Flow analysis

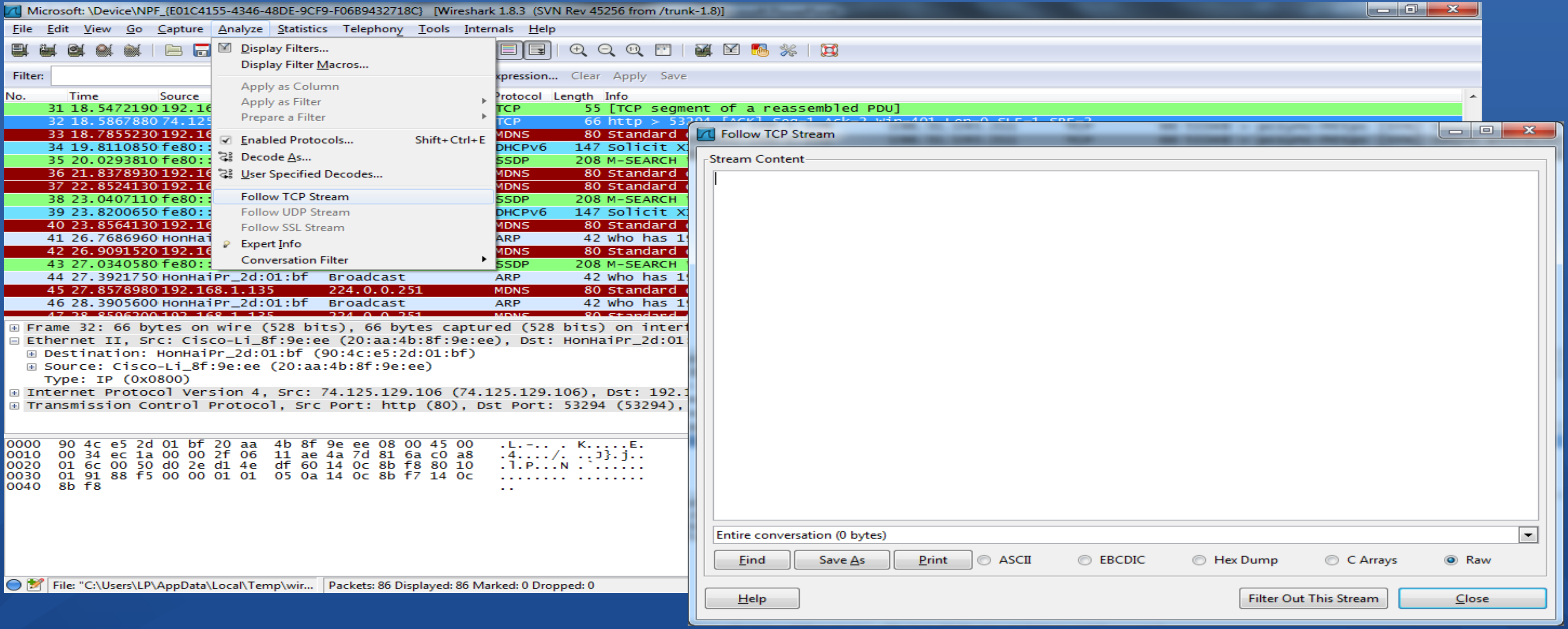
Network Security and Forensics

Flow analysis

- Defined
  - “Examination of sequences of related packets (“flows”). Flow analysis is typically conducted in order to identify traffic patterns, isolate suspicious activity, analyze higher-layer protocols, or extract data.” (Davidoff & Ham, 2012)
- Flow defined
  - “In RFC 3679, a “flow” is defined as “a sequence of packets sent from a particular source to a particular unicast, anycast, or multicast destination that the source desires to label as a flow. A flow could consist of all packets in a specific transport connection or a media stream. However, a flow is not necessarily 1:1 mapped to a transport connection.”” (Davidoff & Ham, 2012)
- Flow and stream are becoming interchangeable

