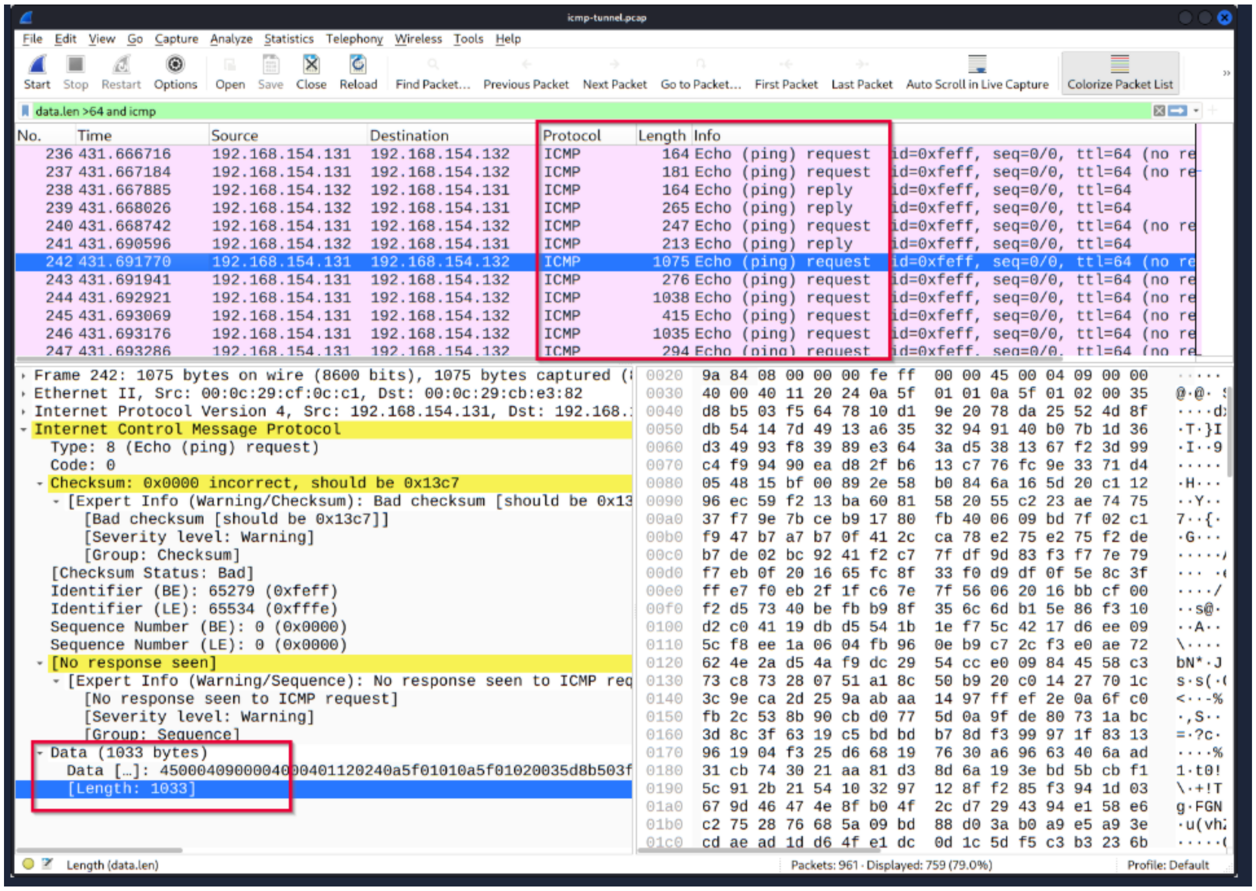
Tunneling Traffic: ICMP and DNS Write Up

- Traffic Tunneling (Port Forwarding) transfers data and resources to a network.
- Provide anonymity and traffic security.
- Adversary can use tunneling to bypass security parameters using trusted protocols that are used in everyday traffic (ICMP and DNS)

ICMP (Internet Control Message Protocol) Analysis

- Reports network communication issues.
- Can be used for DoS (Denial of Services)
- Can be used for data exfiltration and C2 tunneling activities.
- ICMP tunneling attacks can start after a malware execution or vulnerability exploitation.
- Packets can be used to establish a C2 connection (TCP, HTTP, or SSH).

