS packets under various cases.

2.1.1 The .local domain

As described in [RFC 6762], mDNS provides the ability to perform DNS-like operations on the local link in the absence of any conventional unicast DNS server. Names ending in “.local” have only local significance. The same is true for the link-local reverse mapping domains “254.169.in-addr.arpa” for IPv4 and “8-b.e.f.ip6.arpa” for IPv6. Each IP link has its own private “.local”. Any DNS query for a

1 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

2 The (m)DNS TTL should not be confused with the IP TTL.

name ending with “.local” MUST be sent to the multicast addresses 224.0.0.251 and FF02::FB for IPv4 and IPv6 respectively.

When there are not conventional DNS servers available, mDNS queries to resolve global names (i.e. that do not end with “.local”) MAY be sent. Therefore, the option to use mDNS for names not ending in “.local” SHOULD also be supported, but it SHOULD be disabled by default.

2.1.2 Unicast Queries and Resposnes

The most usual questions request multicast responses and for this reason, they are referred as “QM” questions; instead, if they request unicast responses they are referred as “QU” questions. A special flag in each query, the QU bit, denotes if this is a unicast query (the QU bit is set), or not (the QU bit is not set).

However, since it is possible for a unicast query to be received from a machine outside the local link, responders SHOULD check that the source address in the query packet matches the local subnet for that link and silently ignore the packet if not.

2.1.3 Probing and Announcing

Whenever an mDNS responder starts up or its connectivity has changed for any reason, it MUST perform two steps: Probing and Announcing. In *Probing*, the hosts sends an mDNS query asking to see if the resource records (e.g. a host's address record) to be announced are already in use. All probe queries SHOULD be done using query type “ANY” (255) to elicit answers. In the second step, *Announcing*, an mDNS responder sends unsolicited mDNS responses containing in the Answer section all of its newly registered resource records.

In case of a name conflict, [RFC 6762] describes a conflict resolution mechanism (discussed and exploited in the Threat Analysis section).

2.2 How DNS-SD works

DNS-SD allows clients to discover a list of instances of a desired service using standard DNS Queries. When used with mDNS, DNS-SD assists ZeroConf process.

2.2.1 Service Instance Enumeration

Using DNS-SD a client can discover the list of available instances of a given service type by sending a query for a DNS PTR record [RFC 1035] with a name of the form "<*Service*>.<*Domain*>"; such a query returns a set of zero or more PTR records giving service instance names, in the form: "<*Instance*>.<*Service*>.<*Domain*>".

The <Instance> portion of the Service Instance Name is a user-friendly name consisting of arbitrary Net-Unicode text. While this name is initially auto-configured, the user can later define a name of his choice.

The <Service> portion of the Service Instance Name consists of a pair of DNS labels following the convention for SRV records [RFC 2782]. The first label is an underscore character followed by the Service Name [RFC 6335]; the second label is either "_tcp" (for application protocols that run over TCP) or "_udp" (for all others). For further restrictions to Service names, as well as for examples, the reader can refer to Section 7 of [RFC 6763].

The <Domain> portion of the Service Instance Name specifies the DNS subdomain within which those names are registered; it may be ".local" (when mDNS is used), or a conventional unicast DNS domain name learned through some other mechanism, e.g. DHCP.

[RFC 6763] also foresees the usage of selective service instance enumeration, e.g. the use of subtypes, so as to narrow the set of results, when needed. For instance, in order to discover the web pages that provide management for printers, a request for “_printer._sub._http._tcp.<Domain> will provide the desired outcome (whilst a request for _http._tcp.<Domain> would return all the available web pages, not only the ones used for printer management purposes).

A list of registered DNS SRV Service Types can be found in [IANA 2017].

2.2.2 Obtaining Further Information for Available Services

After the service instance enumeration, DNS SRV records [RFC 2782] can be used to provide the target host and port where the service instance can be reached, while DNS TXT records [RFC 1035] provide additional information about this instance. Typically, every DNS-SD service instance has exactly one TXT record. It is possible though to have multiple TXT records to describe a single service instance.

2.2.3 Domain Discovery

DNS-SD can also be used for registration and browsing domain discovery. This can actually be achieved by using either mDNS or even unicast DNS PTR queries (when a DNS server is known) by employing five special Resource Record (RR) names reserved for this purpose. Starting in Mac OS-X v10.4 Tiger and Bonjour for Windows, a network administrator can set up a Bonjour name server to enable wide-area capable devices and applications to discover services anywhere in the world [Cheshire, n.d.b.]. Wide-Area Bonjour uses DNS Service Discovery [Cheshire, n.d.a.] along with DNS Update [RFC 2136] and TSIG security [RFC 2845].

3 mDNS and DNS-SD Security Issues – Related Work

3.1 Security Considerations of Related RFCs

The inherent weaknesses of mDNS (mainly) and DNS-SD were identified from the very beginning, initially from the corresponding RFCs themselves.

First, as explained in [RFC 6762], mDNS, due to its nature and its objective (to assist ZeroConf), assumes cooperating participants. If clients in a network cannot be sure that there are no antagonistic hosts on the same physical link, the cooperating participants need to use IPsec signatures and/or

DNSSEC (Domain Name System Security Extensions) signatures [RFC 4033].

One important note regarding mDNS is the “direct unicast queries to port 5353” case. As described in in [RFC 6762], in such a “rare case”, “responders SHOULD check that the source address in the query packet matches the local subnet of that link”, both for IPv4 and IPv6, and silently ignore the packet if not.

On the other hand, given that DNS-SD is just a specification for how to name and use records in the existing DNS system, it has no additional security requirements other than the ones already applied to DNS queries and updates [RFC 6763]. For instance, for cases where authenticity of information is important, DNSSEC should be used. Of course, when DNS-SD is used in conjunction with mDNS, the corresponding security considerations also apply.

Consequently, when used at the local link, mDNS and DNS-SD assume that all hosts are co-operative; otherwise, it is trivial for any participant at the local link a) to trigger related responses from other clients, and b) to spoof DNS-SD responses from whatever reason, e.g. Man-in-the-Middle (MiTM) attacks at the local link.