Flow analysis tools

- Wireshark: Follow TCP Stream



```
No.    Time       Source                Protocol Length Info
   31 18.5472190 192.16                 TCP        55 [TCP segment of a reassembled PDU]
   32 18.5867880 74.125                 TCP        66 http > 53294
   33 18.7855230 192.16                 MDNS       80 Standard
   34 19.8110850 fe80::                 DHCPv6    147 Solicit X
   35 20.0293810 fe80::                 SSDP      208 M-SEARCH
   36 21.8378930 192.16                 MDNS       80 Standard
   37 22.8524130 192.16                 MDNS       80 Standard
   38 23.0407110 fe80::                 SSDP      208 M-SEARCH
   39 23.8200650 fe80::                 DHCPv6    147 Solicit X
   40 23.8564130 192.16                 MDNS       80 Standard
   41 26.7686960 HonHai                 ARP        42 Who has 1
   42 26.9091520 192.16                 MDNS       80 Standard
   43 27.0340580 fe80::                 SSDP      208 M-SEARCH
   44 27.3921750 HonHaiPr_2d:01:bf    Broadcast     ARP        42 Who has 1
   45 27.8578980 192.168.1.135        224.0.0.251   MDNS       80 Standard
   46 28.3905600 HonHaiPr_2d:01:bf    Broadcast     ARP        42 Who has 1
```

```
⊞ Frame 32: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on inter
⊟ Ethernet II, Src: Cisco-Li_8f:9e:ee (20:aa:4b:8f:9e:ee), Dst: HonHaiPr_2d:01
  ⊞ Destination: HonHaiPr_2d:01:bf (90:4c:e5:2d:01:bf)
  ⊞ Source: Cisco-Li_8f:9e:ee (20:aa:4b:8f:9e:ee)
    Type: IP (0x0800)
⊞ Internet Protocol Version 4, Src: 74.125.129.106 (74.125.129.106), Dst: 192.1
⊞ Transmission Control Protocol, Src Port: http (80), Dst Port: 53294 (53294),
```

```
0000  90 4c e5 2d 01 bf 20 aa  4b 8f 9e ee 08 00 45 00   .L.-.. . K.....E.
0010  00 34 ec 1a 00 00 2f 06  11 ae 4a 7d 81 6a c0 a8   .4..../. ..J}.j..
0020  01 6c 00 50 d0 2e d1 4e  df 60 14 0c 8b f8 80 10   .l.P...N .`......
0030  01 91 88 f5 00 00 01 01  05 0a 14 0c 8b f7 14 0c   ........ ........
0040  8b f8                                              ..
```

File: "C:\Users\LP\AppData\Local\Temp\wir... Packets: 86 Displayed: 86 Marked: 0 Dropped: 0

Follow TCP Stream

Stream Content

Entire conversation (0 bytes)

○ ASCII ○ EBCDIC ○ Hex Dump ○ C Arrays ● Raw

Other tools

- Tshark
- Tcpflow
  - Parses non-fragmented IP packets and reassembles TCP stream into a file
- Pcapcat
  - Lists all of the streams that it sees
  - It can dump individual streams
  - Use magic numbers
    - Magic number is a constant used to identify a file format [1]
- Tcpxtract
  - Using file signatures it extracts and reconstructs payload data
    - Example
      - $ tcpxtract -f capturefile.pcap -o output_dir/

Flow analysis techniques

- Lists Conversations and Flows
- Export a Flow
- File and Data Carving

Lists conversations and flows

- View packet conversations using tshark
  - $ tshark -qn -z conv ,tcp -r evidence01.pcap

```
============================================================================================
TCP Conversations
Filter:<No Filter >
                                              |  <-  |  ->  |  Total |
                                                 Frames Bytes Frames Bytes Frames Bytes
192.168.1.159:1271  <->    205.188.13.12:443     31   29717 16        1451 47   31168
192.168.1.159:1221  <->     64.12.25.91:443 24    4206  16        1799 40    6005
192.168.1.158:51128 <->    64.12.24.50:443       20    2622  20        1681 40    4303
192.168.1.158:5190  <->   192.168.1.159:127       9     1042  15      13100 24  14142
192.168.1.159:1273  <->     64.236.68.246:80      5     1545   5        1964  10   3509
192.168.1.2:54419    <->    192.168.1.157:80       3      206    4         272    7     478
192.168.1.2:55488    <->    192.168.1.30:22   2     292    3        246    5     538
============================================================================================
```