Notes	Wireshark filters
Global search	• `icmp`
"ICMP" options for grabbing the low-hanging fruits: • Packet length. • ICMP destination addresses. • Encapsulated protocol signs in ICMP payload.	• `data.len > 64 and`


icmp-tunnel.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Start Stop Restart Options Open Save Close Reload Find Packet… Previous Packet Next Packet Go to Packet… First Packet Last Packet Auto Scroll in Live Capture Colorize Packet List

```
data.len >64 and icmp
```

```
No.  Time        Source           Destination      Protocol  Length Info
 236 431.666716  192.168.154.131  192.168.154.132  ICMP       164 Echo (ping) request  id=0xfeff, seq=0/0, ttl=64 (no re
 237 431.667184  192.168.154.131  192.168.154.132  ICMP       181 Echo (ping) request  id=0xfeff, seq=0/0, ttl=64 (no re
 238 431.667885  192.168.154.132  192.168.154.131  ICMP       164 Echo (ping) reply    id=0xfeff, seq=0/0, ttl=64
 239 431.668026  192.168.154.132  192.168.154.131  ICMP       265 Echo (ping) reply    id=0xfeff, seq=0/0, ttl=64
 240 431.668742  192.168.154.131  192.168.154.132  ICMP       247 Echo (ping) request  id=0xfeff, seq=0/0, ttl=64 (no re
 241 431.690596  192.168.154.132  192.168.154.131  ICMP       213 Echo (ping) reply    id=0xfeff, seq=0/0, ttl=64
 242 431.691770  192.168.154.131  192.168.154.132  ICMP      1075 Echo (ping) request  id=0xfeff, seq=0/0, ttl=64 (no re
 243 431.691941  192.168.154.131  192.168.154.132  ICMP       276 Echo (ping) request  id=0xfeff, seq=0/0, ttl=64 (no re
 244 431.692921  192.168.154.131  192.168.154.132  ICMP      1038 Echo (ping) request  id=0xfeff, seq=0/0, ttl=64 (no re
 245 431.693069  192.168.154.131  192.168.154.132  ICMP       415 Echo (ping) request  id=0xfeff, seq=0/0, ttl=64 (no re
 246 431.693176  192.168.154.131  192.168.154.132  ICMP      1035 Echo (ping) request  id=0xfeff, seq=0/0, ttl=64 (no re
 247 431.693286  192.168.154.131  192.168.154.132  ICMP       294 Echo (ping) request  id=0xfeff, seq=0/0, ttl=64 (no re
```

```
› Frame 242: 1075 bytes on wire (8600 bits), 1075 bytes captured (
› Ethernet II, Src: 00:0c:29:cf:0c:c1, Dst: 00:0c:29:cb:e3:82
› Internet Protocol Version 4, Src: 192.168.154.131, Dst: 192.168.
⌄ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
  ⌄ Checksum: 0x0000 incorrect, should be 0x13c7
    ⌄ [Expert Info (Warning/Checksum): Bad checksum [should be 0x13
        [Bad checksum [should be 0x13c7]]
        [Severity level: Warning]
        [Group: Checksum]
    [Checksum Status: Bad]
    Identifier (BE): 65279 (0xfeff)
    Identifier (LE): 65534 (0xfffe)
    Sequence Number (BE): 0 (0x0000)
    Sequence Number (LE): 0 (0x0000)
  ⌄ [No response seen]
    ⌄ [Expert Info (Warning/Sequence): No response seen to ICMP req
        [No response seen to ICMP request]
        [Severity level: Warning]
        [Group: Sequence]
  ⌄ Data (1033 bytes)
      Data […]: 4500040900004000401120240a5f01010a5f01020035d8b503f
      [Length: 1033]
```

```
0020  9a 84 08 00 00 00 fe ff  00 00 45 00 04 09 00 00   ·····
0030  40 00 40 11 20 24 0a 5f  01 01 0a 5f 01 02 00 35   @·@·
0040  d8 b5 03 f5 64 78 10 d1  9e 20 78 da 25 52 4d 8f   ····dx
0050  db 54 14 7d 49 13 a6 35  32 94 91 40 b0 7b 1d 36   ·T·}I
0060  d3 49 93 f8 39 89 e3 64  3a d5 38 13 67 f2 3d 99   ·I··9
0070  c4 f9 94 90 ea d8 2f b6  13 c7 76 fc 9e 33 71 d4   ·····
0080  05 48 15 bf 00 89 2e 58  b0 84 6a 16 5d 20 c1 12   ·H···
0090  96 ec 59 f2 13 ba 60 81  58 20 55 c2 23 ae 74 75   ··Y··
00a0  37 f7 9e 7b ce b9 17 80  fb 40 06 09 bd 7f 02 c1   7··{·
00b0  f9 47 b7 a7 b7 0f 41 2c  ca 78 e2 75 e2 75 f2 de   ·G···
00c0  b7 de 02 bc 92 41 f2 c7  7f df 9d 83 f3 f7 7e 79   ·····
00d0  f7 eb 0f 20 16 65 fc 8f  33 f0 d9 df 0f 5e 8c 3f   ··· ·
00e0  ff e7 f0 eb 2f 1f c6 7e  7f 56 06 20 16 bb cf 00   ····/
00f0  f2 d5 73 40 be fb b9 8f  35 6c 6d b1 5e 86 f3 10   ··s@·
0100  d2 c0 41 19 db d5 54 1b  1e f7 5c 42 17 d6 ee 09   ··A··
0110  5c f8 ee 1a 06 04 fb 96  0e b9 c7 2c f3 e0 ae 72   \····
0120  62 4e 2a d5 4a f9 dc 29  54 cc e0 09 84 45 58 c3   bN*·J
0130  73 c8 73 28 07 51 a1 8c  50 b9 20 c0 14 27 70 1c   s·s(·
0140  3c 9e ca 2d 25 9a ab aa  14 97 ff ef 2e 0a 6f c0   <··-%
0150  fb 2c 53 8b 90 cb d0 77  5d 0a 9f de 80 73 1a bc   ·,S··
0160  3d 8c 3f 63 19 c5 bd bd  b7 8d f3 99 97 1f 83 13   =·?c·
0170  96 19 04 f3 25 d6 68 19  76 30 a6 96 63 40 6a ad   ····%
0180  31 cb 74 30 21 aa 81 d3  8d 6a 19 3e bd 5b cb f1   1·t0!
0190  5c 91 2b 21 54 10 32 97  12 8f f2 85 f3 94 1d 03   \·+!T
01a0  67 9d 46 47 4e 8f b0 4f  2c d7 29 43 94 e1 58 e6   g·FGN
01b0  c2 75 28 76 68 5a 09 bd  88 d0 3a b0 a9 e5 a9 3e   ·u(vh
01c0  cd ae ad 1d d6 4f e1 dc  0d 1c 5d f5 c3 b3 23 6b   ·····
```

```
Length (data.len)          Packets: 961 · Displayed: 759 (79.0%)          Profile: Default
```

