

# Forensics in CTF - part 2

oalieno

# What is Forensics

- 在 CTF 比賽中，Forensics 可能會是以下幾種類型
  - File Format Analysis
  - Steganography
  - **Memory Dump Analysis**
  - **Network Packet Analysis**
  - ...



# What is Forensics

[chtsecurity.com/service/m401](http://chtsecurity.com/service/m401)

- Forensics 這個字的意思就是鑑識、取證，原本是用在法律案件中的犯罪鑑識、取證
- 如今，新型態的資安案件的出現，Forensics 也可以指資安事件的鑑識、取證
- 主要目的是在協助客戶緊急應變、入侵管道定位、受影響範圍評估及回復受駭系統



# Memory Dump Analysis

# Dump Memory

檔案總管

本機

內容



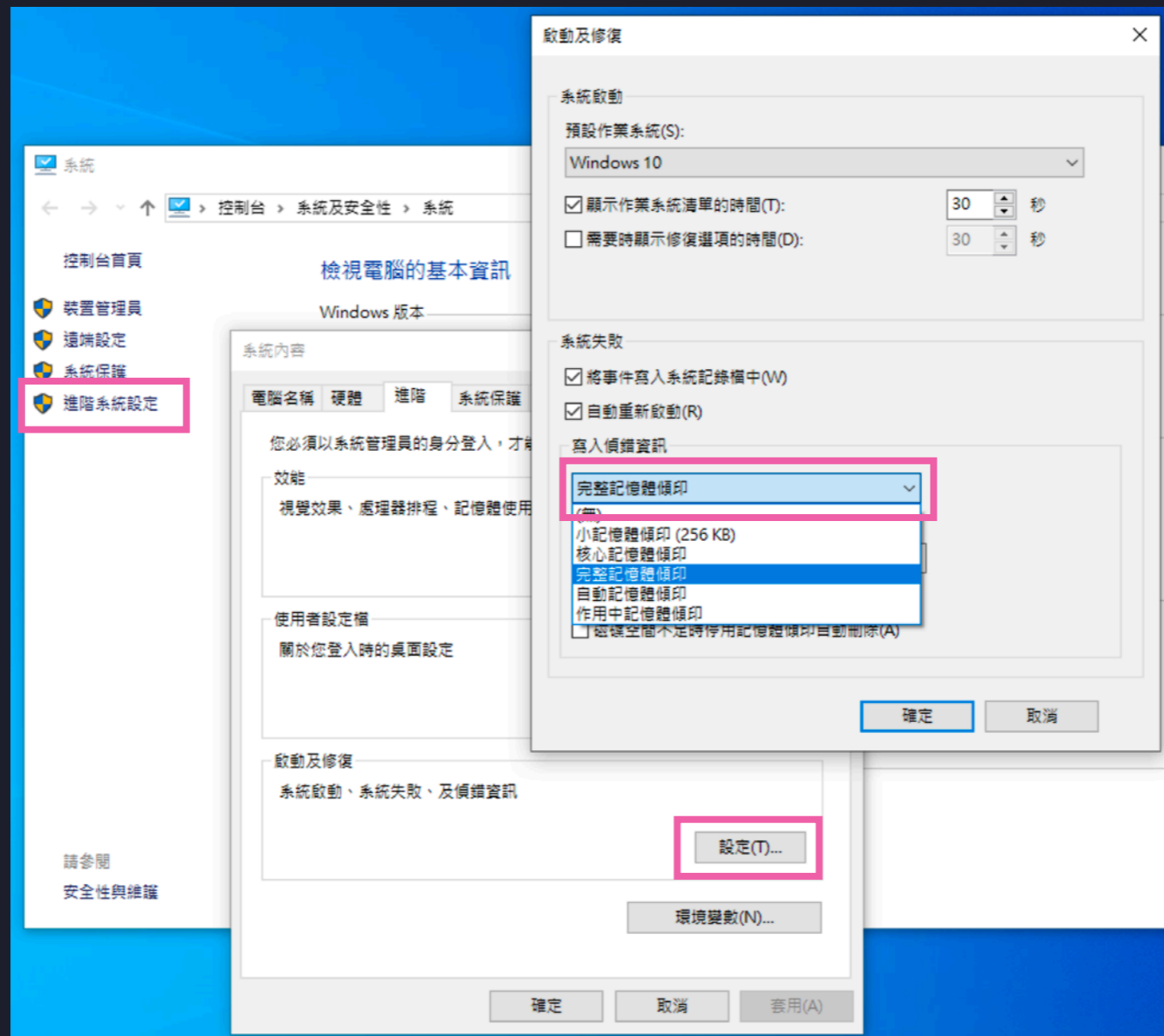
# Dump Memory

進階系統設定

進階

啟動及修復

寫入偵錯資訊



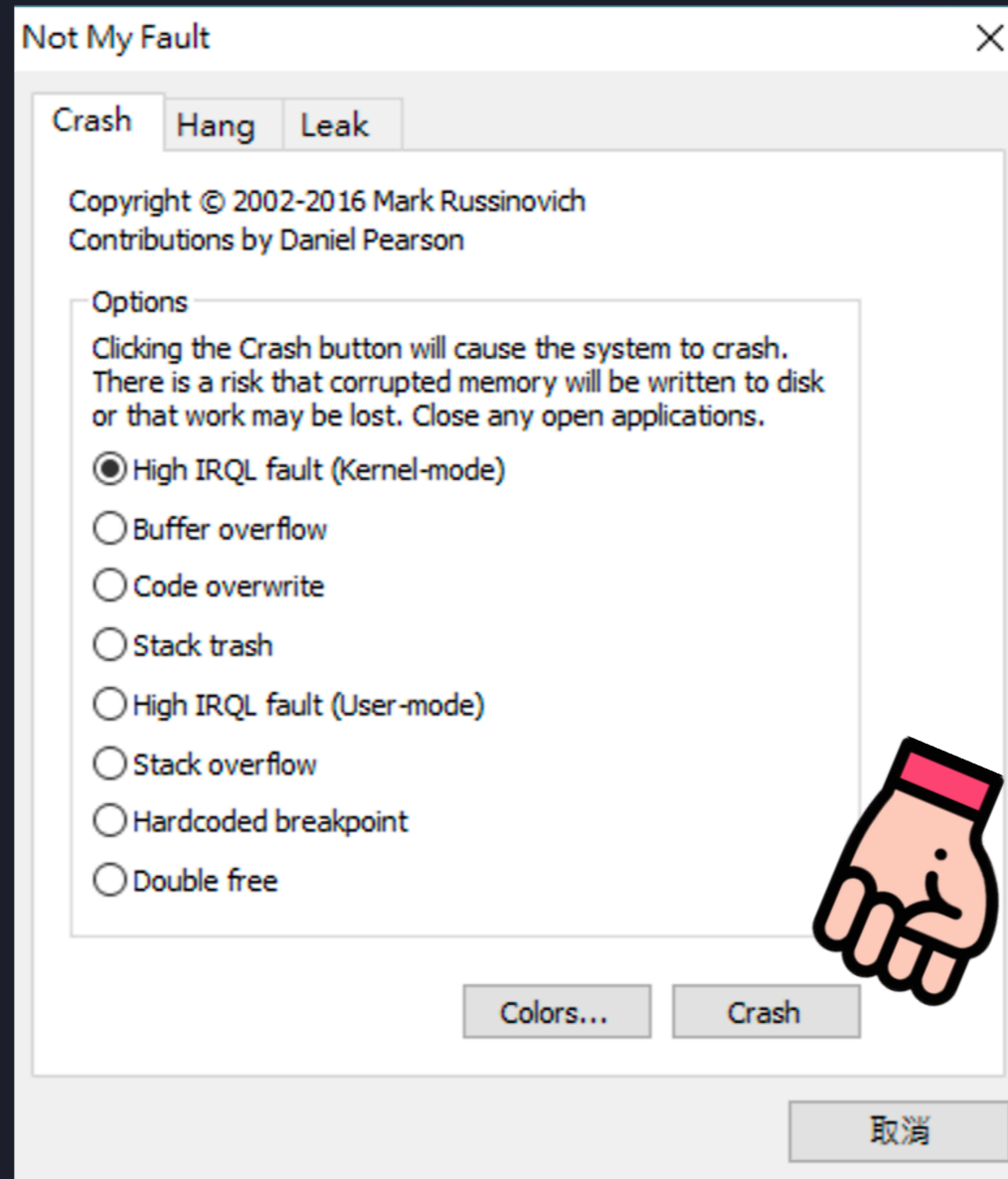
# Dump Memory

- 設定完整記憶體傾印可以拿到比較完整的訊息，不過檔案會變很大
- 可以設定一下傾印檔案到你喜歡的路徑



# Crash It !!!

<https://docs.microsoft.com/en-us/sysinternals/downloads/notmyfault>







您的電腦發生問題，因此必須重新啟動。  
我們剛剛正在  
收集某些錯誤資訊，接著我們會為您重新啟動。

已完成 100%



如需此問題與可能修正的詳細資訊，請瀏覽 <https://www.windows.com/stopcode>

致電支援人員時，請提供此資訊：

停止代碼: DRIVER\_IRQL\_NOT\_LESS\_OR\_EQUAL

失敗的項目: myfault.sys

# Get Dump Memory

- 在電腦掛掉之後，就會根據剛剛的設定產生 memory dump
- 那我們就可以去分析這個 memory dump
- 然後釐清到底發生什麼事情了

# volatility

- Volatile memory extraction utility framework
- RAM 就是一種 Volatile memory
- 有很多 plugin 可以用
- 用這個工具來分析 memory dump 出來的檔案吧