List TCP flows

- Identify specific flow of interest
    - Look for IP and port
        - $ pcapcat -r evidence01.pcap

```
[1] TCP 192.168.1.2:54419 -> 192.168.1.157:80
[2] TCP 192.168.1.159:1271 -> 205.188.13.12:443
[3] TCP 192.168.1.159:1272 -> 192.168.1.158:5190
[4] TCP 192.168.1.159:1273 -> 64.236.68.246:80
Enter the index number of the conversation to dump or press enter to quit:
```

Export a Flow

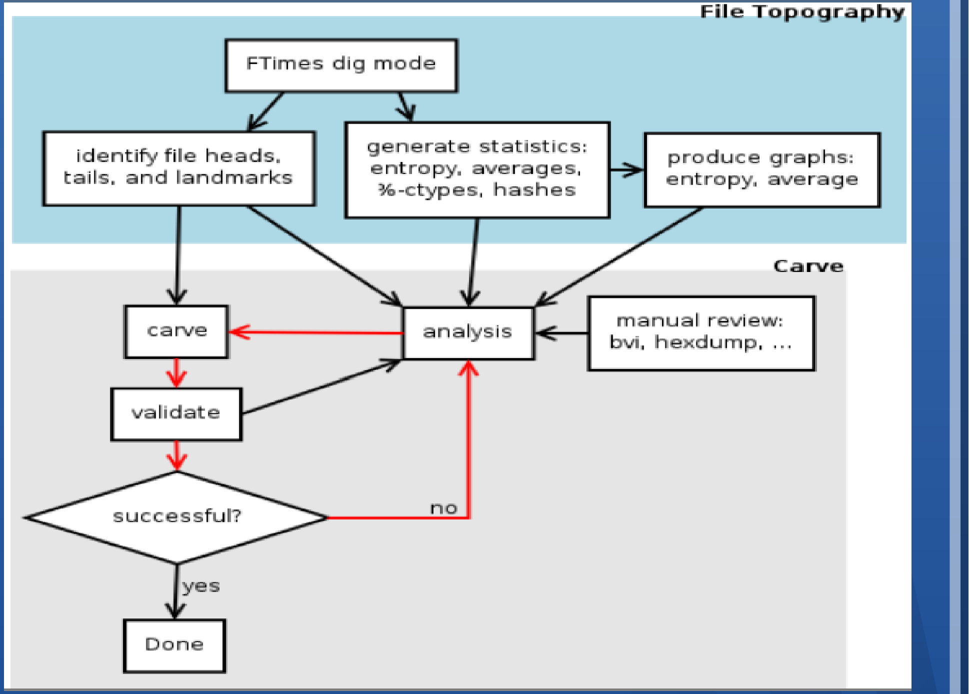
- Identify the file that most likely contains the evidence for export
  - $ pcapcat -r evidence01.pcap -w internal -stream.dump -f 'host 192.168.1.158 and port 5190 '

    ```
    [1] TCP 192.168.1.159:1272 -> 192.168.1.158:5190
    Enter the index number of the conversation to dump or press enter to quit: 1
    Dumping index value 1
    ```
  - $ tcpflow -r evidence01.pcap 'host 192.168.1.158 and port 5190 ‘
    - Example display:

      ```
      tcpflow [25586]: tcpflow version 0.21 by Jeremy Elson <jelson@circlemud.org >
      tcpflow [25586]: looking for handler for datalink type 1 for interface
      evidence01.pcap
      tcpflow [25586]: found max FDs to be 16 using OPEN_MAX
      tcpflow [25586]: 192.168.001.159.01272 -192.168.001.158.05190: new flow
      tcpflow [25586]: 192.168.001.158.05190 -192.168.001.159.01272: new flow
      tcpflow [25586]: 192.168.001.158.05190 -192.168.001.159.01272: opening new
          output file
      tcpflow [25586]: 192.168.001.159.01272 -192.168.001.158.05190: opening new
          output file
      ```
  - Wireshark
    - Click on packet and right-click of “Follow TCP Stream”
    - “Save As” in raw format