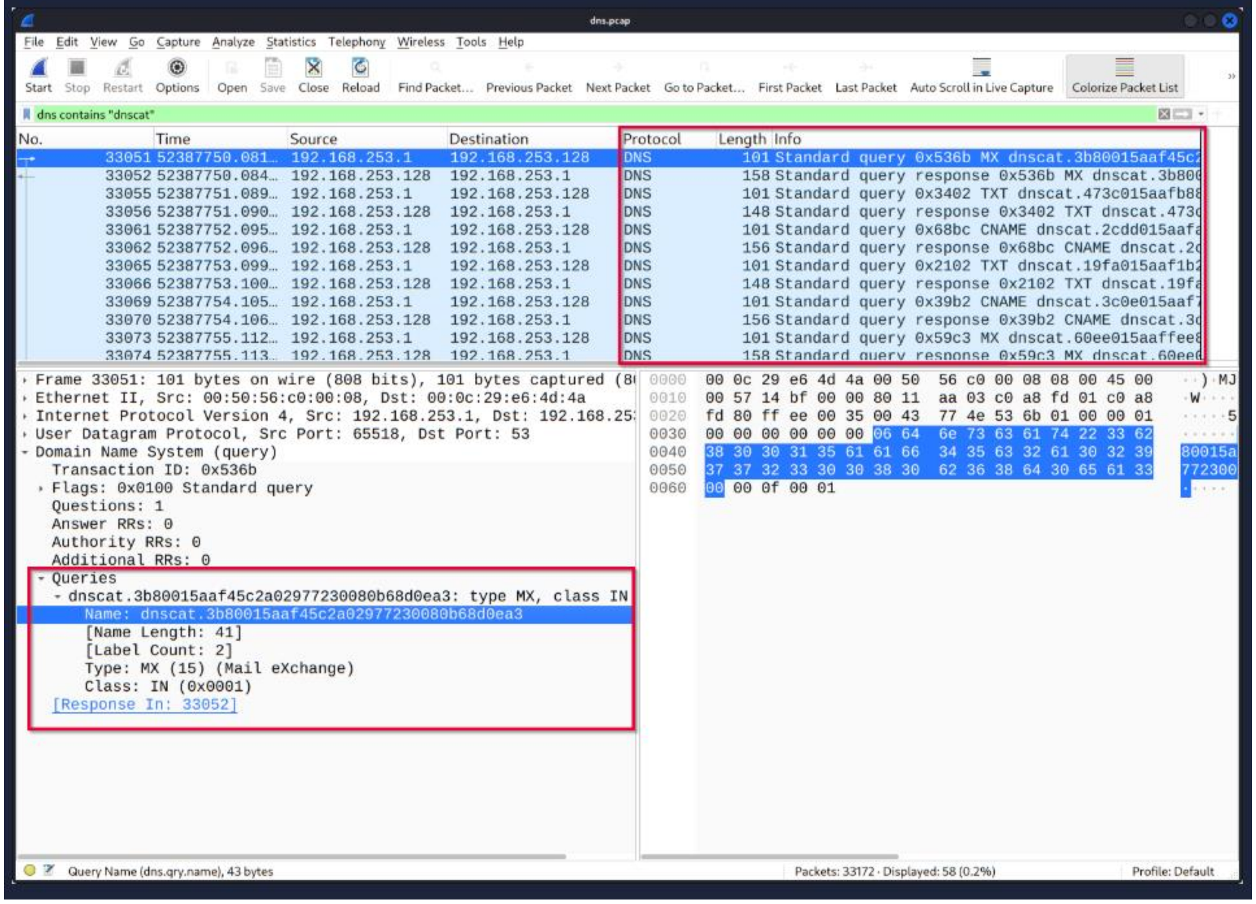
DNS (Domain Name System) Analysis

- Design to translate/convert IP domain addresses to IP addresses.
- Used for exfiltration and C2 activities.
- Attacks start after a malware execution or vulnerability exploitation

Notes	Wireshark Filter
Global search	• dns
"DNS" options for grabbing the low-hanging fruits: • Query length. • Anomalous and non-regular names in DNS addresses. • Long DNS addresses with encoded subdomain addresses.	• `dns contains "dnscat"` • `dns.qry.name.len > 15 and !mdns`

- Known patterns like dnscat and dns2tcp.
- Statistical analysis like the anomalous volume of DNS requests for a particular target.

!mdns: Disable local link device queries.


dns.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Start Stop Restart Options Open Save Close Reload Find Packet... Previous Packet Next Packet Go to Packet... First Packet Last Packet Auto Scroll in Live Capture Colorize Packet List

dns contains "dnscat"

```
No.    Time              Source           Destination      Protocol Length Info
33051 52387750.081…  192.168.253.1    192.168.253.128  DNS      101 Standard query 0x536b MX dnscat.3b80015aaf45c2
33052 52387750.084…  192.168.253.128  192.168.253.1    DNS      158 Standard query response 0x536b MX dnscat.3b800
33055 52387751.089…  192.168.253.1    192.168.253.128  DNS      101 Standard query 0x3402 TXT dnscat.473c015aafb8
33056 52387751.090…  192.168.253.128  192.168.253.1    DNS      148 Standard query response 0x3402 TXT dnscat.473