3.2 Related Work

To the best of the author’s knowledge, the first security concerns regarding mDNS were published in [GNUCitizen 2008a] and [GNUCitizen 2008b]. In these two posts it is described that mDNS can be used at Local Area Networks (LANs) both for reconnaissance/enumeration purposes, as well as for spoofing and MiTM attacks. As it is also explained, due to the multicast nature of the protocol, an attacker in a LAN with a single multicast packet can learn a plethora of useful things such as the available devices' versions and types, etc. Moreover, as far as mDNS injection is concerned, since the attacker does not have to manipulate the request but rather advertise itself as required for the purpose, mDNS attacks can be more effective and nastier than other related attacks. Especially in a situation where DNS requests cannot be sniffed and subsequently forged by an attacker, an mDNS spoofing attack can still work. As a proof of concept, a tool was also made available (whose link, however, nowadays seems to be broken).

The use of mDNS for reconnaissance purposes was also discussed in [Pickett, 2011] and [SpiderLabs, 2012]. A more elaborated and extended study regarding mDNS spoofing and MiTM attacks was presented in [Bai et. al., 2016], but it was mainly focused on Apple products; however, all the described attacks – although examined to a quite deep level – are based on the same concept of multicast unauthenticated messages. That said, other related threats are not examined.

On the contrary, in [VU 550620] an implementation vulnerability was reported. As it is described, in several implementation the systems responded to unicast queries that originated from sources outside of the local link network. Such responses can remotely disclose information about network devices or, even worse, to be used in denial-of-service (DoS) amplification attacks.

A set of mDNS tools for windows was published in [mDNSTools, 2014], and an mDNS Ruby script at [Perez, 2012]. Moreover, *Responder* [Gaffie, 2016] offers some (rather limited) mDNS poisoning capabilities.

4 Threat Analysis

The threat analysis that it will follow is based on a thorough analysis of mDNS and DNS-SD specifications as described in the corresponding RFCs. Inevitably these will include some already identified threats, but also some other completely new. Our final objective is to present a complete treat analysis of these two protocols to the best possible extent. These threats are presented in related groups, not necessarily following the flow of the corresponding RFCs.

The threat analysis is accompanied from experiments performed against several systems (Linux flavours, OS-X, Windows) and devices (Printers, Apple TV, Chromecast, Home speakers, NAS). As a testing tool, *Pholus* is used [Atlasis, 2017], which was specially developed to address all the threats identified in this study.

4.1 Reconnaissance and Network Scanning

4.1.1 Passive Reconnaissance

Passive reconnaissance is always an option. It has been examined in good detail in [Pickett, 2009]. The multicast nature of mDNS in combination with the rather “chatty” behaviour of some Operating Systems (OS) and devices allow an attacker to sit back, sniff the “wire” passively and collect all the provided information. As it will be shown later on as part of active reconnaissance attacks, this information can be as simple as the used IP address(es), up to the specific OS family or the CPU architecture. The same information is sometimes advertised unintentionally by some of the OS.

While passive reconnaissance may seem not very exciting from a technical perspective, the “Bring your Own Device” concept applied in airports, hotels, other public places offering wi-fi connections or even corporate environments with guest features render mDSN passive sniffing a stealthy and at the same time really effective reconnaissance technique. It is really amazing what information can be gathered just be sniffing the “wire” passively in such locations.

4.1.2 Active Reconnaissance

Instead of relying on passive reconnaissance, an impatient attacker can always try to trigger the desired responses for his reconnaissance purposes by sending the appropriate queries. However, taking into account that mDNS messages can easily be spoofed (the provided source MAC address and source IP address do not have to be the real ones, due to the fact that UDP is the underlying layer-4 protocol), an attacker can hide its identity during this scanning process.

These active attacks can get the following forms:

- Discovery of available services.

- Discovering instances of a specific service.
- Discovering DNS-SD clients-only - The _workstation service
- Querying a specific instance of a service.
- Domain enumeration.

The aforementioned attacks are discussed in the next sub-sections in more detail.

Discovery of available services

As explained in [RFC 6763], a DNS query for PTR records with the name "*_services._dns-sd._udp.<Domain>*" yields a set of PTR records where the rdata of each PTR record is the two-label <Service> name plus the same domain, e.g., "_http._tcp.<Domain>". Hence, by sending such a query, specified in the RFC “for problem diagnosis” purposes (i.e. a legitimate user would not try to identify all the available services but would query only for the services he is interested in), an attacker can automatically discover all the services advertised in the network. If he also spoofs this message, he can perform this reconnaissance in a stealthy way.

As far as the *qtype* of the queries is concerned, when responding to queries using *qtype* “ANY” and/or *qclass* ANY, a multicast DNS responder MUST respond with ALL of its records that match the query. Hence, the use of such a *qtype* is convenient for triggering all the corresponding responses.

The hosts that respond, they do so by sending standard query PTR responses of the service, e.g. *_http._tcp.local.* Sample results are depicted in the figure below.

```
*****************************************RESULTS*********************************************
00:08:9b:[redacted] 192.168[redacted] QUERY Answer: _services._dns-sd._udp.local. PTR Class:IN "_workstation._tcp."
00:11:32:[redacted] 192.168[redacted] QUERY Answer: _services._dns-sd._udp.local. PTR Class:IN "_adisk._tcp."
00:11:32:[redacted] 192.168[redacted] QUERY Answer: _services._dns-sd._udp.local. PTR Class:IN "_smb."
00:11:32:[redacted] 192.168[redacted] QUERY Answer: _services._dns-sd._udp.local. PTR Class:IN "_device-info."
00:11:32:[redacted] 192.168[redacted] QUERY Answer: _services._dns-sd._udp.local. PTR Class:IN "_afpovertcp."
00:11:32:[redacted] 192.168[redacted] QUERY Answer: _services._dns-sd._udp.local. PTR Class:IN "_http."
```

Figure 1: Obtaining a list of the services available in the network

As we can see from the above figure, an attacker by sending a single, spoofed packet can identify the types of the services offered by the domain.