# imageinfo

```
volatility -f file imageinfo
```

```
~> volatility -f Mem_Evidence.raw imageinfo
Volatility Foundation Volatility Framework 2.6
INFO      : volatility.debug      : Determining profile based on KDBG search...
          : Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_23418
          : AS Layer1           : WindowsAMD64PagedMemory (Kernel AS)
          : AS Layer2           : FileAddressSpace (/home/oalieno_eecs04/Mem_Evidence.raw)
          : PAE type            : No PAE
          : DTB                 : 0x187000L
          : KDBG                 : 0xf800027f60a0L
          : Number of Processors : 1
          : Image Type (Service Pack) : 1
          : KPCR for CPU 0       : 0xffffffff800027f7d00L
          : KUSER_SHARED_DATA    : 0xffffffff78000000000L
          : Image date and time   : 2019-06-29 07:30:00 UTC+0000
          : Image local date and time : 2019-06-29 13:00:00 +0530
```

# pslist

```
volatility -f file --profile=Win7SP1x64 pslist
```

```
~> volatility -f Mem_Evidence.raw --profile=Win7SP1x64 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)      Name          PID  PPID  Thds  Hnds  Sess  Wow64  Start
-----
0xffffffff8000ca0040 System        4    0    79   509  -----  0  2019-06-29 07:28:07 UTC+0000
0xffffffff80014af950 smss.exe     256  4    3    32  -----  0  2019-06-29 07:28:07 UTC+0000
0xffffffff8001c57b30 csrss.exe   328  320  11   385  0        0  2019-06-29 07:28:14 UTC+0000
0xffffffff8000ca8960 csrss.exe   376  368  7    200  1        0  2019-06-29 07:28:15 UTC+0000
0xffffffff8001c6f760 wininit.exe 384  320  3    75  0        0  2019-06-29 07:28:15 UTC+0000
0xffffffff8001c751f0 winlogon.exe 412  368  6   119  1        0  2019-06-29 07:28:15 UTC+0000
0xffffffff8001bc1b30 services.exe 472  384  13   193  0        0  2019-06-29 07:28:17 UTC+0000
0xffffffff8001cb5940 lsass.exe   480  384  8   582  0        0  2019-06-29 07:28:17 UTC+0000
0xffffffff8001cc1b30 lsm.exe     488  384  12   187  0        0  2019-06-29 07:28:17 UTC+0000
0xffffffff8001d02b30 svchost.exe 580  472  11   358  0        0  2019-06-29 07:28:21 UTC+0000
0xffffffff8001d30b30 VBoxService.ex 640  472  14   137  0        0  2019-06-29 07:28:21 UTC+0000
0xffffffff8001d43a70 svchost.exe 708  472  7    260  0        0  2019-06-29 07:28:22 UTC+0000
0xffffffff8001dacb30 svchost.exe 804  472  19   393  0        0  2019-06-29 07:28:23 UTC+0000
0xffffffff8001db9b30 svchost.exe 840  472  21   431  0        0  2019-06-29 07:28:24 UTC+0000
0xffffffff8001dc6850 svchost.exe 864  472  37   917  0        0  2019-06-29 07:28:24 UTC+0000
0xffffffff8001df1060 audiodg.exe 952  804  7    131  0        0  2019-06-29 07:28:26 UTC+0000
0xffffffff8001e1b890 svchost.exe 220  472  16   323  0        0  2019-06-29 07:28:27 UTC+0000
0xffffffff8001e45630 svchost.exe 484  472  18   376  0        0  2019-06-29 07:28:29 UTC+0000
0xffffffff8001eaab30 spoolsv.exe 1132 472  15   286  0        0  2019-06-29 07:28:32 UTC+0000
0xffffffff8001ed7b30 svchost.exe 1176 472  21   307  0        0  2019-06-29 07:28:33 UTC+0000
0xffffffff8001f452e0 svchost.exe 1276 472  14   220  0        0  2019-06-29 07:28:34 UTC+0000
0xffffffff8001f81b30 taskhost.exe 1804 472  10   161  1        0  2019-06-29 07:28:42 UTC+0000
0xffffffff8001ff9630 taskeng.exe 1824 864  6    82  0        0  2019-06-29 07:28:42 UTC+0000
0xffffffff80020bbb30 dwm.exe     1908 840  5    77  1        0  2019-06-29 07:28:43 UTC+0000
```

# filescan

```
volatility -f file --profile=Win7SP1x64 filescan
```

```
~> volatility -f Mem_Evidence.raw --profile=Win7SP1x64 filescan
Volatility Foundation Volatility Framework 2.6
Offset(P)          #Ptr  #Hnd Access Name
-----
0x000000003e8009a0      1      1 R--rwd \Device\HarddiskVolume2\Windows\System32\catroot
0x000000003e800f20     14      0 R--r-d \Device\HarddiskVolume2\Windows\System32\SearchIndexer.exe
0x000000003e8012d0     14      0 R--r-d \Device\HarddiskVolume2\Windows\System32\esent.dll
0x000000003e8027c0      1      1 R--rw- \Device\HarddiskVolume2\Windows\System32
0x000000003e80d320      1      1 R--rw- \Device\HarddiskVolume2\Windows\winsxs\amd64_microsoft.wir
0x000000003e812070     10      0 R--r-d \Device\HarddiskVolume2\Windows\System32\msidle.dll
0x000000003e812c00      8      0 R--r-d \Device\HarddiskVolume2\Windows\System32\netman.dll
0x000000003e816070      2      0 R--r-d \Device\HarddiskVolume2\Windows\System32\rasdlg.dll
0x000000003e819960     17      0 RW-rwd \Device\HarddiskVolume2\Directory
0x000000003e819f20      8      0 R--r-d \Device\HarddiskVolume2\Windows\System32\ActionCenter.dll
```

