SharkFest '19 Europe

Plaintext Information in Network Protocols

Green Locks Are Not Enough

Simone Mainardi

mainardi@ntop.org

#sf19eu • Palacio Estoril Hotel, Estoril, Portugal • Nov 4 - 8

About me

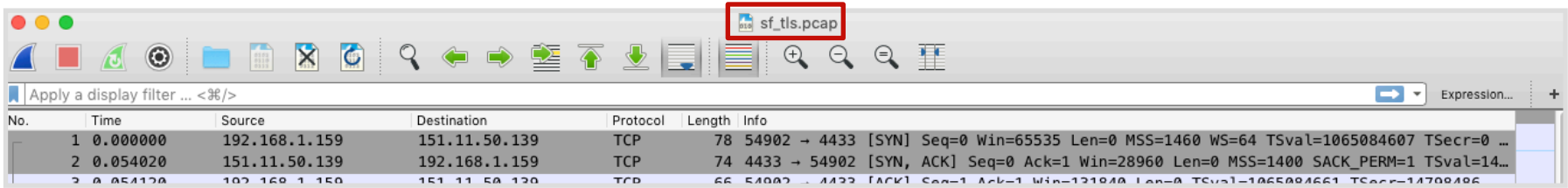
- Simone Mainardi
- Engineer, PhD born in 1986
- Joined Luca Deri and ntop in late 2015
- Used to be a pure data scientist
- Now more close to a software developer

Agenda

- Introduction and motivation
  - Encrypted but not so encrypted
  - Secure but not so secure
- Plaintext information in network protocols
- Discussion and conclusion

Following Along

- Pcaps and docs available at
  - https://bit.ly/328e5As
- Screenshots shown during the presentation, look at the filename!

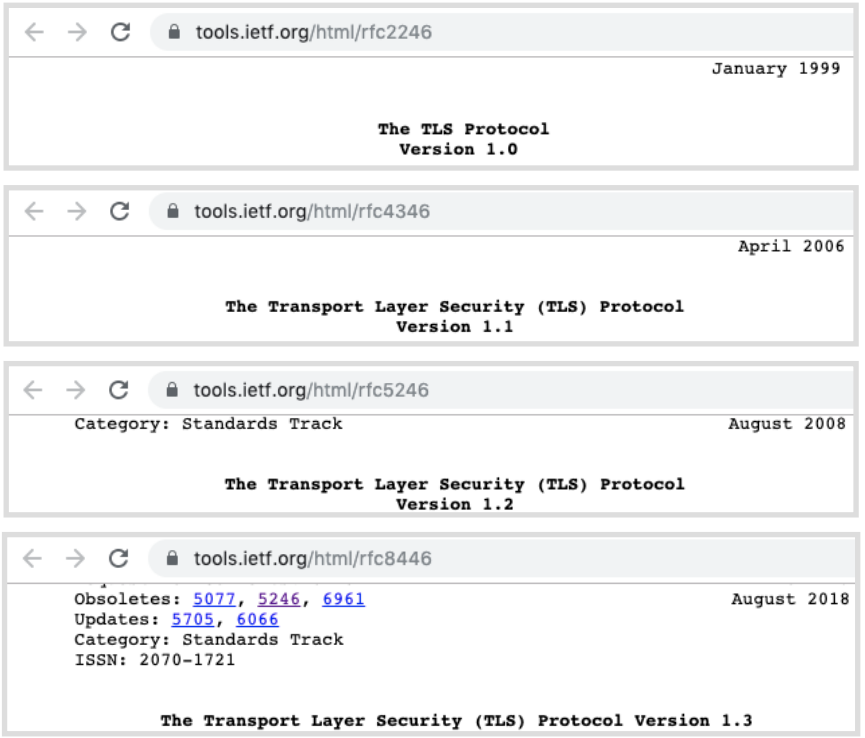


Encrypted Network Traffic

- Encryption is increasingly used in network protocols
- Fundamental to protect
  - Internet browsing
  - Online transactions
  - Instant messaging
  - Email
  - VoIP
  - ...

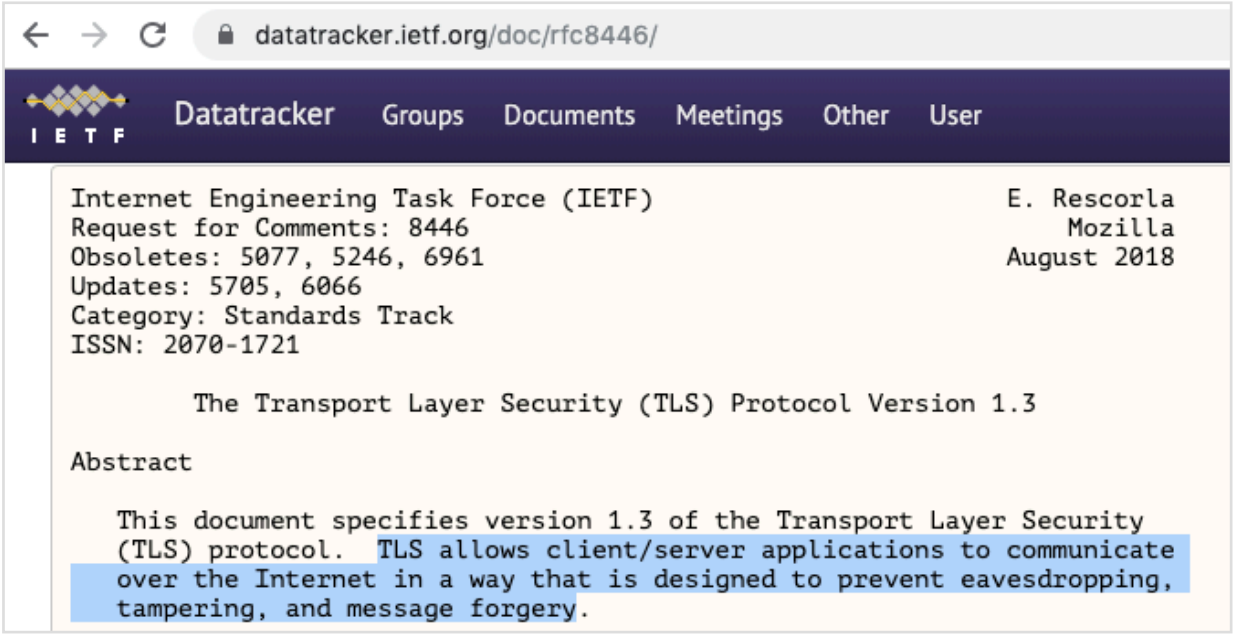
Cryptographic Protocols

- Cryptographic protocols necessary for the encryption of network communications
- Most popular is the Transport Layer Security (TLS)
- ~20 years since TLS 1.0



What is the TLS

- Cryptographic protocol providing end-to-end communication security over the networks

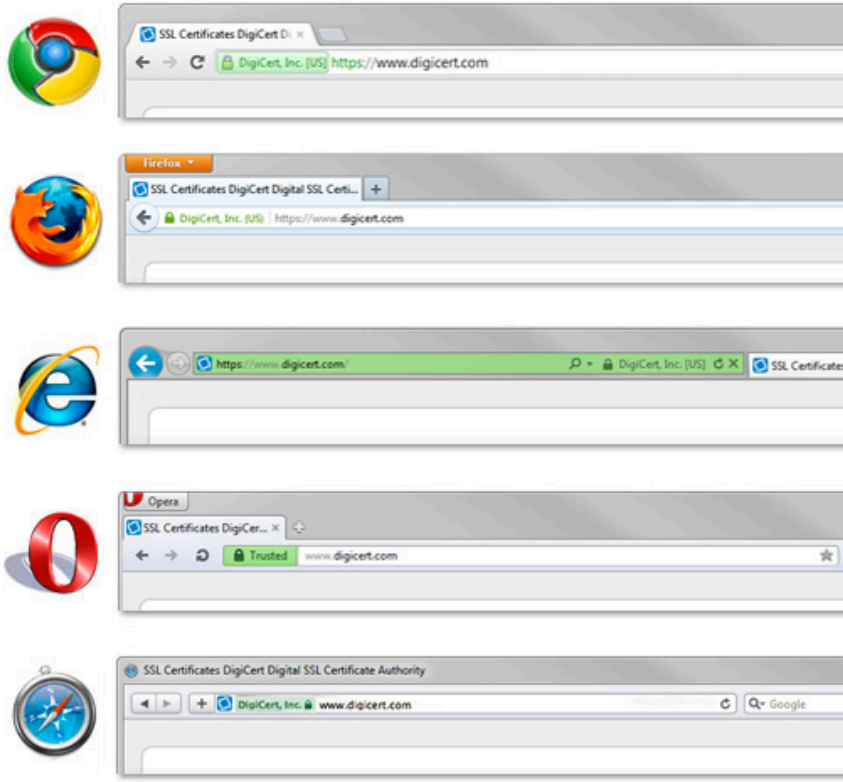


Where is the TLS

- Implemented in libraries and network applications
  - OpenVPN and other VPN tools
  - Quick UDP Internet Connections (QUIC)
  - Web Browsers (Chrome, FF, Opera, IE, ...)
  - Web Servers (Apache2, nginx, ...)
  - …

TLS and HTTPS

- Probably everyone has experience with HTTPS
- HTTPS is HTTP transported over TLS
- Browsers and websites that use HTTPS are employing TLS encryption



Encryption Makes Us Feel Secure

- We feel secure when we know our traffic is encrypted
  - "No one can look at it!"
- We feel secure when we see the locks or a comfortable light-green while browsing the web
  - "It's something private just between me and the website!"
- But actually...

Fact #1: Encrypted != Secure [1/2]

- **Encrypted != Secure**
  - A secure communication must be encrypted
  - An encrypted communication is not necessarily secure
- Security depends on the cryptographic protocol (e.g., TLS), on its implementation (bugs?), on how cryptographic keys are managed, ...

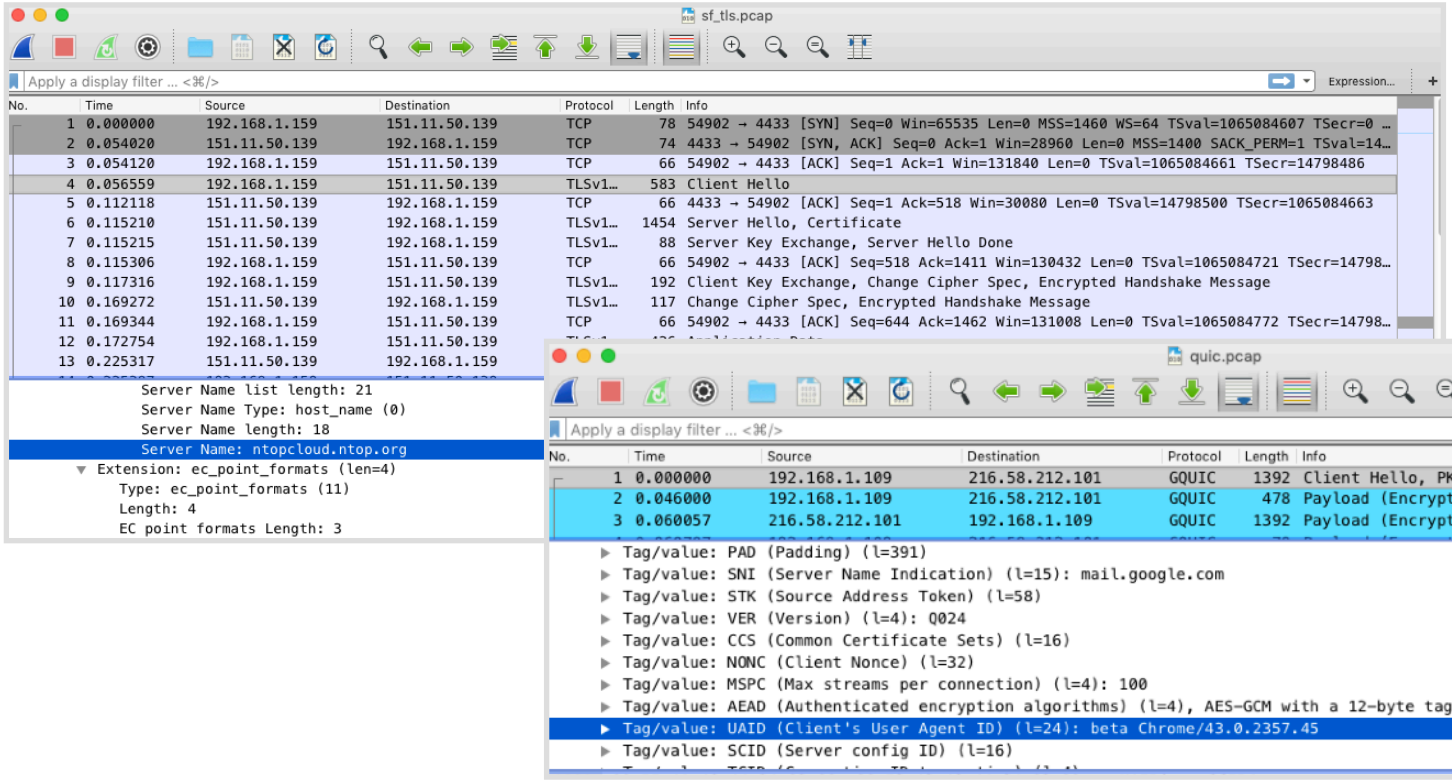
Fact #1: Encrypted != Secure [2/2]

- Secure if...
  - ...the data being transferred is encrypted?
  - ...the parties exchanging information are who they claim to be?
  - ...the data has not been forged or tampered?
- TLS has vulnerabilities and is subject to attacks - as basically any other protocol

Fact #2: Plaintext Information in Encrypted Network Protocols [1/2]

- Cryptographic protocols or protocols that support encryption may carry certain **plaintext information**
- They will do that - almost surely - at least during the initial setup phase
  - Initial TLS handshake
  - Quick UDP Internet Connections (QUIC) or Google quick

Fact #2: Plaintext Information in Encrypted Protocols [2/2]



Fact #3: Plaintext Information in Network Protocols [1/2]

- Still a great deal of network protocols are **plaintext** or carry **plaintext information**
- Computers - and network protocols - have born and evolved when security was not an issue
  - Small, local networks (e.g., university labs) in which all the participants were trusted
  - Build something that 'just works'

Fact #3: Plaintext Information in Network Protocols [2/2]

- Even today when security is a main concern, certain network protocols didn't evolve in that sense
- Among the most common protocols which disseminate plaintext information there are
  - DHCP
  - DNS and mDNS
  - SSDP

This Talk is About...

- Fact #1: **Encrypted != Secure**
- Fact #2: Cryptographic protocols or protocols that support encryption may carry certain plaintext information
- Fact #3: Still a great deal of network protocols carry plaintext information

What is this Talk NOT About

- This talk is NOT about
  - Cryptographic protocols
  - TLS vulnerabilities / attacks / pitfalls
  - Network Encryption / Decryption
  - SSL Man-In-The-Middle

What is this Talk About

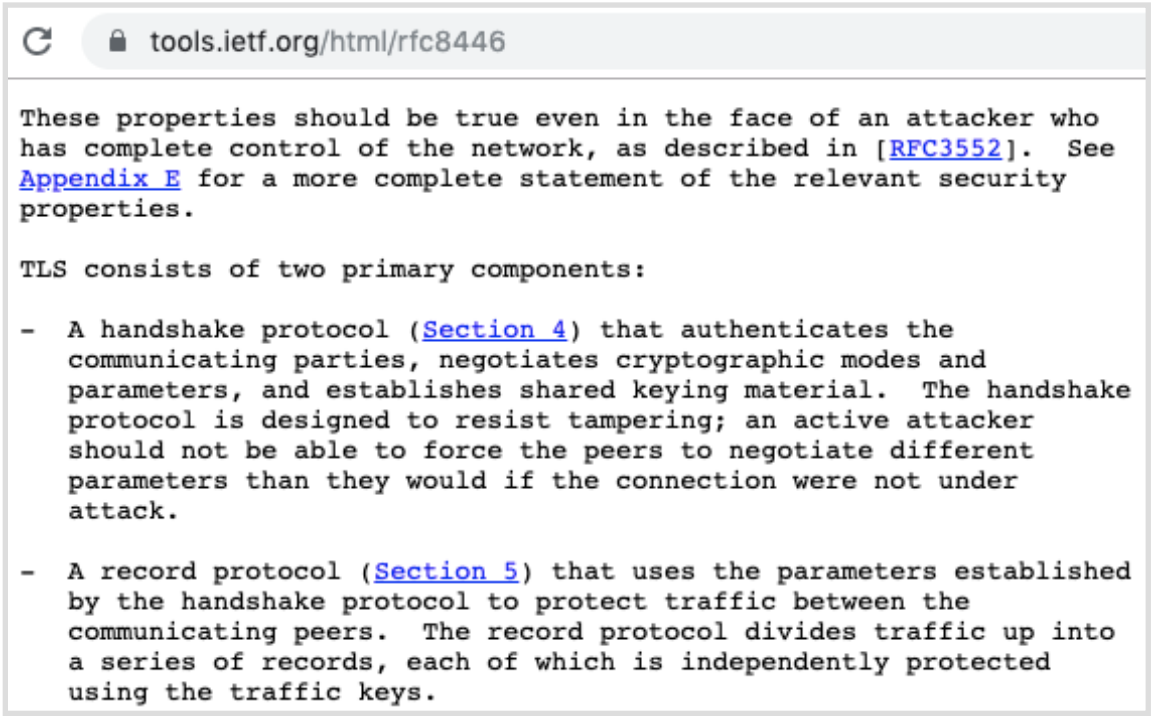
- This talk is about
  - Understanding how certain protocols disseminate plaintext information
  - Seeing which information is actually disseminated in plaintext
  - What it can be done to prevent it