It should be reminded that this behaviour is not really needed by mDNS/DNS-SD, but it is defined in the corresponding RFC just for “problem diagnosis” purposes.

Discovering instances of a specific service

After discovering the available types of services in the network, these can then be used to discover instances of these service types. A client can discover the list of available instances of a given service type using a query for a DNS PTR record [RFC 1035] with a name of the form "<*Service*>.<*Domain*>". Such a query returns a set of zero or more instance names. As an example, if

from the service listed in Figure 1 we would query the smb service, typically used for file sharing, we would get the following information:

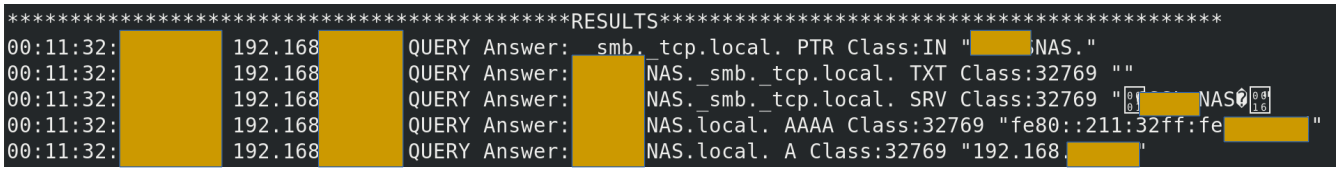

Figure 2: Discovering instances of a specific service

As we would expect, the smb service is used from a NAS device; it is notable that one of the information that we get is its IPv6 address (provided via the AAAA record).

Discovering DNS-SD clients only – The _workstation service

What about if there are no offered services at the local network? There is a service called *_workstation*, called “Workgroup Manager”, which is typically advertised by some OS by default and in such cases, it can be used for host identification. According to the *Avahi-daemon* man page [Avahi, 2017], registering a service of type "_workstation._tcp" on the LAN might be useful for administrative purposes (i.e. browse for all PCs on the LAN). As it is also explained, newer MacOS-X releases register a service of this type.

From an attacker’s perspective, when this service is advertised is really convenient because workstations can be identified even if they do not advertise any other typical DNS-SD service.

However, not all hosts advertise services and hence, their discovery via DNS-SD is a big challenge. One implicit method is described below in the Implicit Network Sweeping section (i.e. section 4.1.3). As it will be explained, this technique has been found to work against some Linux flavours.

Apart from that, for hosts not advertising any services the only chance to discover them is to send fake fake advertisements for popular services interesting for the target OS-es, expecting that these will send queries to identify their instances or information about them. These advertisements can also be spoofed so as to hide attacker’s identity.

Querying a specific instance of a service

After identifying the instances of a specific service, DNS SRV records [RFC 2782] can be used to provide the target host and port. Moreover, going one step further, DNS TXT records [RFC 1035] can be queried to provide additional information about this instance.

It should be noted that this can be achieved by sending a couple of spoofed multicast packets, without having to perform typical noisy port scanning. Hence, not only the attacker can remain undetected, but it is rather difficult even for the defenders to realise that there is an undergoing reconnaissance attack.

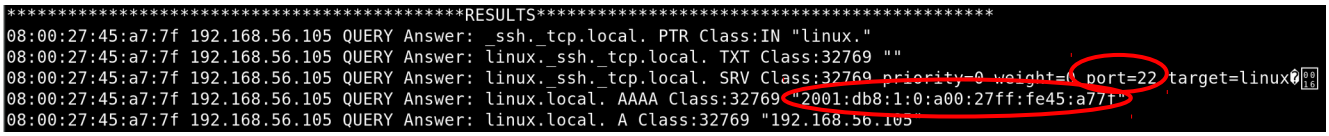

Figure 3.a: Information provided through SRV records

In Figure 3.a above, ssh service has been queried. The outcome shows that it is listening at port 22 (not surprising), the target is a Linux system and its IPv6 address is 2001:db8:1:0:a00:27ff:fe45:a77f.

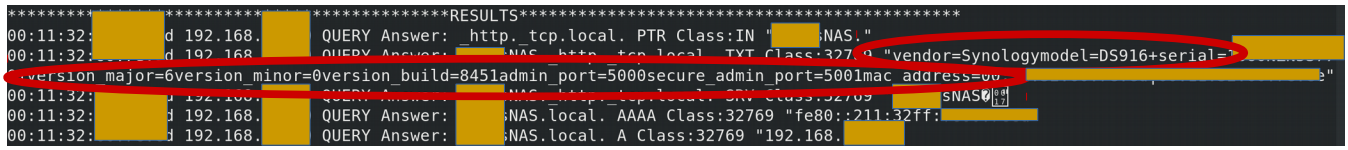

Figure 3.b: Information provided through TXT records

In Figure 3.b the NAS service presented on Figure 2 has been further queried. As we can clearly see, we can identify the vendor, the model, its serial number, the major and minor versions, the build, and the ports where NAS listens for admin connections.

Finally, information regarding the Operating System and the CPU architecture can also be obtained (Figure 3.c). In the presented example Home Speakers using Linux on an ARM7 and a Linux system of a 64-bit architecture are identified.