# plugins

- 除了內建的 plugins 還有很多別人寫的
- 比如這個 [volatility-plugins](#)
- 官方社群的 [github](#) 上面也有很多
- 每年還會有 [plugin contest](#)





# 練習題



InCTF 2019 - Just Do It



InCTF 2019 - Notch It Up

# Network Packet Analysis

# wireshark

- 免費開源的網路封包分析軟體
- command line 版的：tshark
- python package：pyshark



# pcap v.s. pcapng

- pcapng 就是 pcap next generation 的縮寫
- 所以 pcapng 就是強化版的 pcap
- 可以用 editcap 指令把 pcapng 轉成 pcap

```
editcap -F libpcap -T ether file.pcapng file.pcap
```

# Filter

```
http
```

```
ip.src == 192.168.0.1 && ip.dst == 127.0.0.1
```

```
http and http.request.uri contains "index"
```

```
frame contains "flag"
```

# Find Out Field Name

The screenshot shows the Wireshark network protocol analyzer interface. The main pane displays a list of captured packets, with packet 3306 selected. A context menu is open over packet 3306, showing options like 'Copy', 'Show Packet Bytes...', and 'Field Name'. The 'Field Name' option is highlighted. The packet details pane shows the 'Request URI' field with the value 'http://connectivity-check.ubuntu.com/'.

No.	Time	Source	Destination	Protocol	Length	Info
2530	314.627689193	192.168.43.97	151.101.112.246	HTTP	264	GET /image/62c225bdbe30485332b18cf9cbfbaeb010b33b21 HTTP/1.1
2575	314.750904556	151.101.112.246	192.168.43.97	HTTP	284	HTTP/1.1 200 OK (JPEG JFIF image)
2747	335.708348601	192.168.43.97	151.101.112.246	HTTP	264	GET /image/118247f5437b8a527487f5f99ce576b8fbd98fc HTTP/1.1
2796	335.783556996	151.101.112.246	192.168.43.97	HTTP	1148	HTTP/1.1 200 OK (JPEG JFIF image)
2862	357.022596233	192.168.43.97	104.199.64.136	HTTP	291	GET /?time=1566996509&type=accesspoint HTTP/1.1
2866	357.113306916	104.199.64.136	192.168.43.97	HTTP	73	HTTP/1.1 200 OK (application/json)
3079	366.932999909	192.168.43.97	151.101.12.246	HTTP	264	GET /image/606336f26cd23dc672ba3b1ad986f0284e089d7d HTTP/1.1
3166	388.541324909	192.168.43.97	151.101.12.246	HTTP	264	GET /image/9f8bd0efcc5566b42db44064b6a1e0356c5dbdd4 HTTP/1.1
3244	400.374465732	192.168.43.97	151.101.12.246	HTTP	264	GET /image/26a3e45a68317a8e21a08b02bb136335b8963ae1 HTTP/1.1
3281	400.496290242	151.101.12.246	192.168.43.97	HTTP	1548	HTTP/1.1 200 OK (JPEG JFIF image)
3306	404.417468723	192.168.43.97	35.224.99.156	HTTP	155	GET / HTTP/1.1
3310	404.768298211	35.224.99.156	192.168.43.97	HTTP	216	HTTP/1.1 204 No Content
337		192.168.43.97	151.101.12.246	HTTP	264	GET /image/7f3fd84167f5f990f570aed338f8ed31541d085 HTTP/1.1
358		192.168.43.97	151.101.112.246	HTTP	264	GET /image/70b0c1d8ae1d88e5a4ced559b97bc06ab048ee56 HTTP/1.1
375		192.168.43.97	151.101.112.246	HTTP	264	GET /image/1a3f165b0e7a0ea003c9f3a447e05cc2dfa4288a HTTP/1.1
379		192.168.43.97	192.168.43.97	HTTP	1262	HTTP/1.1 200 OK (JPEG JFIF image)
382		192.168.43.97	151.101.112.246	HTTP	264	GET /image/5955b33bd14a312f331bb3c2dc3ac527fc21d6df HTTP/1.1
387		192.168.43.97	192.168.43.97	HTTP	2678	HTTP/1.1 200 OK (JPEG JFIF image)
392		192.168.43.97	151.101.112.246	HTTP	264	GET /image/96b35b373991e847a94926e21f1fddc4a82ec784 HTTP/1.1
394		151.101.112.246	192.168.43.97	HTTP	1382	HTTP/1.1 200 OK (JPEG JFIF image)
398		192.168.43.97	151.101.112.246	HTTP	264	GET /image/77eb7c17cafe550265ac9656051fe4e651a00d70 HTTP/1.1
400		151.101.112.246	192.168.43.97	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
405		192.168.43.97	151.101.112.246	HTTP	264	GET /image/a392f0f7a7dbc6424f769f6f7f7824e40c42a734 HTTP/1.1
408		192.168.43.97	192.168.43.97	HTTP	1453	HTTP/1.1 200 OK (JPEG JFIF image)
429		192.168.43.97	192.168.43.97	HTTP	264	GET /image/45881f936e2aad16355052fdf68cf54ed4cacba HTTP/1.1
437		192.168.43.97	192.168.43.97	HTTP	578	GET /?flag=Did%20you%20got%20the%20flag%20at%20DCTF?%20N0?%20Try%20Harder! HTTP/1.1

