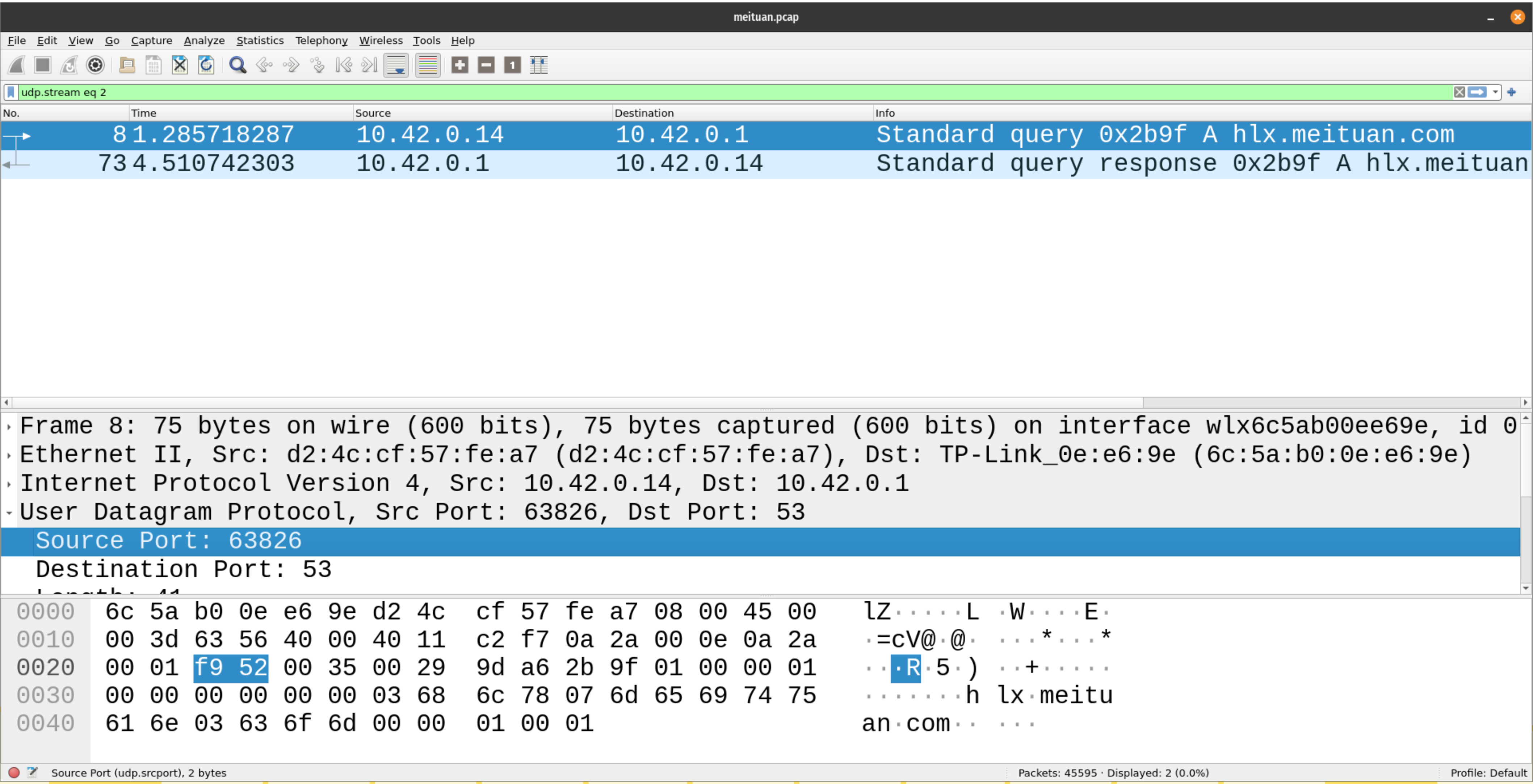
# DNS; introduction to side channels, birthday attacks and signatures

