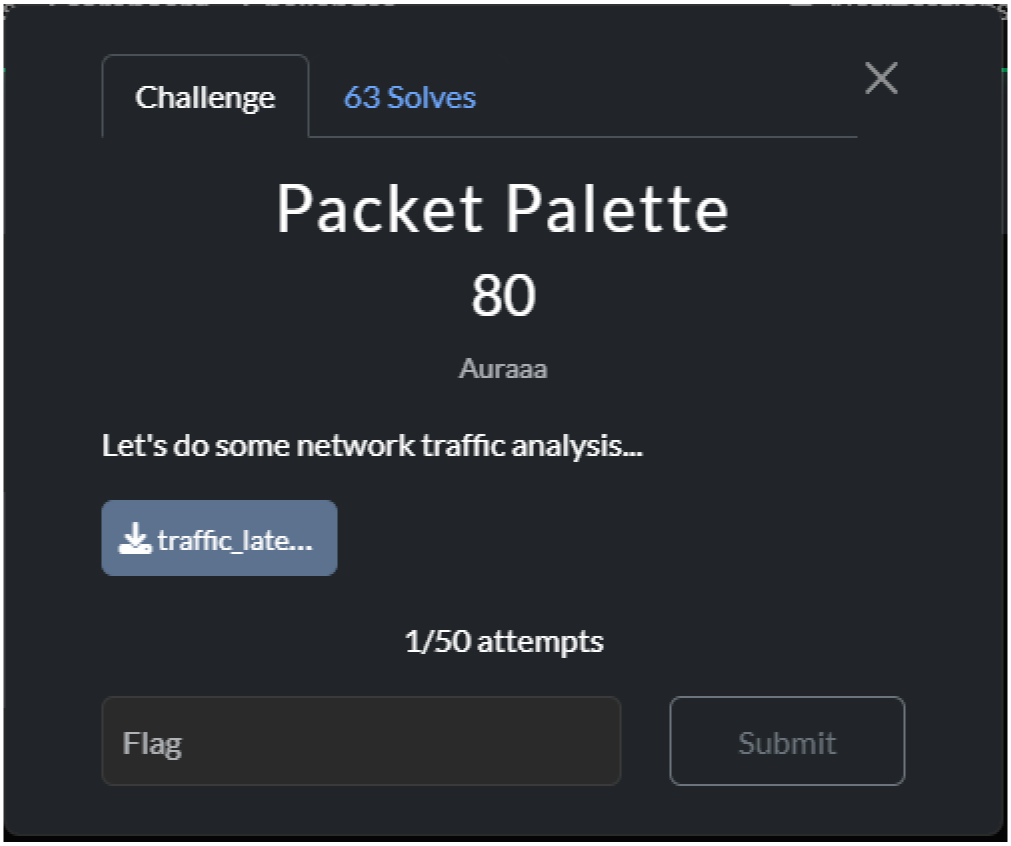
Packet Palette

Types	misc
CTF	SunCTF



Packet Palette

80

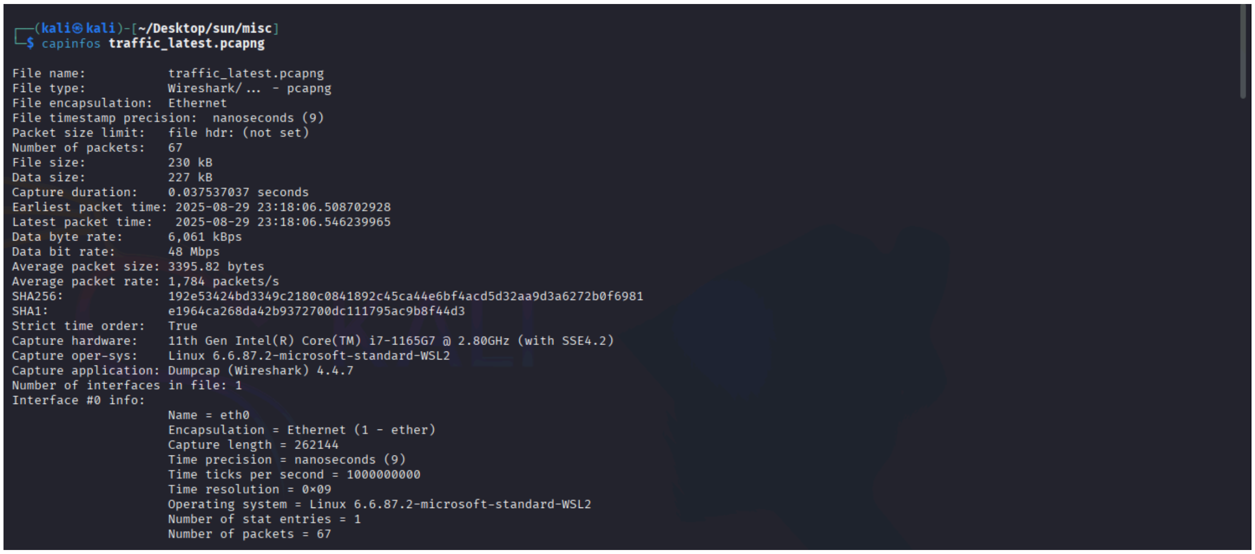
Auraaa

Let's do some network traffic analysis...