Context Menu for Packet 3306:

- Expand Subtrees
- Expand All
- Collapse All
- Apply as Column
- Apply as Filter
- Prepare a Filter
- Conversation Filter
- Colorize with Filter
- Follow
- Copy
  - All Visible Items
  - All Visible Selected Tree Items
  - Description
  - Field Name
  - Value
  - As Filter
- Show Packet Bytes...
- Export Packet Bytes...
- Wiki Protocol Page
- Filter Field Reference
- Protocol Preferences
- Decode As...
- Go to Linked Packet
- Show Linked Packet in New Window

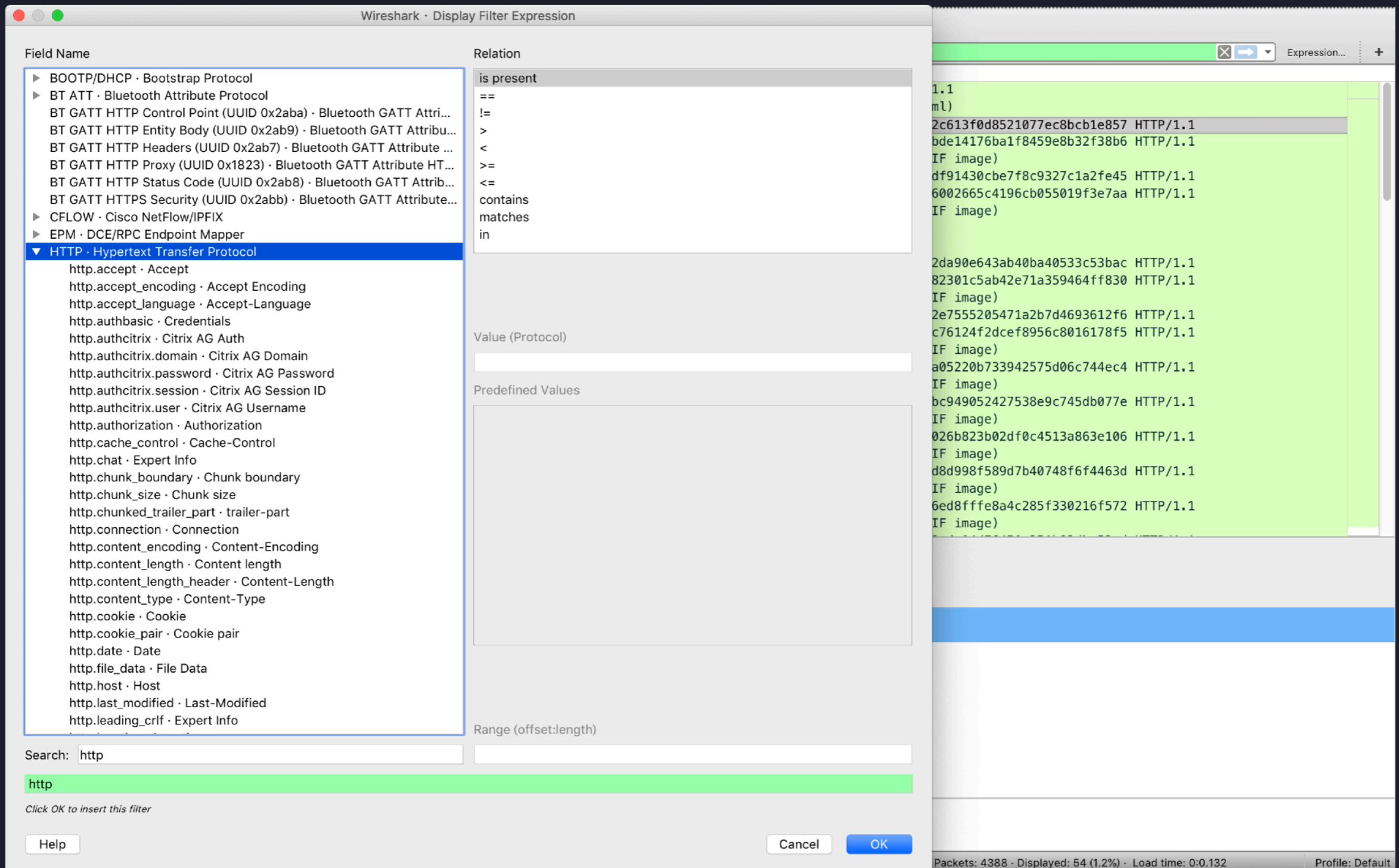
Packet Details for Packet 3306:

Request URI: /  
Request Version: HTTP/1.1  
Host: connectivity-check.ubuntu.com\r\n  
Accept: \*/\*\r\n  
Connection: close\r\n  
\r\n  
[\[Full request URI: http://connectivity-check.ubuntu.com/\]](http://connectivity-check.ubuntu.com/)

Packet Bytes:

```
0040 2c 94 d4 c1 47 45 54 20 2f 20 48 54 54 50 2f 31  ...GET / HTTP/1
0050 2e 31 0d 0a 48 6f 73 74 3a 20 63 6f 6e 6e 65 63  .1..Host : connec
```

# Filter Expression



The image shows the Wireshark 'Display Filter Expression' dialog box overlaid on a packet capture window. The dialog box is titled 'Wireshark - Display Filter Expression' and is divided into several sections:

- Field Name:** A tree view of protocol fields. The 'HTTP' category is expanded, and 'http' is selected. A search box at the bottom left contains 'http', and a list of matching fields is shown below it.
- Relation:** A list of logical operators. 'is present' is selected.
- Value (Protocol):** An empty text input field.
- Predefined Values:** An empty list box.
- Range (offset:length):** An empty text input field.

At the bottom of the dialog box, there are buttons for 'Help', 'Cancel', and 'OK'. A small instruction reads 'Click OK to insert this filter'.

The background window shows a packet capture list with several HTTP packets. The first packet is highlighted in green, indicating it matches the filter. The packet list shows the following details:

- 1.1 (mL)
- 2c613f0d8521077ec8bcb1e857 HTTP/1.1
- bde14176ba1f8459e8b32f38b6 HTTP/1.1
- IF image)
- df91430cbe7f8c9327c1a2fe45 HTTP/1.1
- 6002665c4196cb055019f3e7aa HTTP/1.1
- IF image)
- 2da90e643ab40ba40533c53bac HTTP/1.1
- 82301c5ab42e71a359464ff830 HTTP/1.1
- IF image)
- 2e7555205471a2b7d4693612f6 HTTP/1.1
- c76124f2dcef8956c8016178f5 HTTP/1.1
- IF image)
- a05220b733942575d06c744ec4 HTTP/1.1
- IF image)
- bc949052427538e9c745db077e HTTP/1.1
- IF image)
- 026b823b02df0c4513a863e106 HTTP/1.1
- IF image)
- d8d998f589d7b40748f6f4463d HTTP/1.1
- IF image)
- 6ed8fffe8a4c285f330216f572 HTTP/1.1
- IF image)

The status bar at the bottom of the packet capture window shows: 'Packets: 4388 - Displayed: 54 (1.2%) - Load time: 0:0.132 Profile: Default'.

# Follow Stream

The image shows a Wireshark network traffic analysis tool interface. The main pane displays a list of captured packets. Packet 4006 is selected, and a context menu is open over it, with the 'Follow' option highlighted. A sub-menu is also open, showing 'HTTP Stream' as the selected option. Below the packet list, the packet details pane shows the structure of the selected packet, including the frame, Linux cooked capture, Internet Protocol Version 4, Transmission Control Protocol, and Hypertext Transfer Protocol. The bottom status bar shows the current protocol as Hypertext Transfer Protocol: Protocol, with 4388 packets displayed (54 shown, 1.2% load time: 0:0.216) and a default profile.

No.	Time	Source	Destination	Protocol	Length	Info
4381	557.130450247	93.184.216.34	192.168.43.97	HTTP	1034	HTTP/1.1 200 OK (text/html)
4379	556.867668969	192.168.43.97	93.184.216.34	HTTP	578	GET /?flag=Did%20you%20got%20the%20flag%20at%20DCTF?%20NO?%20Try%20Harder! HTTP/
4296	528.086287350	192.168.43.97	151.101.12.246	HTTP	264	GET /image/45881f936e2aadb16355052fdf68cf54ed4cacba HTTP/1.1
4088	498.733914002	151.101.112.246	192.168.43.97	HTTP	1453	HTTP/1.1 200 OK (JPEG JFIF image)
4059	498.674253287	192.168.43.97	151.101.112.246	HTTP	264	GET /image/a392f0f7a7dbc6424f769f6f7f7824e40c42a734 HTTP/1.1
4006	485.465222028	151.101.112.246	192.168.43.97	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3983	485.401050060	192.168.43.97	151.101.112.246	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3942	475.886457428	151.101.112.246	192.168.43.97	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3923	475.820925330	192.168.43.97	151.101.112.246	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3873	461.797234942	151.101.112.246	192.168.43.97	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3824	461.711862725	192.168.43.97	151.101.112.246	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3796	452.689978522	151.101.112.246	192.168.43.97	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3750	452.595306313	192.168.43.97	151.101.112.246	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3588	438.428590835	192.168.43.97	151.101.112.246	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3379	420.315978347	192.168.43.97	151.101.12.246	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3310	404.768298211	35.224.99.156	192.168.43.97	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3306	404.417468723	192.168.43.97	35.224.99.156	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3281	400.496290242	151.101.12.246	192.168.43.97	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3244	400.374465732	192.168.43.97	151.101.12.246	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3166	388.541324909	192.168.43.97	151.101.12.246	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
3079	366.932999909	192.168.43.97	151.101.12.246	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
2866	357.113306916	104.199.64.136	192.168.43.97	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
2862	357.022596233	192.168.43.97	104.199.64.136	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
2796	335.783556996	151.101.112.246	192.168.43.97	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
2747	335.708348601	192.168.43.97	151.101.112.246	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)
2575	314.750904556	151.101.112.246	192.168.43.97	HTTP	1106	HTTP/1.1 200 OK (JPEG JFIF image)