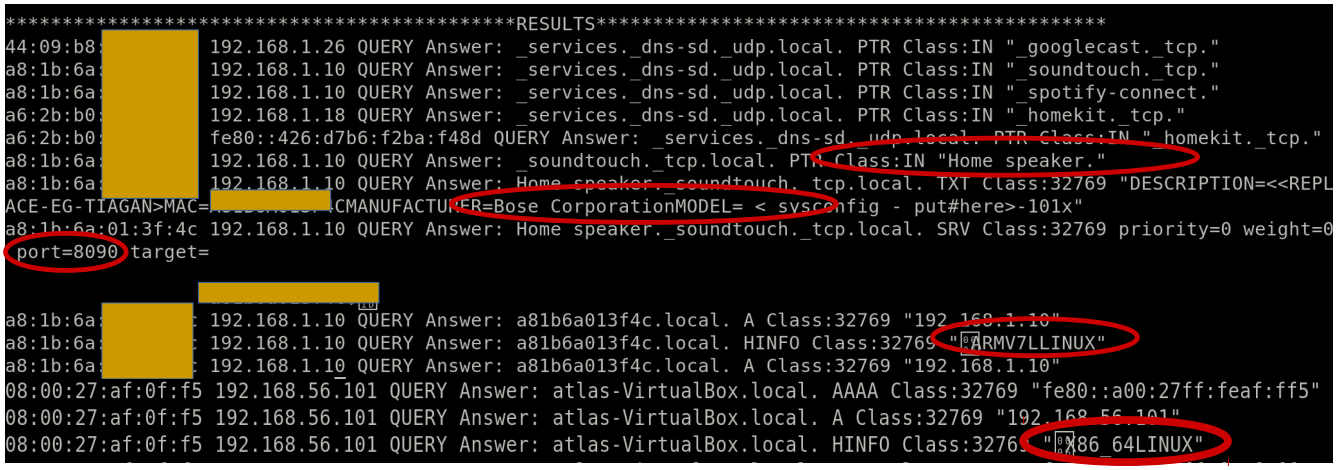

Figure 3.c: Information provided through HINFO records

Domain Enumeration

Using wide-area Bonjour Domains available for Browsing and Registration can be discovered by using PTR DNS queries over unicast or multicast DNS messages. By using the appropriate PTR queries, as explained in [RFC 6763] a client can learn:

- A list of domains recommended for browsing or a single recommended default domain for browsing.
- A list of domains recommended for registering services using Dynamic Update or a single recommended default domain for registering services.
- The "legacy browsing" or "automatic browsing" domain(s).

As an example, to discover the recommended automatic browsing domain(s) for devices on this subnet, the host issues a DNS PTR query for the name:

"lb._dns-sd._udp.0.0.168.192.in-addr.arpa."

Equivalent address-derived Domain Enumeration queries should also be done for the host's IPv6 address(es).

However, as further defined in [RFC 6763], address-derived Domain Enumeration queries SHOULD NOT be done for IPv4 link-local addresses [RFC 3927] or IPv6 link-local addresses [RFC 4862].

4.1.3 Implicit Network Sweeping

Some OS advertise their IP reverse mapping as a service. This can be exploited to find alive hosts in the network which do not advertise any DNS-SD services. Using single mDNS datagrams (which, however, incorporate multiple queries) an attacker can perform an implicit network sweeping of subnets and identify some potentially hiding hosts (e.g. which block ping or other requests). To achieve this, queries regarding DNS reverse mapping for IP addresses (e.g. for the “in-addr.arpa” domain for

IPv4) are sent. Specifically, an attacker can send mDNS requests by querying all the IP addresses in a subnet so as to get the corresponding response and hence, doing so to scan the whole subnet. As said, the mDNS queries can be included in a single mDNS datagram. However, to scan even a /24 subnet, the size of the mDNS packet will exceed that of the Ethernet MTU (1500 bytes); this is not really a problem, since mDNS packets (either over IPv4 or IPv6) can also be fragmented.

Examples of DNS reverse mapping information (both for IPv4 and IPv6) advertised through mDNS messages are depicted in Figure 4.

