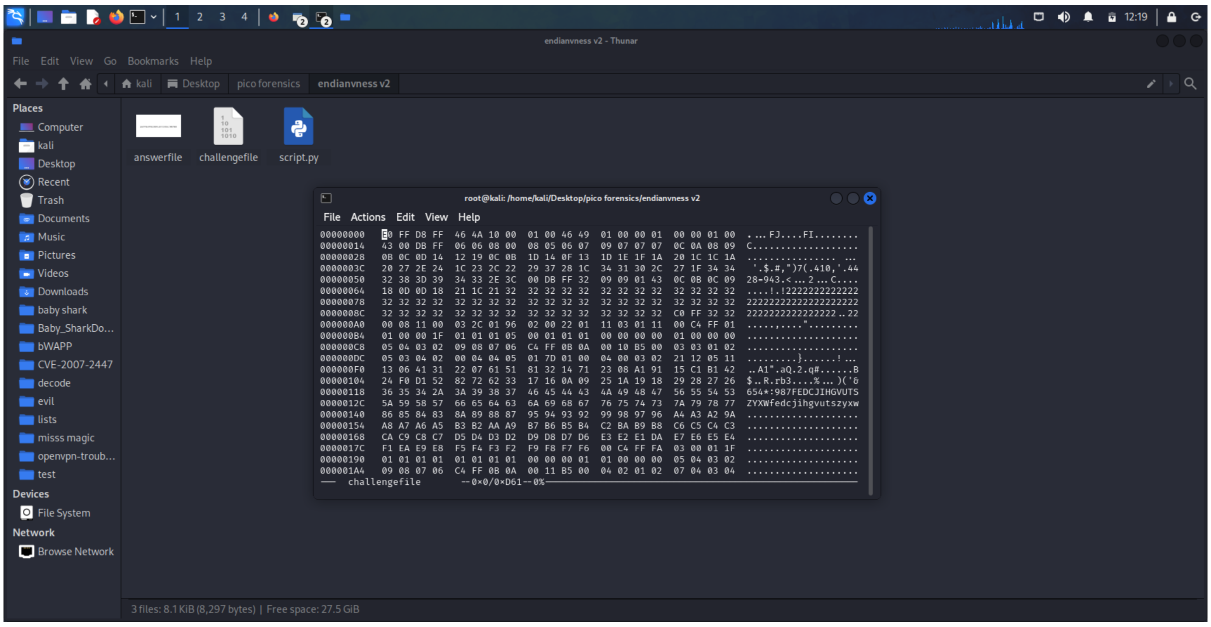
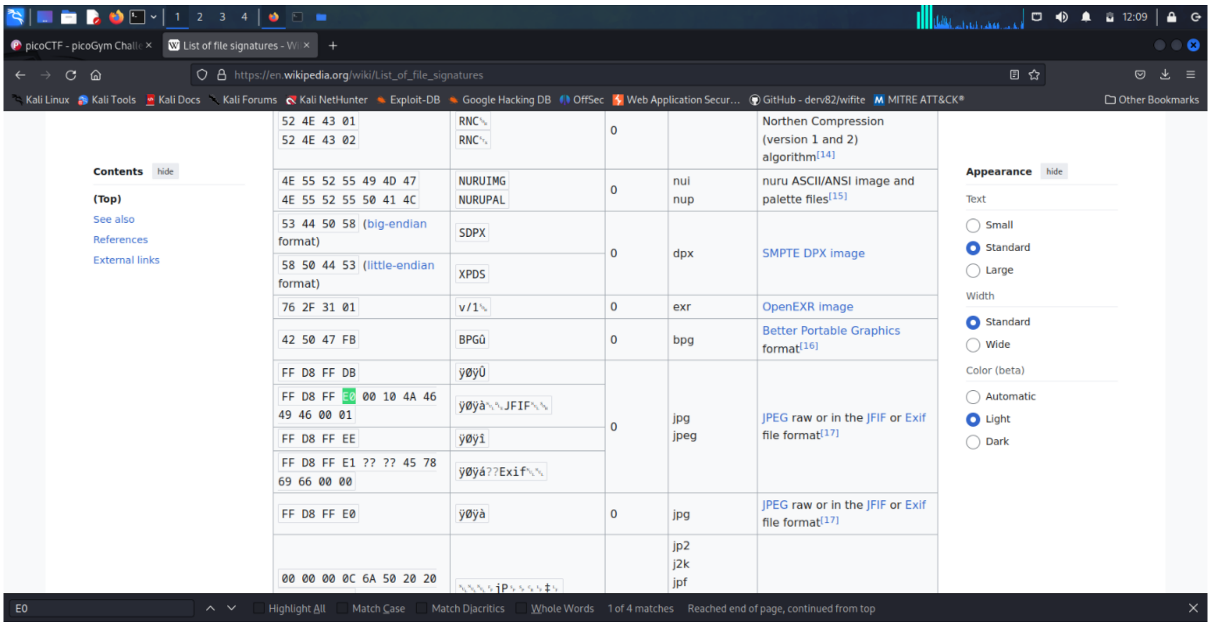