1/50 attempts

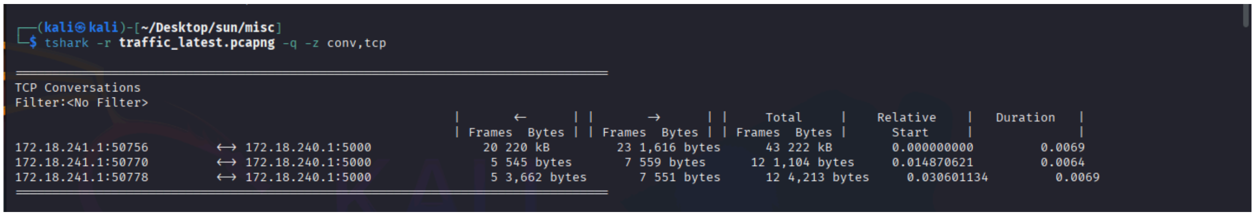
given file is in pcap file

Step 1 — Quick reconnaissance



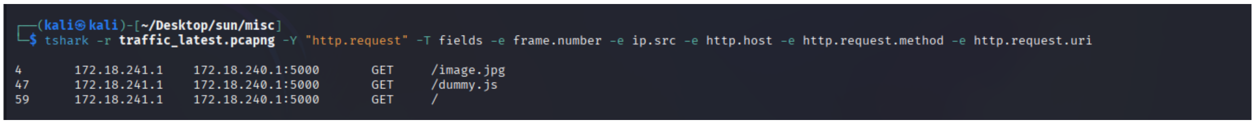
```
┌──(kali㉿kali)-[~/Desktop/sun/misc]
└─$ capinfos traffic_latest.pcapng

File name:           traffic_latest.pcapng
File type:           Wireshark/... - pcapng
File encapsulation:  Ethernet
File timestamp precision:  nanoseconds (9)
Packet size limit:   file hdr: (not set)
Number of packets:   67
File size:           230 kB
Data size:           227 kB
Capture duration:    0.037537037 seconds
Earliest packet time: 2025-08-29 23:18:06.508702928
Latest packet time:   2025-08-29 23:18:06.546239965
Data byte rate:      6,061 kBps
Data bit rate:       48 Mbps
Average packet size: 3395.82 bytes
Average packet rate: 1,784 packets/s
SHA256:              192e53424bd3349c2180c0841892c45ca44e6bf4acd5d32aa9d3a6272b0f6981
SHA1:                e1964ca268da42b9372700dc111795ac9b8f44d3
Strict time order:   True
Capture hardware:    11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz (with SSE4.2)
Capture oper-sys:    Linux 6.6.87.2-microsoft-standard-WSL2
Capture application: Dumpcap (Wireshark) 4.4.7
Number of interfaces in file: 1
Interface #0 info:
                     Name = eth0
                     Encapsulation = Ethernet (1 - ether)
                     Capture length = 262144
                     Time precision = nanoseconds (9)
                     Time ticks per second = 1000000000
                     Time resolution = 0×09
                     Operating system = Linux 6.6.87.2-microsoft-standard-WSL2
                     Number of stat entries = 1
                     Number of packets = 67
```

```
capinfos traffic_latest.pcapng
# basic metadata (packets, duration, file size)
```



```
┌──(kali㉿kali)-[~/Desktop/sun/misc]
└─$ tshark -r traffic_latest.pcapng -q -z conv,tcp

================================================================================
TCP Conversations
Filter:<No Filter>
                                                           |       ←      | |       →      | |     Total    |    Relative    |   Duration   |
                                                           | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |              |
172.18.241.1:50756         ←→ 172.18.240.1:5000              20 220 kB         23 1,616 bytes      43 222 kB      0.000000000         0.0069
172.18.241.1:50770         ←→ 172.18.240.1:5000               5 545 bytes       7 559 bytes      12 1,104 bytes   0.014870621         0.0064
172.18.241.1:50778         ←→ 172.18.240.1:5000               5 3,662 bytes       7 551 bytes      12 4,213 bytes     0.030601134         0.0069
================================================================================
```

```
# list TCP conversations to see large transfers
tshark -r traffic_latest.pcapng -q -z conv,tcp
```



```
┌──(kali㉿kali)-[~/Desktop/sun/misc]
└─$ tshark -r traffic_latest.pcapng -Y "http.request" -T fields -e frame.number -e ip.src -e http.host -e http.request.method -e http.request.uri

4       172.18.241.1    172.18.240.1:5000       GET     /image.jpg
47      172.18.241.1    172.18.240.1:5000       GET     /dummy.js
59      172.18.241.1    172.18.240.1:5000       GET     /
```