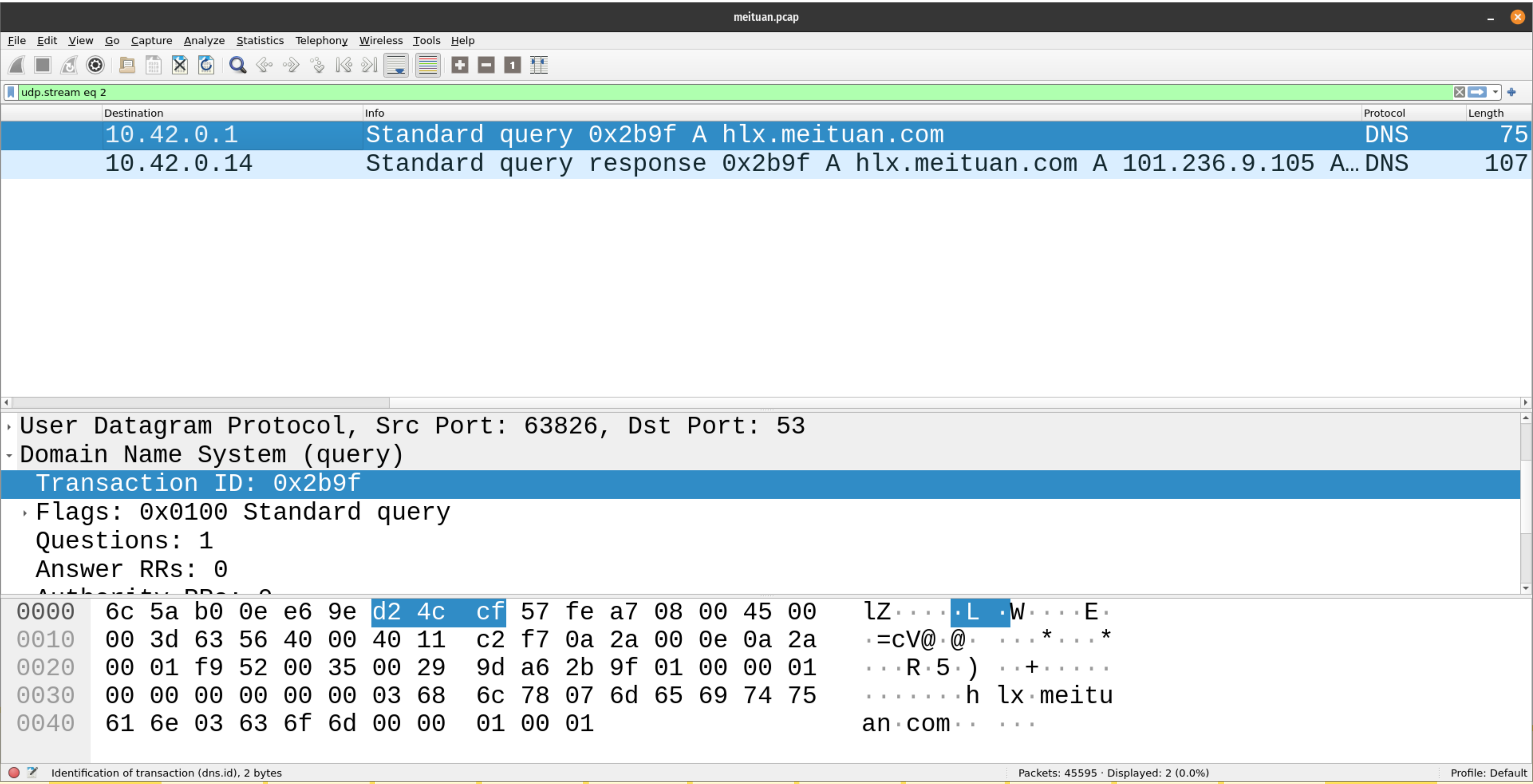
CSE 468 Fall 2025

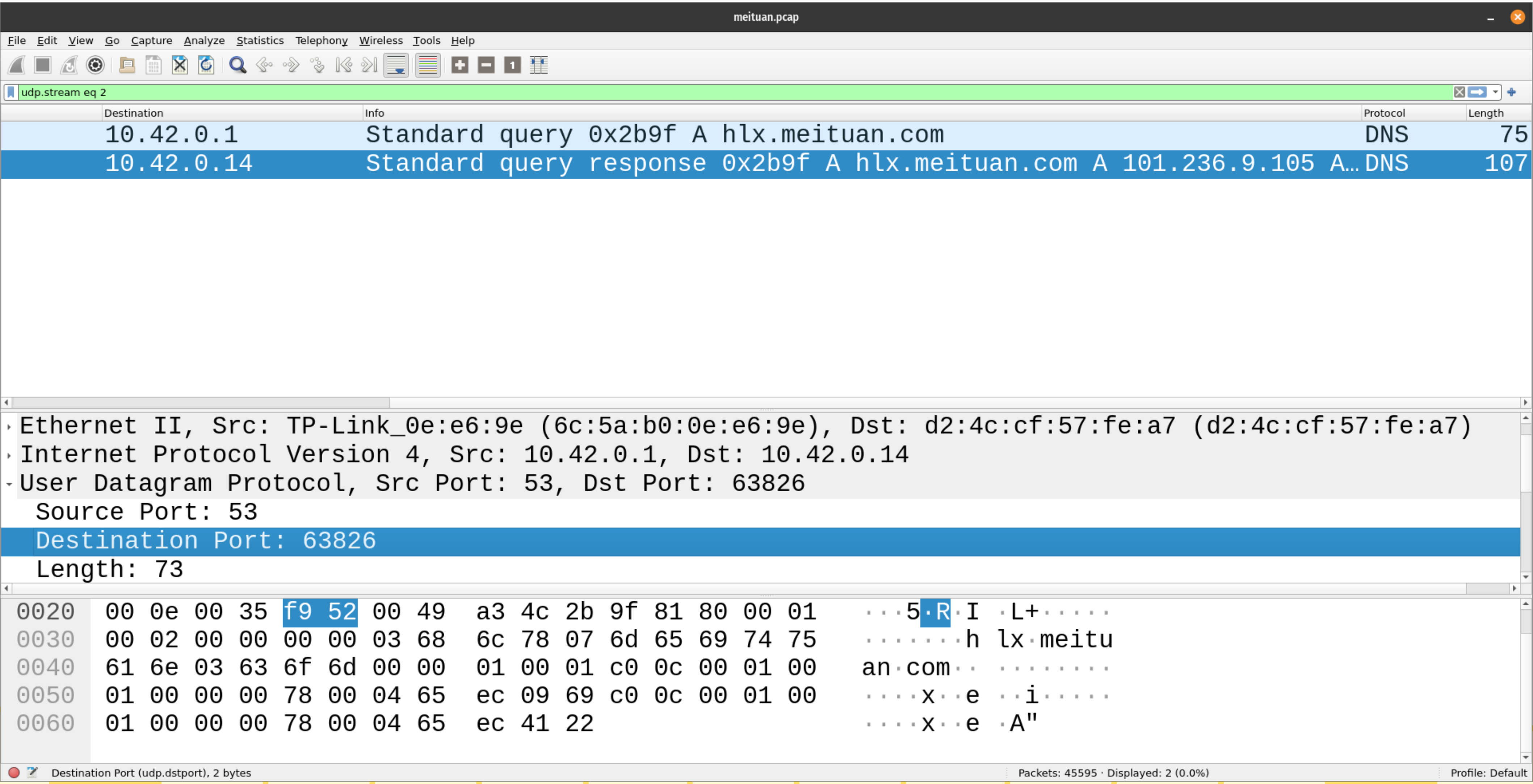
jedimaestro@asu.edu

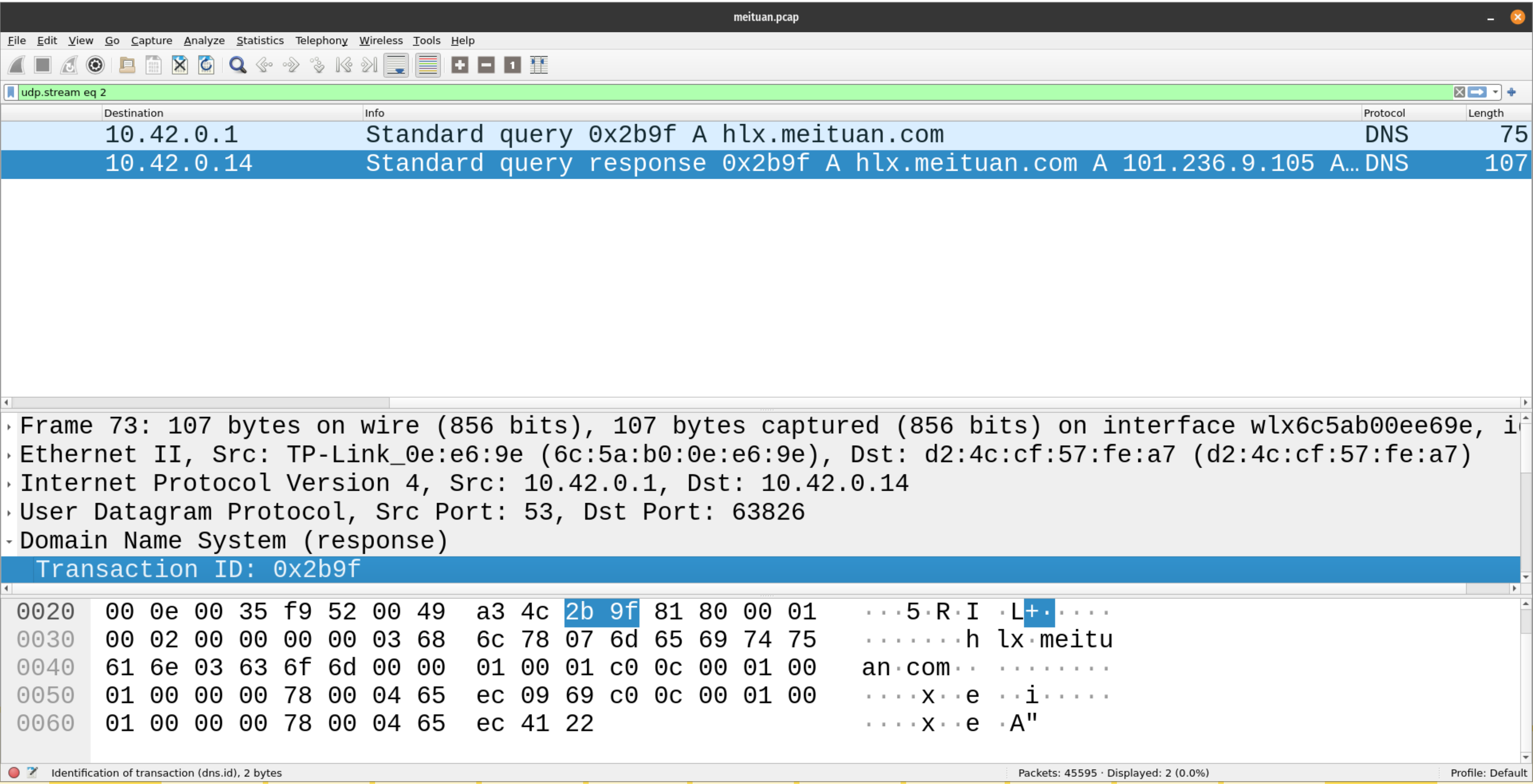
# Outline

- DNS basics
- On-path *vs.* in-path *vs.* off-path
- Birthday attacks
  - Example: Wagner Sacramento's birthday attack on DNS (2002)
- Dan Kaminsky's DNS poisoning attack (2008)
- Side channel attacks (information theory)
  - Example: Fragmentation attack
- Solution: signatures
  - Important ingredient for signatures: extended Euclidean algorithm

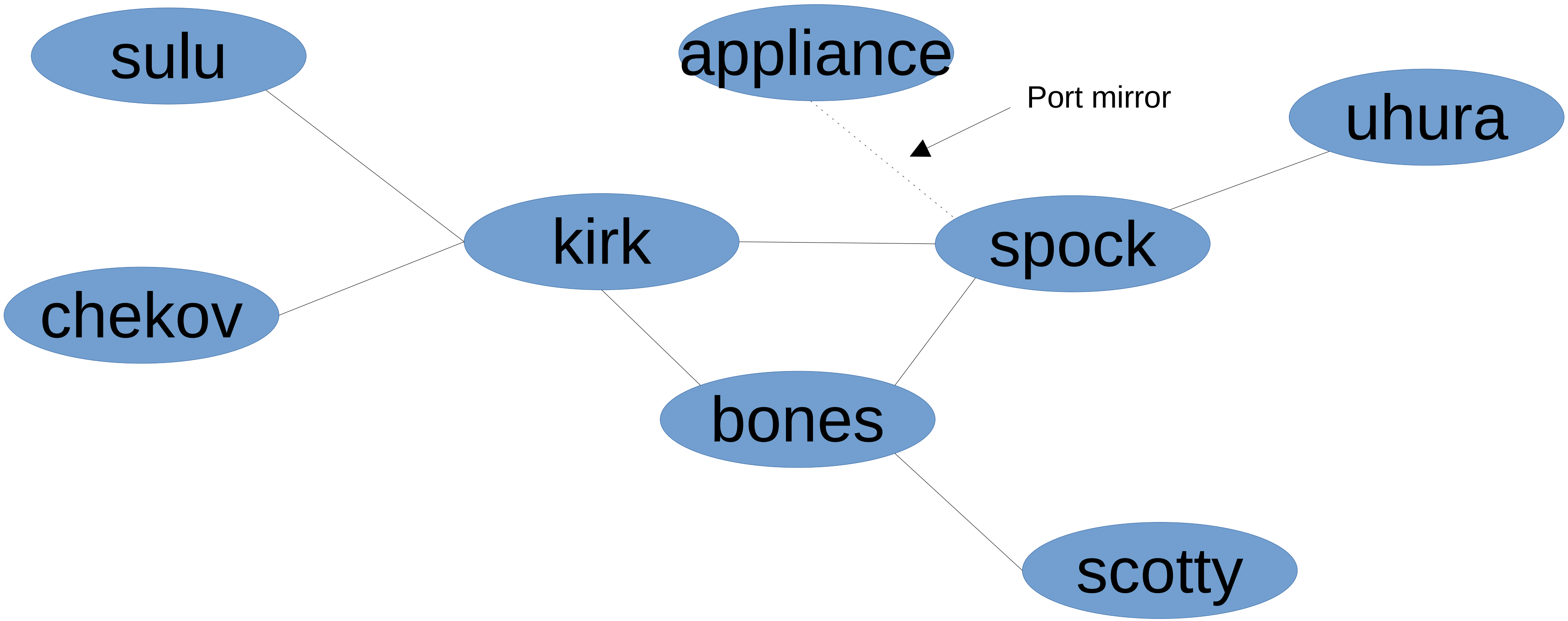

meituan.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
udp.stream eq 2
No. Time Source Destination Info
8 1.285718287 10.42.0.14 10.42.0.1 Standard query 0x2b9f A hlx.meituan.com
73 4.510742303 10.42.0.1 10.42.0.14 Standard query response 0x2b9f A hlx.meituan
Frame 8: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface wlx6c5ab00ee69e, id 0
Ethernet II, Src: d2:4c:cf:57:fe:a7 (d2:4c:cf:57:fe:a7), Dst: TP-Link_0e:e6:9e (6c:5a:b0:0e:e6:9e)
Internet Protocol Version 4, Src: 10.42.0.14, Dst: 10.42.0.1
User Datagram Protocol, Src Port: 63826, Dst Port: 53
Source Port: 63826
Destination Port: 53
0000 6c 5a b0 0e e6 9e d2 4c cf 57 fe a7 08 00 45 00 lZ·····L ·W····E·
0010 00 3d 63 56 40 00 40 11 c2 f7 0a 2a 00 0e 0a 2a ·=cV@·@· ···*···*
0020 00 01 f9 52 00 35 00 29 9d a6 2b 9f 01 00 00 01 ···R·5·) ··+·····
0030 00 00 00 00 00 00 03 68 6c 78 07 6d 65 69 74 75 ·······h lx·meitu
0040 61 6e 03 63 6f 6d 00 00 01 00 01 an·com·· ···
Source Port (udp.srcport), 2 bytes
Packets: 45595 · Displayed: 2 (0.0%)
Profile: Default