Protocols Shown in this Talk

- Protocols
  - TLS
  - DNS
  - mDNS
  - DNS-SD
  - SSDP
  - DHCP
- ~10 minutes per protocol
  - Basic overview with real examples
  - No deep-dive

The Building Blocks of the TLS

- TLS actually consists of two protocols
- Only one actually carry encrypted application data
- TLS v 1.3, 1.2, 1.1, 1.0

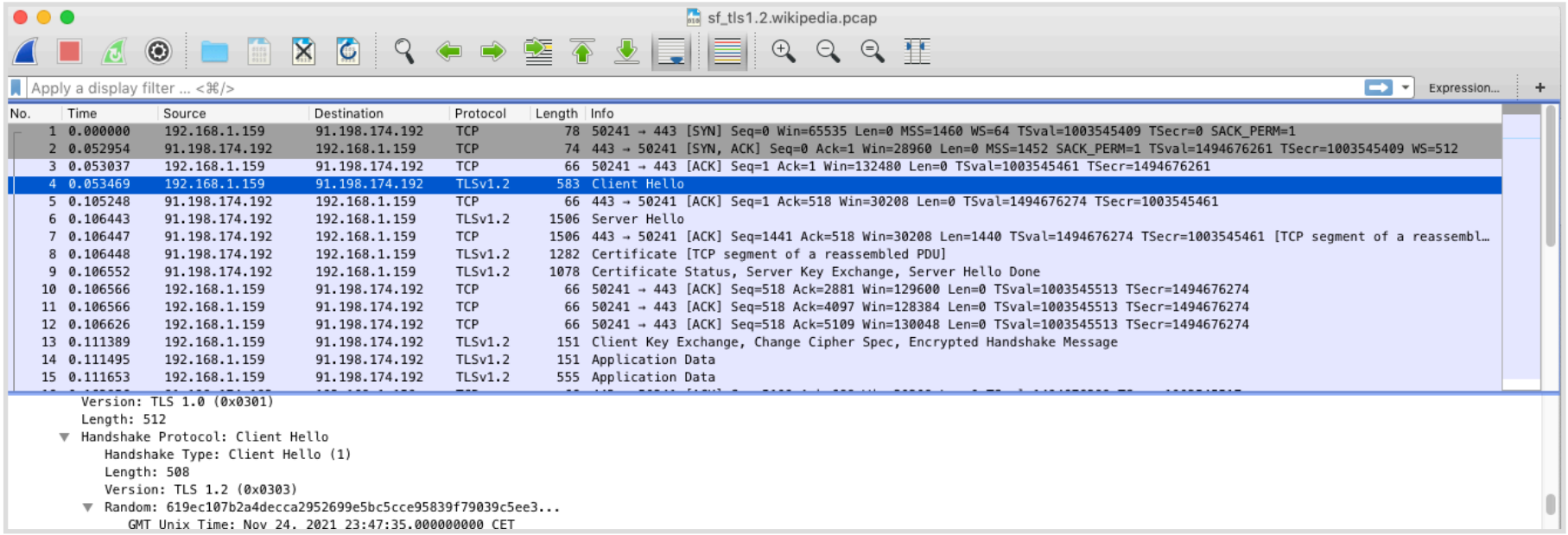


The TLS Handshake

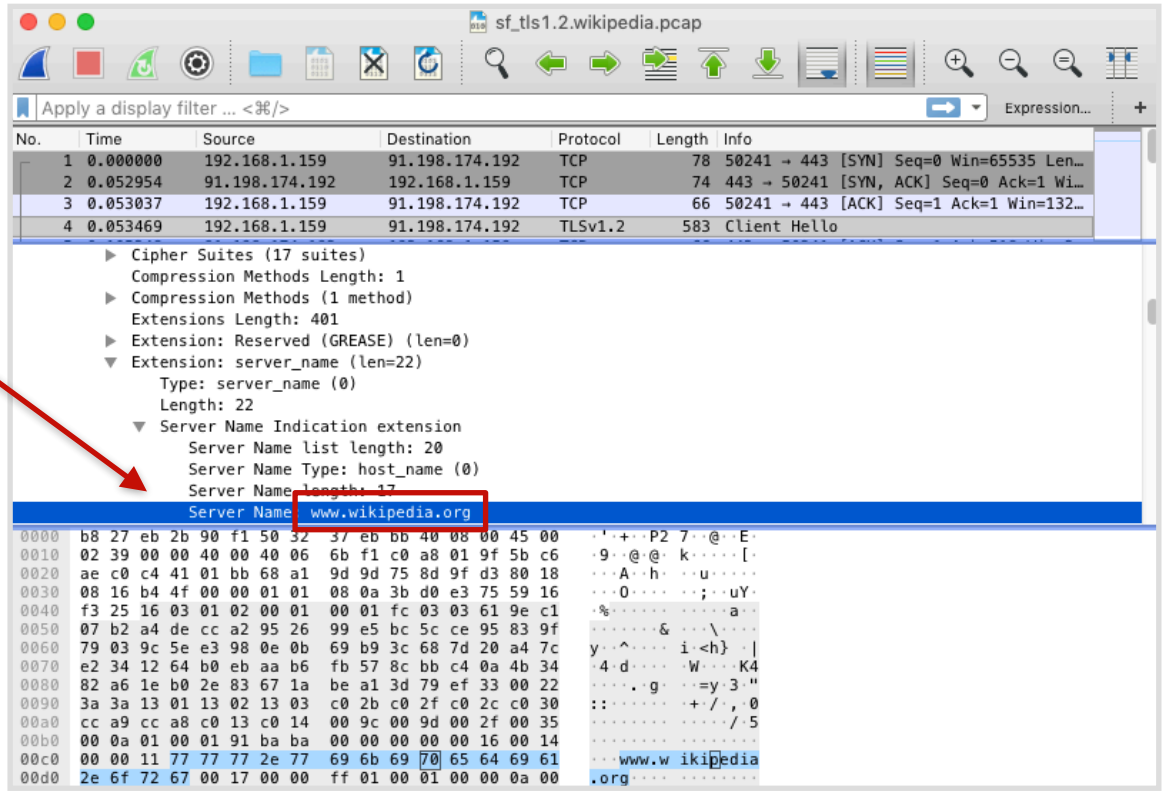
- Before actually exchanging encrypted data, two parties willing to use TLS must perform an handshake
- Allows the server and client to
  - Authenticate each other
  - Negotiate an encryption algorithm and cryptographic keys
- Involves a series of back-and-forth packets between client and server

The TLS Handshake in Packets

- Shown TLS v 1.2, 1.3 fewer packets but still plaintext

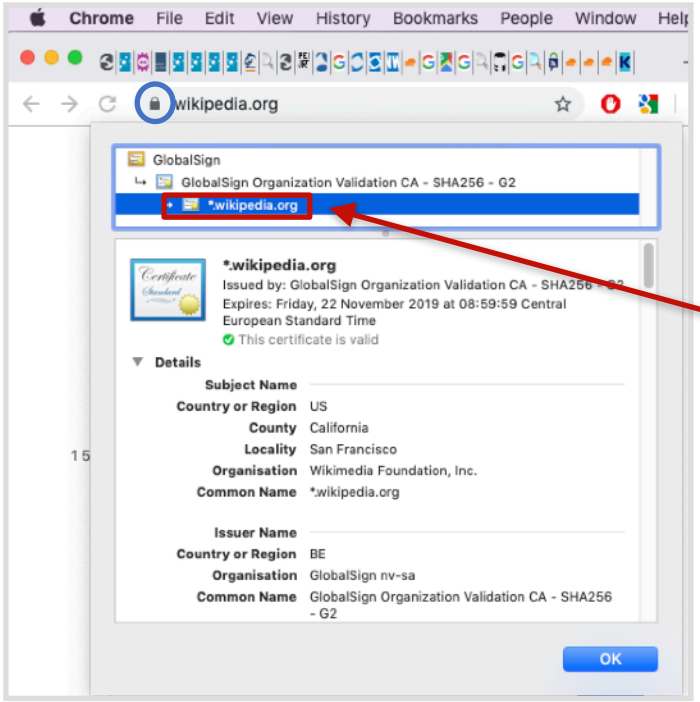


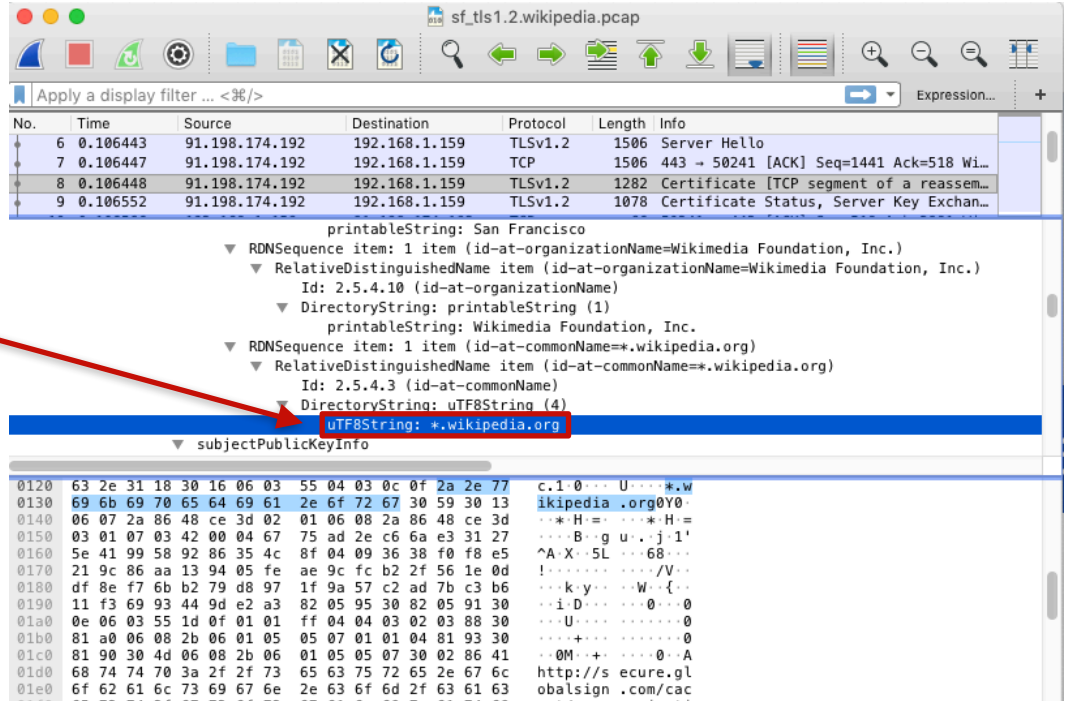
TLS Client Handshake: Example



- Open page https://www.wikipedia.org
- Host name is sent in plaintext, along with other information

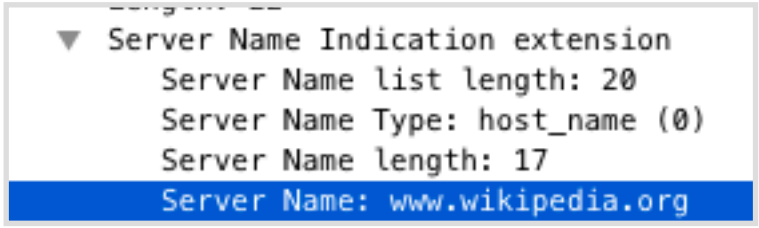
TLS Server Handshake: Example

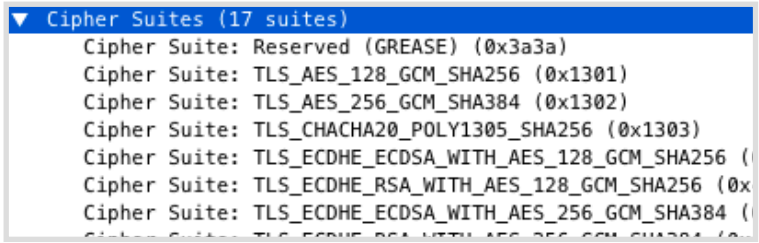




TLS Handshake: Plaintext Information [1/2]

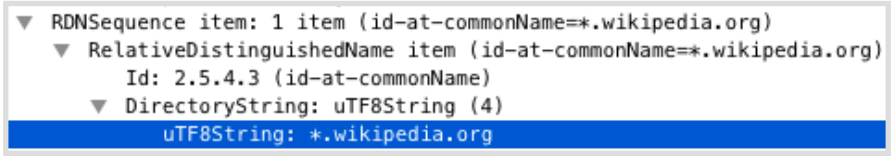
- Server Name Indication (SNI)
  - From the browser
  - Similar to the HTTP virtual hosts
- Cipher Suites
  - Sets of (more or less secure) algorithms to secure the communication

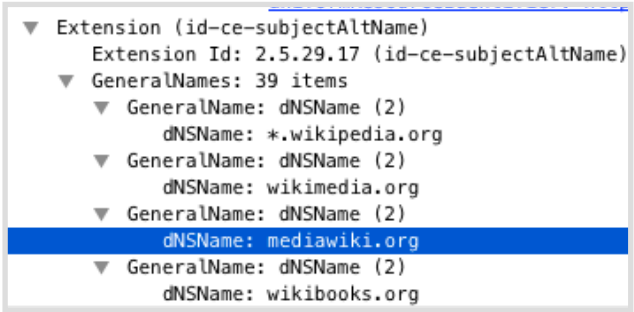


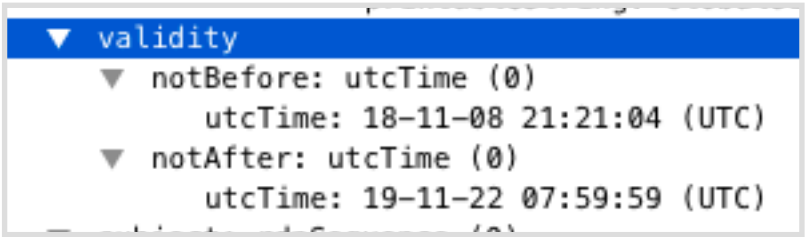


TLS Handshake: Plaintext Information [2/2]

- Server Certificate
  - Common Name
  - Alternative Names
  - Validity
  - Plaintext in TLS 1.2
  - Encrypted in TLS 1.3

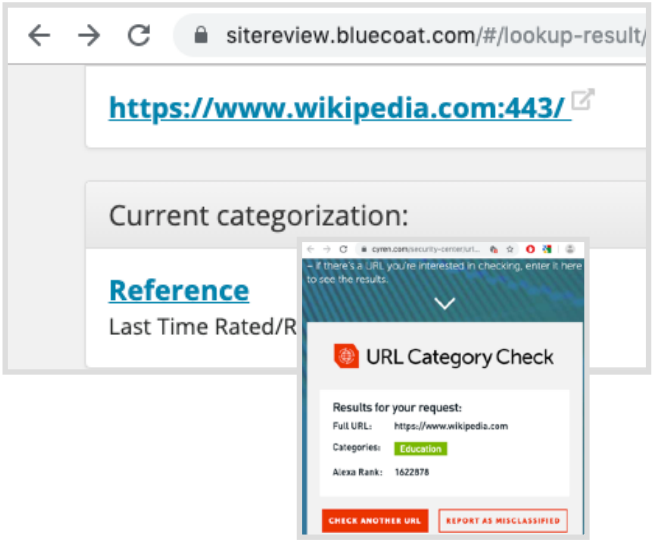






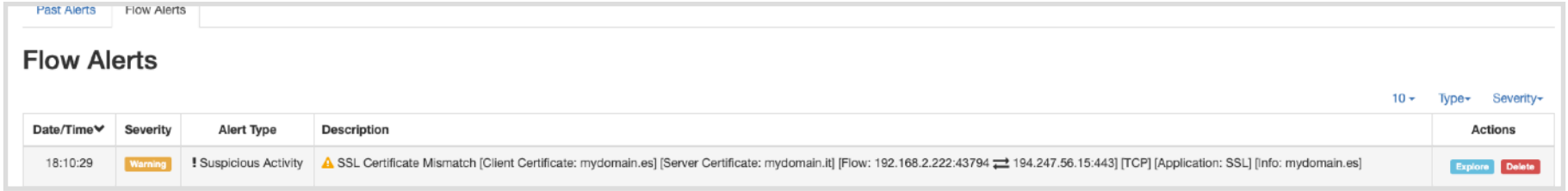
How to Use TLS Handshake Data: SNI [1/2]

- SNI to profile users
  - *.facebook.com -> social media
  - *.bloomberg.com -> news
  - Services
    - SimilarWeb, Webshrinker, Symantec, Cyren
- Censorship in Korea