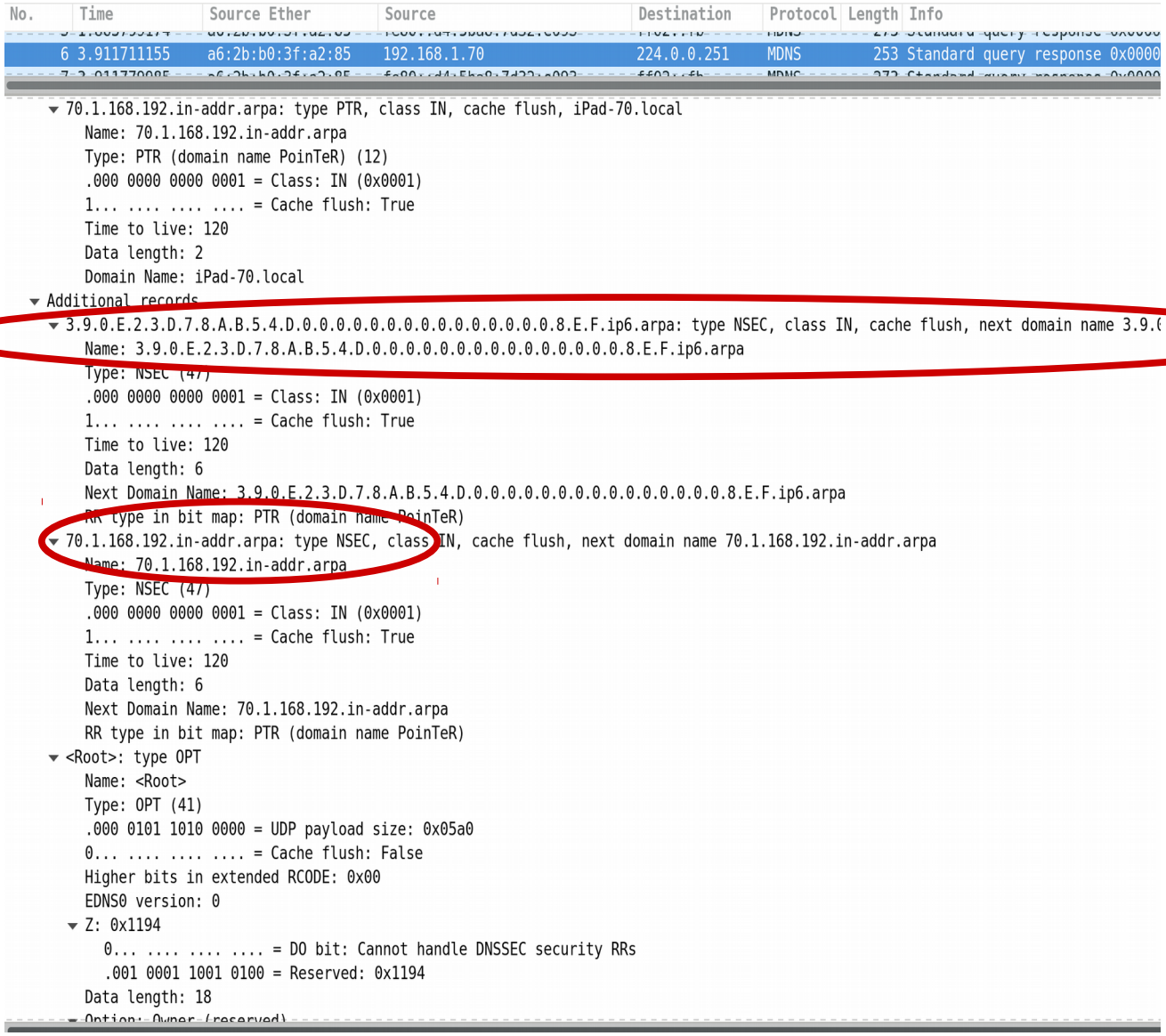

Figure 4: DNS reverse mapping advertised via mDNS and DNS-SD

4.2 Spoofed mDNS Responses and MiTM Attacks

This is an inherent problem of zero-configuration. By definition this issue is inevitable and has already been discussed widely. An mDNS querier typically takes the first response it receives and hence, the attacker has to win the race with the legitimate advertised service (this should not be that difficult for a determined attacker).

If this race is won be the attacker, it can lead to various related Man-in-the-Middle (MiTM) attacks. Of course, except from winning this race, the attacker has also to emulate/provide the fake service so as to complete his attack to a full extent. mDNS MiTM attacks are extensively discussed in [Bai et. al.,2016]

and therefore, they are not analyzed here in detail.

Some tricks that an attacker can use to increase the odds of achieving a successful MiTM attacks are the following:

- Send mDNS packets where the so called “cache-flush bit” (the most significant bit of the *rrclass* field of the Resource Record) of the newly announced records is set so as to cause old *rdata* to be flushed from peer caches.
- Maximise the *Priority* and the *Weight* values of the corresponding fields in the SRV records.
- Send periodically unsolicited spoofed responses.

In some real-life use cases asymmetric key verification is used to stop rogue services. Such an example is the verification of an AirPlay TV of an Airport Express from iTunes. However, with some effort the related keys can be extracted (see for example [Laird, 2001]) and the attack can be downgraded to a simple spoofing attack.

4.3 DNS Resolution of Global Names using mDNS

DNS queries that do not end with “.local” can also be sent using mDNS [RFC 6762]. Hence, resolving global names via local multicast is also allowed, e.g. as a fallback mechanism when unicast DNS resolution is not available.

In such a case, given that an mDNS querier typically takes the first response it receives, if a queried host deliberately transmits false information and wins the race condition, the querier will be deceived. Winning this race condition should not be that difficult though since OS are typically configured not to respond to mDNS queried for global domains.

Moreover, DNS (and consequently mDNS) responders may put for efficiency purposes additional records in the additional section of the DNS messages (i.e. records that a client did not explicitly request); it is client responsibility whether or not to trust them.

However, typically modern systems ignore DNS records passed back which are not directly relevant to a query. Finally, usually by default modern OS do not accept mDNS packets for unicast DNS resolution of non-local services. Therefore, the DNS cache poisoning risk by abusing mDNS is rather limited.

4.4 Denial of Service Possibilities

In this subsection, the Denial of Service (DoS) possibilities by abusing mDNS specifications defined in [RFC 6762] are discussed. DDoS (Distributed DoS) capabilities shall be discussed in subsection 4.6.

4.4.1 Setting DNS TTL=0

mDNS answers contain a Time-to-Live (TTL) value that indicates for how many seconds this answer is valid. A DNS TTL value equal to zero (0) indicates that the corresponding record has been deleted. For

instance, [RFC 6762] foresees that in the case of shared records, a host MUST send a "*goodbye*" announcement with an RR TTL equal to zero for the old rdata (i.e. an unsolicited mDNS response packet giving the same resource record name, rrtype, rrclass, and rdata, but an RR TTL of zero).

Given that the responses are unauthenticated and "spoofable" (i.e. they can easily be spoofed), this means that an attacker can easily DoS some services advertised via mDNS by creating fake responses "advertising" legitimate (existing) services just by setting DNS TTL equal to 0.

The attack can be more effective if combined with setting the so called *cache-flush* bit (i.e. the most significant bit of the rrclass field of the Resource Record, that is the cache-flush bit), especially as far as shared records are concerned.

Outcome of the Experiments

This DoS techniques was tested successfully against various systems and devices. It should be noted though that, as it is usually the case, this is a race between an attacker and the legitimate service. From an attacker's perspective, flooding the network with such TTL=0 mDNS messages may help him to win this race.

4.4.2 DoS using negative DNS records

As explained in section 6.1 of RFC 6762, there can also be cases where a responder has to respond with asserting the nonexistence of a record (that is, sending back a negative response) using a DNS NSEC record [RFC 4034]. In this case the NSEC record is not used for its usual DNSSEC security properties but, as a way of expressing which records do or do not exist with a given name.

According to the RFC, a responder MUST only generate negative responses to queries for which it has legitimate ownership of the name, rrtype and rrclass in the section. However, nothing stops an attacker from creating such fake negative response to denote their nonexistence and therefore, creating indirect DoS for the legitimate service.

4.4.3 Causing mDNS Suppression

RFC 6762 foresees the suppression of mDNS packets under various cases. Specifically, as described, mDNS packets MUST be suppressed in the following cases:

- An mDNS responder receives an mDNS query which in its "Known Answer" section includes the answer with an RR TTL at least half of the correct value the answer that this responder would send.
- An mDNS responder receives a response from another host that contains the same answer record and the TTL of that record is not less than the one this responder would send.
- An mDNS querier receives a query from another host that contains the "QM" question it wants to send and the "Known-Answer" section of that record does not contain any records that this host would not also put on it own "Known-Answer" section (i.e. when the already query queries

for the same resource records).