▶ Frame 4006: 1106 bytes on wire (8848 bits), 1106 bytes captured (8848 bits) on interface 0  
▶ Linux cooked capture  
▶ Internet Protocol Version 4, Src: 151.101.112.246, Dst: 192.168.43.97  
▶ Transmission Control Protocol, Src Port: 80, Dst Port: 52176, Seq: 99951, Ack: 981, Len: 1038  
▶ [8 Reassembled TCP Segments (17901 bytes): #3992(1408), #3994(2816), #3996(2816), #4000(1408), #3998(223), #4002(4224), #4004(3968), #4006(1038)]  
▶ Hypertext Transfer Protocol  
▶ JPEG File Interchange Format

0000 00 00 00 01 00 06 48 2c a0 6a b3 15 e2 48 08 00 .....H. .j...H..  
0010 45 00 04 42 cf 09 40 00 b2 06 01 47 97 65 70 f6 E..B..@. ...G.ep.

Frame (1106 bytes) Reassembled TCP (17901 bytes)

Hypertext Transfer Protocol: Protocol Packets: 4388 - Displayed: 54 (1.2%) - Load time: 0:0.216 Profile: Default



# Save Stream Data

The screenshot shows the Wireshark interface with a packet list pane at the top displaying an HTTP 200 OK response. The packet details pane shows the following headers:

```
HTTP/1.1 200 OK
Etag: "I1h1kVC3/BzCjQRReBqn6xNKK/FnCyxje7QPDBksng"
Fastly-Io-Info: ifsz=25084 idim=300x300 ifmt=jpeg ofsz=15087 odim=300x300 ofmt=jpeg
Fastly-Stats: io=1
Cache-Control: max-age=315360000
Accept-Ranges: bytes
Date: Wed, 28 Aug 2019 12:43:55 GMT
Age: 91209
Connection: keep-alive
X-Served-By: cache-ord1742-ORD, cache-hhn4067-HHN
X-Cache: HIT, HIT
X-Cache-Hits: 1, 1
Access-Control-Allow-Origin: *
Content-Type: image/jpeg
Content-Length: 15087
```

The packet bytes pane shows the start of a JPEG image header:

```
.....JFIF.....ICC_PROFILE.....lcms.....mnrRGB XYZ .....
desc.....^cprt...
\....wptp...h....bkpt...|....rXYZ.....gXYZ.....bXYZ.....rT
.....text....FB..XYZ .....-XYZ .....XYZ .....$.
2.;.F.Qw].kpz....|.i.)...0.....C.....
...
.....) .. )/'%/9339GDG]]}...C.....
...
.....) .. )/'%/9339GDG]]}....."
.....}.....!1A..Qa."q.2....#B...R..s3br.
...%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxz.....
.....w.....!1..AQ.aq."2...B.... #3R..br.
.$4.%.....&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxz.....?....ZZ..H(..E.8R
u&R
z...J.E&Qm{U..Z..EZ..V..=>Dm...-..jf.j..B.UH.j..9.qQ....QE.D..Q@.4..R....P(.85%...i..
..P+...S...a.>.Pj|.c..i.Zq..MjD..?J(.b'.)...g...@..p.kQE.QH...i)6Z.Y^..j>..U^..f>...U..F...
..n..B...u!a.T5d.Usz..bQE.b
(..
(..
(..
(..
i.R...m..Q..(u.S.M...PE%...*3RTf.).LJ(...S.)....h...@.....- ..(R.....,j..G...j..J.....MD.3u5.R]J..a.Z.5.{V....QE.b
(..
```

A 'Save As' dialog box is open in the center of the screen, with the following fields:

- Save As:
- Tags:
- Where:  (with a dropdown arrow)
- Buttons: Cancel, Save

At the bottom of the Wireshark interface, there is a status bar showing '0 client pkts, 2 server pkts, 0 turns.' and a network address range '151.101.112.246:80 → 192.168.43.97:52142 (46 kB)'. A 'Find:' search bar is also present, with a mouse cursor pointing to the 'Save as...' button in the bottom toolbar.