Manual File and Data carving

- Carve the file out of the exported flow
    - Open in hex editor
    - Look for the magic numbers (file signatures)
        - Examples:
            - Jpeg beginning 0xffd8 - end 0xffd9
            - .docx beginning 0x504B
    - Figure file size to find end of file –
        - add initial byte offset to expected size
- Gather hashes
    - Example:
        - $ sha256sum filename
        - $ md5sum filename
- Confirm file size
- Open a copy and confirm the file is correct


File Topography
FTimes dig mode
identify file heads, tails, and landmarks
generate statistics: entropy, averages, %-ctypes, hashes
produce graphs: entropy, average
Carve
carve
analysis
manual review: bvi, hexdump, ...
validate
successful?
no
yes
Done

1.http://www.korelogic.com/Resources/Projects/dfrws_challenge_2006/DFRWS_2006_File_Carving_Challenge.pdf

Automatic file carving

- $ tcpxtract -f evidence01.pcap

```
...
Found file of type "zip" in session [192.168.1.158:17940 ->
192.168.1.159:63492] , exporting to 00000023. zip
Found file of type "zip" in session [192.168.1.158:17940 ->
192.168.1.159:63492] , exporting to 00000024. zip
Found file of type "zip" in session [192.168.1.158:17940 ->
192.168.1.159:63492] , exporting to 00000025. zip
```

- $ ls -l

```
...
-rwx ------ 1 student student 12020 2011 -01 -08 11:22 00000023. zip
-rwx ------ 1 student student 11068 2011 -01 -08 11:22 00000024. zip
-rwx ------ 1 student student 10264 2011 -01 -08 11:22 00000025. zip
```

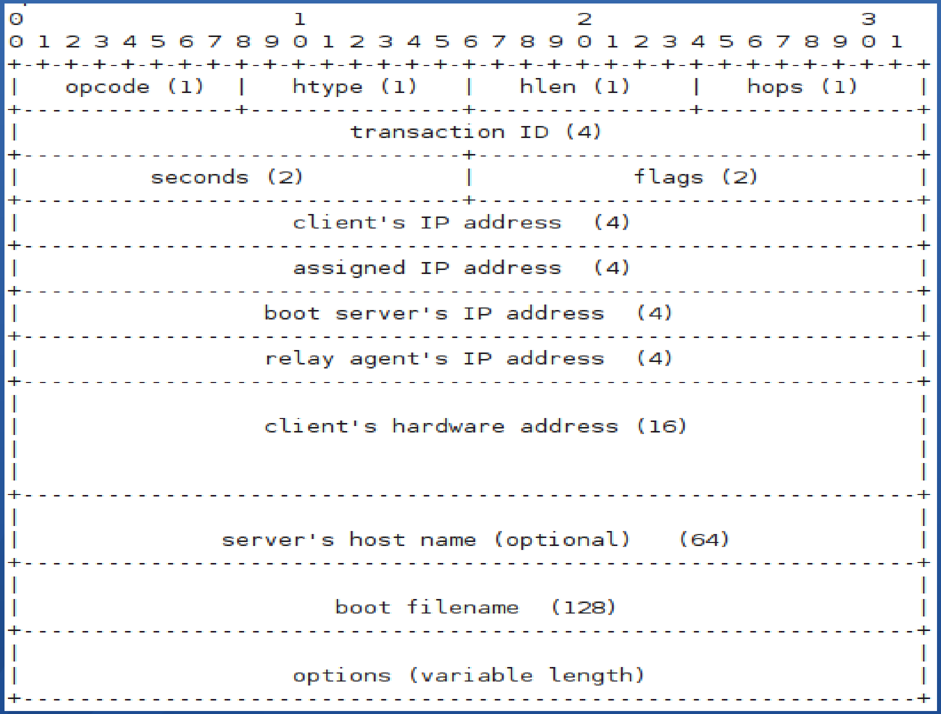
Higher-layer traffic analysis

- Hypertext Transfer Protocol (HTTP)
- Simple Mail Transfer Protocol (SMTP)
- Domain Name System (DNS)
- Dynamic Host Configuration Protocol (DHCP)
- Etc