- When an mDNS query is sent with the TC (Truncated) bit set (this happens when "Known-Answers" do not fit in one mDNS packet), potential responders should wait for about half a second so as to give the margin for further related queries.

An attacker could combine the aforementioned use cases with unicast interaction so as to suppress specific mDNS responses from being sent to the network. For instance, knowing responses sent by specific mDNS responder, he could include these responses to a spoofed mDNS response using the proper TTL value. This fake response should be sent to the unicast address of the real mDNS responder, so as to ensure that other hosts will not receive that message.

4.4.4 Abusing the Probing Process

As explained in subsection 2.1.3, whenever an mDNS responder starts up, or its connectivity has changed for any reason, it sends a probe mDNS query asking to see if the resource records (e.g. a host's address record) going to announce are already in use by using query type "ANY" (255). Any answer containing a record in question MUST be considered as a conflicting one.

If a conflicting mDNS response is received, the probing host SHOULD choose new names for its conflicting records, as appropriate. If fifteen conflicts occur within any 10-second period, the host MUST wait at least five seconds before each successive additional attempt. After one minute of probing, if the mDNS responder has been unable to find any unused name, it should log an error message to inform the user or the operator.

By taking advantage of this procedure, an attacker can automatically send responses containing the record in question so as to force querier's configuration change. To make it effective, he can repeat this process continuously for any new queries it receives.

Outcome of the Experiments

Experiments showed that an attacker, by creating conflicting responses during the probing process of other hosts can cause:

a) Continuous name changes of the hosts being in the Probing process;

b) continuous flooding of the network, since some implementations do not give up and they continuously try to find out new names;

c) finally, implicit DoS of the specific services.

4.4.5 Setting the Cache-Flush Bit

The *cache-flush* bit (i.e. the most significant bit of the rrclass field of the Resource Record), is used to inform the recipients of a response that "this is an assertion that this information is the truth and the whole truth, and anything you may have heard more than a second ago regarding records of this name/rrtype/rrclass is no longer true" [RFC 6762]. Recipients MUST wait 1 second before flushing

from their cache the expired mDNS records (to give the chance to a sender to send many back-to-back related responses with a cache-flush bit in a burst, if needed).

An attacker can simply send several spoofed mDNS responses for existing records so as to override them. These new spoofed services may be implemented by the attacker, or not; in this last case, this results, once more, in DoS of this service. If it is implemented, it can enhance MiTM attacks (see subsection 4.2).

4.5 Flooding Attacks

4.5.1 Flood the Network with Queries for existing responses

As explained in [RFC 6762], a host can use a single message to probe for all of its resource records instead of needing a separate message for each. For example, a host can can simultaneously probe for uniqueness of its “A” record and all its SRV records in the same query message. When responding to queries using qtype “ANY” and/or qclass ANY, a multicast DNS responder MUST respond with ALL of its records that match the query.

Moreover, [RFC 6762] also foresees that in some cases mDNS responders MUST send at least two responses, one second apart, whilst to provide increased robustness, responders MAY send up to eight responses.

An attacker can exploit the aforementioned recommended behaviours, trying to combine the trigger of responses from all responders regarding the full set of their records, which, if combined with the responders’ “robustness” behaviour, will result in increased network traffic and flooding of the network.

Outcome of the Experiments

Experiments confirmed that under specific scenarios for some OS an amplification factor up to 8x can be achieved at the local link.

4.5.2 Flooding Using Responses with Low TTL

When the TTL value of a record cached to a host is about to expire, to perform cache maintenance, this host should plan to issue a query at 80% of the record lifetime, and then if no answer is received, at 85%, 90%, and 95%.

To exploit this behaviour, an attacker can advertise records with very low TTL value, so as to trigger frequent queries. These records can be updated regularly and frequently so as to continue triggering queries. Of course, the attacker can spoof its advertised records so as to avoid detection.

4.6 Interacting with a Target Off-Link

As explained, all mDNS responses (including responses sent via unicast) SHOULD be sent with IP TTL set to 255; however, non-conforming packets do not have to be discarded. To ensure that off-link

mDNS messages are not received and accepted, [RFC 6762] defines that a host sending mDNS queries to a link-local destination address (including the 224.0.0.251 and FF02::FB link-local multicast addresses) MUST only accept responses to that query that originate from the local link, and silently discard any other response packets.

The test whether a response has originated on the local link is performed as following:

- All responses received with a destination address in the IP header that is the mDNS IPv4 link-local multicast address 224.0.0.251 or the mDNS IPv6 link-local multicast address FF02::FB are necessarily deemed to have originated on the local link, regardless of source IP address.
- For responses received with a unicast destination address in the IP header, the source IP address in the packet is checked to see if it is an address on a local subnet.

So, an attacker, to interact with a target remotely using an mDNS, the following conditions has to be met:

1. The target to accept mDNS packets with IP TTL < 255; this should not be a problem if the OS conform with the RFC recommendation.
2. The target to accept (unsolicited) unicast mDNS responses.
3. The target to accept IP packets coming from a non link-local address (in case of IPv6).
4. The target not to conform with the RFC statement that they should only accept mDNS packets originating from the local link, and silently discard any other packets. Actually this is the only condition where the RFC recommendation must be violated.

Of course, in addition to the above, for remote (off-link) interactions, the intermediate firewalls has to allow incoming mDNS packets (UDP port 5353); however, in this study we examine the related threats from the protocol perspective only.

4.6.1 Outcome of Performed Experiments

A first report regarding off-link (remote) mDNS interaction was reported with [VU 550620]. As it was described, some implementations of mDNS (like Avahi mDNS, Canon, Hewlett-Packard, IBM and Synology systems) do however respond to unicast queries originating outside the local link.

Several performed tests confirmed that nowadays the majority of the issues have been resolved, for instance as far as Avahi daemon is concerned. However, it was found out that there are still embedded systems, e.g. some home speakers from a very prestigious vendor (CVE-2017-6520) or other home devices which are still vulnerable.