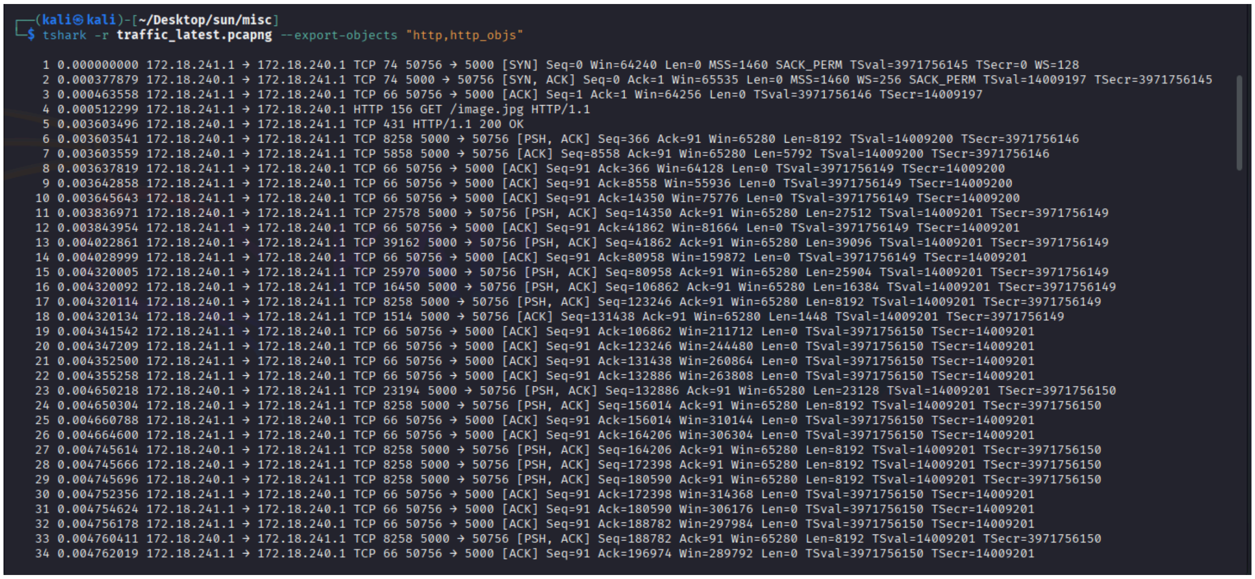
```
# list HTTP requests (frames with GETs)
tshark -r traffic_latest.pcapng -Y "http.request" -T fields -e frame.number -e i
```

```
p.src -e http.host -e http.request.method -e http.request.uri
```

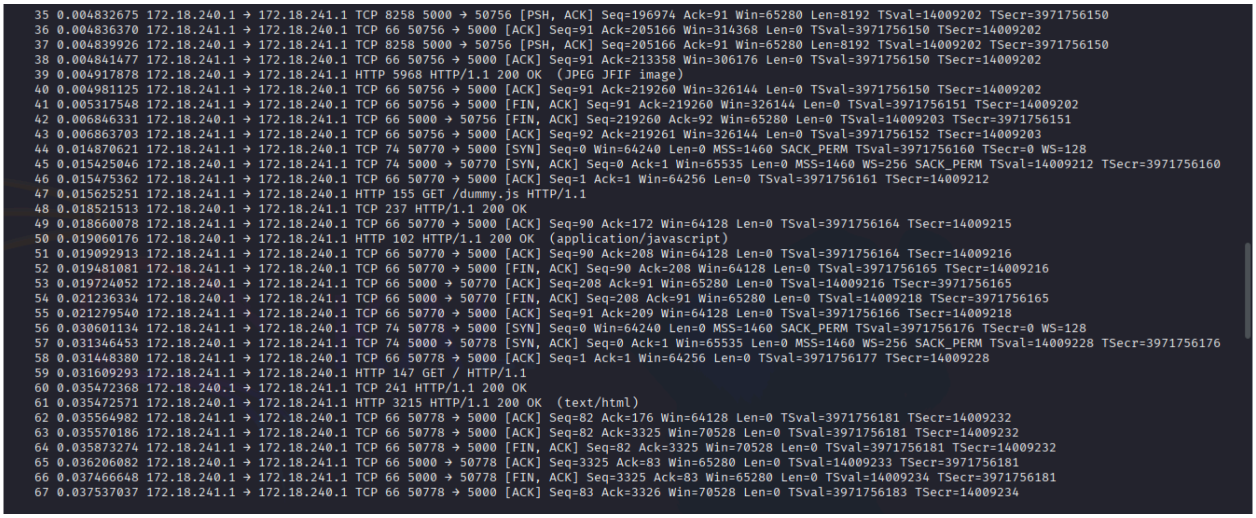
Why: this shows which resources were fetched. In this capture there were three HTTP requests: `/image.jpg`, `/dummy.js`, and `/` (the HTML page).

Screenshot tip: capture the terminal showing the `tshark ... http.request` output (this proves we found the interesting resources).