http

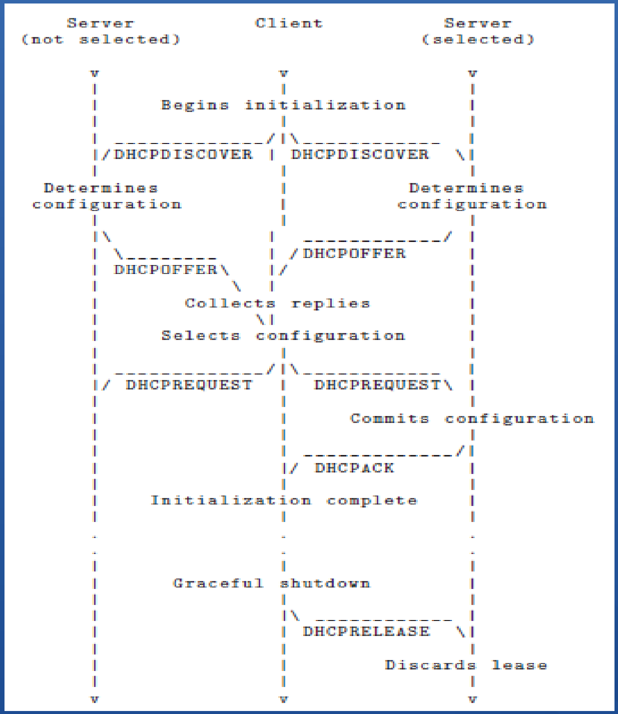
- RFC 2616 defined methods
    - OPTIONS – obtain information about communication
    - GET – retrieve information ID by Uniform Resource Identifier (URI)
    - HEAD – retrieves information without message body
    - POST – send data to URI for processing
    - PUT – upload information to specified URI
    - DELETE – delete resource specified
    - TRACE – echo request message back to client, helpful for debugging
    - CONNECT - reserved

DHCP



```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   opcode (1)  |   htype (1)   |   hlen (1)    |   hops (1)    |
+---------------+---------------+---------------+---------------+
|                      transaction ID (4)                       |
+-------------------------------+-------------------------------+
|          seconds (2)          |           flags (2)           |
+-------------------------------+-------------------------------+
|                     client's IP address  (4)                  |
+---------------------------------------------------------------+
|                     assigned IP address  (4)                  |
+---------------------------------------------------------------+
|                   boot server's IP address  (4)               |
+---------------------------------------------------------------+
|                   relay agent's IP address  (4)               |
+---------------------------------------------------------------+
|                                                               |
|                   client's hardware address (16)              |
|                                                               |
|                                                               |
+---------------------------------------------------------------+
|                                                               |
|                server's host name (optional)   (64)           |
+---------------------------------------------------------------+
|                                                               |
|                     boot filename  (128)                      |
+---------------------------------------------------------------+
|                                                               |
|                   options (variable length)                   |
+---------------------------------------------------------------+
```

1.



```
    Server           Client           Server
 (not selected)                     (selected)

       v                v                v
       |                |                |
       |     Begins initialization       |
       |                |                |
       | _____________/|\_____________   |
       |/DHCPDISCOVER | DHCPDISCOVER  \|
       |                |                |
   Determines           |           Determines
  configuration         |          configuration
       |                |                |
       |\              |  ____________/ |
       | \_________    | /DHCPOFFER      |
       | DHCPOFFER\    |/                |
       |           \   |                 |
       |       Collects replies          |
       |             \|                  |
       |     Selects configuration       |
       |                |                |
       | _____________/|\_____________   |
       |/ DHCPREQUEST   |  DHCPREQUEST\  |
       |                |                |
       |                |     Commits configuration
       |                |                |
       |                | _____________/|
       |                |/ DHCPACK       |
       |                |                |
       |    Initialization complete      |
       |                |                |
       .                .                .
       .                .                .
       |                |                |
       |      Graceful shutdown          |
       |                |                |
       |                |\ _____________ |
       |                | DHCPRELEASE  \|
       |                |                |
       |                |        Discards lease
       |                |                |
       v                v                v
```

2.