root@kali: /home/kali/Desktop/pico forensics/endianvness v2

```
00000000   E0 FF D8 FF  46 4A 10 00  01 00 46 49  01 00 00 01  00 00 01 00  ....FJ....FI........
00000014   43 00 DB FF  06 06 08 00  08 05 06 07  09 07 07 07  0C 0A 08 09  C...................
00000028   0B 0C 0D 14  12 19 0C 0B  1D 14 0F 13  1D 1E 1F 1A  20 1C 1C 1A  ................  ...
0000003C   20 27 2E 24  1C 23 2C 22  29 37 28 1C  34 31 30 2C  27 1F 34 34   '.$.#,")7(.410,'.44
00000050   32 38 3D 39  34 33 2E 3C  00 DB FF 32  09 09 01 43  0C 0B 0C 09  28=943.<...2...C....
00000064   18 0D 0D 18  21 1C 21 32  32 32 32 32  32 32 32 32  32 32 32 32  ....!.!2222222222222
00000078   32 32 32 32  32 32 32 32  32 32 32 32  32 32 32 32  32 32 32 32  22222222222222222222
0000008C   32 32 32 32  32 32 32 32  32 32 32 32  32 32 32 32  C0 FF 32 32  2222222222222222..22
000000A0   00 08 11 00  03 2C 01 96  02 00 22 01  11 03 01 11  00 C4 FF 01  .....,....".........
000000B4   01 00 00 1F  01 01 01 05  00 01 01 01  00 00 00 00  01 00 00 00  ....................
000000C8   05 04 03 02  09 08 07 06  C4 FF 0B 0A  00 10 B5 00  03 03 01 02  ....................
000000DC   05 03 04 02  00 04 04 05  01 7D 01 00  04 00 03 02  21 12 05 11  .........}......!...
000000F0   13 06 41 31  22 07 61 51  81 32 14 71  23 08 A1 91  15 C1 B1 42  ..A1".aQ.2.q#......B
00000104   24 F0 D1 52  82 72 62 33  17 16 0A 09  25 1A 19 18  29 28 27 26  $..R.rb3....%...)('&
00000118   36 35 34 2A  3A 39 38 37  46 45 44 43  4A 49 48 47  56 55 54 53  654*:987FEDCJIHGVUTS
0000012C   5A 59 58 57  66 65 64 63  6A 69 68 67  76 75 74 73  7A 79 78 77  ZYXWfedcjihgvutszyxw
00000140   86 85 84 83  8A 89 88 87  95 94 93 92  99 98 97 96  A4 A3 A2 9A  ....................
00000154   A8 A7 A6 A5  B3 B2 AA A9  B7 B6 B5 B4  C2 BA B9 B8  C6 C5 C4 C3  ....................
00000168   CA C9 C8 C7  D5 D4 D3 D2  D9 D8 D7 D6  E3 E2 E1 DA  E7 E6 E5 E4  ....................
0000017C   F1 EA E9 E8  F5 F4 F3 F2  F9 F8 F7 F6  00 C4 FF FA  03 00 01 1F  ....................
00000190   01 01 01 01  01 01 01 01  00 00 00 01  01 00 00 00  05 04 03 02  ....................
000001A4   09 08 07 06  C4 FF 0B 0A  00 11 B5 00  04 02 01 02  07 04 03 04  ....................
---  challengefile       --0x0/0xD61--0%
```