# Export Objects

The screenshot shows the Wireshark interface with the 'File' menu open and 'Export Objects' selected. The main pane displays a list of captured packets, with packet 1645 highlighted. The 'Export Objects' submenu is visible, listing various protocols like DICOM, HTTP, IMF, SMB, and TFTP. The packet list table is as follows:

No.	Destination	Protocol	Length	Info
2335	192.168.43.97	HTTP	2765	HTTP/1.1 200 OK (JPEG JFIF image)
2299	151.101.112.246	HTTP	264	GET /image/ba9f2138015f926ed8fffe8a4c285f330216f572 HTTP/1.1
2241	192.168.43.97	HTTP	1086	HTTP/1.1 200 OK (JPEG JFIF image)
2187	151.101.112.246	HTTP	264	GET /image/8b96f771abe2f3d8d998f589d7b40748f6f4463d HTTP/1.1
2151	192.168.43.97	HTTP	5264	HTTP/1.1 200 OK (JPEG JFIF image)
2109	151.101.112.246	HTTP	264	GET /image/77eb7c17cafe55026b823b02df0c4513a863e106 HTTP/1.1
2061	192.168.43.97	HTTP	691	HTTP/1.1 200 OK (JPEG JFIF image)
2001	151.101.112.246	HTTP	264	GET /image/474115d5ea751cbc949052427538e9c745db077e HTTP/1.1
1771	192.168.43.97	HTTP	329	HTTP/1.1 200 OK (JPEG JFIF image)
1726	151.101.12.246	HTTP	264	GET /image/e08b756820a7c5a05220b733942575d06c744ec4 HTTP/1.1
1680	192.168.43.97	HTTP	1243	HTTP/1.1 200 OK (JPEG JFIF image)
1645	151.101.12.246	HTTP	264	GET /image/336471918174b6c76124f2dcef8956c8016178f5 HTTP/1.1
1573	151.101.12.246	HTTP	264	GET /image/d3b33d8067e34f2e7555205471a2b7d4693612f6 HTTP/1.1
1459	151.101.12.246	HTTP	2696	HTTP/1.1 200 OK (JPEG JFIF image)
1420	12.246	HTTP	264	GET /image/c588ccad32d0d482301c5ab42e71a359464ff830 HTTP/1.1
1195	112.246	HTTP	264	GET /image/7e79bbb6f3c7e32da90e643ab40ba40533c53bac HTTP/1.1
1153	43.97	HTTP	216	HTTP/1.1 204 No Content
1151	35.5	HTTP	155	GET / HTTP/1.1
1118	192.168.43.97	HTTP	1302	HTTP/1.1 200 OK (JPEG JFIF image)
1071	151.101.112.246	HTTP	264	GET /image/41d223339d4a7b6002665c4196cb055019f3e7aa HTTP/1.1
1009	151.101.112.246	HTTP	264	GET /image/43b85702779d9cdf91430cbe7f8c9327c1a2fe45 HTTP/1.1
937	192.168.43.97	HTTP	1331	HTTP/1.1 200 OK (JPEG JFIF image)
912	151.101.112.246	HTTP	264	GET /image/0843f93df10815bde14176ba1f8459e8b32f38b6 HTTP/1.1
538	151.101.112.246	HTTP	264	GET /image/20d5ccc04cba362c613f0d8521077ec8bc1e857 HTTP/1.1
26	192.168.43.97	HTTP	1040	HTTP/1.1 200 OK (text/html)
24	93.184.216.34	HTTP	523	GET /?flag=welcome! HTTP/1.1

The packet details pane for frame 1645 shows:

- Frame 1645: 264 bytes on wire (2112 bits), 264 bytes captured (2112 bits) on interface 0
- Linux cooked capture
- Internet Protocol Version 4, Src: 192.168.43.97, Dst: 151.101.12.246
- Transmission Control Protocol, Src Port: 35840, Dst Port: 80, Seq: 393, Ack: 53909, Len: 196
- Hypertext Transfer Protocol

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```
0000 00 04 00 01 00 06 50 76 af 24 eb ef 00 00 08 00 .....Pv .$......
0010 45 00 00 f8 aa 63 40 00 40 06 ff 37 c0 a8 2b 61 E....c@. @.7...+
0020 97 65 0c f6 8c 00 00 50 b9 3e 4d 45 7a fd bb b9 .e.....P .>MEZ...
```

# 練習題



DCTF 2019 - radio station

# tshark example

```
tshark -r radio.pcapng -Y "http and http.request.uri contains image" -T fields -e 'http.request.uri'
```

- -r : 要分析的 pcap 檔案
- -Y : filter 規則
- -e : 要 extract 的欄位
- -T : extract 的格式

# pyshark example

```
#!/usr/bin/env python3
import pyshark

cap = pyshark.FileCapture('./radio.pcapng', display_filter = "http")

for packet in cap:
    try:
        if 'image' in packet.http.request_uri:
            print(packet.http.request_uri)
    except AttributeError:
        pass
```