meituan.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
udp.stream eq 2
Destination Info Protocol Length
10.42.0.1 Standard query 0x2b9f A hlx.meituan.com DNS 75
10.42.0.14 Standard query response 0x2b9f A hlx.meituan.com A 101.236.9.105 A… DNS 107
User Datagram Protocol, Src Port: 63826, Dst Port: 53
Domain Name System (query)
Transaction ID: 0x2b9f
Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
0000 6c 5a b0 0e e6 9e d2 4c cf 57 fe a7 08 00 45 00 lZ····L·W····E·
0010 00 3d 63 56 40 00 40 11 c2 f7 0a 2a 00 0e 0a 2a ·=cV@·@····*···*
0020 00 01 f9 52 00 35 00 29 9d a6 2b 9f 01 00 00 01 ···R·5·)··+·····
0030 00 00 00 00 00 00 03 68 6c 78 07 6d 65 69 74 75 ·······hlx·meitu
0040 61 6e 03 63 6f 6d 00 00 01 00 01 an·com·····
Identification of transaction (dns.id), 2 bytes
Packets: 45595 · Displayed: 2 (0.0%)
Profile: Default

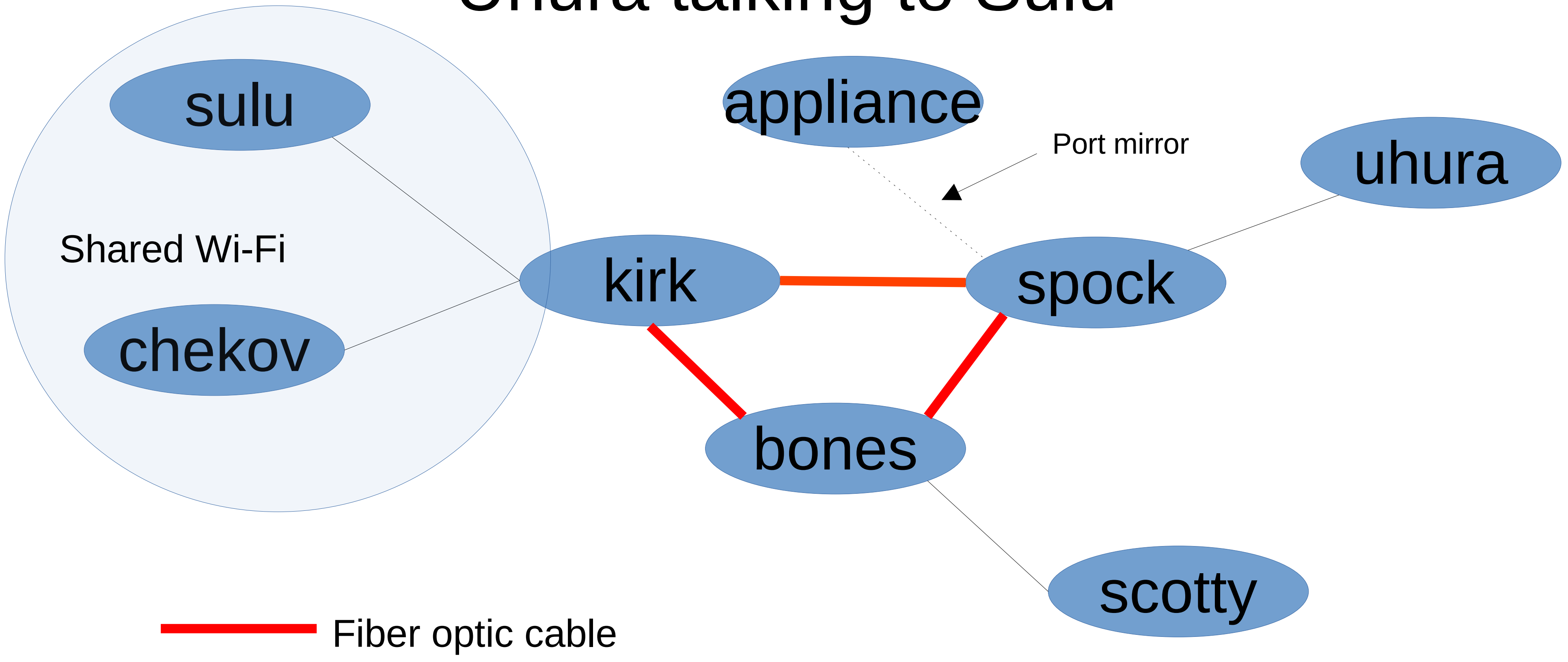

meituan.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
udp.stream eq 2
Destination Info Protocol Length
10.42.0.1 Standard query 0x2b9f A hlx.meituan.com DNS 75
10.42.0.14 Standard query response 0x2b9f A hlx.meituan.com A 101.236.9.105 A… DNS 107
Ethernet II, Src: TP-Link_0e:e6:9e (6c:5a:b0:0e:e6:9e), Dst: d2:4c:cf:57:fe:a7 (d2:4c:cf:57:fe:a7)
Internet Protocol Version 4, Src: 10.42.0.1, Dst: 10.42.0.14
User Datagram Protocol, Src Port: 53, Dst Port: 63826
Source Port: 53
Destination Port: 63826
Length: 73
0020 00 0e 00 35 f9 52 00 49 a3 4c 2b 9f 81 80 00 01 ···5·R·I ·L+·····
0030 00 02 00 00 00 00 03 68 6c 78 07 6d 65 69 74 75 ·······h lx·meitu
0040 61 6e 03 63 6f 6d 00 00 01 00 01 c0 0c 00 01 00 an·com·· ········
0050 01 00 00 00 78 00 04 65 ec 09 69 c0 0c 00 01 00 ····x··e ··i·····
0060 01 00 00 00 78 00 04 65 ec 41 22 ····x··e ·A"
Destination Port (udp.dstport), 2 bytes
Packets: 45595 · Displayed: 2 (0.0%)
Profile: Default


meituan.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
udp.stream eq 2
Destination Info Protocol Length
10.42.0.1 Standard query 0x2b9f A hlx.meituan.com DNS 75
10.42.0.14 Standard query response 0x2b9f A hlx.meituan.com A 101.236.9.105 A… DNS 107
Frame 73: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface wlx6c5ab00ee69e, i
Ethernet II, Src: TP-Link_0e:e6:9e (6c:5a:b0:0e:e6:9e), Dst: d2:4c:cf:57:fe:a7 (d2:4c:cf:57:fe:a7)
Internet Protocol Version 4, Src: 10.42.0.1, Dst: 10.42.0.14
User Datagram Protocol, Src Port: 53, Dst Port: 63826
Domain Name System (response)
Transaction ID: 0x2b9f
0020 00 0e 00 35 f9 52 00 49 a3 4c 2b 9f 81 80 00 01 ···5·R·I ·L+·····
0030 00 02 00 00 00 00 03 68 6c 78 07 6d 65 69 74 75 ·······h lx·meitu
0040 61 6e 03 63 6f 6d 00 00 01 00 01 c0 0c 00 01 00 an·com·· ········
0050 01 00 00 00 78 00 04 65 ec 09 69 c0 0c 00 01 00 ····x··e ··i·····
0060 01 00 00 00 78 00 04 65 ec 41 22 ····x··e ·A"
Identification of transaction (dns.id), 2 bytes
Packets: 45595 · Displayed: 2 (0.0%)
Profile: Default

# Uhura talking to Sulu



# Uhura talking to Sulu



# sulu == DNS client, uhura == DNS server

- kirk and spock are in-path
- appliance is on-path
  - Gets a <u>copy</u> of the packets from the port mirror on kirk
- chekov is on-path
  - Shared Wi-Fi with sulu, kirk has a wireless interface and two fiber optic interfaces
- scotty and bones are off-path

# On-path attack

- Need to respond faster than the DNS server
  - Not hard, 3 seconds (example above) is an eternity
  - Maybe DoS the DNS server
- Need to get the TXID and source port correct
  - Trivial, just read them from the packet

# In-path attack

- ~~Need to respond faster than the DNS server~~
  - ~~Not hard, 3 seconds (example above) is an eternity~~
  - ~~Maybe DoS the DNS server~~
- Need to get the TXID and source port correct
  - Trivial, just read them from the packet
- Just don’t forward the request to the DNS server
  - Or, do and then modify the response on its way back