3 files: 8.1 KiB (8,297 bytes) | Free space: 27.5 GiB

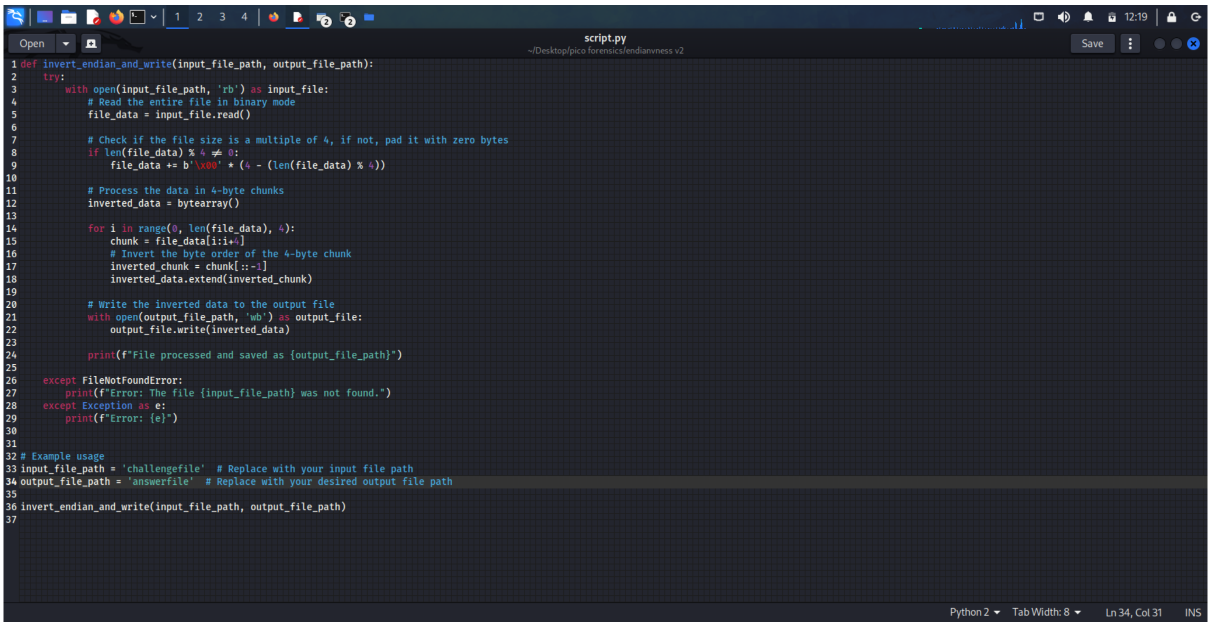
First open it with hexedit take the first symbol and search it online in the magical bytes section


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

52 4E 43 01 52 4E 43 02	RNC␁ RNC␂	0		Northen Compression (version 1 and 2) algorithm[14]
4E 55 52 55 49 4D 47 4E 55 52 55 50 41 4C	NURUIMG NURUPAL	0	nui nup	nuru ASCII/ANSI image and palette files[15]
53 44 50 58 (big-endian format)	SDPX	0	dpx	SMPTE DPX image
58 50 44 53 (little-endian format)	XPDS			
76 2F 31 01	v/1␁	0	exr	OpenEXR image
42 50 47 FB	BPGû	0	bpg	Better Portable Graphics format[16]
FF D8 FF DB	ÿØÿÛ	0	jpg jpeg	JPEG raw or in the JFIF or Exif file format[17]
FF D8 FF E0 00 10 4A 46 49 46 00 01	ÿØÿà␀␐JFIF␀␁			
FF D8 FF EE	ÿØÿî			
FF D8 FF E1 ?? ?? 45 78 69 66 00 00	ÿØÿá??Exif␀␀			
FF D8 FF E0	ÿØÿà	0	jpg	JPEG raw or in the JFIF or Exif file format[17]
00 00 00 0C 6A 50 20 20	␀␀␀␌jP␠␠␍␊‡␊		jp2 j2k jpf	