Therefore, it is anticipated that there are a lot of embedded “Internet-of-Things” devices which are still vulnerable to off-link mDNS interaction. As the results of a popular search engine, Shodan[3], showed as

3 https://www.shodan.io/

of March 2017, there are more than 959000 hosts which listen to UDP port 5353. Of course, this does not necessarily mean that all these devices are vulnerable; however, we believe that there should be quite good chances to find among them a considerable amount of vulnerable devices.

Moreover, it has historically been proven (e.g. ping-of-death issue, or IP fragmentation) that sometimes problems re-appear over a different protocol. Indeed, during the experiments it was found out that implementations which were found to be vulnerable in the past [VU 550620] and are not vulnerable any more over IPv4, they are still vulnerable when mDNS is used over IPv6 (CVE-2017-6519). Therefore, it seems that the specific vendors fixed the issue when IPv4 is used, but not for the latest IP protocol. More information will be published soon at https://www.secfu.net/advisories-1/ .

4.6.2 Consequences of an Off-Link Interaction

If an OS or any other system is proven to be susceptible to such a remote, off-link mDNS interaction, the following security consequences are possible.

- Remote DDoS (Distributed Denial of Service) amplification attacks, by combining spoofed requests from a single host (which will be the target of the attack) to multiple ones. The request receivers will answer with their responses which, eventually, will flood the spoofed originator.
- Remote information leakage (like the results demonstrated in 4.1.2, e.g. supported services and ports, OS used and corresponding architectures, etc.).
- Remote DoS of specific services by using various means described in 4.4 (e.g. by setting TTL equal to 0).

4.7 Overflow Attacks

As explained in [RFC 6762], an mDNS packet larger than the interface MTU which is sent using fragments MUST not contain more than one resource record. However, even when fragmentation is used, an mDNS packet MUST NOT exceed 9000 bytes.

The aforementioned limitation restricts a DNS TXT record, which can be up to 65535 bytes long, to 8900 bytes [RFC 6763]. Nevertheless, using TXT records larger than 1300 bytes is not recommended.

Typically, every DNS-SD service instance has exactly one TXT record. It is possible though, but not often, to have multiple TXT records to describe a single service instance. Specifically, in such a case each TXT record describes a different variant of the same logical service, which is offered using the same underlying protocol on the same port, and it is described by the same SRV record.

4.7.1 Outcome of related experiments

In practice, it was found out that despite the aforementioned RFC limitations and the related recommendations, at least Avahi daemon responds to about 59000 bytes queries at a minimum. This implies that in practice there is no limitation regarding neither the size of TXT records nor the size of the mDNS packets. Moreover, it was found out that these large mDNS packets could be comprised of

either one huge TXT records or of too many ones.

The aforementioned behaviours inevitably leaves a lot of room regarding data exfiltration, command and control activities (when combined with off-link interaction), etc.

5 Mitigation Techniques and Countermeasures

In the previous chapter several attacks were discussed, from simple reconnaissance, to MiTM attacks, remote interaction, DoS and DDoS attacks.

The provision of a wealth of information regarding a system is inevitable, as long as mDNS is used. However, when Zeroconf is not required, mDNS should be disabled. For instance, as discussed Linux systems come with Avahi daemon installed and enabled by default and in some cases, the workstation service is advertised. However, this is not typically needed. Therefore, this service should be disabled. Similar hardening preventive measures should also be taken for other systems as well, when needed.

To protect your network from vulnerable mDNS implementations that allow off-link interaction, the most effective mitigation technique continues to be the simplest one: the filtering of UDP port 5353, both for incoming and outgoing connections. This simple but effective countermeasure will prevent a set of threats exploited over the remote interaction. Of course, vendors should also ensure that they implement the related specifications following closely the RFC recommendations. No preventive measure should be underestimated and ignored.

Furthermore, to avoid device impersonations and MiTM attacks, devices offering mDNS/DNS-SD services (like printers, multimedia devices, etc.) should use signed certificates from a trusted PKI Certificate Authoity. Applications should only connect to devices with valid certificates. This should not be difficult for devices manufactured from well-known vendors, which can also force developers to validate their certificates upon connection.

Finally, as far as end-user devices are concerned, and in order to avoid deliberate conflicts or other related attacks, a unique host identifier is needed such as when the same identifier is detected to be used from other devices, it should be assumed with a great confidence that this is a deliberate attempt. To this end IPv6 Unique Local Addresses (ULAs) [RFC 4193] encoded as an AAAA Resource Record could be used, in a way similar to the one used in [RFC 6281].

6 Conclusions

In this paper a complete, in-depth threat analysis of two protocols used for ZeroConf, mDNS and DNS-SD, was presented. As it was explained in detail and demonstrated using related experiments, the related attacks cover a wide range, from detailed reconnaissance, to Denial of Service, Distribute Denial of Service, network flooding, remote unicast interaction and Man-in-the-Middle attacks. Therefore, despite the uselessness of these two protocols for ZeroConf, taking into account that the “cooperating participants” environment cannot be guaranteed in the “Internet of Things” and “Bring

your own device” era, it is suggested that their design and implementation should be reconsidered. Whilst some of the typical mitigation techniques, like blocking the corresponding network ports, are also applicable, it is suggested that to handle properly some the discussed attacks, additional measures that enhance security but do not break ZeroConf should be added to the specifications. For instance, the use of trusted certificates from devices advertising services over mDNS and DNS-SD should be considered. Moreover, automated unique host identification methods like IPv6 ULAs should also be examined. The last one is already used by services like “Back to my MAC”. We believe that this type of measures could reduce or even eliminate the risk of related threats without breaking ZeroConf. It depends on the major vendors and the IETF community whether such security enhancements should be added to the specifications.

References

[Apple, 2016], Apple Inc. https://support.apple.com/downloads/Bonjour_for_Windows (last accessed in 2nd April 2017).

[Atlasis, 2017], “Pholus: An mDNS and DNS-SD security assessment tool”, https://www.secfu.net/tools-scripts/ (last accessed in 26th March 2017).

[Avahi, 2017], avahi-daemon.conf - Linux man page, https://linux.die.net/man/5/avahi-daemon.conf (last accessed in 26th March 2017).