33061 52387752.095…  192.168.253.1    192.168.253.128  DNS      101 Standard query 0x68bc CNAME dnscat.2cdd015aafa
33062 52387752.096…  192.168.253.128  192.168.253.1    DNS      156 Standard query response 0x68bc CNAME dnscat.2c
33065 52387753.099…  192.168.253.1    192.168.253.128  DNS      101 Standard query 0x2102 TXT dnscat.19fa015aaf1b2
33066 52387753.100…  192.168.253.128  192.168.253.1    DNS      148 Standard query response 0x2102 TXT dnscat.19fa
33069 52387754.105…  192.168.253.1    192.168.253.128  DNS      101 Standard query 0x39b2 CNAME dnscat.3c0e015aaf7
33070 52387754.106…  192.168.253.128  192.168.253.1    DNS      156 Standard query response 0x39b2 CNAME dnscat.3c
33073 52387755.112…  192.168.253.1    192.168.253.128  DNS      101 Standard query 0x59c3 MX dnscat.60ee015aaffee8
33074 52387755.113…  192.168.253.128  192.168.253.1    DNS      158 Standard query response 0x59c3 MX dnscat.60ee0
```

```
› Frame 33051: 101 bytes on wire (808 bits), 101 bytes captured (8
› Ethernet II, Src: 00:50:56:c0:00:08, Dst: 00:0c:29:e6:4d:4a
› Internet Protocol Version 4, Src: 192.168.253.1, Dst: 192.168.25
› User Datagram Protocol, Src Port: 65518, Dst Port: 53
˅ Domain Name System (query)
    Transaction ID: 0x536b
  › Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ˅ Queries
    ˅ dnscat.3b80015aaf45c2a02977230080b68d0ea3: type MX, class IN
        Name: dnscat.3b80015aaf45c2a02977230080b68d0ea3
        [Name Length: 41]
        [Label Count: 2]
        Type: MX (15) (Mail eXchange)
        Class: IN (0x0001)
    [Response In: 33052]
```