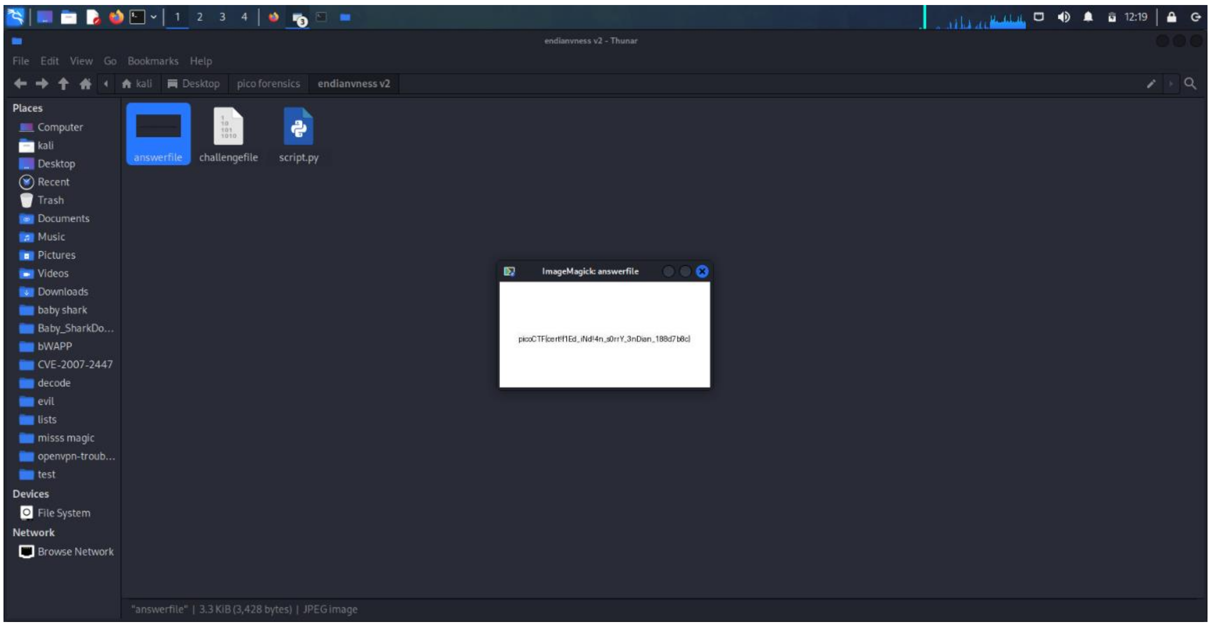
E0 — Highlight All, Match Case, Match Diacritics, Whole Words, 1 of 4 matches, Reached end of page, continued from top

You will find it in the magic bytes section which is jpeg now if you see it close evey 4 bytes / symbols are inverted or reverse so write a program to del with it



```python
def invert_endian_and_write(input_file_path, output_file_path):
    try:
        with open(input_file_path, 'rb') as input_file:
            # Read the entire file in binary mode
            file_data = input_file.read()

            # Check if the file size is a multiple of 4, if not, pad it with zero bytes
            if len(file_data) % 4 ≠ 0:
                file_data += b'\x00' * (4 - (len(file_data) % 4))

            # Process the data in 4-byte chunks
            inverted_data = bytearray()

            for i in range(0, len(file_data), 4):
                chunk = file_data[i:i+4]
                # Invert the byte order of the 4-byte chunk
                inverted_chunk = chunk[::-1]
                inverted_data.extend(inverted_chunk)

            # Write the inverted data to the output file
            with open(output_file_path, 'wb') as output_file:
                output_file.write(inverted_data)

            print(f"File processed and saved as {output_file_path}")

    except FileNotFoundError:
        print(f"Error: The file {input_file_path} was not found.")
    except Exception as e:
        print(f"Error: {e}")


# Example usage
input_file_path = 'challengefile'  # Replace with your input file path
output_file_path = 'answerfile'  # Replace with your desired output file path

invert_endian_and_write(input_file_path, output_file_path)
```

```python
def invert_endian_and_write(input_file_path, output_file_path):
  try:
    with open(input_file_path, 'rb') as input_file:
      # Read the entire file in binary mode
      file_data = input_file.read()

      # Check if the file size is a multiple of 4, if not, pad it with zero bytes
      if len(file_data) % 4 != 0:
        file_data += b'\x00' * (4 - (len(file_data) % 4))

      # Process the data in 4-byte chunks
      inverted_data = bytearray()

      for i in range(0, len(file_data), 4):
        chunk = file_data[i:i+4]
        # Invert the byte order of the 4-byte chunk
        inverted_chunk = chunk[::-1]
        inverted_data.extend(inverted_chunk)
      # Write the inverted data to the output file
      with open(output_file_path, 'wb') as output_file:
        output_file.write(inverted_data)

      print(f"File processed and saved as {output_file_path}")

  except FileNotFoundError:
    print(f"Error: The file {input_file_path} was not found.")
  except Exception as e:
    print(f"Error: {e}")


# Example usage
input_file_path = 'challengefile'  # Replace with your input file path
output_file_path = 'answerfile'  # Replace with your desired output file path

invert_endian_and_write(input_file_path, output_file_path)
```



```
picoCTF{cert!f1Ed_iNd!4n_s0rrY_3nDian_188d7b8c}
```

"answerfile" | 3.3 KiB (3,428 bytes) | JPEG image