[Bai et. al., 2016], X. Bai, L. Xing, N. Zhang, X. Wang, X. Liao, T. Li, S. Hu, “Staying Secure and Unprepared: Understanding and Mitigating the Security Risks of Apple ZeroConf”, IEEE Symposium on Security and Privacy, 2016.

[Cheshire, 2009], Stuart Cheshire, “Bonjour’s DNS-Based Service Discovery”, Applications Area Open Meeting, 22nd March 2009.

[Cheshire, 2016], Stuart Cheshire, “DNS SRV (RFC 2782) Service Types”, http://www.dns-sd.org/ServiceTypes.html (last accessed in 23rd June 2016).

[Cheshire, n.d.a.], Stuart Cheshire, “DNS Service Discovery (DNS-SD)”, http://www.dns-sd.org (last accessed in 14th November 2016).

[Cheshire, n.d.b.], http://www.dns-sd.org/ServerSetup.html], “Setting up a Bonjour Name Server”, http://www.dns-sd.org/ServerSetup.html (last accessed in 14th November 2016).

[Gaffie, 2016], L. Gaffie, “Responder”, https://github.com/SpiderLabs/Responder, September 2016 (last accessed in 14th November 2016).

[GNUCitizen 2008a], GNUCitizen, “Name (mDNS) Poisoning Attacks Inside The LAN”, 23 Jan 2008, http://www.gnucitizen.org/blog/name-mdns-poisoning-attacks-inside-the-lan (last accessed in 14th November 2016).

[GNUCitizen 2008b], GNUCitizen, “DHCP/mDNS Injection Issues”, 27 Jan 2008, http://www.gnucitizen.org/blog/dhcpmdns-injection-issues/ (last accessed in 14th November 2016).

[IANA, 2017], IANA, “Service Name and Transport Protocol Port Number Registry”, https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml, 17 March 2017.

[Laird, 2011], James Laird, “[vlc-devel] RAOP/Airtunes”, https://mailman.videolan.org/pipermail/vlc-devel/2011-April/079148.html, 8 April 2011 (last accessed in 26th March 2017).

[mDNSTools, 2014], (unknown), https://sourceforge.net/projects/mdnstools/, October 2014 (last accessed in 14th November 2016).

[MITMf, 2016], (unknown), “MITMf - Framework for Man-In-The-Middle attacks”, https://github.com/byt3bl33d3r/MITMf, June 2016 (last accessed in 14th November 2016).

[Perez, 2012], C. Perez, “MDNSRecon”, http://www.darkoperator.com/blog/2012/7/11/mdnsrecon.html, July 2012 (last accessed in 14th November 2016).

[Pickett, 2011], G. Pickett, “Port Scanning Without Sending Packets”, DEFCON-19, Las Vegas, 2011.

[RFC 1035], P. Mockapetris, “Domain Names – Implementation and Specification”, IETF RFC 1035, November 1987.

[RFC 2119], S. Bradner, “Key words for use in RFCs to Indicate Requirement Levels”, IETF RFC 2119, March 1997.

[RFC 2132], S. Alexander, R. Droms, “DHCP Options and BOOTP Vendor Extensions”, IETF RFC 2132, March 1997.

[RFC 2136], P. Vixie, S. Thomson, Y. Rekhter, J. Bound, “Dynamic Updates in the Domain Name System (DNS UPDATE)”, IETF RFC 2136, April 1997.

[RFC 2782], A. Gulbrandsen, P. Vixie, L. Esibov, “A DNS RR for specifying the location of services (DNS SRV)”, IETF RFC 2782, February 2000.

[RFC 2845], P. Vixie, O. Gudmundsson, D. Eastlake 3rd, B. Wellington, “Secret Key Transaction Authentication for DNS (TSIG)”, IETF RFC 2845, May 2000.

[RFC 3397], B. Aboba, S. Cheshire, “Dynamic Host Configuration Protocol (DHCP) Domain Search Option”, IETF RFC 3397, November 2002.

[RFC 3927], S. Cheshire, B. Aboba, E. Guttman, “Dynamic Configuration of IPv4 Link-Local Addresses”, IETF RFC 3927, May 2005.

[RFC 4033], R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, “DNS Security Introduction and Requirements”, IETF RFC 4035, March 2005.

[RFC 4034], R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, “Resource Records for the DNS Security Extensions”, IETF RFC 4034, March 2005.

[RFC 4193], R. Hinden, B. Haberman, “Unique Local IPv6 Unicast Addresses”, IETF RFC 4193, October 2005.

[RFC 4862], S. Thomson, T. Narten, T. Jinmei, “IPv6 Stateless Address Autoconfiguration”, IETF RFC 4862, September 2007.

[RFC 6106], J. Jeong, S. Park, L. Beloeil, S. Madanapalli, “IPv6 Router Advertisement Options for DNS Configuration”, IETF RFC 6106, November 2010.

[RFC 6281], S. Cheshire, Z. Zhu, R. Wakikawa, L. Zhang, “Understanding Apple's Back to My Mac (BTMM) Service”, IETF RFC 6281, June 2011.

[RFC 6335], M. Cotton, L. Eggert, J. Touch, M. Westerlund, S. Cheshire, “Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry”, IETF RFC 6335, August 2011.

[RFC 6760], S. Cheshire, M. Krochmal, “Requirements for a Protocol to Replace the AppleTalk Name Binding Protocol (NBP)”, February 2013.

[RFC 6762], S. Cheshire, M. Krochmal, “Multicast DNS”, IETF RFC 6762, February 2013.

[RFC 6763], S. Cheshire, M. Krochmal, “DNS-Based Service Discovery”, IETF RFC 6763, February 2013.

[SpiderLabs, 2012], “mDNS - Telling the world about you (and your device)”, 10 October 2012, https://www.trustwave.com/Resources/SpiderLabs-Blog/mDNS---Telling-the-world-about-you-(and-your-device) (last accessed in 14th November 2016).

[VU 550620] Vulnerability Note VU#550620, “Multicast DNS (mDNS) implementations may respond to unicast queries originating outside the local link”, https://www.kb.cert.org/vuls/id/550620, CERT, 15 May 2015.