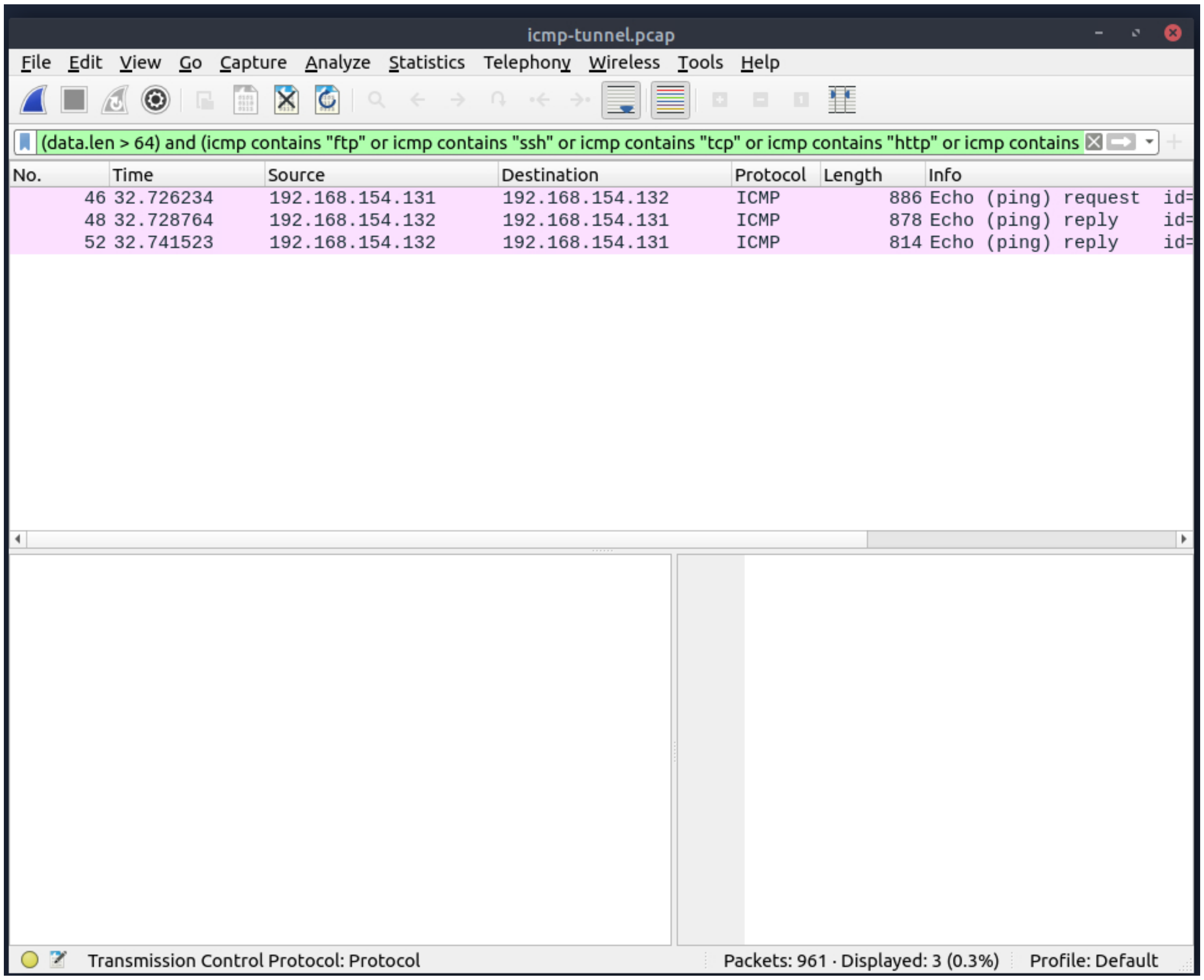
```
0000  00 0c 29 e6 4d 4a 00 50  56 c0 00 08 08 00 45 00   ··)·MJ
0010  00 57 14 bf 00 00 80 11  aa 03 c0 a8 fd 01 c0 a8   ·W····
0020  fd 80 ff ee 00 35 00 43  77 4e 53 6b 01 00 00 01   ·····5
0030  00 00 00 00 00 00 06 64  6e 73 63 61 74 22 33 62   ······
0040  38 30 30 31 35 61 61 66  34 35 63 32 61 30 32 39   80015a
0050  37 37 32 33 30 30 38 30  62 36 38 64 30 65 61 33   772300
0060  00 00 0f 00 01                                     ·····
```

```
Query Name (dns.qry.name), 43 bytes      Packets: 33172 · Displayed: 58 (0.2%)      Profile: Default
```