How to Use TLS Handshake Data: SNI [2/2]

- SNI for HTTPS blocking / throttling
  - ntop's ntopng Edge
  - Trustwave's Web Filter
  - Sophos UTM
- SNI for Alerting
  - Suspicious or malicious host names

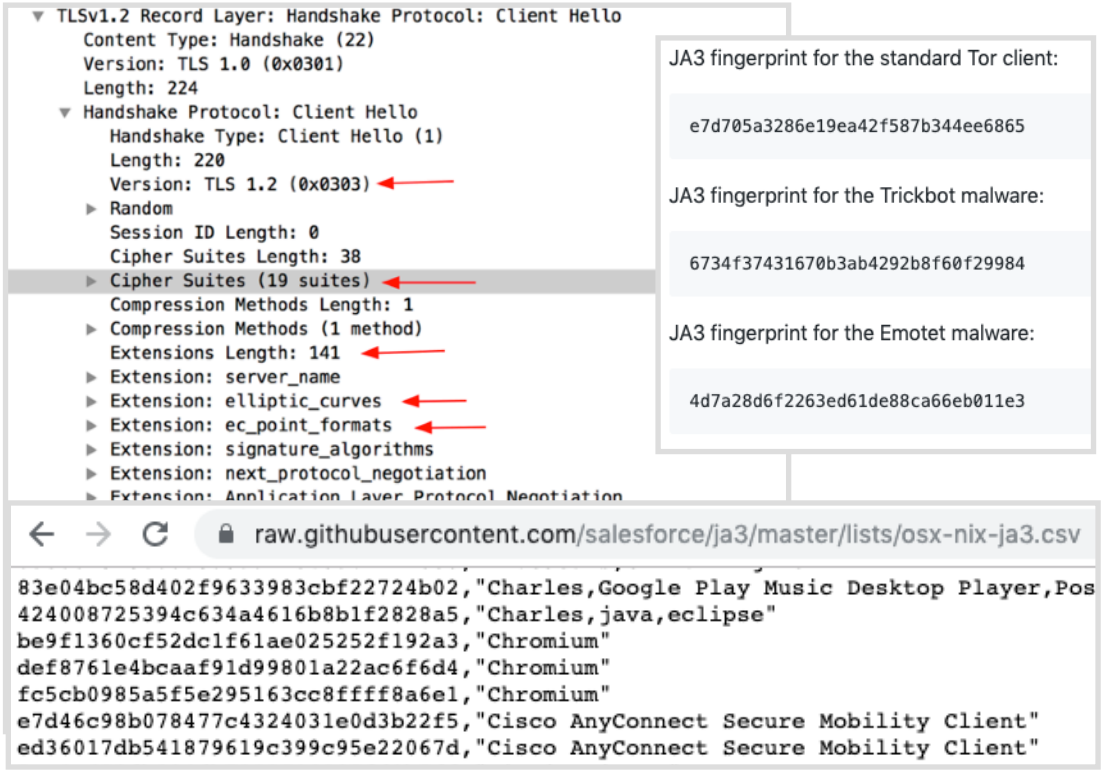


How to Use TLS Handshake Data: Certificate and Cipher Suites

- Sever Certificate validity
- Cipher Suites to check if hosts in your network are using algorithms which are (deemed to be) secure
  - Entities maintain guidelines for TLS with regard to network security

How to Use TLS Handshake Data: Fingerprinting

- **Fingerprinting** to profile SSL/TLS Clients
  - Good, bad, expected, unexpected, unsecure
- A fingerprint (almost surely) identify a client
- **JA3** (https://github.com/salesforce/ja3)
  - Uses fields in the client hello

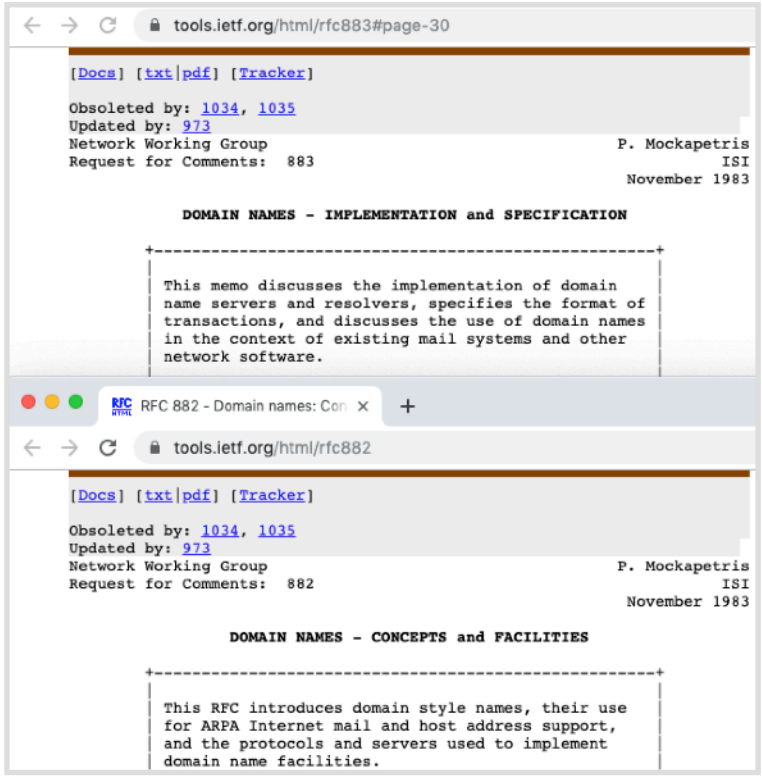


Protection Against TLS Handshake Eavesdroppers

- Encrypted SNI as an extension of TLS v 1.3
  - The server publishes a public key on a well-known DNS record
  - The client then replaces the plaintext SNI with an encrypted SNI, encrypted using a symmetric encryption key derived using the server's public key

The Domain Name System (DNS)

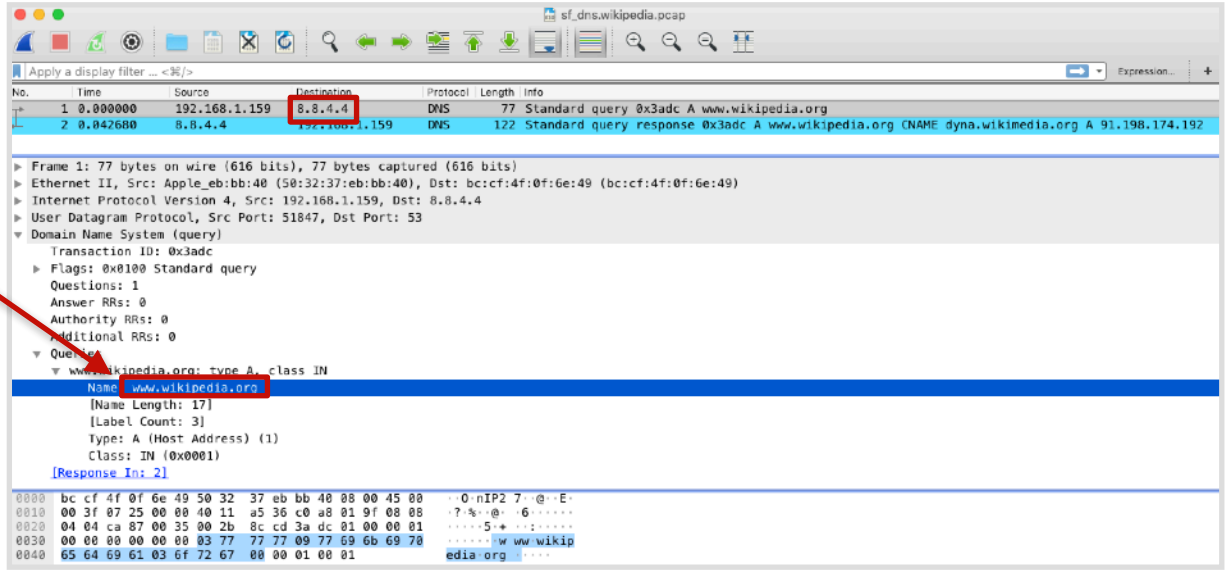
- System to map symbolic names to IP addresses
  - e.g., wikipedia.com -> 1.2.3.4
- Hierarchical and distributed architecture
- Defines the **DNS protocol**
- Ultra-long history
  - Tens of RFCs



The DNS Protocol

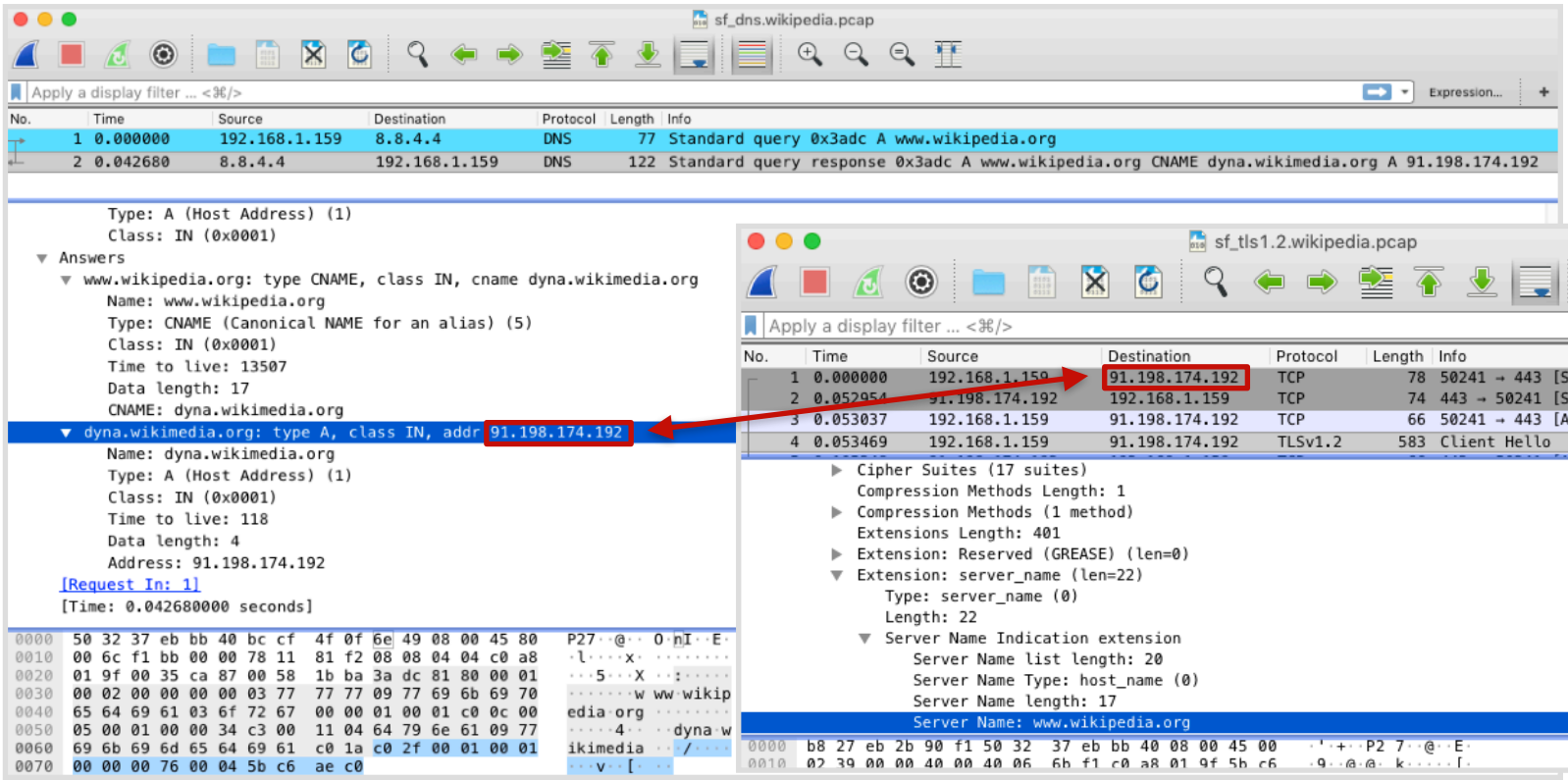
- The DNS protocol (almost surely) kicks in every time a **name** is used to identify Internet resource
  - ping google.it
  - curl www.wikipedia.org
- To **resolve** a to an IP address, the **DNS client** queries a **DNS server**
  - The DNS client issues a **DNS query**
  - The DNS server responds with a **DNS query response**
- DNS server IP address is known to the DNS client

DNS Query: Example



- Open page https://www.wikipedia.org
- DNS query and query response before initiating the TLS connection

DNS Query Response: Example



DNS Protocol: Plaintext Information

- All the resolved names are plaintext
  - Even if all the subsequent communications are encrypted

How to use DNS Data

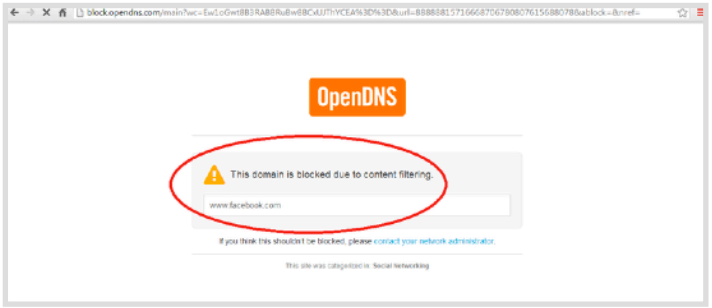
- Names to passively profile users similar to what has been seen with the TLS SNI
- The ISP or even a Free-Wifi bar can easily get their hands into the DNS traffic

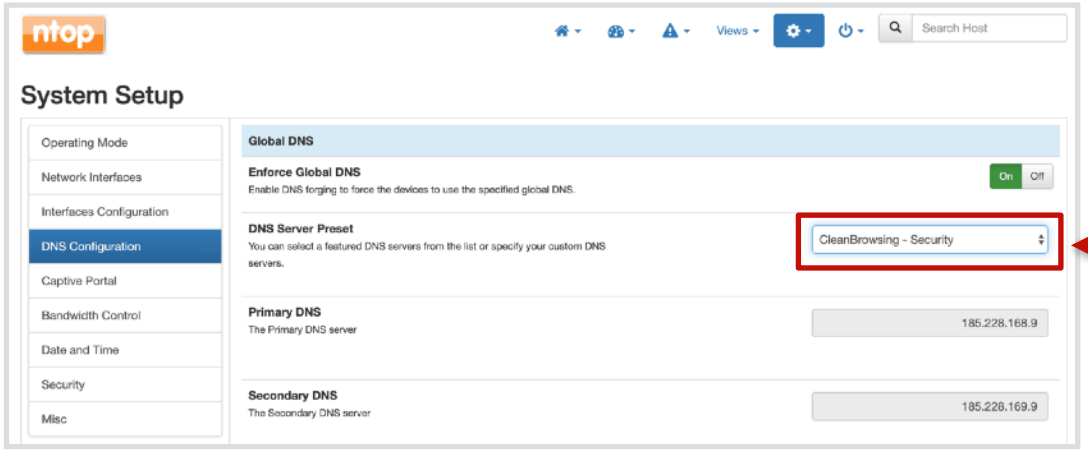
DNS Queries Interception [1/2]

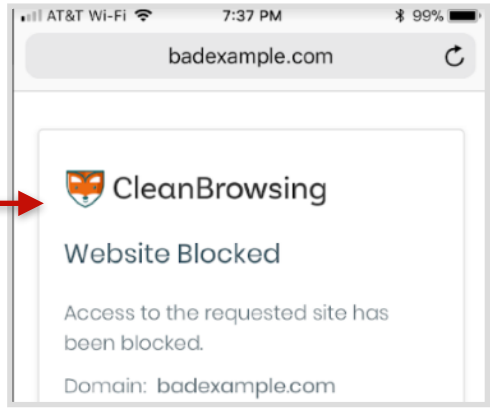
- As there is no encryption / authentication, queries can be intercepted
  - Transparently redirect the DNS queries to a DNS server chosen by the ISP (or an attacker)
  - The DNS server can respond with arbitrary IP addresses
- Interceptions can be made for various purposes
  - Censorship
  - Displaying ads
  - Collecting statistics
  - Blocking malware
- The point is that they are not authorized by users and are difficult to spot

DNS Queries Interception [2/2]