# Off-path attack

- Need to respond faster than the DNS server
  - ~~Not hard, 3 seconds (example above) is an eternity~~
  - Maybe DoS the DNS server
- Need to get the TXID and source port correct
  - **Not easy**, being off path means you’re *blind* to these values
  - Guessing might work ($2^{16} * 2^{16} = 2^{32}$)
    - Side channels and birthday attacks even better
- Need to know what was queried and when
  - Cache poisoning (you know these things because you caused it)

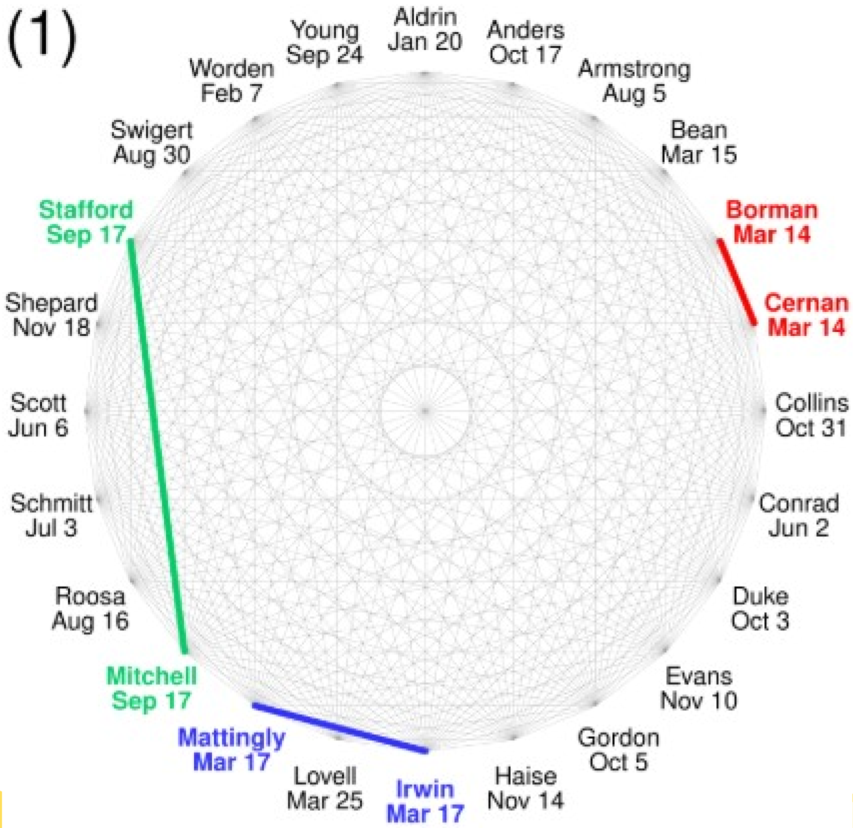
# Birthday Attacks

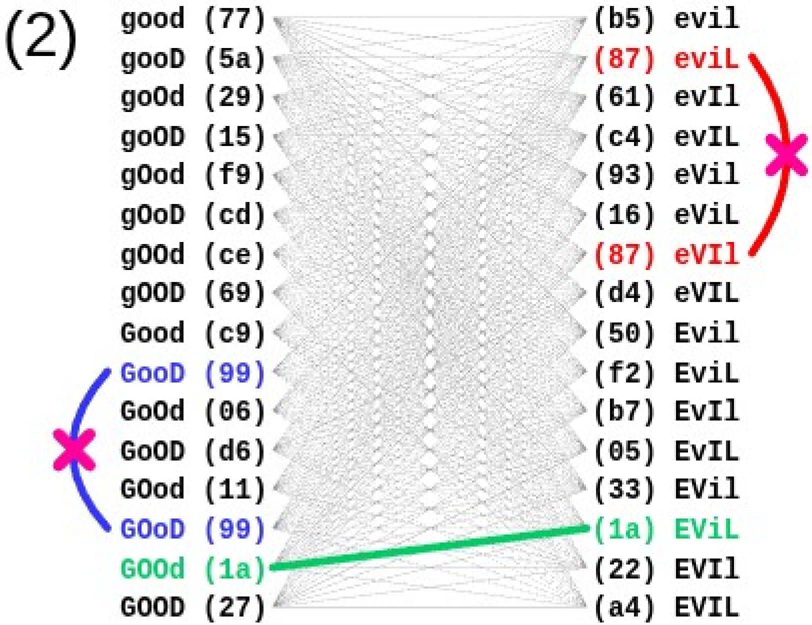
- https://www.kb.cert.org/vuls/id/457
- 2002

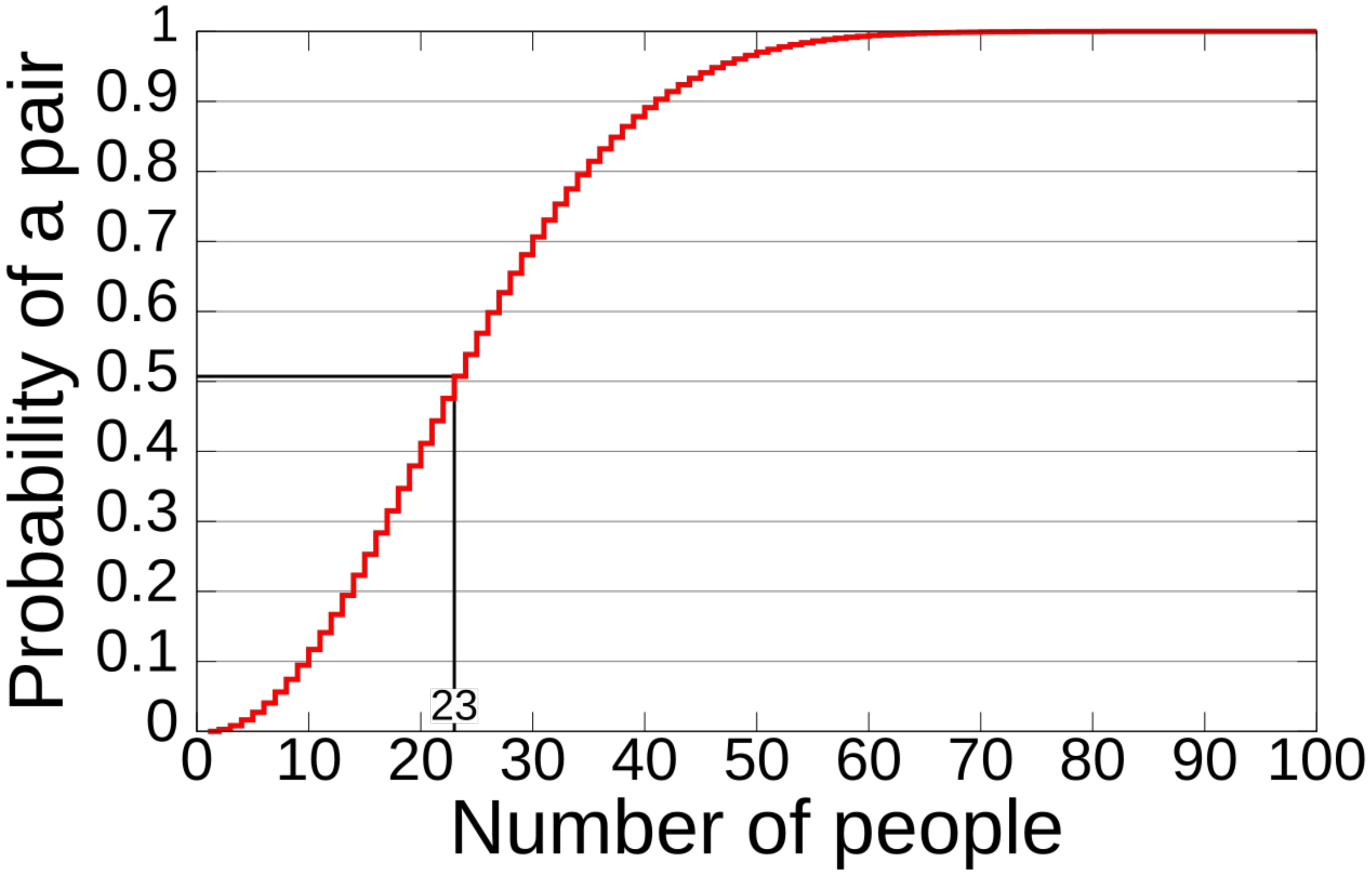
| If the attacker has to guess... | ...and is limited to the following number of open requests... | ...it will take the following number of packets to achieve a 50% success rate (includes both requests and responses) |
|---|---|---|
| TID only (16bits) | 1 | 32.7 k ($2^{15}$) |
| TID only (16bits) | 4 | 10.4 k |
| TID only (16bits) | 200 | 427 |
| TID only (16bits) | unlimited | 426 |
| TID and port (32 bits) | 1 | 2.1 billion ($2^{31}$) |
| TID and port (32 bits) | 4 | 683 million |
| TID and port (32 bits) | 200 | 15 million |
| TID and port (32 bits) | unlimited | 109 k |

***Table 1: Number of packets required to reach 50% success probability for various numbers of open queries***

https://en.wikipedia.org/wiki/Birthday_attack




Probability of a pair
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
23
0 10 20 30 40 50 60 70 80 90 100
Number of people

This process can be generalized to a group of $n$ people, where $p(n)$ is the probability of at least two of the $n$ people sharing a birthday. It is easier to first calculate the probability $\bar{p}(n)$ that all $n$ birthdays are *different*. According to the pigeonhole principle, $\bar{p}(n)$ is zero when $n > 365$. When $n \leq 365$:

$$\bar{p}(n) = 1 \times \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \cdots \times \left(1 - \frac{n-1}{365}\right)$$

The Taylor series expansion of the exponential function (the constant $e \approx 2.718\,281\,828$)