1. **Which protocol is used in ICMP tunneling?**

   - Since we are looking for a protocol that is used in the ICMP tunneling, there are several protocols to look for – ftp, http, tcp, and ssh.
   - We can input a filter that filters traffic to identify which protocol is being used.
   - (data.len > 64) and (icmp contains “ftp” or icmp contains “ssh” or icmp contains “http” or icmp contains “tcp”) <--- this is filtering traffic with packets that have more than 64 bytes of data. Since ICMP echo request uses small payloads,

payloads larger than 64 bytes is a sign of unauthorized data being carried. And we are looking for protocols that can be used for tunneling exfiltration.

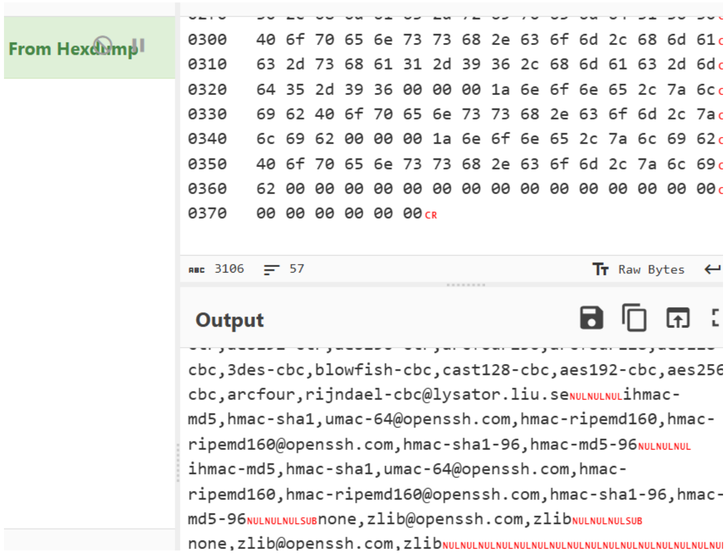

icmp-tunnel.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
(data.len > 64) and (icmp contains "ftp" or icmp contains "ssh" or icmp contains "tcp" or icmp contains "http" or icmp contains
```

```
No.   Time       Source           Destination      Protocol Length Info
   46 32.726234  192.168.154.131  192.168.154.132  ICMP        886 Echo (ping) request  id=
   48 32.728764  192.168.154.132  192.168.154.131  ICMP        878 Echo (ping) reply    id=
   52 32.741523  192.168.154.132  192.168.154.131  ICMP        814 Echo (ping) reply    id=
```

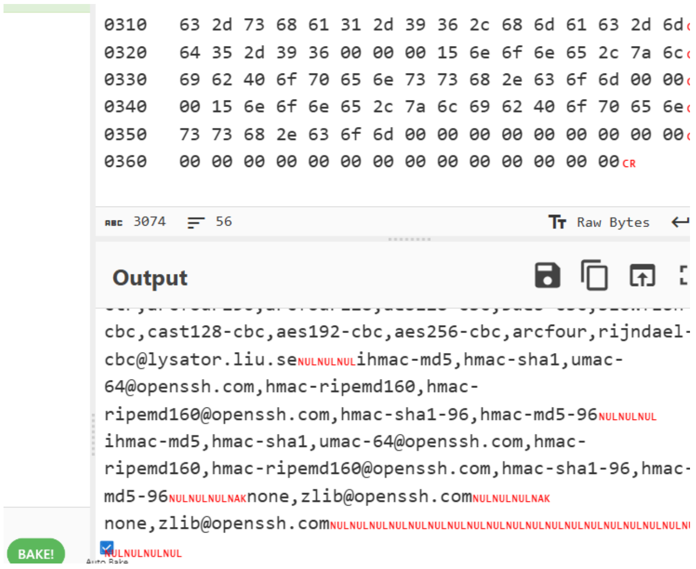
```
Transmission Control Protocol: Protocol        Packets: 961 · Displayed: 3 (0.3%)    Profile: Default
```

- We have 3 packets displayed. Now we must inspect each of those packets to see which protocol is being used.
- From the first traffic, using cyber chef to decode the hex dump, we see multiple ssh protocols being executed.


From Hexdump