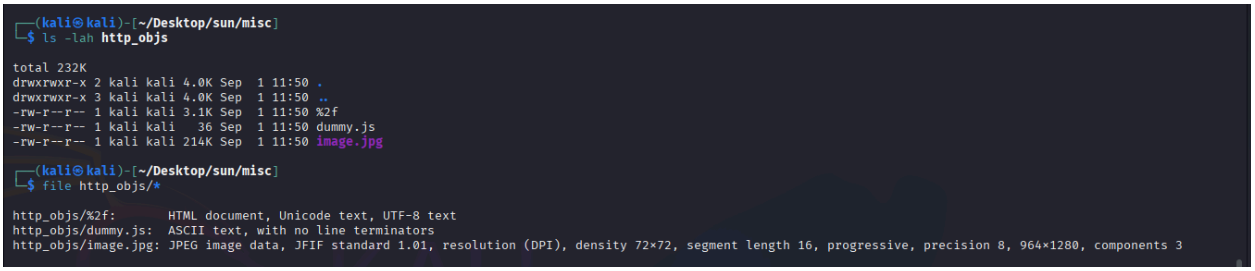
Step 2 — Export HTTP objects



```
┌──(kali㉿kali)-[~/Desktop/sun/misc]
└─$ tshark -r traffic_latest.pcapng --export-objects "http,http_objs"

    1 0.000000000 172.18.241.1 → 172.18.240.1 TCP 74 50756 → 5000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3971756145 TSecr=0 WS=128
    2 0.000377879 172.18.240.1 → 172.18.241.1 TCP 74 5000 → 50756 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM TSval=14009197 TSecr=3971756145
    3 0.000463558 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3971756146 TSecr=14009197
    4 0.000512299 172.18.241.1 → 172.18.240.1 HTTP 156 GET /image.jpg HTTP/1.1
    5 0.003603496 172.18.240.1 → 172.18.241.1 TCP 431 HTTP/1.1 200 OK
    6 0.003603541 172.18.240.1 → 172.18.241.1 TCP 8258 5000 → 50756 [PSH, ACK] Seq=366 Ack=91 Win=65280 Len=8192 TSval=14009200 TSecr=3971756146
    7 0.003603559 172.18.240.1 → 172.18.241.1 TCP 5858 5000 → 50756 [ACK] Seq=8558 Ack=91 Win=65280 Len=5792 TSval=14009200 TSecr=3971756146
    8 0.003637819 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=366 Win=64128 Len=0 TSval=3971756149 TSecr=14009200
    9 0.003642858 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=8558 Win=55936 Len=0 TSval=3971756149 TSecr=14009200
   10 0.003645643 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=14350 Win=75776 Len=0 TSval=3971756149 TSecr=14009200
   11 0.003836971 172.18.240.1 → 172.18.241.1 TCP 27578 5000 → 50756 [PSH, ACK] Seq=14350 Ack=91 Win=65280 Len=27512 TSval=14009201 TSecr=3971756149
   12 0.003843954 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=41862 Win=81664 Len=0 TSval=3971756149 TSecr=14009201
   13 0.004022861 172.18.240.1 → 172.18.241.1 TCP 39162 5000 → 50756 [PSH, ACK] Seq=41862 Ack=91 Win=65280 Len=39096 TSval=14009201 TSecr=3971756149
   14 0.004028999 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=80958 Win=159872 Len=0 TSval=3971756149 TSecr=14009201
   15 0.004320005 172.18.240.1 → 172.18.241.1 TCP 25970 5000 → 50756 [PSH, ACK] Seq=80958 Ack=91 Win=65280 Len=25904 TSval=14009201 TSecr=3971756149
   16 0.004320092 172.18.240.1 → 172.18.241.1 TCP 16450 5000 → 50756 [PSH, ACK] Seq=106862 Ack=91 Win=65280 Len=16384 TSval=14009201 TSecr=3971756149
   17 0.004320114 172.18.240.1 → 172.18.241.1 TCP 8258 5000 → 50756 [PSH, ACK] Seq=123246 Ack=91 Win=65280 Len=8192 TSval=14009201 TSecr=3971756149
   18 0.004320134 172.18.240.1 → 172.18.241.1 TCP 1514 5000 → 50756 [ACK] Seq=131438 Ack=91 Win=65280 Len=1448 TSval=14009201 TSecr=3971756149
   19 0.004341542 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=106862 Win=211712 Len=0 TSval=3971756150 TSecr=14009201
   20 0.004347209 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=123246 Win=244480 Len=0 TSval=3971756150 TSecr=14009201
   21 0.004352500 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=131438 Win=260864 Len=0 TSval=3971756150 TSecr=14009201
   22 0.004355258 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=132886 Win=263808 Len=0 TSval=3971756150 TSecr=14009201
   23 0.004650218 172.18.240.1 → 172.18.241.1 TCP 23194 5000 → 50756 [PSH, ACK] Seq=132886 Ack=91 Win=65280 Len=23128 TSval=14009201 TSecr=3971756150
   24 0.004650304 172.18.240.1 → 172.18.241.1 TCP 8258 5000 → 50756 [PSH, ACK] Seq=156014 Ack=91 Win=65280 Len=8192 TSval=14009201 TSecr=3971756150
   25 0.004660788 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=156014 Win=310144 Len=0 TSval=3971756150 TSecr=14009201
   26 0.004664600 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=164206 Win=306304 Len=0 TSval=3971756150 TSecr=14009201
   27 0.004745614 172.18.240.1 → 172.18.241.1 TCP 8258 5000 → 50756 [PSH, ACK] Seq=164206 Ack=91 Win=65280 Len=8192 TSval=14009201 TSecr=3971756150
   28 0.004745666 172.18.240.1 → 172.18.241.1 TCP 8258 5000 → 50756 [PSH, ACK] Seq=172398 Ack=91 Win=65280 Len=8192 TSval=14009201 TSecr=3971756150
   29 0.004745696 172.18.240.1 → 172.18.241.1 TCP 8258 5000 → 50756 [PSH, ACK] Seq=180590 Ack=91 Win=65280 Len=8192 TSval=14009201 TSecr=3971756150
   30 0.004752356 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=172398 Win=314368 Len=0 TSval=3971756150 TSecr=14009201
   31 0.004754624 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=180590 Win=306176 Len=0 TSval=3971756150 TSecr=14009201
   32 0.004756178 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=188782 Win=297984 Len=0 TSval=3971756150 TSecr=14009201
   33 0.004760411 172.18.240.1 → 172.18.241.1 TCP 8258 5000 → 50756 [PSH, ACK] Seq=188782 Ack=91 Win=65280 Len=8192 TSval=14009201 TSecr=3971756150
   34 0.004762019 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=196974 Win=289792 Len=0 TSval=3971756150 TSecr=14009201
```



```
   35 0.004832675 172.18.240.1 → 172.18.241.1 TCP 8258 5000 → 50756 [PSH, ACK] Seq=196974 Ack=91 Win=65280 Len=8192 TSval=14009202 TSecr=3971756150
   36 0.004836370 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=205166 Win=314368 Len=0 TSval=3971756150 TSecr=14009202
   37 0.004839926 172.18.240.1 → 172.18.241.1 TCP 8258 5000 → 50756 [PSH, ACK] Seq=205166 Ack=91 Win=65280 Len=8192 TSval=14009202 TSecr=3971756150
   38 0.004841477 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=213358 Win=306176 Len=0 TSval=3971756150 TSecr=14009202
   39 0.004917878 172.18.240.1 → 172.18.241.1 HTTP 5968 HTTP/1.1 200 OK  (JPEG JFIF image)
   40 0.004981125 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=91 Ack=219260 Win=326144 Len=0 TSval=3971756150 TSecr=14009202
   41 0.005317548 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [FIN, ACK] Seq=91 Ack=219260 Win=326144 Len=0 TSval=3971756151 TSecr=14009202
   42 0.006846331 172.18.240.1 → 172.18.241.1 TCP 66 5000 → 50756 [FIN, ACK] Seq=219260 Ack=92 Win=65280 Len=0 TSval=14009203 TSecr=3971756151
   43 0.006863703 172.18.241.1 → 172.18.240.1 TCP 66 50756 → 5000 [ACK] Seq=92 Ack=219261 Win=326144 Len=0 TSval=3971756152 TSecr=14009203
   44 0.014870621 172.18.241.1 → 172.18.240.1 TCP 74 50770 → 5000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3971756160 TSecr=0 WS=128
   45 0.015425046 172.18.240.1 → 172.18.241.1 TCP 74 5000 → 50770 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM TSval=14009212 TSecr=3971756160
   46 0.015475362 172.18.241.1 → 172.18.240.1 TCP 66 50770 → 5000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3971756161 TSecr=14009212
   47 0.015625251 172.18.241.1 → 172.18.240.1 HTTP 155 GET /dummy.js HTTP/1.1
   48 0.018521513 172.18.240.1 → 172.18.241.1 TCP 237 HTTP/1.1 200 OK
   49 0.018660078 172.18.241.1 → 172.18.240.1 TCP 66 50770 → 5000 [ACK] Seq=90 Ack=172 Win=64128 Len=0 TSval=3971756164 TSecr=14009215
   50 0.019060176 172.18.240.1 → 172.18.241.1 HTTP 102 HTTP/1.1 200 OK  (application/javascript)
   51 0.019092913 172.18.241.1 → 172.18.240.1 TCP 66 50770 → 5000 [ACK] Seq=90 Ack=208 Win=64128 Len=0 TSval=3971756164 TSecr=14009216
   52 0.019481081 172.18.241.1 → 172.18.240.1 TCP 66 50770 → 5000 [FIN, ACK] Seq=90 Ack=208 Win=64128 Len=0 TSval=3971756165 TSecr=14009216
   53 0.019724052 172.18.240.1 → 172.18.241.1 TCP 66 5000 → 50770 [ACK] Seq=208 Ack=91 Win=65280 Len=0 TSval=14009216 TSecr=3971756165
   54 0.021236334 172.18.240.1 → 172.18.241.1 TCP 66 5000 → 50770 [FIN, ACK] Seq=208 Ack=91 Win=65280 Len=0 TSval=14009218 TSecr=3971756165
   55 0.021279540 172.18.241.1 → 172.18.240.1 TCP 66 50770 → 5000 [ACK] Seq=91 Ack=209 Win=64128 Len=0 TSval=3971756166 TSecr=14009218
   56 0.030601134 172.18.241.1 → 172.18.240.1 TCP 74 50778 → 5000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3971756176 TSecr=0 WS=128
   57 0.031346453 172.18.240.1 → 172.18.241.1 TCP 74 5000 → 50778 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM TSval=14009228 TSecr=3971756176
   58 0.031448380 172.18.241.1 → 172.18.240.1 TCP 66 50778 → 5000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3971756177 TSecr=14009228
   59 0.031609293 172.18.241.1 → 172.18.240.1 HTTP 147 GET / HTTP/1.1
   60 0.035472368 172.18.240.1 → 172.18.241.1 TCP 241 HTTP/1.1 200 OK
   61 0.035472571 172.18.240.1 → 172.18.241.1 HTTP 3215 HTTP/1.1 200 OK  (text/html)
   62 0.035564982 172.18.241.1 → 172.18.240.1 TCP 66 50778 → 5000 [ACK] Seq=82 Ack=176 Win=64128 Len=0 TSval=3971756181 TSecr=14009232
   63 0.035570186 172.18.241.1 → 172.18.240.1 TCP 66 50778 → 5000 [ACK] Seq=82 Ack=3325 Win=70528 Len=0 TSval=3971756181 TSecr=14009232
   64 0.035873274 172.18.241.1 → 172.18.240.1 TCP 66 50778 → 5000 [FIN, ACK] Seq=82 Ack=3325 Win=70528 Len=0 TSval=3971756181 TSecr=14009232
   65 0.036206082 172.18.240.1 → 172.18.241.1 TCP 66 5000 → 50778 [ACK] Seq=3325 Ack=83 Win=65280 Len=0 TSval=14009233 TSecr=3971756181
   66 0.037466648 172.18.240.1 → 172.18.241.1 TCP 66 5000 → 50778 [FIN, ACK] Seq=3325 Ack=83 Win=65280 Len=0 TSval=14009234 TSecr=3971756181
   67 0.037537037 172.18.241.1 → 172.18.240.1 TCP 66 50778 → 5000 [ACK] Seq=83 Ack=3326 Win=70528 Len=0 TSval=3971756183 TSecr=14009234
```



```
┌──(kali㉿kali)-[~/Desktop/sun/misc]
└─$ ls -lah http_objs

total 232K
drwxrwxr-x 2 kali kali 4.0K Sep  1 11:50 .
drwxrwxr-x 3 kali kali 4.0K Sep  1 11:50 ..
-rw-r--r-- 1 kali kali 3.1K Sep  1 11:50 %2f
-rw-r--r-- 1 kali kali   36 Sep  1 11:50 dummy.js
-rw-r--r-- 1 kali kali 214K Sep  1 11:50 image.jpg

┌──(kali㉿kali)-[~/Desktop/sun/misc]
└─$ file http_objs/*

http_objs/%2f:       HTML document, Unicode text, UTF-8 text
http_objs/dummy.js:  ASCII text, with no line terminators
http_objs/image.jpg: JPEG image data, JFIF standard 1.01, resolution (DPI), density 72×72, segment length 16, progressive, precision 8, 964×1280, components 3
```