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots$$

provides a first-order approximation for $e^x$ for $|x| \ll 1$:

$$e^x \approx 1 + x.$$

To apply this approximation to the first expression derived for $\bar{p}(n)$, set $x = -\frac{a}{365}$. Thus,

$$e^{-a/365} \approx 1 - \frac{a}{365}.$$

Then, replace $a$ with non-negative integers for each term in the formula of $\bar{p}(n)$ until $a = n - 1$, for example, when $a = 1$,

$$e^{-1/365} \approx 1 - \frac{1}{365}.$$

The first expression derived for $\bar{p}(n)$ can be approximated as

$$\bar{p}(n) \approx 1 \cdot e^{-1/365} \cdot e^{-2/365} \cdots e^{-(n-1)/365}$$

$$= e^{-\left(1+2+\cdots+(n-1)\right)/365}$$

$$= e^{-\frac{n(n-1)/2}{365}} = e^{-\frac{n(n-1)}{730}}.$$

Therefore,

$$p(n) = 1 - \bar{p}(n) \approx 1 - e^{-\frac{n(n-1)}{730}}.$$

$$p(n,d) \approx 1 - e^{-\frac{n(n-1)}{2d}}$$

An even coarser approximation is given by

$$p(n) \approx 1 - e^{-\frac{n^2}{730}},$$

A good rule of thumb which can be used for mental calculation is the relation

$$p(n,d) \approx \frac{n^2}{2d}$$

which can also be written as

$$n \approx \sqrt{2d \times p(n)}$$

which works well for probabilities less than or equal to $\frac{1}{2}$. In these equations, $d$ is the number of days in a year.

For instance, to estimate the number of people required for a $\frac{1}{2}$ chance of a shared birthday, we get

$$n \approx \sqrt{2 \times 365 \times \tfrac{1}{2}} = \sqrt{365} \approx 19$$

Which is not too far from the correct answer of 23.

Solution to the specific birthday attack on DNS above… Don’t allow multiple queries for the same domain at the same time.

# Dan Kaminsky’s attack (2008)

- https://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-Kaminsky/BlackHat-Japan-08-Kaminsky-DNS08-BlackOps.pdf

# DNS is distributed

- Three possible answers to any question
  - “Here’s your answer”
  - “Go away”
  - “I don’t know, ask that guy over there”
    - This is delegation. You start with a request, and then get bounced around all over the place.
    - 13 root servers: “www.foo.com? I don’t know, go ask the com server, it’s at 1.2.3.4”
    - Com server: “www.foo.com? I don’t know, go ask the foo.com server, it’s at 2.3.4.5”
    - Foo.com server: “www.foo.com? Yeah, that’s at 3.4.5.6.”

- If the bad guy can reply 100 times before the good guy returns, that 65536 to 1 advantage drops to 655 to 1.
  - Alas…still long odds. And when he loses, he has to wait the TTL. That could be 655 days – almost 2 years!
  - Or maybe not.

# Finally, the bad guy doesn't actually need to wait to try again.

- If the bad guy asks the name server to look up www.foo.com ten times, there will only be one race with the good guy
  - The first race will be lost (most likely), and then the other nine will be suppressed by the TTL
    - No new races on this name for one more day! Here, use the answer from a while ago
    - So, can we race on other names?
- If the bad guy asks the name server to look up 1.foo.com, 2.foo.com, 3.foo.com, and so on, for ten names, there will be 10 races with the good guy
  - TTL only stops repeated races for the same name!
- Eventually, the bad guy will guess the right TXID before the good guy shows up with it
  - And now…the bad guy is the proud spoofer of … 83.foo.com
  - So? He didn't *want* to poison 83.foo.com. He wanted www.foo.com