```
0300    40 6f 70 65 6e 73 73 68 2e 63 6f 6d 2c 68 6d 61
0310    63 2d 73 68 61 31 2d 39 36 2c 68 6d 61 63 2d 6d
0320    64 35 2d 39 36 00 00 00 1a 6e 6f 6e 65 2c 7a 6c
0330    69 62 40 6f 70 65 6e 73 73 68 2e 63 6f 6d 2c 7a
0340    6c 69 62 00 00 00 1a 6e 6f 6e 65 2c 7a 6c 69 62
0350    40 6f 70 65 6e 73 73 68 2e 63 6f 6d 2c 7a 6c 69
0360    62 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0370    00 00 00 00 00 00CR

3106   57                                   Raw Bytes

Output
cbc,3des-cbc,blowfish-cbc,cast128-cbc,aes192-cbc,aes256
cbc,arcfour,rijndael-cbc@lysator.liu.seNULNULNULihmac-
md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160,hmac-
ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96NULNULNUL
ihmac-md5,hmac-sha1,umac-64@openssh.com,hmac-
ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-
md5-96NULNULNULSUBnone,zlib@openssh.com,zlibNULNULNULSUB
none,zlib@openssh.com,zlibNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
```

- Let's look at the second hex dump to be sure that protocol ssh is still being used.



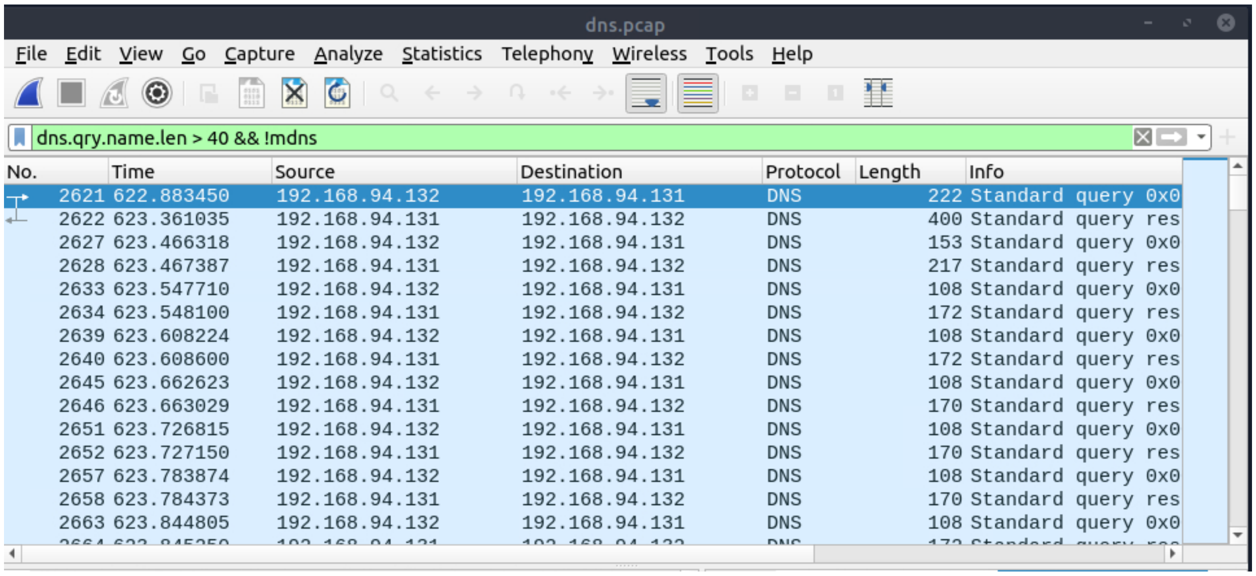
```
0310    63 2d 73 68 61 31 2d 39 36 2c 68 6d 61 63 2d 6d
0320    64 35 2d 39 36 00 00 00 15 6e 6f 6e 65 2c 7a 6c
0330    69 62 40 6f 70 65 6e 73 73 68 2e 63 6f 6d 00 00
0340    00 15 6e 6f 6e 65 2c 7a 6c 69 62 40 6f 70 65 6e
0350    73 73 68 2e 63 6f 6d 00 00 00 00 00 00 00 00 00
0360    00 00 00 00 00 00 00 00 00 00 00 00 00 00CR