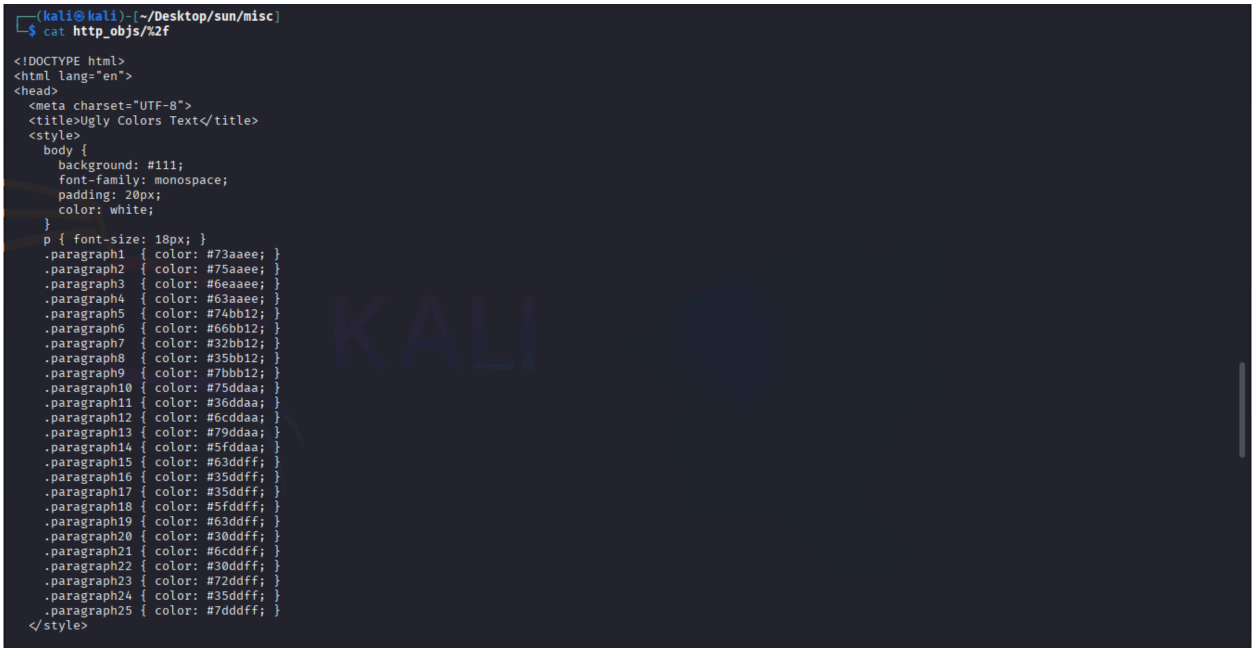
Export all HTTP-delivered files so we can inspect them locally.

```
mkdir -p http_objs
tshark -r traffic_latest.pcapng --export-objects "http,http_objs"
ls -lah http_objs
file http_objs/*
```