- DNS-based content filtering
  - OpenDNS, CleanBrowsing DNS and other services

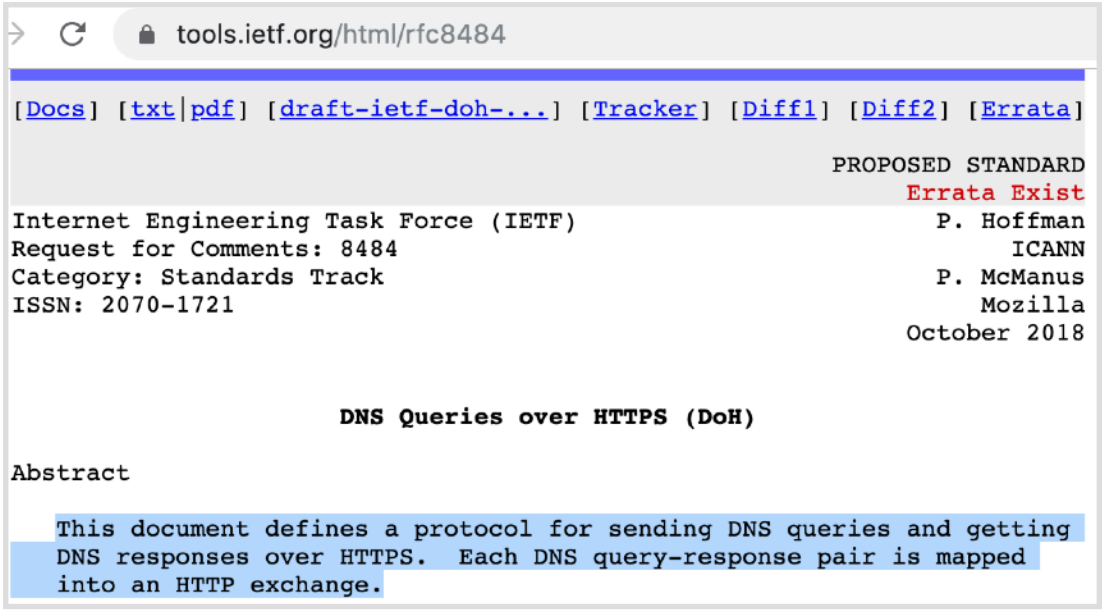




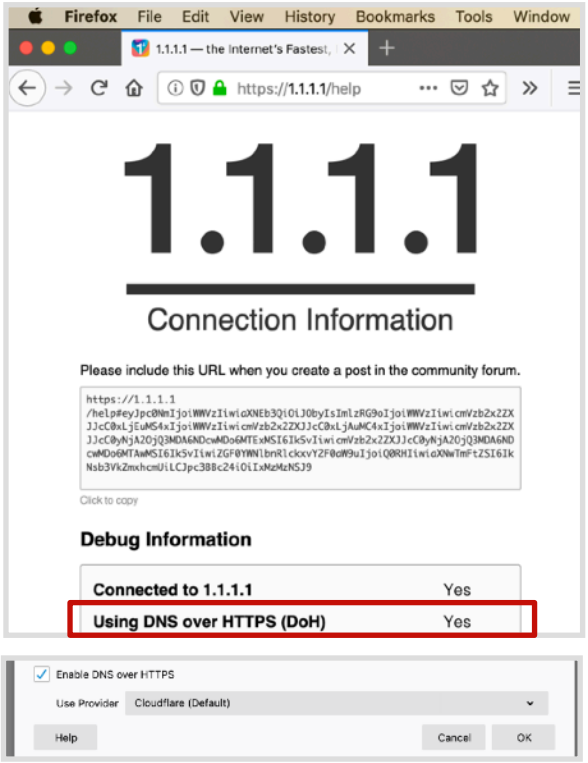


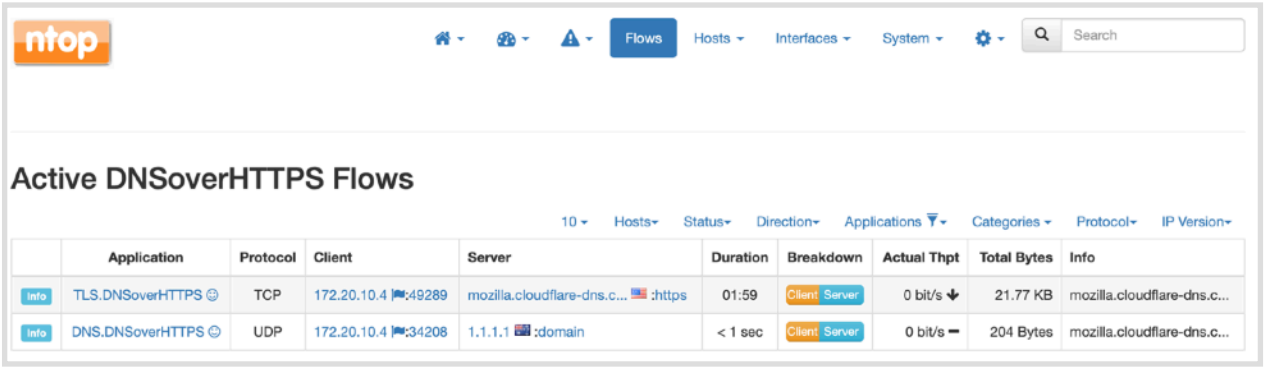
Protection Against DNS Eavesdroppers: DoH

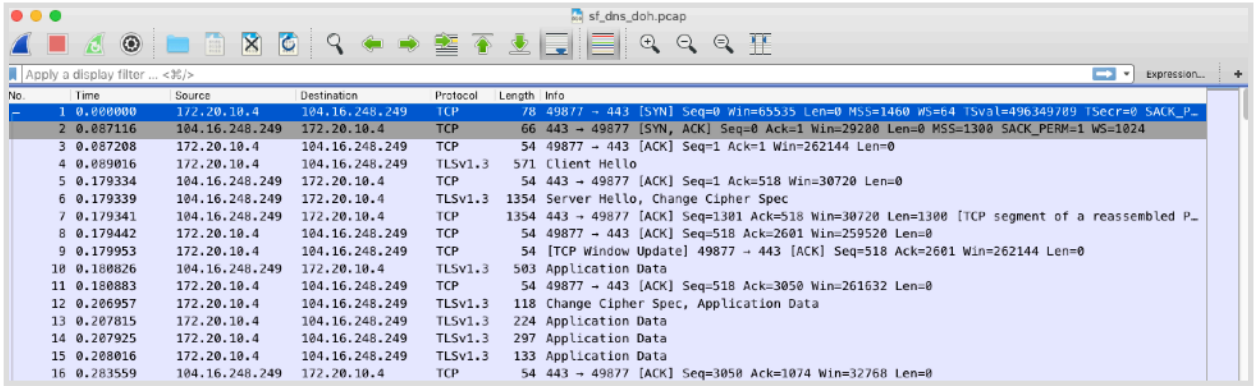
- DNS over HTTPS (DoH)
- TCP port 443
- Third-party observers can't look at DNS requests
- Supported by recent browsers (FF, Chrome)



Protection Against DNS Eavesdroppers: Firefox DoH

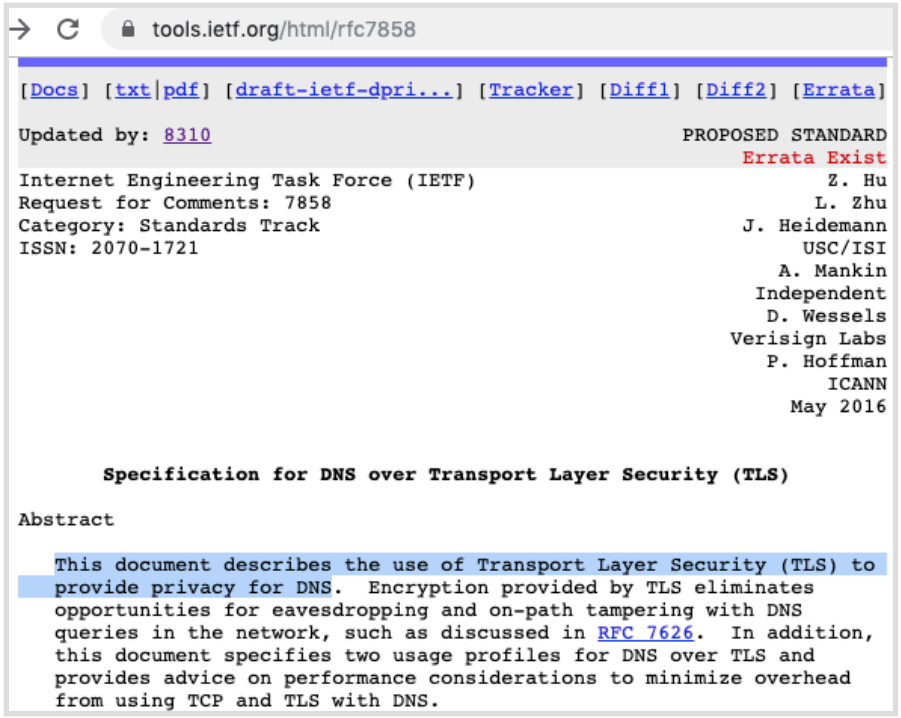






Protection Against DNS Eavesdroppers: DoT

- DNS over TLS (DoT)
- TCP port 8**53**
- System-wide
- Linux: `systemd-resolved` (`systemd >= 239`)
- Linux/Win/OS X: DNS Privacy Daemon - `stubby`



tools.ietf.org/html/rfc7858

[Docs] [txt|pdf] [draft-ietf-dpri...] [Tracker] [Diff1] [Diff2] [Errata]

Updated by: 8310 — PROPOSED STANDARD — Errata Exist

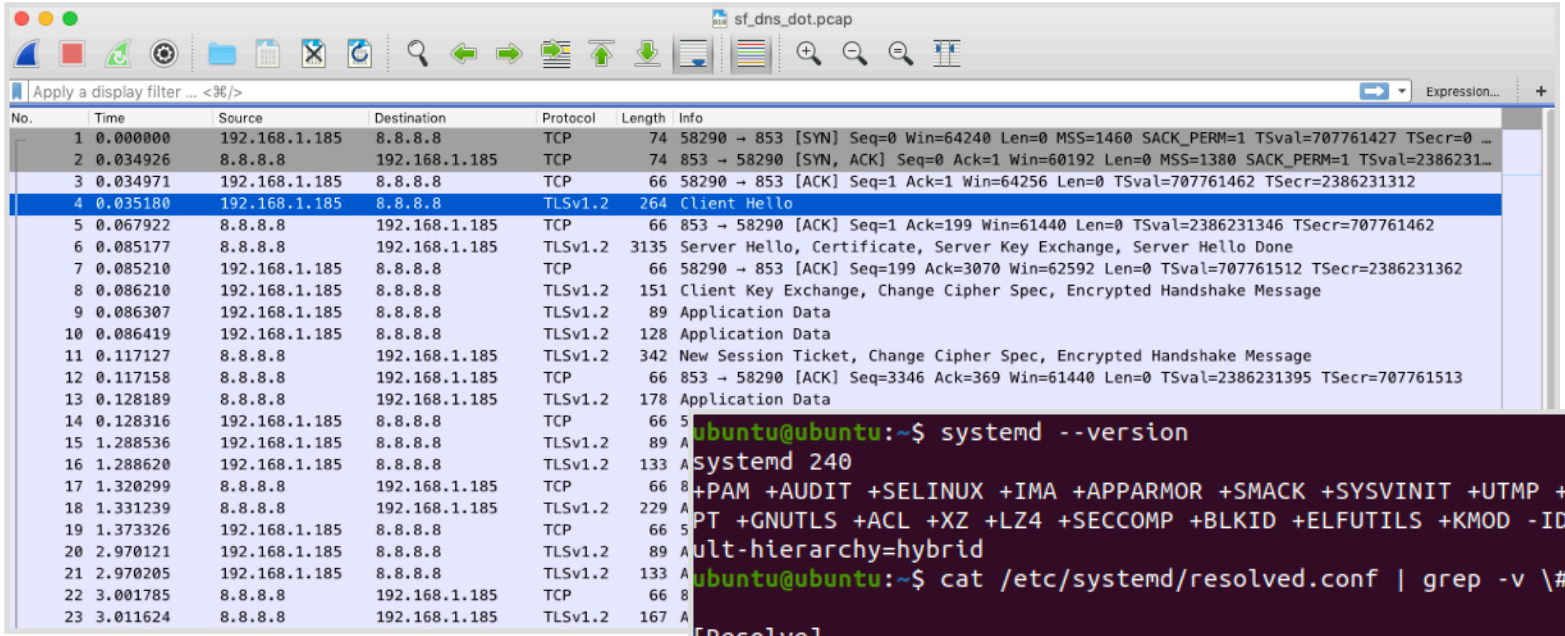
Internet Engineering Task Force (IETF) — Z. Hu
Request for Comments: 7858 — L. Zhu
Category: Standards Track — J. Heidemann
ISSN: 2070-1721 — USC/ISI
A. Mankin
Independent
D. Wessels
Verisign Labs
P. Hoffman
ICANN
May 2016

Specification for DNS over Transport Layer Security (TLS)

Abstract

This document describes the use of Transport Layer Security (TLS) to provide privacy for DNS. Encryption provided by TLS eliminates opportunities for eavesdropping and on-path tampering with DNS queries in the network, such as discussed in RFC 7626. In addition, this document specifies two usage profiles for DNS over TLS and provides advice on performance considerations to minimize overhead from using TCP and TLS with DNS.

Protection Against DNS Eavesdroppers: `systemd-resolved` DoT




```
ubuntu@ubuntu:~$ systemd --version
systemd 240
+PAM +AUDIT +SELINUX +IMA +APPARMOR +SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP +GCRY
PT +GNUTLS +ACL +XZ +LZ4 +SECCOMP +BLKID +ELFUTILS +KMOD -IDN2 +IDN -PCRE2 defa
ult-hierarchy=hybrid
ubuntu@ubuntu:~$ cat /etc/systemd/resolved.conf | grep -v \#

[Resolve]
DNS=8.8.8.8
Domains=~.
DNSOverTLS=opportunistic
ubuntu@ubuntu:~$ systemctl restart systemd-resolved
```

The multicast DNS (mDNS)

- Resolve host names to IP addresses in (small) networks
- No need for a DNS server
- IP UDP multicast packets
- Only resolves host names ending with **.local**

tools.ietf.org/html/rfc6762

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.

```
simone@devel:~$ avahi-resolve-host-name -4 Simones-MacBook-Pro.local
Simones-MacBook-Pro.local       192.168.2.126
simone@devel:~$ ping -c1 Simones-MacBook-Pro.local
PING Simones-MacBook-Pro.local (192.168.2.126) 56(84) bytes of data.
64 bytes from 192.168.2.126: icmp_seq=1 ttl=64 time=1.00 ms

--- Simones-MacBook-Pro.local ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.009/1.009/1.009/0.000 ms
simone@devel:~$
```

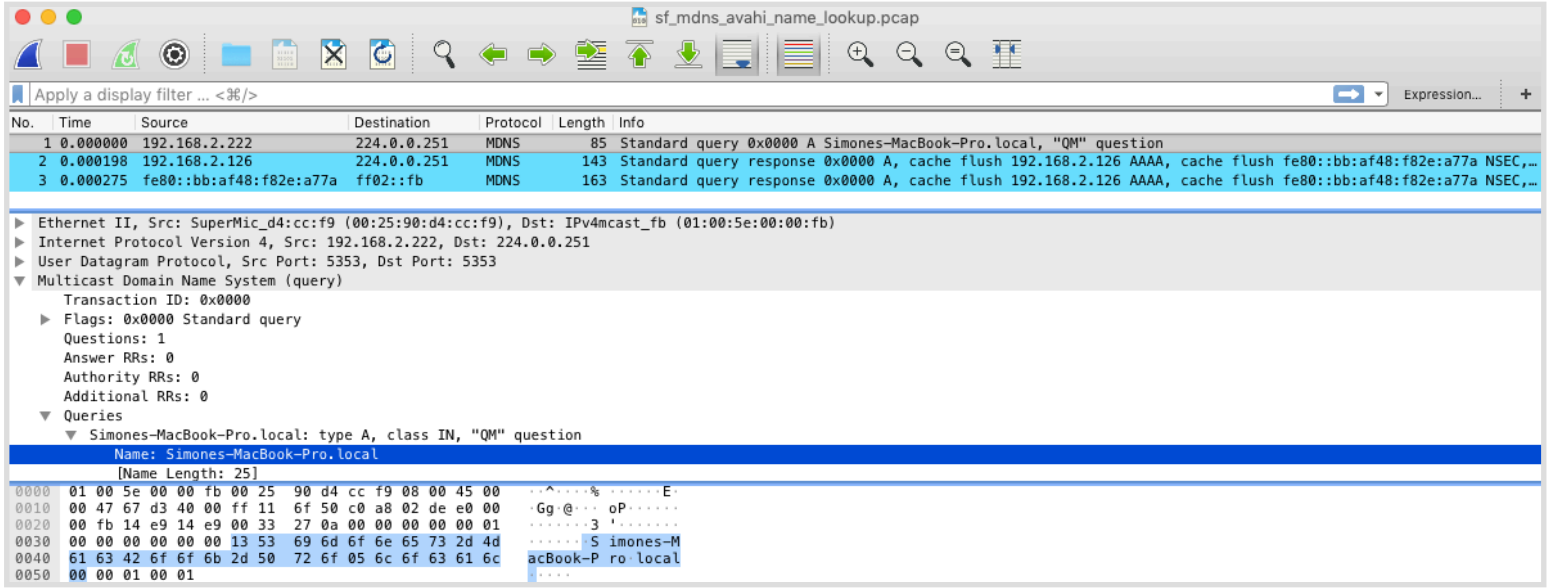
mDNS Implementations

- Apple
  - Bonjour (`mDNSResponder`)
- Linux & BSDs
  - Avahi (`avahi-daemon`)
  - `systemd-resolved`
- Windows
  - Bonjour for Windows (`mDNSResponder.exe`)
  - Link-local Multicast Name Resolution (LLMNR) - not actually mDNS but similar

mDNS Query: Example

```
simone@devel:~$ avahi-resolve-host-name -4 Simones-MacBook-Pro.local
Simones-MacBook-Pro.local       192.168.2.126
```