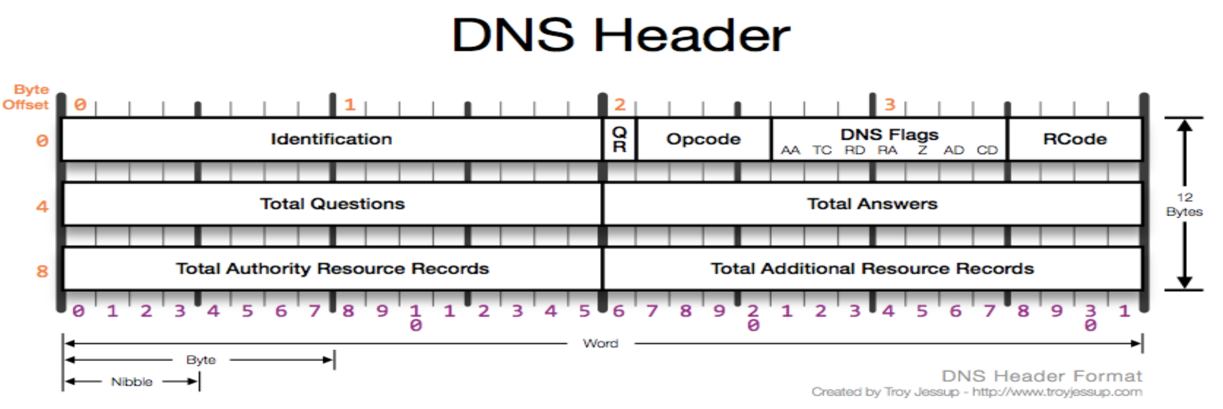
1.http://www.tiwoc.de/blog/2008/05/dynamic-host-configuration-protocol/

SMTP

- Important vocabulary
  - Mail User Agent (MUA) – end-users mail client
  - Mail Submission Agent ((MSA) – Local mail submissions
  - Mail Transfer Agent (MTA) – transfers mail between mail servers
  - Mail eXchanger (MX) – accepts incoming messages for a domain
  - Mail Delivery Agent (MDA) – local mail delivery
- Basic commands
  - HELO – opens connection
  - MAIL – identifies return address
  - RCPT – identifies recipient address
  - DATA – message content

DNS

- Query-response protocol
  - Client question = single UDP packet
  - Server response = single UDP packet
  -

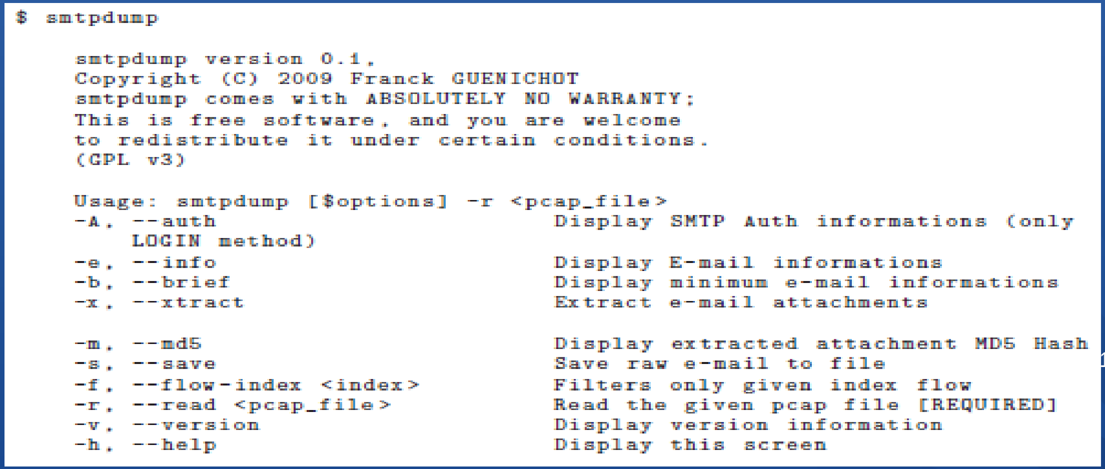


[1]