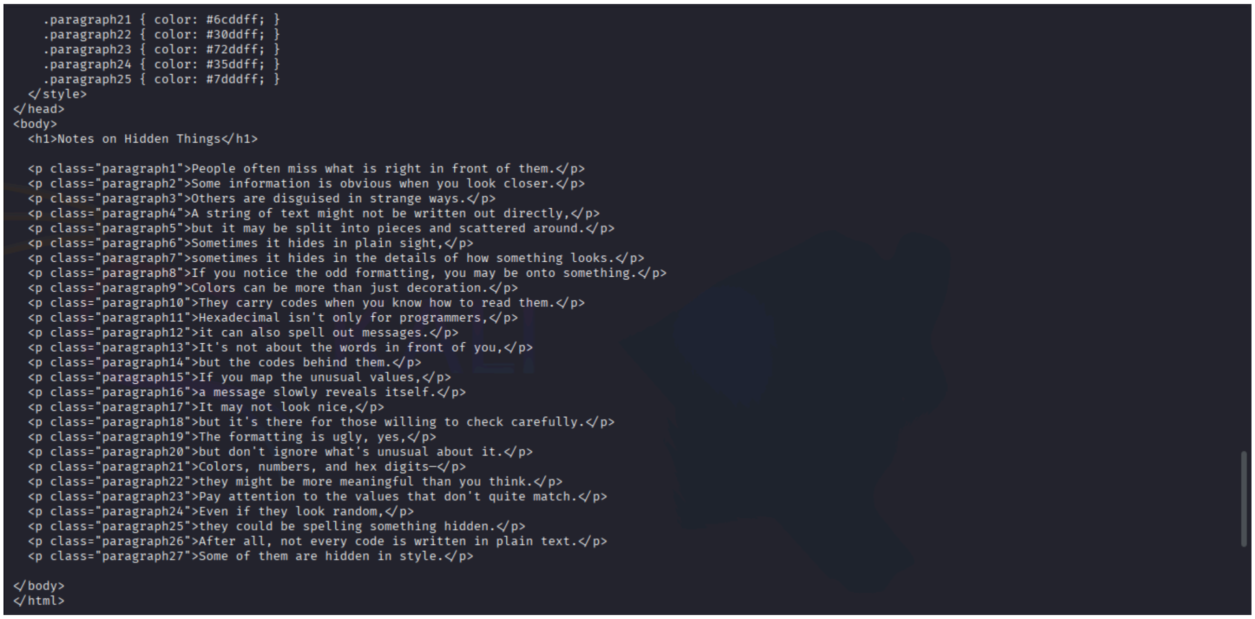
Expected results: you should see three files (or similarly named): the HTML from `/` (often named `%2f`), `dummy.js`, and `image.jpg`.