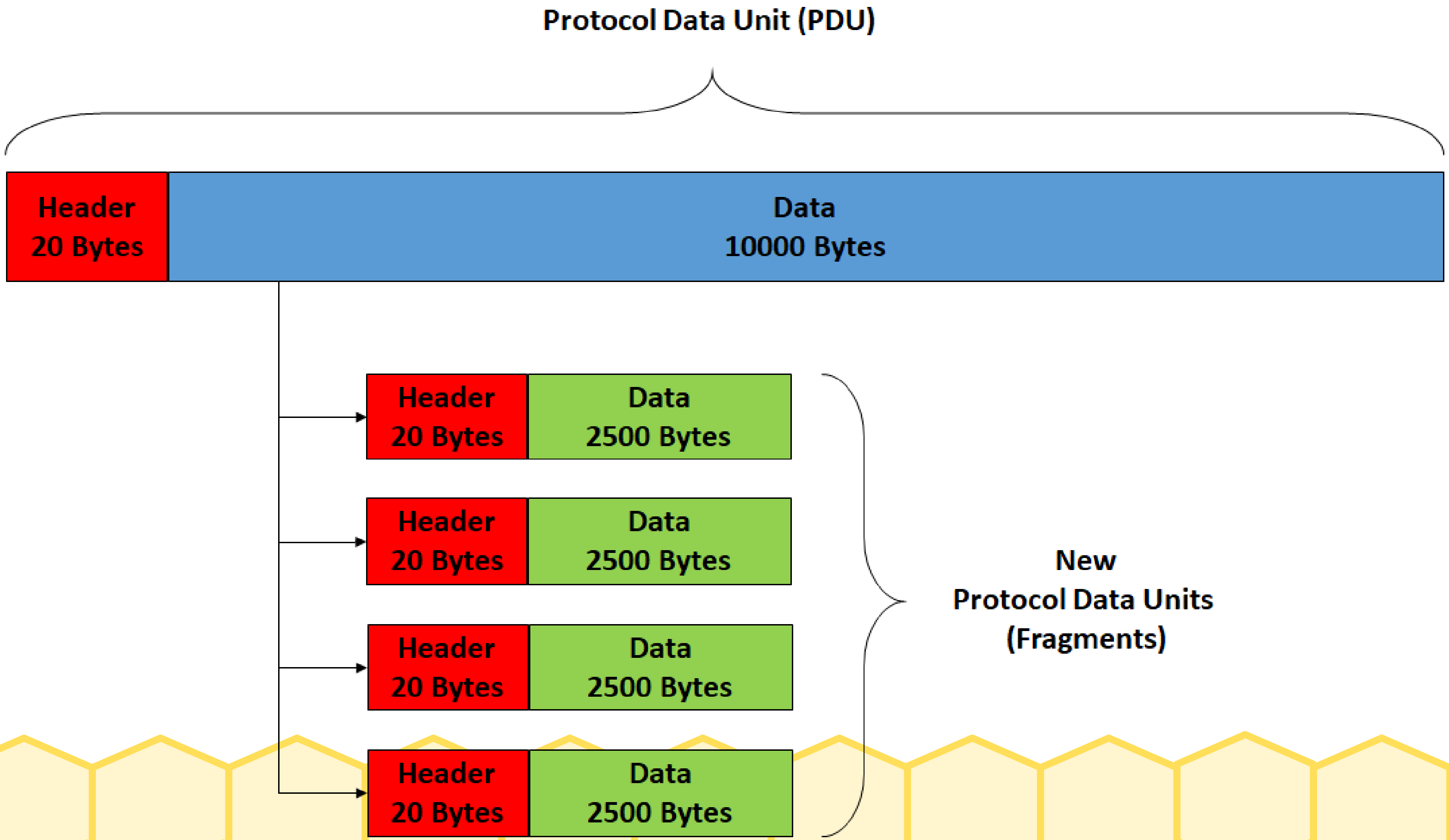
# Bait and Switch

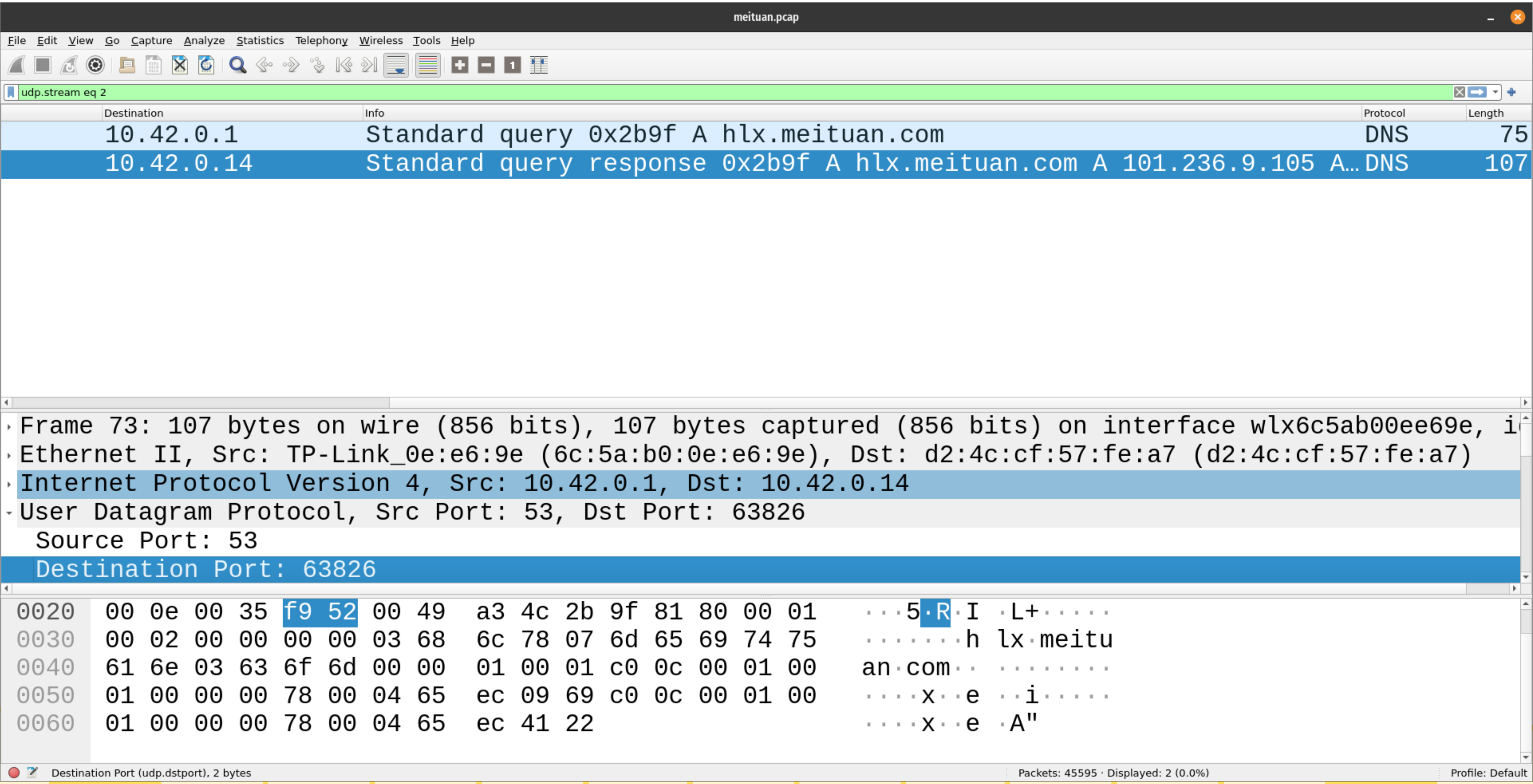
- Is it possible for a bad guy, who has won the race for 83.foo.com, to end up stealing www.foo.com as well?
  - He has three possible replies that can be associated with correctly guessed TXID
  - 1) “Here’s your answer for 83.foo.com – it’s 6.6.6.6”
  - 2) “I don’t know the answer for 83.foo.com.”
  - 3) “83.foo.com? I don’t know, go ask the www.foo.com server, it’s at 6.6.6.6”
    - This has to work – it’s just another delegation
      - 13 root servers: “83.foo.com? I don’t know, go ask the com server, it’s at 1.2.3.4”
      - Com server: “83.foo.com? I don’t know, go ask the foo.com server, it’s at 2.3.4.5”
      - Foo.com server: “83.foo.com? I don’t know, go ask the www.foo.com server, it’s at 6.6.6.6”

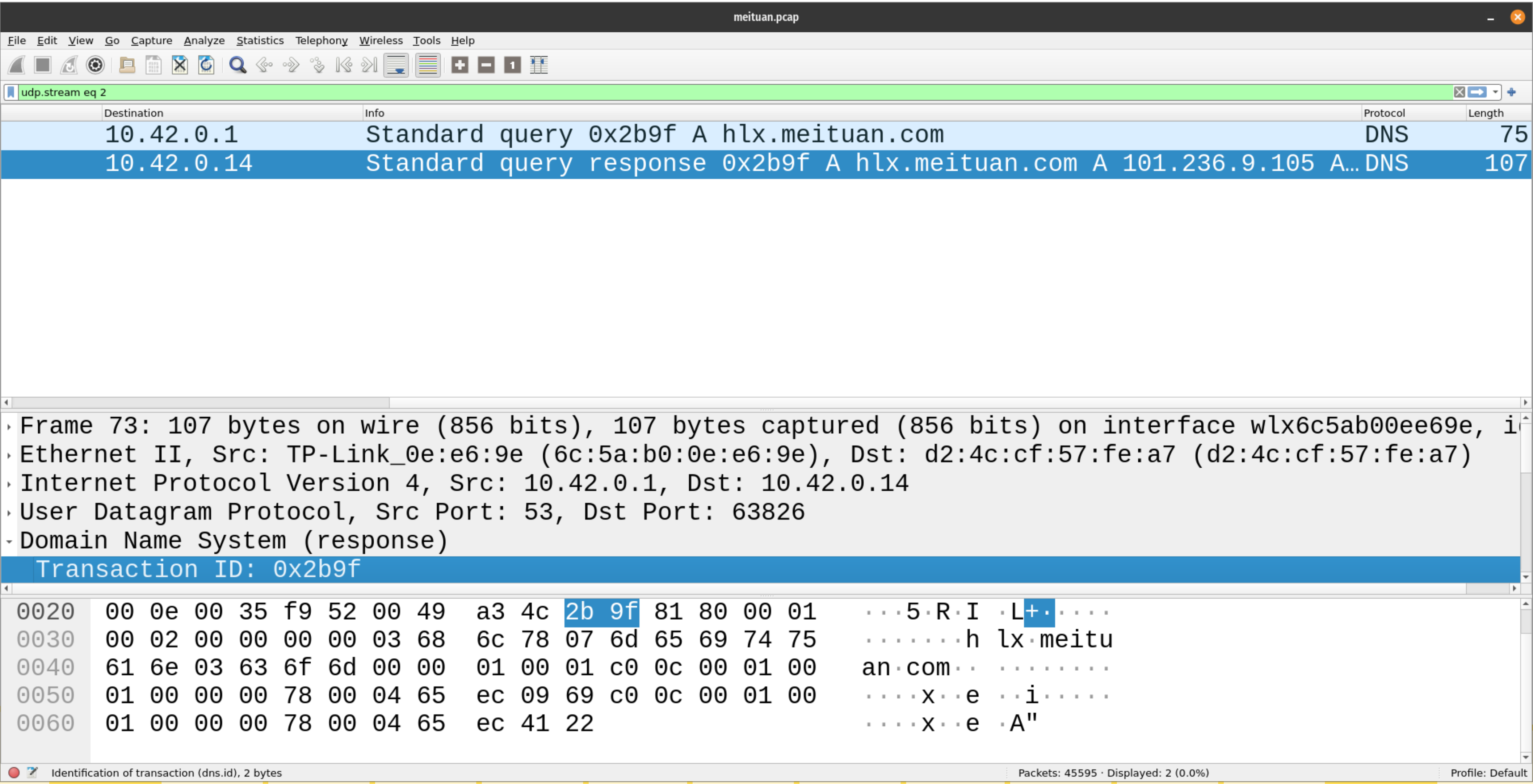
Solution to the Kaminsky attack… OSes now randomize source ports.

But, what if we didn't have to guess the TXID or source port?