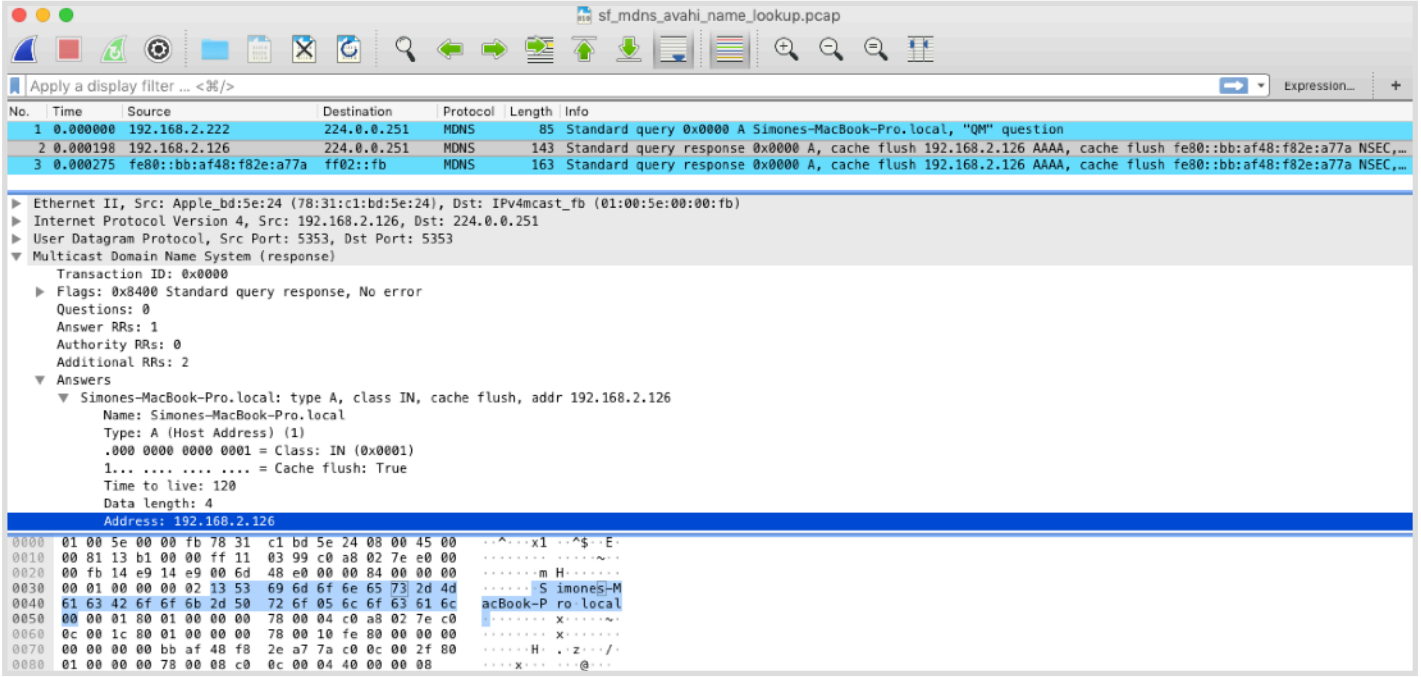
- Query sent to 224.0.0.251
- UDP with src/dst ports 5353
- Name is carried in plaintext in a standard DNS packet



mDNS Query Response: Example

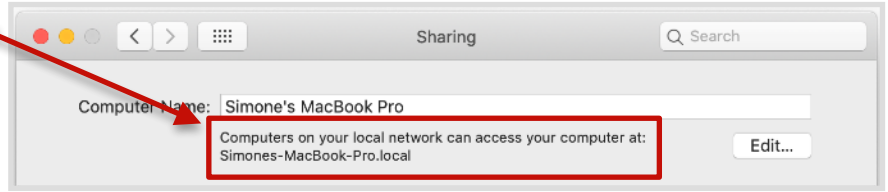
```
simone@devel:~$ avahi-resolve-host-name -4 Simones-MacBook-Pro.local
Simones-MacBook-Pro.local       192.168.2.126
```

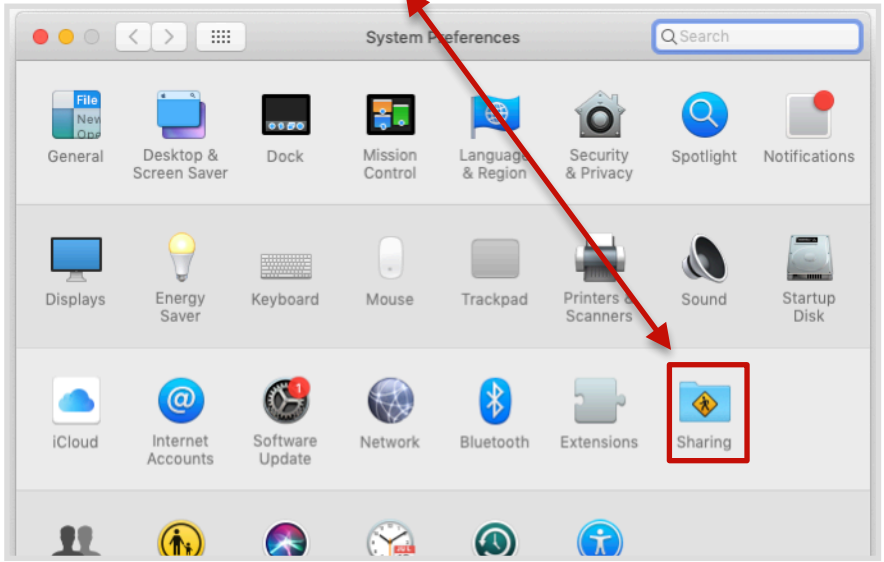
- Response sent to 224.0.0.251
- UDP with src/ dst ports 5353
- Name and IP address carried in plaintext in a standard DNS packet



mDNS: Tips & Tricks to Play with It

```
simone@devel:~$ avahi-resolve-host-name -4 Simones-MacBook-Pro.local
Simones-MacBook-Pro.local       192.168.2.126
```





- Setting the name on OS X
  - System Prefs->Sharing
- Can use dig

```
$ dig @224.0.0.251 -p5353 +short \
"Simones-MacBook-Pro.local"
192.168.2.126
```

- Can reverse lookup

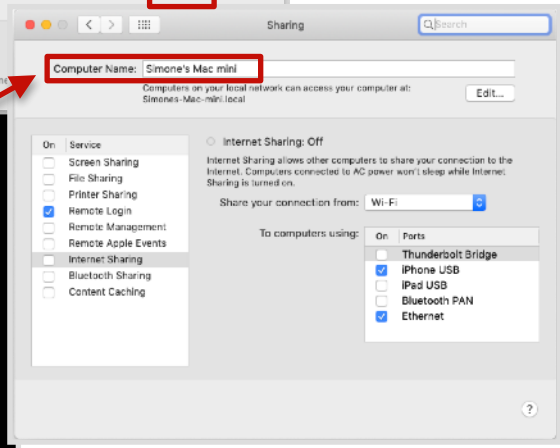
```
$ avahi-resolve-address 192.168.2.126
192.168.2.126     Simones-MacBook-Pro.local
```

Advertising Services with mDNS

- mDNS per-se does not provide information device types and services
- **Advertise** information about **network services** that a device offers
- DNS Service-Discovery (**DNS-SD**) - RFC 6763
  - Allows clients to discover services, and to resolve those services to host names using standard DNS queries

Discovering Services With DNS-SD

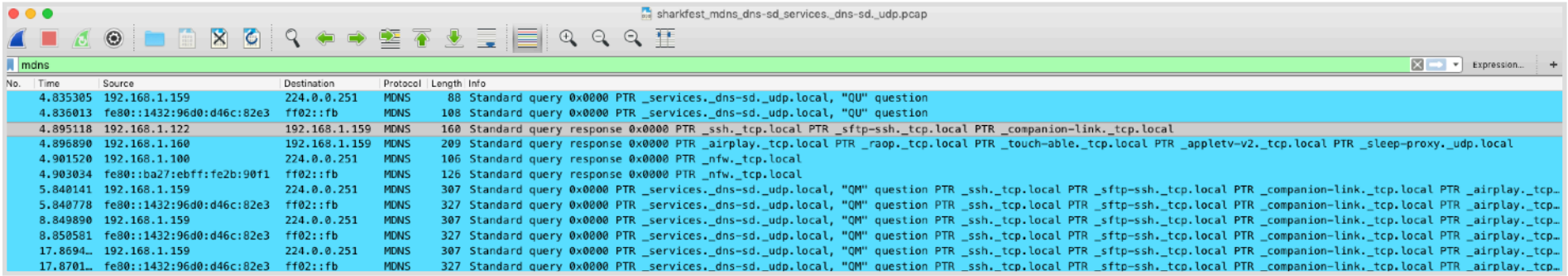
```
Simones-Mac-mini:~ simone$ dns-sd -B _services._dns-sd._udp
Browsing for _services._dns-sd._udp
DATE: ---Sun 15 Sep 2019---
17:34:30.426  ...STARTING...
Timestamp     A/R    Flags  if Domain               Service Type         Instance Name
17:34:30.426  Add        3   6 .                    _tcp.local.          _nfw
17:34:30.426  Add        3   6 .                    _tcp.local.          _airplay
17:34:30.426  Add        3   6 .                    _tcp.local.          _raop
17:34:30.426  Add        3   6 .                    _tcp.local.          _touch-able
17:34:30.426  Add        3   6 .                    _tcp.local.          _appletv-v2
17:34:30.426  Add        3   6 .                    _udp.local.          _sleep-proxy
17:34:30.426  Add        3   6 .                    _tcp.local.          _ssh
17:34:30.426  Add        3   6 .                    _tcp.local.          _sftp-ssh
17:34:30.426  Add        2   6 .                    _tcp.local.          _companion-link
^C
Simones-Mac-mini:~ simone$ dns-sd -B _ssh._tcp
Browsing for _ssh._tcp
DATE: ---Sun 15 Sep 2019---
17:34:37.338  ...STARTING...
Timestamp     A/R    Flags  if Domain               Service Type         Instance Name
17:34:37.339  Add        3   6 local.               _ssh._tcp.           Simone's Mac mini
17:34:37.339  Add        2   6 local.               _ssh._tcp.           Simone's MacBook Pro
^C
Simones-Mac-mini:~ simone$ dns-sd -L "Simone's MacBook Pro" _ssh._tcp
Lookup Simone's MacBook Pro._ssh._tcp.local
DATE: ---Sun 15 Sep 2019---
17:34:49.436  ...STARTING...
17:34:49.437  Simone's\032MacBook\032Pro._ssh._tcp.local. can be reached at Simones-MacBook-Pro.local.:22 (interface 6)
```



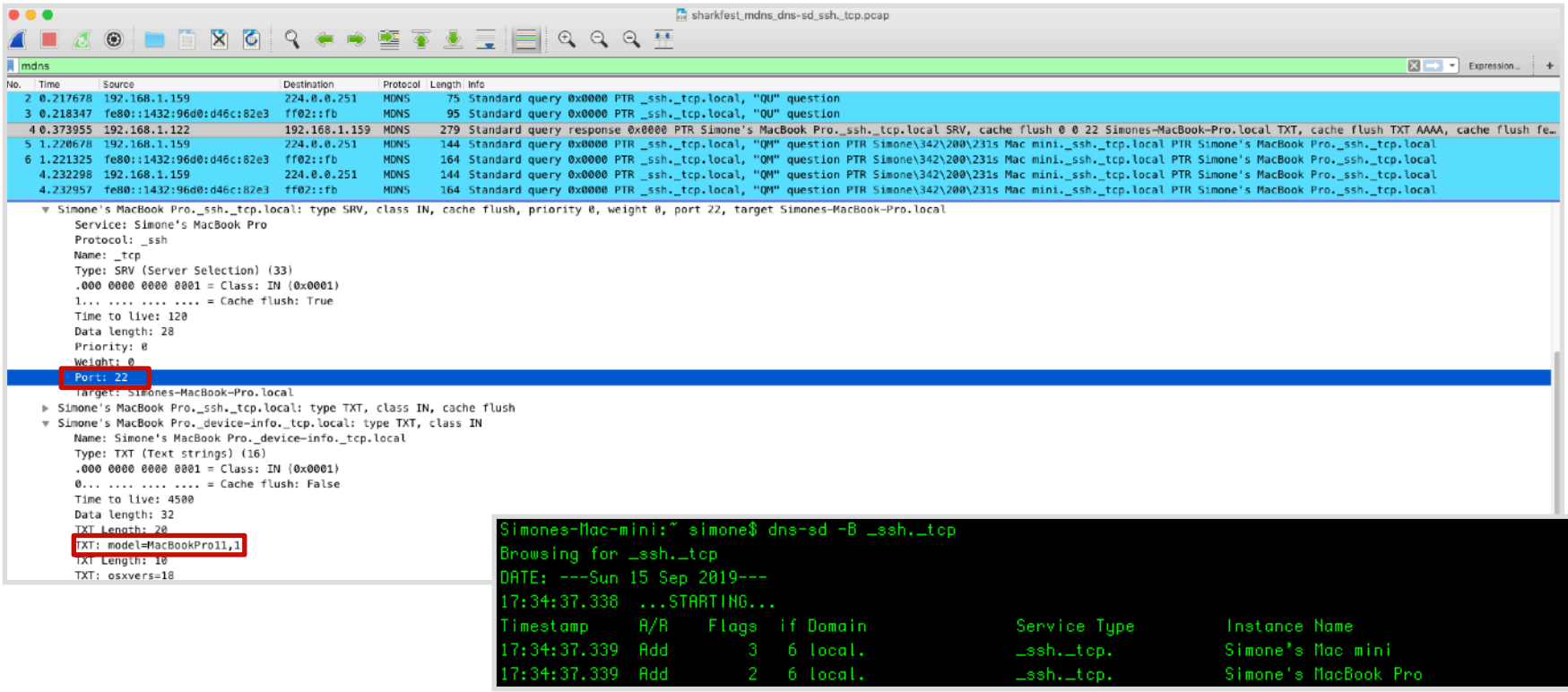
DNS-SD Service Discovery: Example [1/3]

- Queries
- Responses
- Known-Answer Suppression to avoid wasting network capacity with repeated transmission of those answers

```
Simones-Mac-mini:~ simone$ dns-sd -B _services._dns-sd._udp
Browsing for _services._dns-sd._udp
DATE: ---Sun 15 Sep 2019---
17:34:30.426  ...STARTING...
Timestamp     A/R    Flags  if Domain               Service Type         Instance Name
17:34:30.426  Add        3   6 .                    _tcp.local.          _nfw
17:34:30.426  Add        3   6 .                    _tcp.local.          _airplay
17:34:30.426  Add        3   6 .                    _tcp.local.          _raop
17:34:30.426  Add        3   6 .                    _tcp.local.          _touch-able
17:34:30.426  Add        3   6 .                    _tcp.local.          _appletv-v2
17:34:30.426  Add        3   6 .                    _udp.local.          _sleep-proxy
17:34:30.426  Add        3   6 .                    _tcp.local.          _ssh
17:34:30.426  Add        3   6 .                    _tcp.local.          _sftp-ssh
17:34:30.426  Add        2   6 .                    _tcp.local.          _companion-link
```