Screenshot tip: take a screenshot of `ls -lah http_objs` and `file http_objs/*`.

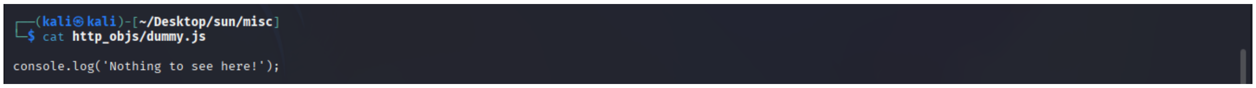
Step 3 — Inspect the HTML & JS



```
┌──(kali㉿kali)-[~/Desktop/sun/misc]
└─$ cat http_objs/%2f

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Ugly Colors Text</title>
  <style>
    body {
      background: #111;
      font-family: monospace;
      padding: 20px;
      color: white;
    }
    p { font-size: 18px; }
    .paragraph1  { color: #73aaee; }
    .paragraph2  { color: #75aaee; }
    .paragraph3  { color: #6eaaee; }
    .paragraph4  { color: #63aaee; }
    .paragraph5  { color: #74bb12; }
    .paragraph6  { color: #66bb12; }
    .paragraph7  { color: #32bb12; }
    .paragraph8  { color: #35bb12; }
    .paragraph9  { color: #7bbb12; }
    .paragraph10 { color: #75ddaa; }
    .paragraph11 { color: #36ddaa; }
    .paragraph12 { color: #6cddaa; }
    .paragraph13 { color: #79ddaa; }
    .paragraph14 { color: #5fddaa; }
    .paragraph15 { color: #63ddff; }
    .paragraph16 { color: #35ddff; }
    .paragraph17 { color: #35ddff; }
    .paragraph18 { color: #5fddff; }
    .paragraph19 { color: #63ddff; }
    .paragraph20 { color: #30ddff; }
    .paragraph21 { color: #6cddff; }
    .paragraph22 { color: #30ddff; }
    .paragraph23 { color: #72ddff; }
    .paragraph24 { color: #35ddff; }
    .paragraph25 { color: #7dddff; }
  </style>
```



```
    .paragraph21 { color: #6cddff; }
    .paragraph22 { color: #30ddff; }
    .paragraph23 { color: #72ddff; }
    .paragraph24 { color: #35ddff; }
    .paragraph25 { color: #7dddff; }
  </style>
</head>
<body>
  <h1>Notes on Hidden Things</h1>

  <p class="paragraph1">People often miss what is right in front of them.</p>
  <p class="paragraph2">Some information is obvious when you look closer.</p>
  <p class="paragraph3">Others are disguised in strange ways.</p>
  <p class="paragraph4">A string of text might not be written out directly,</p>
  <p class="paragraph5">but it may be split into pieces and scattered around.</p>
  <p class="paragraph6">Sometimes it hides in plain sight,</p>
  <p class="paragraph7">sometimes it hides in the details of how something looks.</p>
  <p class="paragraph8">If you notice the odd formatting, you may be onto something.</p>
  <p class="paragraph9">Colors can be more than just decoration.</p>
  <p class="paragraph10">They carry codes when you know how to read them.</p>
  <p class="paragraph11">Hexadecimal isn't only for programmers,</p>
  <p class="paragraph12">it can also spell out messages.</p>
  <p class="paragraph13">It's not about the words in front of you,</p>
  <p class="paragraph14">but the codes behind them.</p>
  <p class="paragraph15">If you map the unusual values,</p>
  <p class="paragraph16">a message slowly reveals itself.</p>
  <p class="paragraph17">It may not look nice,</p>
  <p class="paragraph18">but it's there for those willing to check carefully.</p>
  <p class="paragraph19">The formatting is ugly, yes,</p>
  <p class="paragraph20">but don't ignore what's unusual about it.</p>
  <p class="paragraph21">Colors, numbers, and hex digits—</p>
  <p class="paragraph22">they might be more meaningful than you think.</p>
  <p class="paragraph23">Pay attention to the values that don't quite match.</p>
  <p class="paragraph24">Even if they look random,</p>
  <p class="paragraph25">they could be spelling something hidden.</p>
  <p class="paragraph26">After all, not every code is written in plain text.</p>
  <p class="paragraph27">Some of them are hidden in style.</p>

</body>
</html>
```



```
┌──(kali㉿kali)-[~/Desktop/sun/misc]
└─$ cat http_objs/dummy.js

console.log('Nothing to see here!');
```

Open the exported HTML (the `/` resource). It often contains hints.