https://en.wikipedia.org/wiki/IP_fragmentation



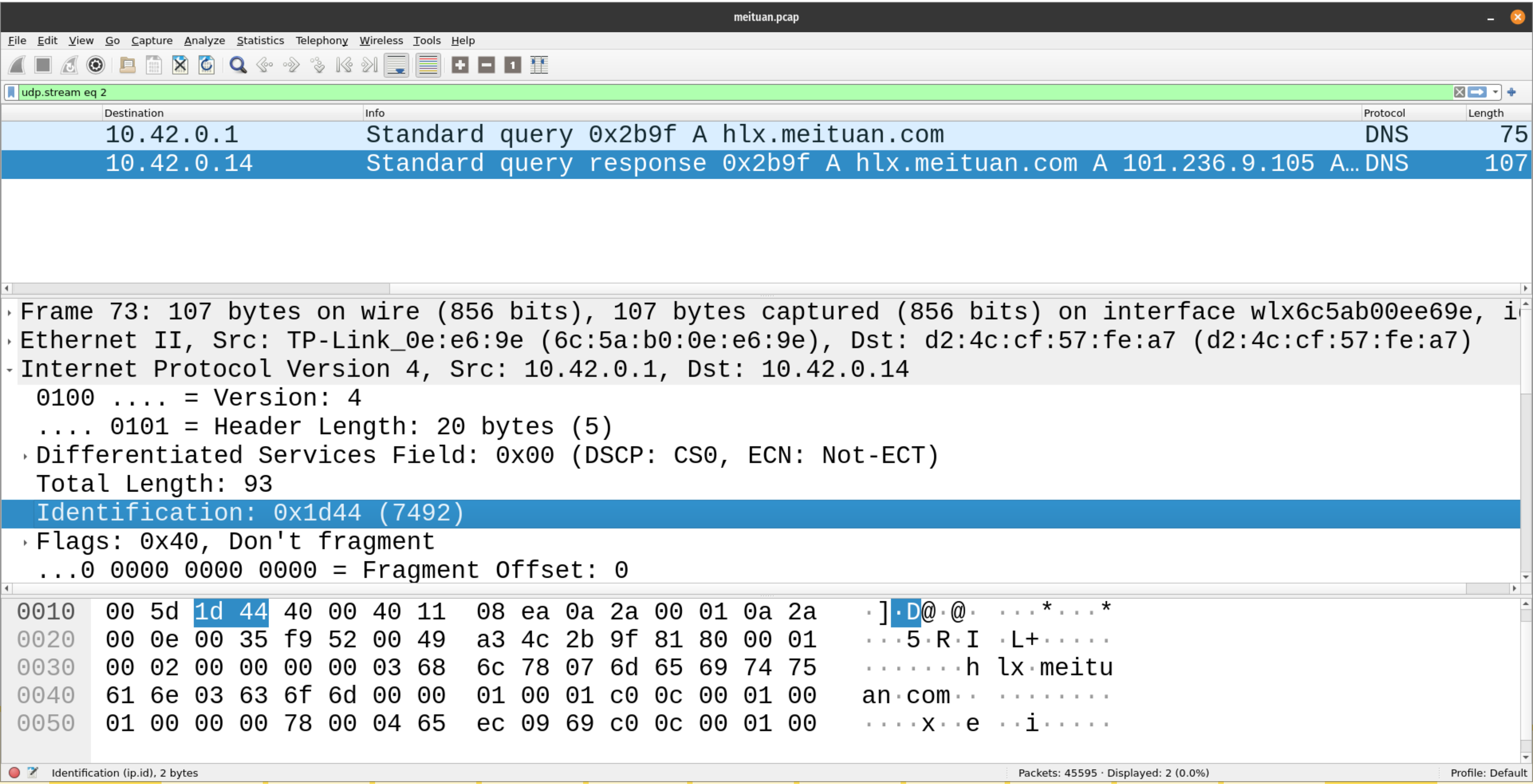

meituan.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
udp.stream eq 2
Destination Info Protocol Length
10.42.0.1 Standard query 0x2b9f A hlx.meituan.com DNS 75
10.42.0.14 Standard query response 0x2b9f A hlx.meituan.com A 101.236.9.105 A… DNS 107
Frame 73: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface wlx6c5ab00ee69e, i
Ethernet II, Src: TP-Link_0e:e6:9e (6c:5a:b0:0e:e6:9e), Dst: d2:4c:cf:57:fe:a7 (d2:4c:cf:57:fe:a7)
Internet Protocol Version 4, Src: 10.42.0.1, Dst: 10.42.0.14
User Datagram Protocol, Src Port: 53, Dst Port: 63826
Source Port: 53
Destination Port: 63826
0020 00 0e 00 35 f9 52 00 49 a3 4c 2b 9f 81 80 00 01 ···5·R·I ·L+·····
0030 00 02 00 00 00 00 03 68 6c 78 07 6d 65 69 74 75 ·······h lx·meitu
0040 61 6e 03 63 6f 6d 00 00 01 00 01 c0 0c 00 01 00 an·com·· ········
0050 01 00 00 00 78 00 04 65 ec 09 69 c0 0c 00 01 00 ····x··e ··i·····
0060 01 00 00 00 78 00 04 65 ec 41 22 ····x··e ·A"
Destination Port (udp.dstport), 2 bytes
Packets: 45595 · Displayed: 2 (0.0%)
Profile: Default


meituan.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
udp.stream eq 2
Destination Info Protocol Length
10.42.0.1 Standard query 0x2b9f A hlx.meituan.com DNS 75
10.42.0.14 Standard query response 0x2b9f A hlx.meituan.com A 101.236.9.105 A… DNS 107
Frame 73: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface wlx6c5ab00ee69e, i
Ethernet II, Src: TP-Link_0e:e6:9e (6c:5a:b0:0e:e6:9e), Dst: d2:4c:cf:57:fe:a7 (d2:4c:cf:57:fe:a7)
Internet Protocol Version 4, Src: 10.42.0.1, Dst: 10.42.0.14
User Datagram Protocol, Src Port: 53, Dst Port: 63826
Domain Name System (response)
Transaction ID: 0x2b9f
0020 00 0e 00 35 f9 52 00 49 a3 4c 2b 9f 81 80 00 01 ···5·R·I ·L+·····
0030 00 02 00 00 00 00 03 68 6c 78 07 6d 65 69 74 75 ·······h lx·meitu
0040 61 6e 03 63 6f 6d 00 00 01 00 01 c0 0c 00 01 00 an·com·· ········
0050 01 00 00 00 78 00 04 65 ec 09 69 c0 0c 00 01 00 ····x··e ··i·····
0060 01 00 00 00 78 00 04 65 ec 41 22 ····x··e ·A"
Identification of transaction (dns.id), 2 bytes
Packets: 45595 · Displayed: 2 (0.0%)
Profile: Default

https://arxiv.org/pdf/1205.4011

# Fragmentation Considered Poisonous

*Amir Herzberg*[†] *and Haya Shulman*[‡]
*Dept. of Computer Science, Bar Ilan University*
[†]*amir.herzberg@gmail.com*, [‡]*haya.shulman@gmail.com*


meituan.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
udp.stream eq 2
Destination Info Protocol Length
10.42.0.1 Standard query 0x2b9f A hlx.meituan.com DNS 75
10.42.0.14 Standard query response 0x2b9f A hlx.meituan.com A 101.236.9.105 A… DNS 107
Frame 73: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface wlx6c5ab00ee69e, i
Ethernet II, Src: TP-Link_0e:e6:9e (6c:5a:b0:0e:e6:9e), Dst: d2:4c:cf:57:fe:a7 (d2:4c:cf:57:fe:a7)
Internet Protocol Version 4, Src: 10.42.0.1, Dst: 10.42.0.14
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 93
Identification: 0x1d44 (7492)
Flags: 0x40, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
0010 00 5d 1d 44 40 00 40 11 08 ea 0a 2a 00 01 0a 2a ·]·D@·@· ···*···*
0020 00 0e 00 35 f9 52 00 49 a3 4c 2b 9f 81 80 00 01 ···5·R·I ·L+·····
0030 00 02 00 00 00 00 03 68 6c 78 07 6d 65 69 74 75 ·······h lx·meitu
0040 61 6e 03 63 6f 6d 00 00 01 00 01 c0 0c 00 01 00 an·com·· ········
0050 01 00 00 00 78 00 04 65 ec 09 69 c0 0c 00 01 00 ····x··e ··i·····
Identification (ip.id), 2 bytes
Packets: 45595 · Displayed: 2 (0.0%)
Profile: Default

# IPIDs

- Used to identify fragments and put them back together
  - Should never be repeated for a given destination
- Different strategies
  - Globally incrementing counter that wraps around at $2^{16}$
  - Pick at random without replacement
  - Per-destination
  - Bucket-based
    - Can add noise

# How much entropy?

- Globally incrementing counter?
- Pick at random?


15.99997798605273604449798348692167764035 70...

# How much entropy?

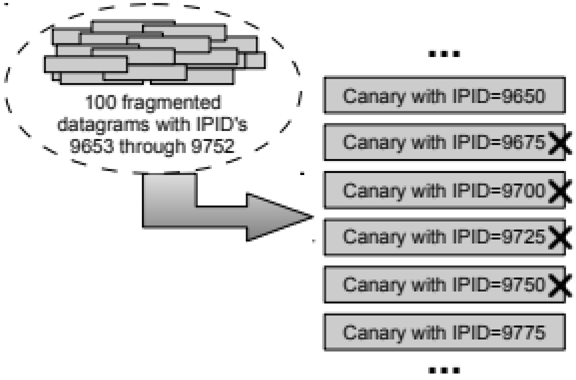
- Per-destination?
  - Think about a noisy server that is talking to other clients
- Bucket-based?

https://www.usenix.org/system/files/conference/foci14/foci14-knockel.pdf

**Counting Packets Sent Between Arbitrary Internet Hosts**


Jeffrey Knockel
*Dept. of Computer Science*
*University of New Mexico*
*jeffk@cs.unm.edu*

Jedidiah R. Crandall
*Dept. of Computer Science*
*University of New Mexico*
*crandall@cs.unm.edu*