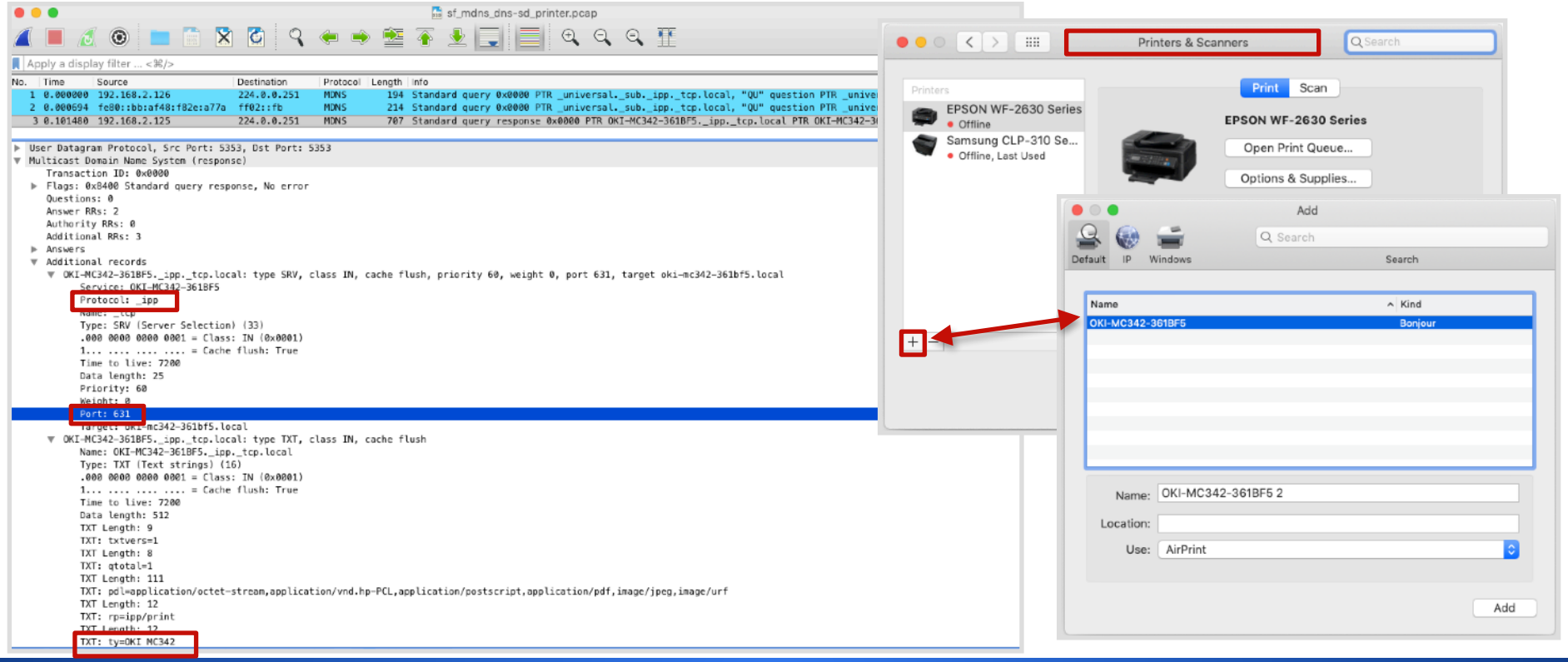


DNS-SD Service Discovery: Example [2/3]




```
Simones-Mac-mini:~ simone$ dns-sd -B _ssh._tcp
Browsing for _ssh._tcp
DATE: ---Sun 15 Sep 2019---
17:34:37.338  ...STARTING...
Timestamp     A/R    Flags  if Domain               Service Type         Instance Name
17:34:37.339  Add        3   6 local.               _ssh._tcp.           Simone's Mac mini
17:34:37.339  Add        2   6 local.               _ssh._tcp.           Simone's MacBook Pro
```

DNS-SD Service Discovery: Example [3/3]



DNS-SD: Tips & Tricks to Play with It

- Can use avahi-browse

```
$ avahi-browse --all
+ docker0 IPv4 apt-cacher-ng proxy on devel                  _apt_proxy._tcp      local
+   eno1 IPv4 apt-cacher-ng proxy on devel                  _apt_proxy._tcp      local
+   eno1 IPv4 Simone's MacBook Pro                          _companion-link._tcp local
+   eno1 IPv4 Simone's MacBook Pro                          SFTP File Transfer   local
+   eno1 IPv4 Simone's MacBook Pro                          SSH Remote Terminal  local
```

- Can use dig

```
$ dig @224.0.0.251 -p 5353 -t ptr _ssh._tcp.local
[…]
;; ANSWER SECTION:
_ssh._tcp.local.     10          IN          PTR         Simone's\032MacBook\032Pro._ssh._tcp.local.

;; ADDITIONAL SECTION:
Simone's\032MacBook\032Pro._ssh._tcp.local. 10 IN SRV 0 0 22 Simones-MacBook-Pro.local.
Simone's\032MacBook\032Pro._ssh._tcp.local. 10 IN TXT ""
Simone's\032MacBook\032Pro._device-info._tcp.local. 10 IN TXT "model=MacBookPro11,1" "osxvers=18"
Simones-MacBook-Pro.local. 10    IN          AAAA        fe80::bb:af48:f82e:a77a
Simones-MacBook-Pro.local. 10    IN          A           192.168.2.126
```

How to Use mDNS Data

- **Names to passively profile users**
  - Apple devices are particularly open in their default hostname choice of the users' first and last names

How to Use DNS-SD Data

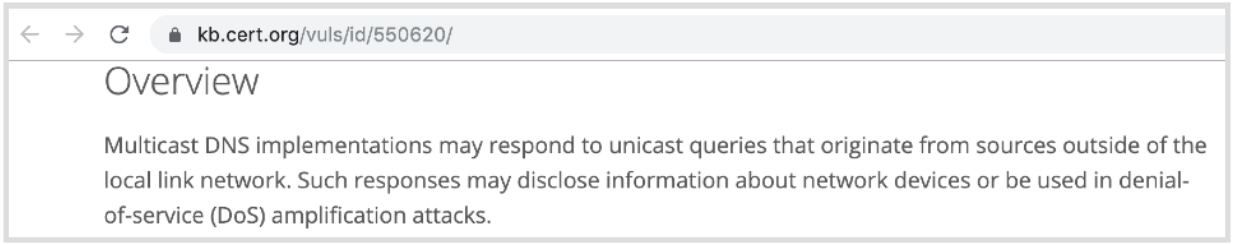
- **Port scanning**
  - _ssh._tcp
- **Service type enumeration**
  - Meta-query: "_services._dns-sd._udp.<domain>"
  - `$ dns-sd -B _services._dns-sd._udp`
- **OS versions, details, information**
  - Sent in TXT and SRV records

mDNS and DNS-SD: Secure? [1/2]

- mDNS and DNS-SD are just specifications for how to name and use records in the existing DNS system, it has no specific additional security requirements over and above those that already apply to DNS queries and DNS updates

mDNS and DNS-SD: Secure? [2/2]

- An **attacker** can respond to typo-ed domains, race against valid domains, and advertise services that don't really exist
- If not properly configured, mDNS may reply to queries from outside the link local network!
  - Publicly (Internet!) disclose software and services, as well as other potentially sensitive information, suchlike hostname, internal network configuration settings, model number, etc
  - Amplification attacks: requests for all services with a spoofed source IP address



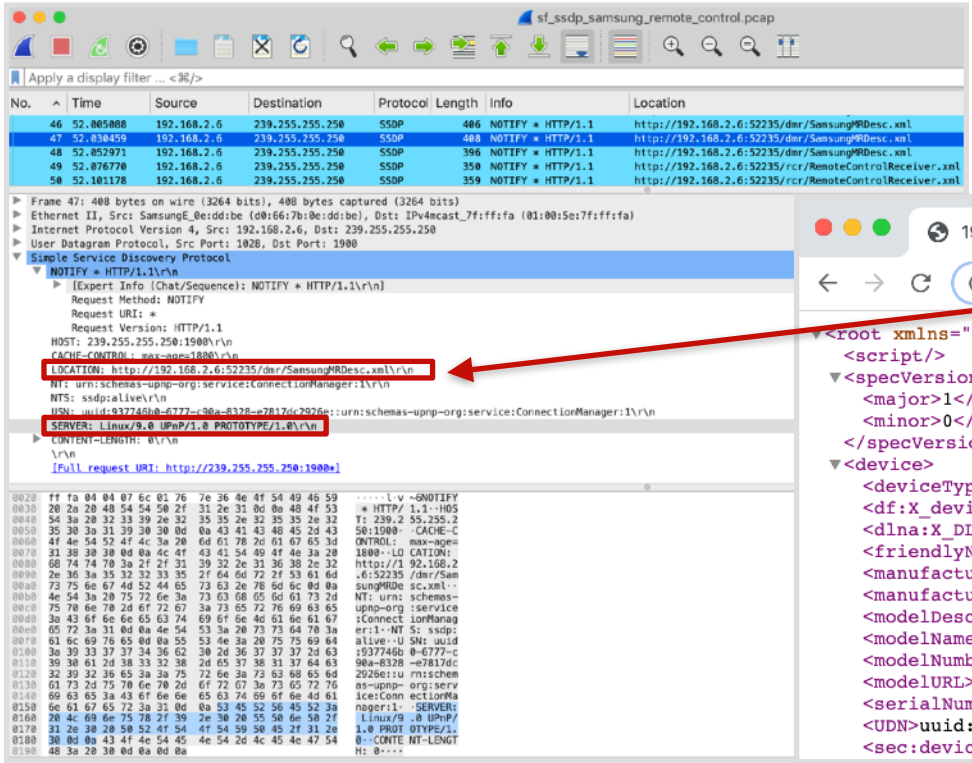
The Simple Service Discovery Protocol (SSDP) [1/2]

- Similar in spirit to mDNS-SD, SSDP is used for the advertisement/ discovery of network devices and services
  - Step 1 (**Discovery**) in the Universal Plug and Play (UPnP) technology which enables "seamless proximity networking in addition to control and data transfer among networked devices"
- Likely that home devices support UPnP and hence SSDP
  - They can be easily discovered by your computer or phone
- Devices, for example when they join the network, can query for specific devices and their services
  - Internet gateways, audio systems, TVs, or printers

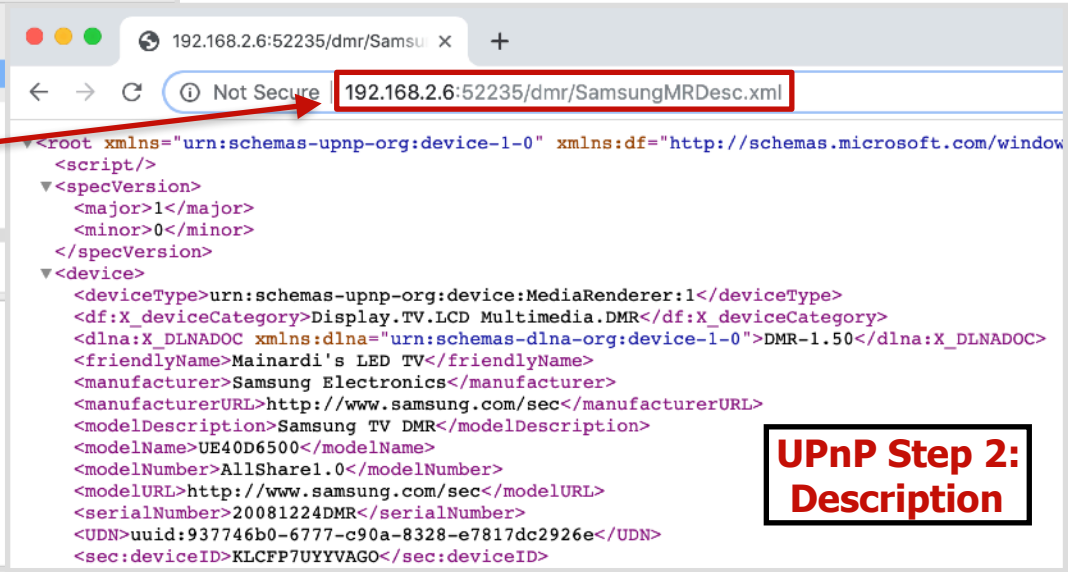
The Simple Service Discovery Protocol (SSDP) [2/2]

- IP UDP (port 1900) multicast packets carrying HTTP
- Discovery
  - Advertisement
    - For example when a device is newly connected to the network
  - Search
    - Look for available devices and offered services

SSDP Advertisement: Example

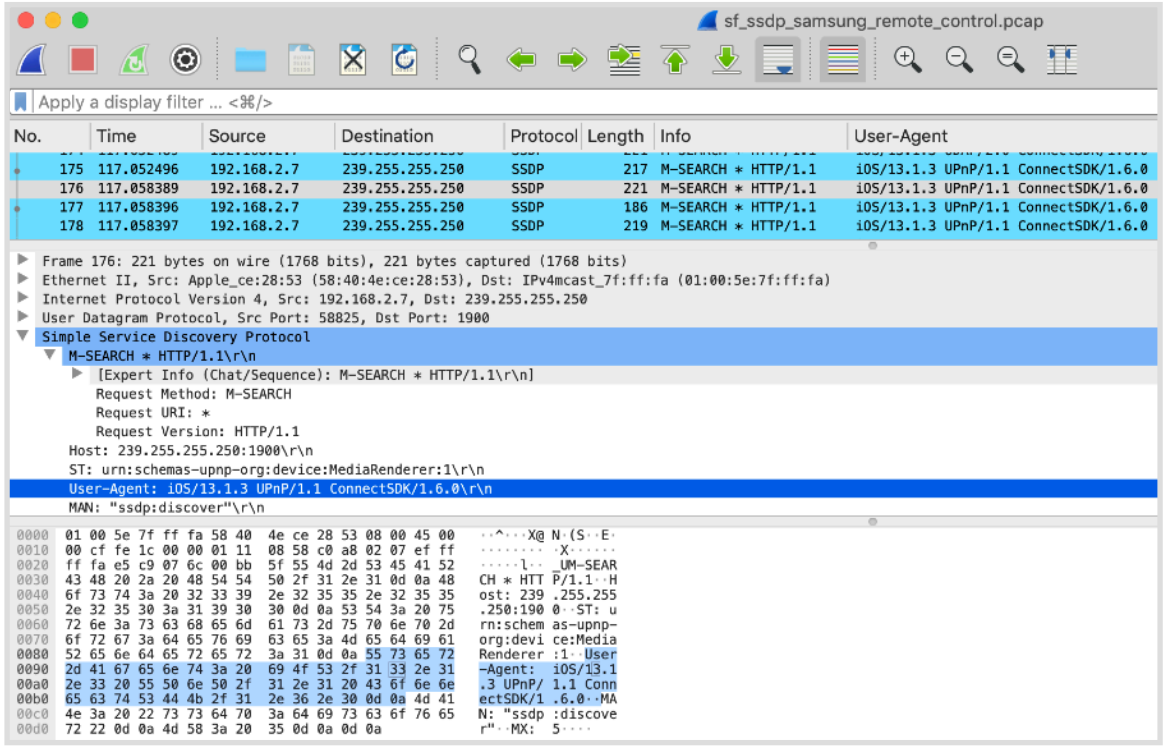


- Multicast/Unicast NOTIFY message
- Notification type and subtype (NT and NTS), Unique Service Name (USN), Server, ...

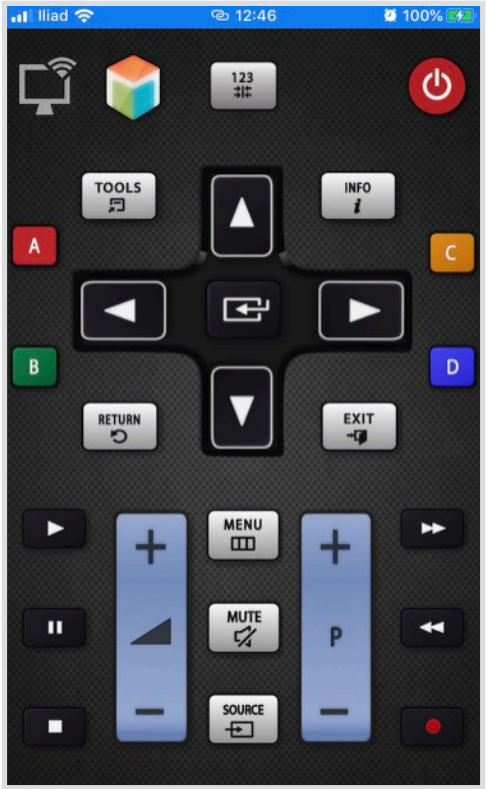


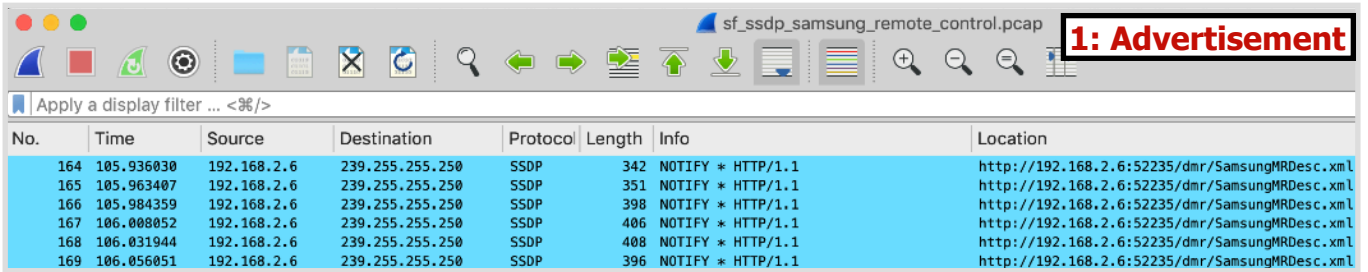
SSDP Search: Example

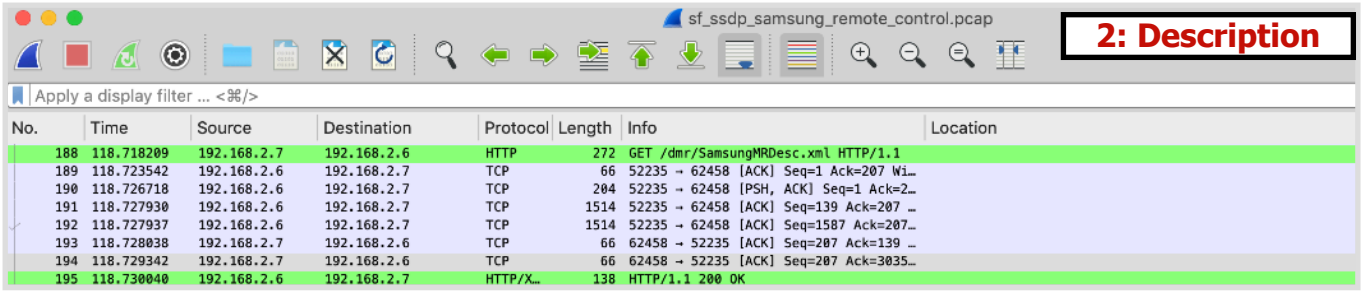
- M-SEARCH HTTP multicast request
- Namespace (fixed, MAN), Search Target (ST), User Agent
- Example is an iPhone looking for remotely-controllable TVs

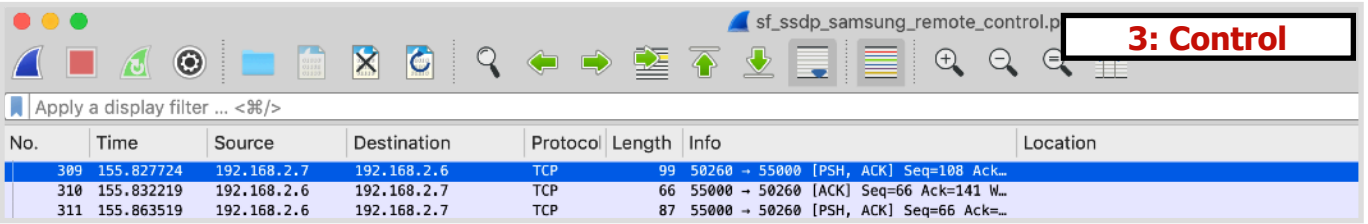


Discover a Remotely-Controllable TV with SSDP: Example









SSDP: Plaintext Information

- Plaintext information which can unveil devices types, characteristics and software version
  - User Agents
    - iOS/13.1.3 UPnP/1.1 ConnectSDK/1.6.0
  - Servers
    - Linux/9.0 UPnP/1.0 PROTOTYPE/1.0
  - USNs
    - 937746b0-6777-c90a-8328-e7817dc2926e::upnp:rootdevice

How to Use SSDP Data

- **Services Enumeration**
  - Advertised in NOTIFY messages
  - Perform queries with M-SEARCH
- **OS and other applications versions, details, information**
  - Advertised both in M-SEARCH and NOTIFY messages

SSDP: Secure?