3074   56                                   Raw Bytes

Output
cbc,cast128-cbc,aes192-cbc,aes256-cbc,arcfour,rijndael-
cbc@lysator.liu.seNULNULNULihmac-md5,hmac-sha1,umac-
64@openssh.com,hmac-ripemd160,hmac-
ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96NULNULNUL
ihmac-md5,hmac-sha1,umac-64@openssh.com,hmac-
ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-
md5-96NULNULNULNAKnone,zlib@openssh.comNULNULNULNAK
none,zlib@openssh.comNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNUL
```

Answer: ssh

2. **What is the suspicious main domain address that receives anomalous DNS queries? (defang the address)**

   - We can use the dns.qry.name.len > 15 mdns, however since we are looking for a suspicious domain, we should increase the size to reduce the amount of network traffic that will be displayed from the names of packets with less than 40 characters.

- dns.qry.name.len > 40 && !mdns <--- filters packets with domain names greater than 40 characters and we want to exclude local devices "!mdns" (Multicast DNS)

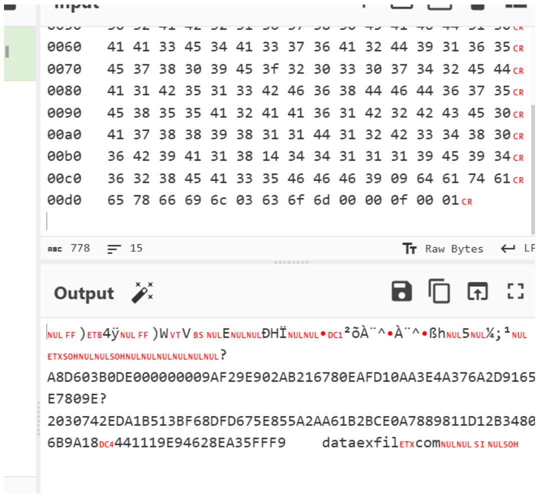

dns.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

dns.qry.name.len > 40 && !mdns

```
No.   Time        Source          Destination     Protocol Length Info
 2621 622.883450 192.168.94.132  192.168.94.131  DNS         222 Standard query 0x0
 2622 623.361035 192.168.94.131  192.168.94.132  DNS         400 Standard query res
 2627 623.466318 192.168.94.132  192.168.94.131  DNS         153 Standard query 0x0
 2628 623.467387 192.168.94.131  192.168.94.132  DNS         217 Standard query res
 2633 623.547710 192.168.94.132  192.168.94.131  DNS         108 Standard query 0x0
 2634 623.548100 192.168.94.131  192.168.94.132  DNS         172 Standard query res
 2639 623.608224 192.168.94.132  192.168.94.131  DNS         108 Standard query 0x0
 2640 623.608600 192.168.94.131  192.168.94.132  DNS         172 Standard query res
 2645 623.662623 192.168.94.132  192.168.94.131  DNS         108 Standard query 0x0
 2646 623.663029 192.168.94.131  192.168.94.132  DNS         170 Standard query res
 2651 623.726815 192.168.94.132  192.168.94.131  DNS         108 Standard query 0x0
 2652 623.727150 192.168.94.131  192.168.94.132  DNS         170 Standard query res
 2657 623.783874 192.168.94.132  192.168.94.131  DNS         108 Standard query 0x0
 2658 623.784373 192.168.94.131  192.168.94.132  DNS         170 Standard query res
 2663 623.844805 192.168.94.132  192.168.94.131  DNS         108 Standard query 0x0
```

- We can see here that there is alot of traffic coming from IP "192[.]168[.]94[.]xxx"

- Now let's decode the hex dump using cyber chef to determine what is the domain name of the adversary



```
0060    41 41 33 45 34 41 33 37 36 41 32 44 39 31 36 35
0070    45 37 38 30 39 45 3f 32 30 33 30 37 34 32 45 44
0080    41 31 42 35 31 33 42 46 36 38 44 46 44 36 37 35
0090    45 38 35 35 41 32 41 41 36 31 42 32 42 43 45 30
00a0    41 37 38 38 39 38 31 31 44 31 32 42 33 34 38 30
00b0    36 42 39 41 31 38 14 34 34 31 31 31 39 45 39 34
00c0    36 32 38 45 41 33 35 46 46 46 39 09 64 61 74 61
00d0    65 78 66 69 6c 03 63 6f 6d 00 00 0f 00 01

778   15
```

```
)4ÿ)WVEÐHÏ²õÀ¨^•À¨^•ßh5¼;¹
?
A8D603B0DE000000009AF29E902AB216780EAFD10AA3E4A376A2D9165
E7809E?
2030742EDA1B513BF68DFD675E855A2AA61B2BCE0A7889811D12B3480
6B9A18441119E94628EA35FFF9    dataexfilcom
```

Answer: dataexfil[.]com