[1] http://www.troyjessup.com/headers/DNS_Header.png

Higher-layer analysis tools

- Oftcat
  - Input = reassembled single flow of transport layer payload (ex: tcpflow or pcapcat)
  - Output = protocol summary of all OFT activity and any recovered files transferred
  - http://blog.kiddaland.net/dw/oftcat
- Smtpdump



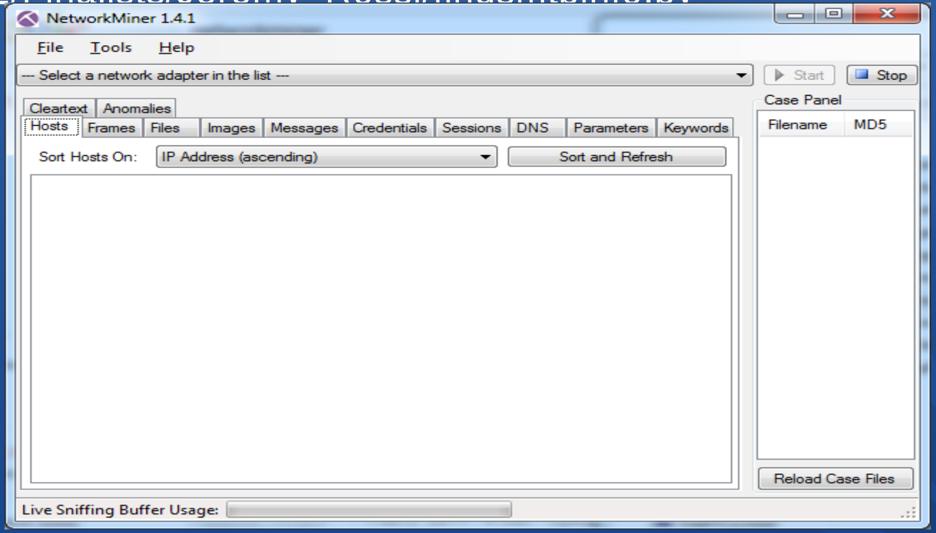
```
$ smtpdump

    smtpdump version 0.1,
    Copyright (C) 2009 Franck GUENICHOT
    smtpdump comes with ABSOLUTELY NO WARRANTY;
    This is free software, and you are welcome
    to redistribute it under certain conditions.
    (GPL v3)

    Usage: smtpdump [$options] -r <pcap_file>
    -A, --auth                              Display SMTP Auth informations (only
        LOGIN method)
    -e, --info                              Display E-mail informations
    -b, --brief                             Display minimum e-mail informations
    -x, --xtract                            Extract e-mail attachments

    -m, --md5                               Display extracted attachment MD5 Hash
    -s, --save                              Save raw e-mail to file
    -f, --flow-index <index>                Filters only given index flow
    -r, --read <pcap_file>                  Read the given pcap file [REQUIRED]
    -v, --version                           Display version information
    -h, --help                              Display this screen
```

1.

2. Image/s clipped from work cited

Higher-layer analysis tools

- Findsmtpinfo.py
  - Input = pcap file
  - Output = extracted authentication data, credentials, mail header info, attachments, MD5 sum and produces a report
  - http://forensicscontest.com/contest02/Finalists/Jeremy_Rossi/findsmtpinfo.py
- NetworkMiner
  - Multipurpose traffic analyzer



Higher-layer analysis techniques

- Small specialized tools
  - Great for higher-layer protocol analysis
  - Best to use if you have a good idea of what the packet contains
  - Most interface easily with other tools
  - Example:
    - Oftcat
    - smtpdump
- Multipurpose tools
  - Best when a wide range of information is needed
  - Gather lots of different information
  - Example:
    - NetworkMiner

Works Cited

Davidoff, S., & Ham, J. (2012). *Network Forensics Tracking Hackers Through Cyberspace.* Boston: Prentice Hall.