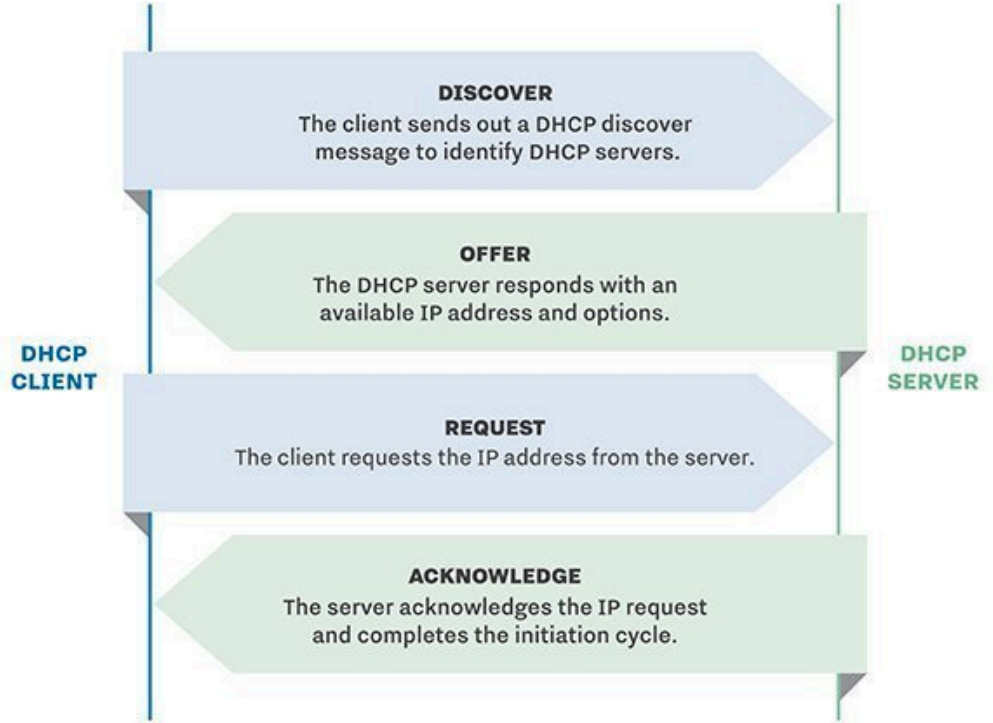
- "To be found by a network search, a device shall send a unicast UDP response to the source IP address and port that sent the request to the multicast address."
- Amplification attacks: requests for all services with a spoofed source IP address

The Dynamic Host Configuration Protocol (DHCP)

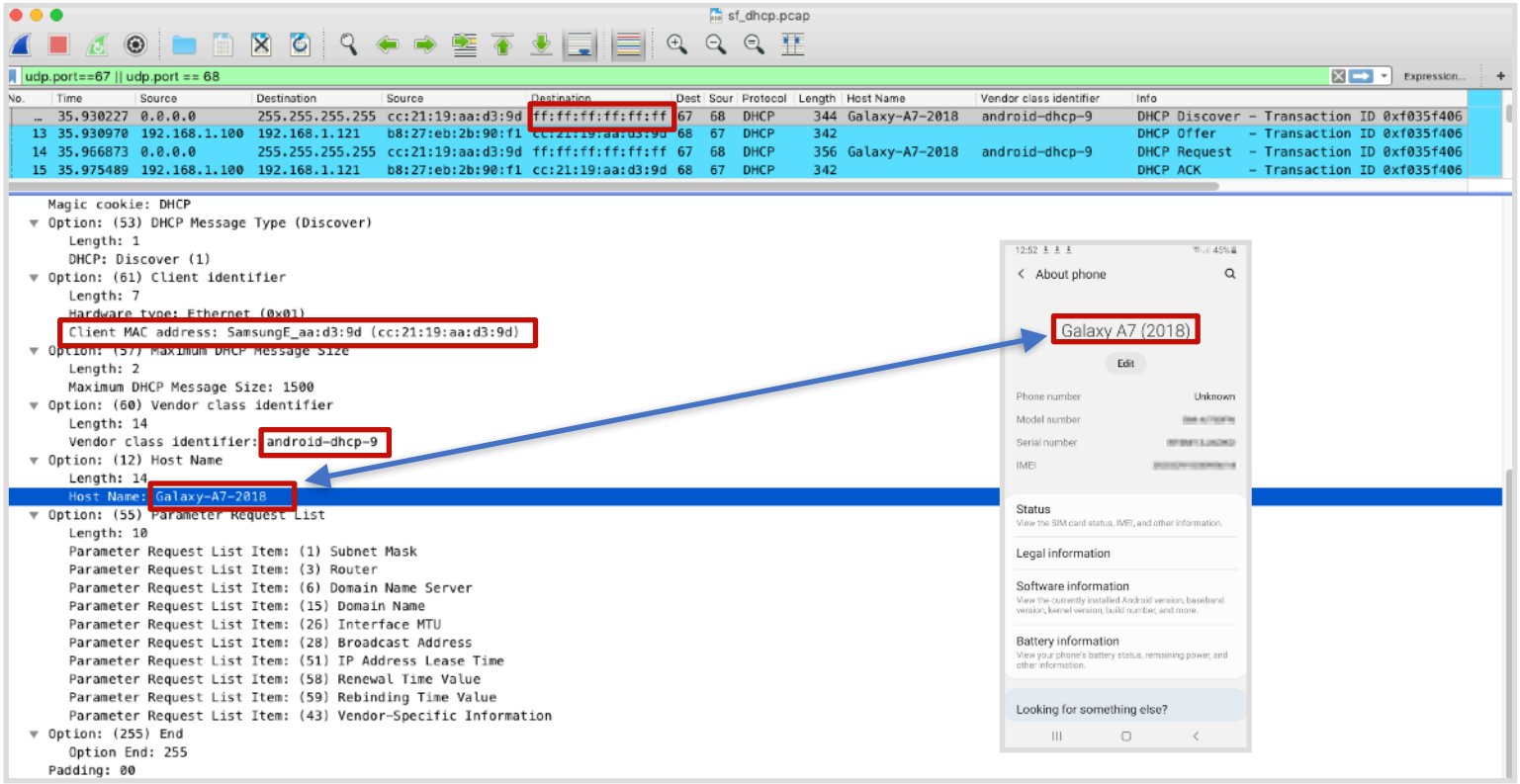
- What happens right after a host has connected to the network?
  - After the **ethernet** cable has been plugged
  - After the **WiFi** has been successfully joined
- To use the network an host typically needs at least to
  - Have an IP address
  - Know the IP address of someone who is in charge of carrying its traffic to the internet (i.e., the **gateway**)
  - Know the IP address of the **DNS server**
- The Dynamic Host Configuration Protocol (**DHCP**) is used to tell the newly connected host all the necessary information to use the joined network

DHCP: Basics

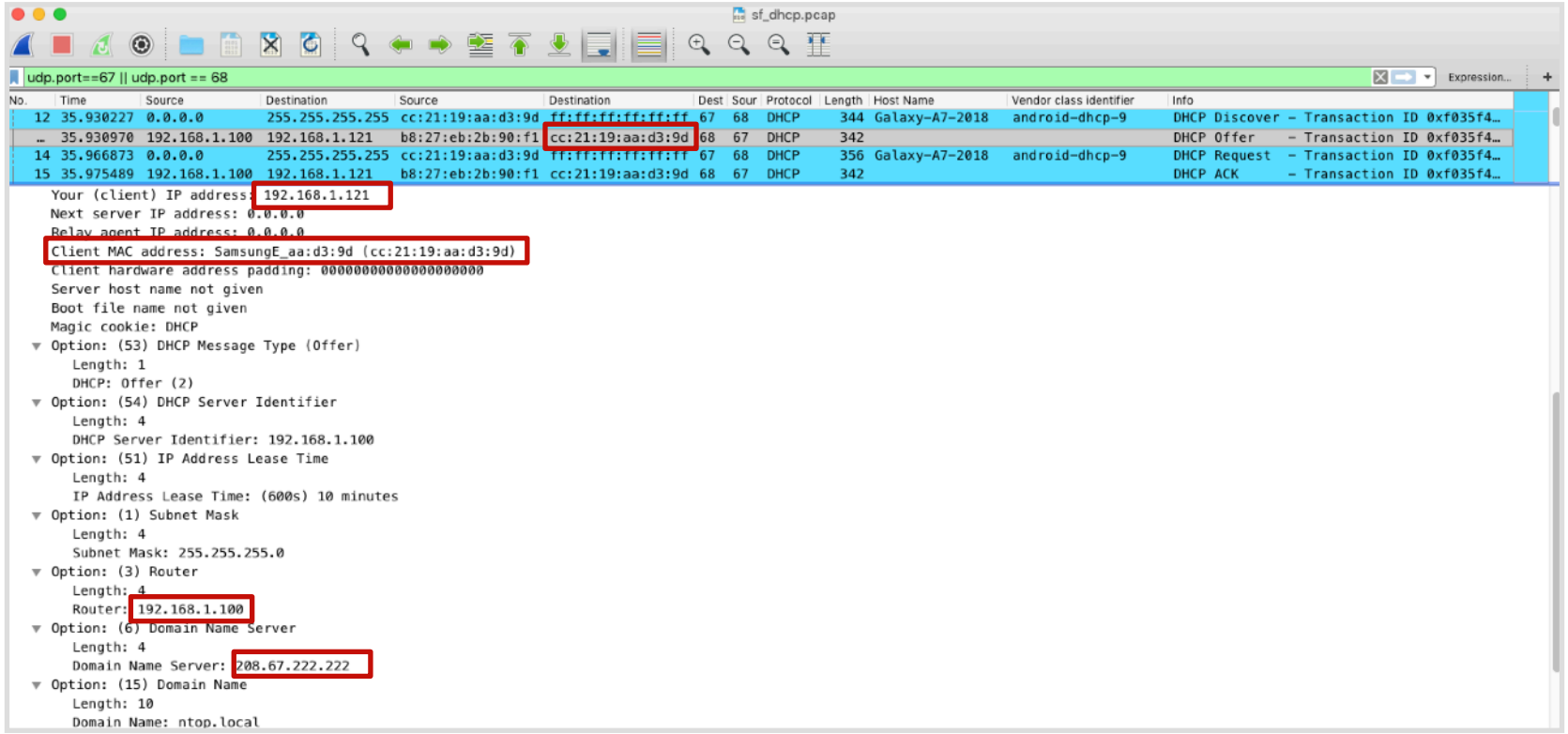
- DHCP client on the host
- DHCP server on the network
- UDP, 4 phases (**DORA**)
  - **D**iscover
  - **O**ffer
  - **R**equest
  - **A**cknowledgement