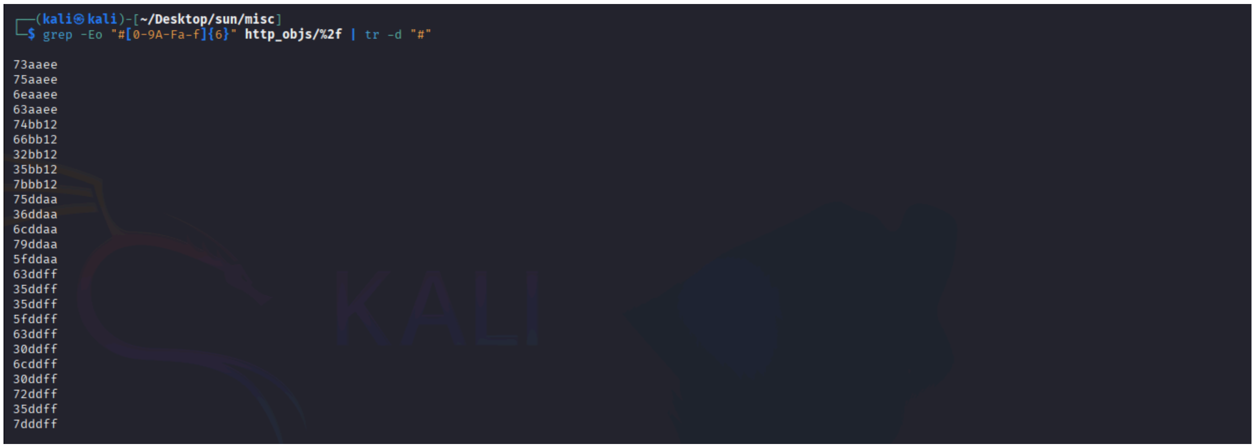
```
cat http_objs/%2f # or cat the filename exported for '/'
cat http_objs/dummy.js
```

What to look for:

- Comments or text mentioning "colors", "hex", "palette".
- Lots of `#RRGGBB` CSS color values.

In this challenge the HTML had many `.paragraphN { color: #RRGGBB; }` entries and text hinting that colors hide a message.

Extract all hex color values from the HTML



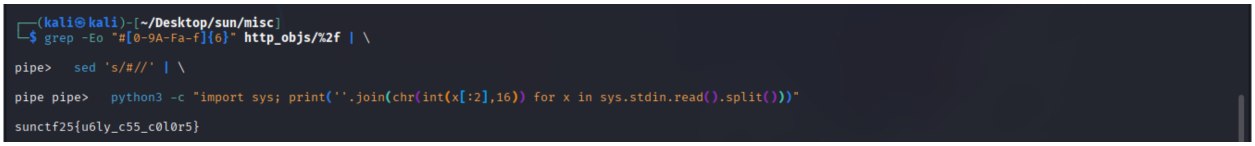
```
┌──(kali㉿kali)-[~/Desktop/sun/misc]
└─$ grep -Eo "#[0-9A-Fa-f]{6}" http_objs/%2f | tr -d "#"

73aaee
75aaee
6eaaee
63aaee
74bb12
66bb12
32bb12
35bb12
7bbb12
75ddaa
36ddaa
6cddaa
79ddaa
5fddaa
63ddff
35ddff
35ddff
5fddff
63ddff
30ddff
6cddff
30ddff
72ddff
35ddff
7dddff
```

```
# list the hex color codes (with leading #)
grep -Eo "#[0-9A-Fa-f]{6}" http_objs/%2f
```

You will see many colors like `73aaee`, `75aaee`, `6eaaee`, etc.

Why this matters: the CSS uses repeated suffixes (e.g. `aaee`, `bb12`, `ddaa`, `ddff`) so the only byte that really changes between entries is the **first pair** (`73`, `75`, `6e`, ...). That first pair is the red channel and is being used as ASCII codes.

Decode the red channel (first byte) into ASCII



```
┌──(kali㉿kali)-[~/Desktop/sun/misc]
└─$ grep -Eo "#[0-9A-Fa-f]{6}" http_objs/%2f | \

pipe>   sed 's/#//' | \

pipe pipe>   python3 -c "import sys; print(''.join(chr(int(x[:2],16)) for x in sys.stdin.read().split()))"

sunctf25{u6ly_c55_c0l0r5}
```

We take the first two hex digits of each `RRGGBB` value, convert from hex to an integer, then to a character. A compact one-liner (bash + python) that was used:

```
grep -Eo "#[0-9A-Fa-f]{6}" http_objs/%2f | sed 's/#//' | \
python3 -c "import sys; print(''.join(chr(int(x[:2],16)) for x in sys.stdin.read().split()))"
```

- `grep` extracts colors like `#73aaee`.

- `sed 's/#//'` removes the leading `#` so each line is `73aaee`.
- Python reads each hex string (`73aaee`) and converts `x[:2]` (`'73'`) from hex to an int then to its ASCII character.

```
sunctf25{u6ly_c55_c0l0r5}
```

we found the flag!