# https://jedcrandall.github.io/INFOCOM2018.pdf

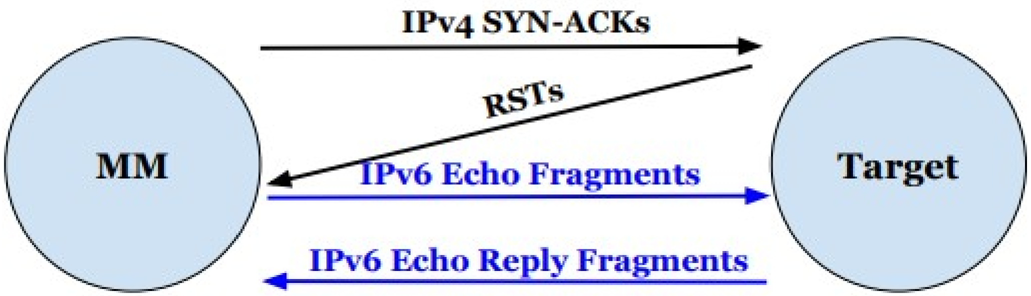


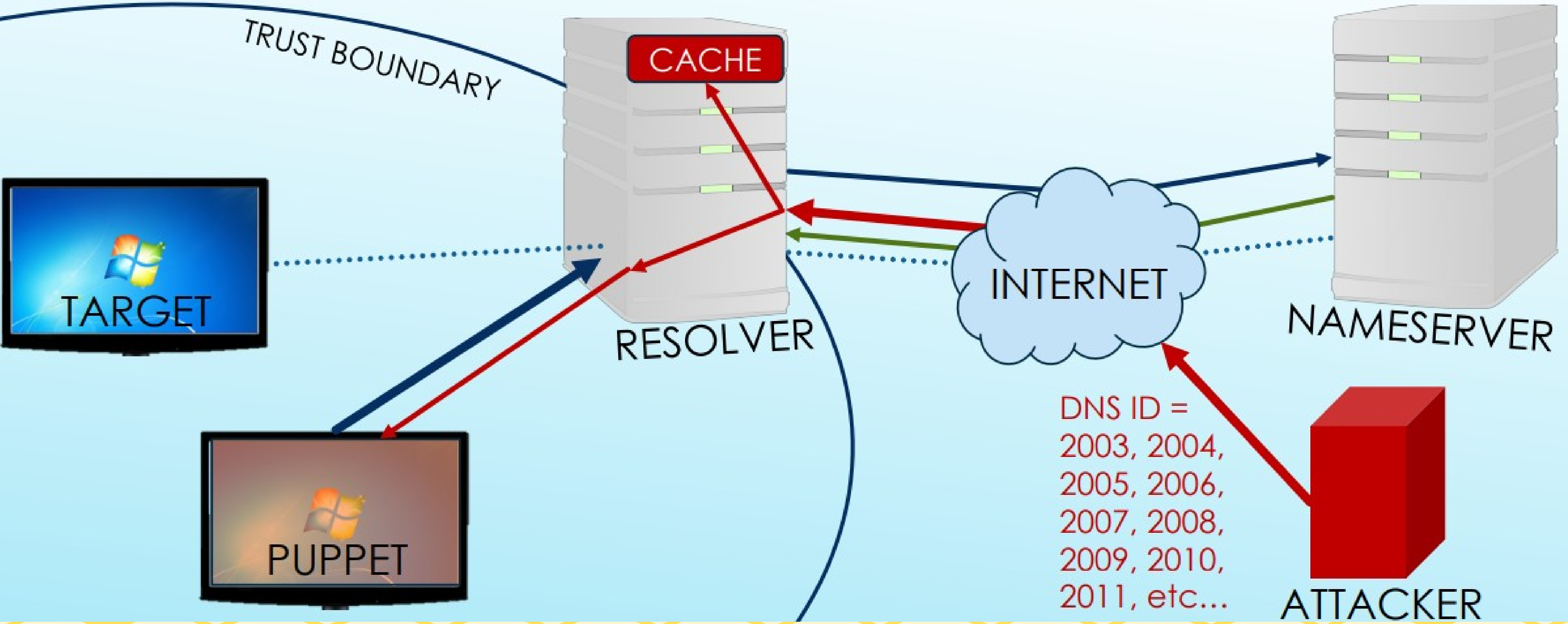

Fig. 3. IPv4 and IPv6 alias resolution.

# Fragmentation attacks on Linux resolvers

- https://media.defcon.org/DEF%20CON%2027/DEF%20CON%2027%20presentations/DEFCON-27-Travis-Palmer-First-try-DNS-Cache-Poisoning-with-IPv4-and-IPv6-Fragmentation.pdf

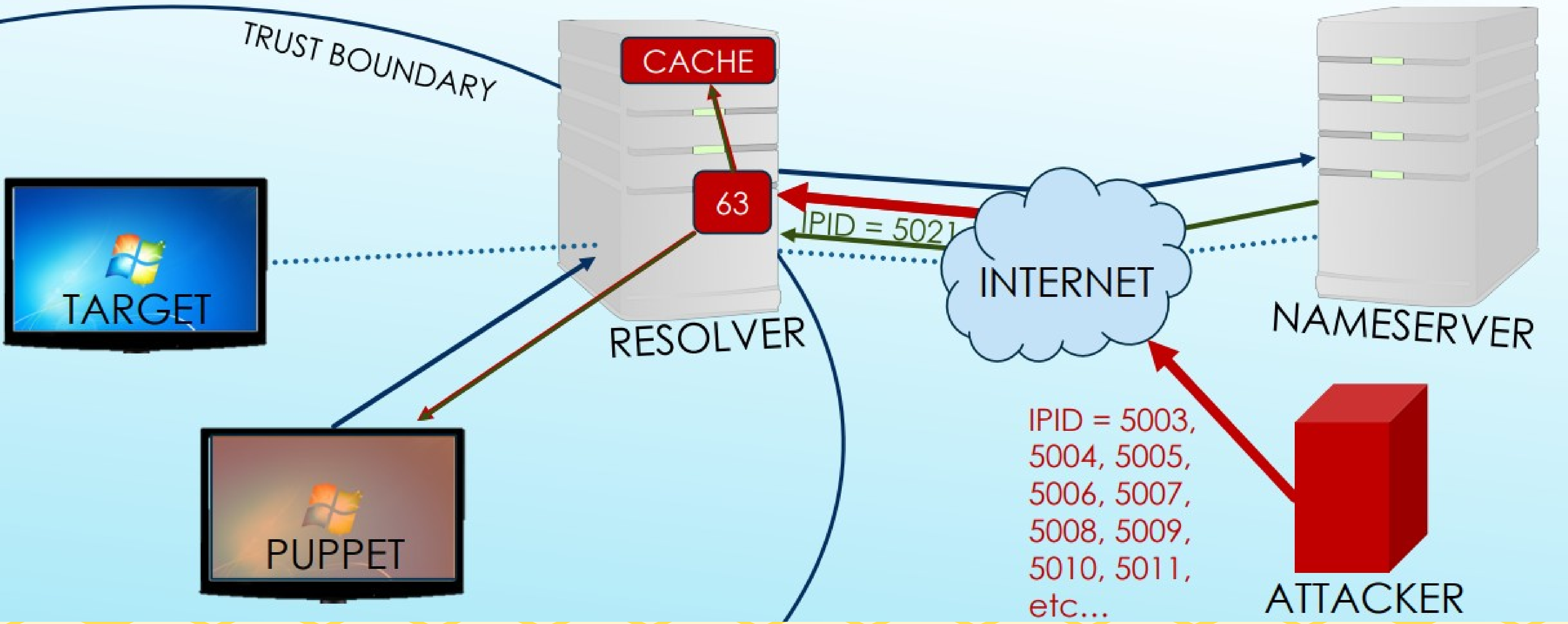
Kaminsky’s attack, assuming source port is completely predictable and you only need to guess the TXID...

# IDEAL POISONING SCENARIO



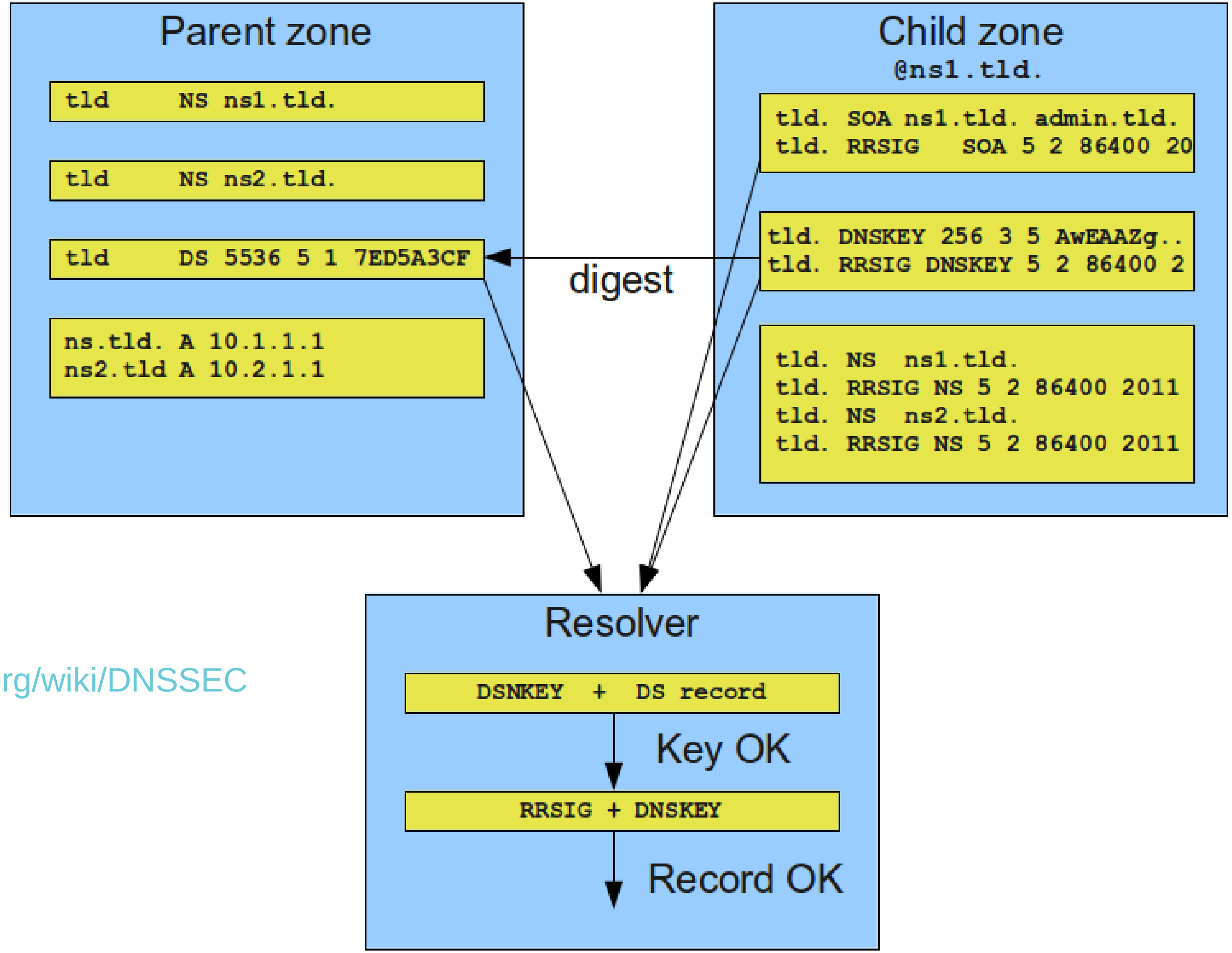
Fragmentation attacks, only need to guess IPID (TXID and source port are in existing fragment from the DNS server)...

# IDEAL POISONING SCENARIO



A real solution would be a real form of authentication, like signatures...

# DNSSEC




https://ru.wikipedia.org/wiki/DNSSEC

# Homework 2 (will be assigned soon)

- Information theory
  - Word problems this time (side channels and NIDS)
- Birthday attacks
  - Word problems
- Extended Euclidean algorithm
  - Then, later in the semester, you’ll have two of the most important ingredients for RSA and signatures (the other being modular exponentiation *via* repeated squaring)