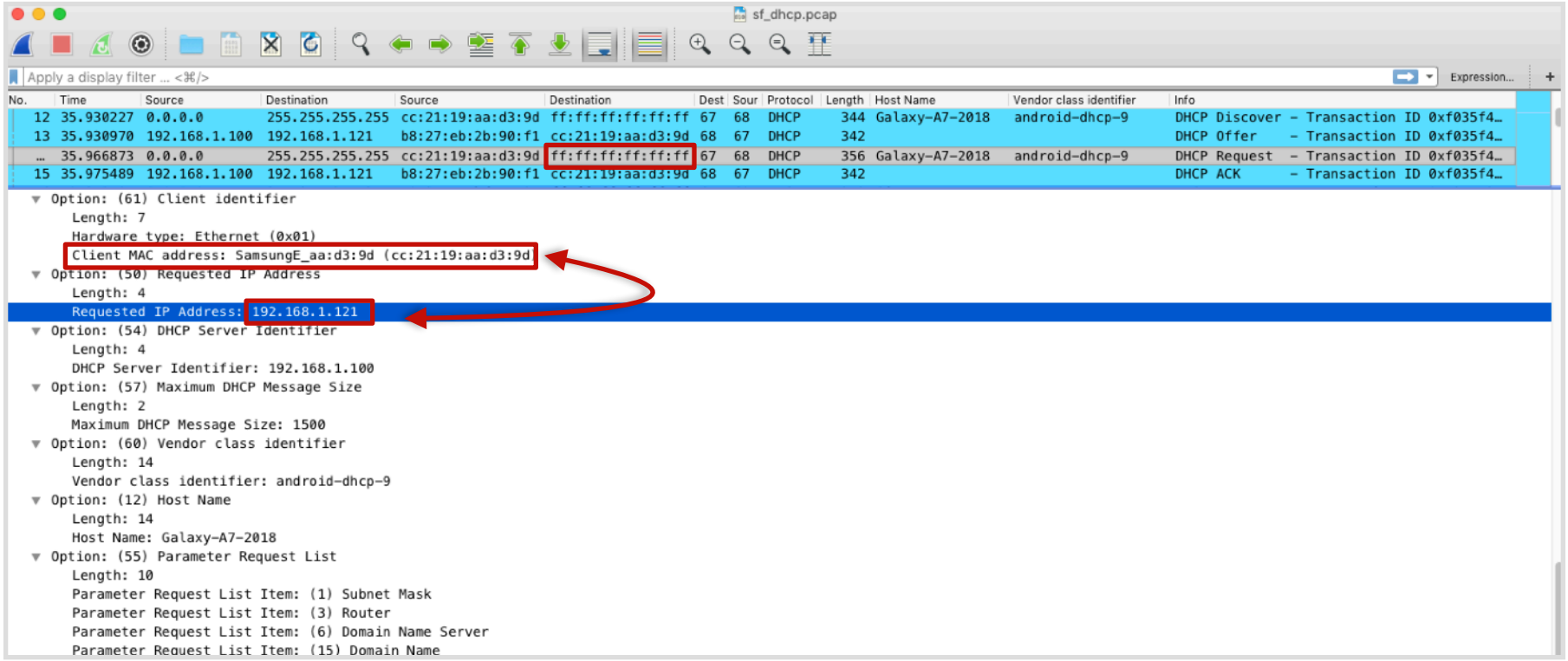
DHCP Discover: Example



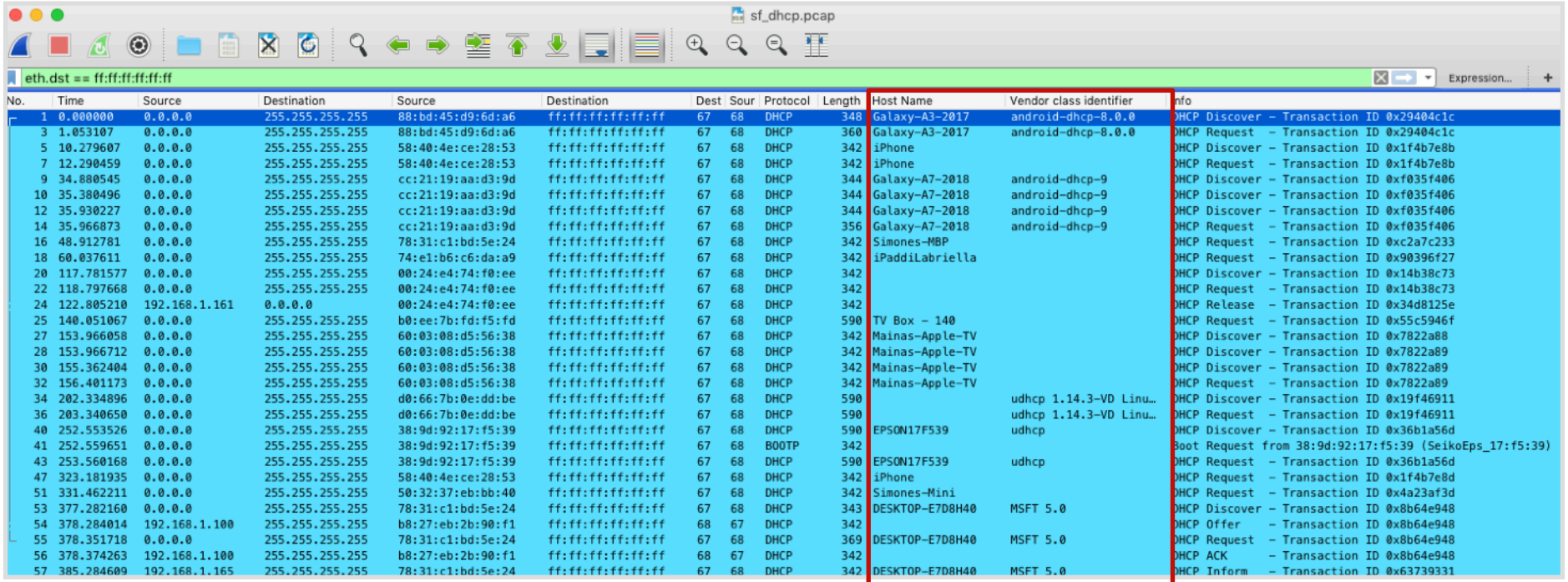
DHCP Offer: Example



DHCP Request: Example



Broadcasted DHCP in a Small Network



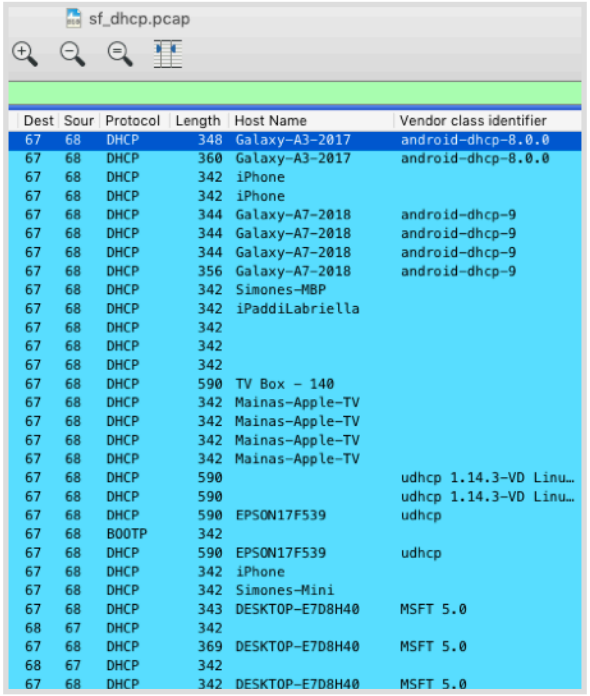
No.	Time	Source	Destination	Source	Destination	Dest	Sour	Protocol	Length	Host Name	Vendor class identifier	nfo
1	0.000000	0.0.0.0	255.255.255.255	88:bd:45:d9:6d:a6	ff:ff:ff:ff:ff:ff	67	68	DHCP	348	Galaxy-A3-2017	android-dhcp-8.0.0	DHCP Discover - Transaction ID 0x29404c1c
3	1.053107	0.0.0.0	255.255.255.255	88:bd:45:d9:6d:a6	ff:ff:ff:ff:ff:ff	67	68	DHCP	360	Galaxy-A3-2017	android-dhcp-8.0.0	DHCP Request - Transaction ID 0x29404c1c
5	10.279607	0.0.0.0	255.255.255.255	58:40:4e:ce:28:53	ff:ff:ff:ff:ff:ff	67	68	DHCP	342	iPhone		DHCP Discover - Transaction ID 0x1f4b7e8b
7	12.290459	0.0.0.0	255.255.255.255	58:40:4e:ce:28:53	ff:ff:ff:ff:ff:ff	67	68	DHCP	342	iPhone		DHCP Request - Transaction ID 0x1f4b7e8b
9	34.880545	0.0.0.0	255.255.255.255	cc:21:19:aa:d3:9d	ff:ff:ff:ff:ff:ff	67	68	DHCP	344	Galaxy-A7-2018	android-dhcp-9	DHCP Discover - Transaction ID 0xf035f406
10	35.380496	0.0.0.0	255.255.255.255	cc:21:19:aa:d3:9d	ff:ff:ff:ff:ff:ff	67	68	DHCP	344	Galaxy-A7-2018	android-dhcp-9	DHCP Discover - Transaction ID 0xf035f406
12	35.930227	0.0.0.0	255.255.255.255	cc:21:19:aa:d3:9d	ff:ff:ff:ff:ff:ff	67	68	DHCP	344	Galaxy-A7-2018	android-dhcp-9	DHCP Discover - Transaction ID 0xf035f406
14	35.966873	0.0.0.0	255.255.255.255	cc:21:19:aa:d3:9d	ff:ff:ff:ff:ff:ff	67	68	DHCP	356	Galaxy-A7-2018	android-dhcp-9	DHCP Request - Transaction ID 0xf035f406
16	48.912781	0.0.0.0	255.255.255.255	78:31:c1:bd:5e:24	ff:ff:ff:ff:ff:ff	67	68	DHCP	342	Simones-MBP		DHCP Request - Transaction ID 0xc2a7c233
18	60.037611	0.0.0.0	255.255.255.255	74:e1:b6:c6:da:a9	ff:ff:ff:ff:ff:ff	67	68	DHCP	342	iPaddiLabriella		DHCP Request - Transaction ID 0x90396f27
20	117.781577	0.0.0.0	255.255.255.255	00:24:e4:74:f0:ee	ff:ff:ff:ff:ff:ff	67	68	DHCP	342			DHCP Discover - Transaction ID 0x14b38c73
22	118.797668	0.0.0.0	255.255.255.255	00:24:e4:74:f0:ee	ff:ff:ff:ff:ff:ff	67	68	DHCP	342			DHCP Request - Transaction ID 0x14b38c73
24	122.805210	192.168.1.161	0.0.0.0	00:24:e4:74:f0:ee	ff:ff:ff:ff:ff:ff	67	68	DHCP	342			DHCP Release - Transaction ID 0x34d8125e
25	140.051067	0.0.0.0	255.255.255.255	b0:ee:7b:fd:f5:fd	ff:ff:ff:ff:ff:ff	67	68	DHCP	590	TV Box - 140		DHCP Request - Transaction ID 0x55c5946f
27	153.966058	0.0.0.0	255.255.255.255	60:03:08:d5:56:38	ff:ff:ff:ff:ff:ff	67	68	DHCP	342	Mainas-Apple-TV		DHCP Discover - Transaction ID 0x7822a88
28	153.966712	0.0.0.0	255.255.255.255	60:03:08:d5:56:38	ff:ff:ff:ff:ff:ff	67	68	DHCP	342	Mainas-Apple-TV		DHCP Discover - Transaction ID 0x7822a89
30	155.362404	0.0.0.0	255.255.255.255	60:03:08:d5:56:38	ff:ff:ff:ff:ff:ff	67	68	DHCP	342	Mainas-Apple-TV		DHCP Discover - Transaction ID 0x7822a89
32	156.401173	0.0.0.0	255.255.255.255	60:03:08:d5:56:38	ff:ff:ff:ff:ff:ff	67	68	DHCP	342	Mainas-Apple-TV		DHCP Request - Transaction ID 0x7822a89
34	202.334896	0.0.0.0	255.255.255.255	d0:66:7b:0e:dd:be	ff:ff:ff:ff:ff:ff	67	68	DHCP	590		udhcp 1.14.3-VD Linu…	DHCP Discover - Transaction ID 0x19f46911
36	203.340650	0.0.0.0	255.255.255.255	d0:66:7b:0e:dd:be	ff:ff:ff:ff:ff:ff	67	68	DHCP	590		udhcp 1.14.3-VD Linu…	DHCP Request - Transaction ID 0x19f46911
40	252.553526	0.0.0.0	255.255.255.255	38:9d:92:17:f5:39	ff:ff:ff:ff:ff:ff	67	68	DHCP	590	EPSON17F539	udhcp	DHCP Discover - Transaction ID 0x36b1a56d
41	252.559651	0.0.0.0	255.255.255.255	38:9d:92:17:f5:39	ff:ff:ff:ff:ff:ff	67	68	BOOTP	342			Boot Request from 38:9d:92:17:f5:39 (SeikoEps_17:f5:39)
43	253.560168	0.0.0.0	255.255.255.255	38:9d:92:17:f5:39	ff:ff:ff:ff:ff:ff	67	68	DHCP	590	EPSON17F539	udhcp	DHCP Request - Transaction ID 0x36b1a56d
47	323.181935	0.0.0.0	255.255.255.255	58:40:4e:ce:28:53	ff:ff:ff:ff:ff:ff	67	68	DHCP	342	iPhone		DHCP Request - Transaction ID 0x1f4b7e8d
51	331.462211	0.0.0.0	255.255.255.255	50:32:37:eb:bb:40	ff:ff:ff:ff:ff:ff	67	68	DHCP	342	Simones-Mini		DHCP Request - Transaction ID 0x4a23af3d
53	377.282160	0.0.0.0	255.255.255.255	78:31:c1:bd:5e:24	ff:ff:ff:ff:ff:ff	67	68	DHCP	343	DESKTOP-E7D8H40	MSFT 5.0	DHCP Discover - Transaction ID 0x8b64e948
54	378.284014	192.168.1.100	255.255.255.255	b8:27:eb:2b:90:f1	ff:ff:ff:ff:ff:ff	68	67	DHCP	342			DHCP Offer - Transaction ID 0x8b64e948
55	378.351718	0.0.0.0	255.255.255.255	78:31:c1:bd:5e:24	ff:ff:ff:ff:ff:ff	67	68	DHCP	369	DESKTOP-E7D8H40	MSFT 5.0	DHCP Request - Transaction ID 0x8b64e948
56	378.374263	192.168.1.100	255.255.255.255	b8:27:eb:2b:90:f1	ff:ff:ff:ff:ff:ff	68	67	DHCP	342			DHCP ACK - Transaction ID 0x8b64e948
57	385.284609	192.168.1.165	255.255.255.255	78:31:c1:bd:5e:24	ff:ff:ff:ff:ff:ff	67	68	DHCP	342	DESKTOP-E7D8H40	MSFT 5.0	DHCP Inform - Transaction ID 0x63739331

How to Use DHCP Data: Discovers and Request [1/2]

- **DHCP Discovers** and **Requests** are sent in **broadcast**
- Every host on the same subnet sees all the DHCP discovers and requests
- Passively determine
  - All the MAC addresses connected to the network
  - All the host names of all the devices connected to the network
  - Associations between IP and MAC addresses

How to Use DHCP Data: Discovers and Request [2/2]

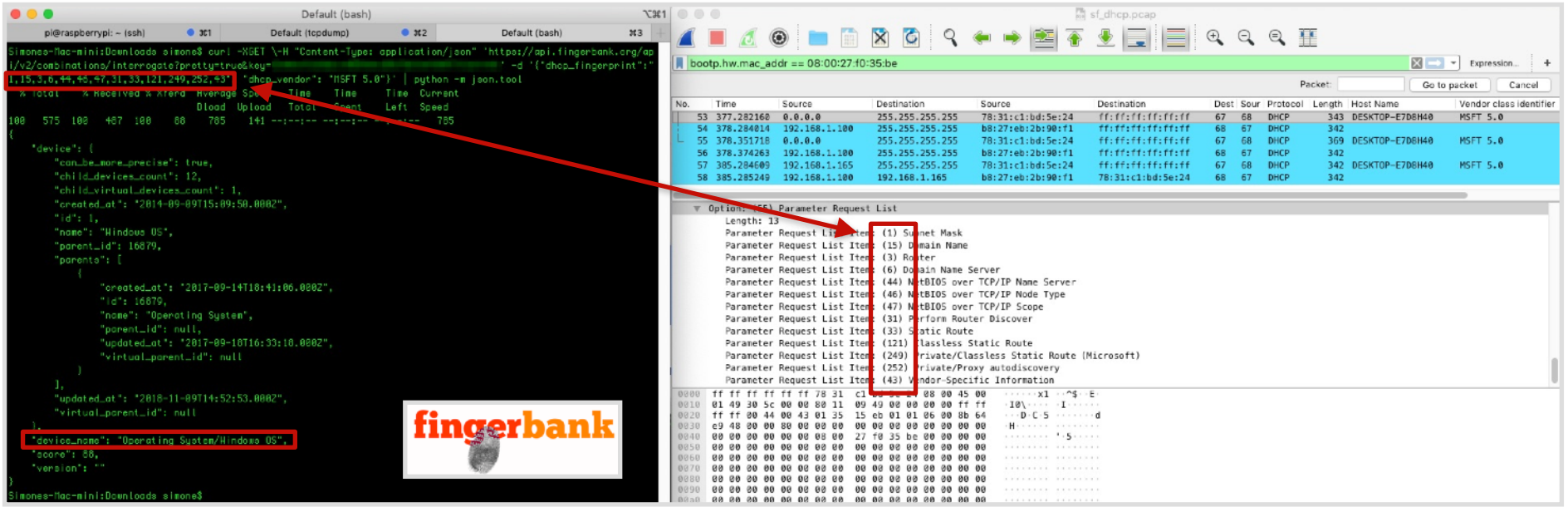
- Host Name / MAC address
  - Associate devices to people (Simones-Mini: the Mac Mini of Simone)
  - Determine device types (Galaxy A7)
- Vendor class identifier
  - Determine the DHCP client and, thus, the operating system



Dest	Sour	Protocol	Length	Host Name	Vendor class identifier
67	68	DHCP	348	Galaxy-A3-2017	android-dhcp-8.0.0
67	68	DHCP	360	Galaxy-A3-2017	android-dhcp-8.0.0
67	68	DHCP	342	iPhone	
67	68	DHCP	342	iPhone	
67	68	DHCP	344	Galaxy-A7-2018	android-dhcp-9
67	68	DHCP	344	Galaxy-A7-2018	android-dhcp-9
67	68	DHCP	344	Galaxy-A7-2018	android-dhcp-9
67	68	DHCP	356	Galaxy-A7-2018	android-dhcp-9
67	68	DHCP	342	Simones-MBP	
67	68	DHCP	342	iPaddiLabriella	
67	68	DHCP	342		
67	68	DHCP	342		
67	68	DHCP	342		
67	68	DHCP	590	TV Box - 140	
67	68	DHCP	342	Mainas-Apple-TV	
67	68	DHCP	342	Mainas-Apple-TV	
67	68	DHCP	342	Mainas-Apple-TV	
67	68	DHCP	342	Mainas-Apple-TV	
67	68	DHCP	590		udhcp 1.14.3-VD Linu…
67	68	DHCP	590		udhcp 1.14.3-VD Linu…
67	68	DHCP	590	EPSON17F539	udhcp
67	68	BOOTP	342		
67	68	DHCP	590	EPSON17F539	udhcp
67	68	DHCP	342	iPhone	
67	68	DHCP	342	Simones-Mini	
67	68	DHCP	343	DESKTOP-E7D8H40	MSFT 5.0
68	67	DHCP	342		
67	68	DHCP	369	DESKTOP-E7D8H40	MSFT 5.0
68	67	DHCP	342		
67	68	DHCP	342	DESKTOP-E7D8H40	MSFT 5.0

How to Use DHCP Data: Fingerprinting

- **Fingerprinting** to guess the OS
  - The order in which the DHCP client asks for certain options is relatively unique and identifies the specific operating system version



DHCP: Secure?

- DHCP **does not include** any mechanism for **authentication**
- Vulnerable to attacks
  - Cannot really trust the response (a 'rouge' DHCP server could respond and tell hosts malicious information such as a DNS server or gateway)
  - Malicious clients can easily exhaust DHCP server resources such as the pool of available IP addresses

Take-Home [1/2]

- Facts
  - Cryptographic protocols or protocols that support encryption may carry certain plaintext information
  - Still a great deal of network protocols carry plaintext information
- Plaintext information can expose information about you, your habits, the devices you use, their features and software

Take-Home [2/2]

- TLS, DNS, mDNS, DNS-SD, SSDP, DHCP are just a few examples
- Make sure you trust the networks you connect to, and you trust those who connect to your networks
- Remove personal information from your devices (e.g., Simone's MacBook Pro)
- Use of VPN and DoH/DoT at minimum