

CS 798: Digital Forensics and Incident Response

Lecture 5 - File Forensics

Diogo Barradas

Winter 2025

University of Waterloo

Recall evidence collection...







































































































1. A primer on file formats
2. Syntactic analysis of files
3. Semantic analysis of files

Shifting our focus to Analysis



- How to analyze evidence from computers and networks?

Files! (*bazillions* of them)

vector	 AI	 EPS	 CDR	 SVG	 WMF	 ART	 CGM	 EMF	 VSD	 PS
image	 TIF	 PSD	 JPG	 GIF	 PNG	 BMP	 TGA	 ICO	 HDR	 RAW
text	 PDF	 TXT	 DOC	 RTF	 ODT	 SUB	 UNX	 ORT	 CHM	 WPD
audio	 MP3	 WAV	 AAC	 FLAC	 CDA	 MIDI	 RMF	 OGG	 VOC	 WMA
video	 QT	 FLV	 MP4	 AVI	 3GP	 MPG	 MKV	 MOV	 ASF	 VOB
ebook	 FB2	 DJVU	 MOBI	 EPUB	 IW4	 PRC	 CHM	 TCR	 EBK	 AZW
archive	 ZIP	 RAR	 ISO	 JAR	 TAR	 ACE	 LZH	 ARJ	 CAB	 ZOO
internet	 JS	 HTM	 CSS	 XML	 MHT	 PSP	 EML	 PHP	 JAVA	 PY
other	 INI	 SYS	 KEY	 PPT	 NFO	 XLS	 CSV	 CAB	 MDB	 COM
bonus	 SWF	 EXE	 DAT	 HLP	 DLL	 FAQ	 RSS	 FON	 TTF	 OTF

A primer on file formats

File formats and file forensics

- A **file format** is a standard way that information is encoded for storage in a computer file
- In **file forensics**, we aim at recovering information from files by decoding their raw data from storage
- There are two broad file format families:
 - Text files: Essential to determine the text encoding scheme and structure (if any)
 - Binary files: Essential to determine the file format (including endianness)

Text files

- Raw bytes represent characters using an encoding
 - Some are printable, others non-printable (e.g., linefeed, end of file)
 - ASCII and Unicode are the most common encoding schemes
- ASCII is the common code text representation
 - American Standard Code for Information Interchange
 - Proposed by ANSI in 1963, finalized in 1968
- Assigns a numerical value to characters in American English
 - Originally, 1 character was encoded in 7 bits
 - But limited, e.g., for European languages or mathematical symbols
- Extended ASCII Character Set
 - Extended ASCII uses 8-bits to encode a single character

Standard ASCII The first 32 characters are control codes.												Extended ASCII (80s)											
DH		DH		DH		DH		DH		DH		DH		DH		DH		DH		DH		DH	
00	Null	33	21	81	Q	128	0	174	AE	220	DC	129	1	175	AF	221	DD	130	2	176	AP	222	DE
1	Start of heading	34	22	82	R	129	81	177	AG	223	DF	131	82	178	AH	224	EE	132	83	179	AI	225	E1
2	Start of text	35	23	83	S	130	82	176	AO	226	E2	131	83	177	BI	227	E3	132	84	178	B2	228	E4
3	End of text	36	24	84	T	131	83	177	BI	229	E5	132	84	178	B2	230	E6	133	85	179	BJ	231	E7
4	End of transmit	37	25	85	U	132	84	178	B2	232	E8	133	85	179	BJ	233	E9	134	86	180	B3	234	EA
5	Enquiry	38	26	86	V	133	85	179	BJ	235	EB	134	86	180	B3	236	EC	135	87	181	B4	237	ED
6	Acknowledge	39	27	87	W	134	86	180	B3	238	EE	135	87	181	B4	239	EF	136	88	182	B5	240	F0
7	Audible bell	40	28	88	X	135	87	181	B5	241	F1	136	88	182	B5	242	F2	137	89	183	B6	243	F3
8	Backspace	41	29	89	Y	136	88	182	B5	244	F4	137	89	183	B6	245	F5	138	90	184	B7	246	F6
9	Horizontal tab	42	30	90	Z	137	89	183	B6	247	F7	138	90	184	B7	248	F8	139	91	185	B8	249	F9
10	Line feed	43	31	91	[138	90	184	B7	250	FA	139	91	185	B8	251	FB	140	92	186	B9	252	FC
11	Vertical tab	44	32	92	\	139	91	185	B8	253	FD	140	92	186	B9	254	FE	141	93	187	BA	255	FF
12	Form feed	45	20	93]	140	92	186	B9	256	100	141	93	187	BA	257	101	142	94	188	BB	258	102
13	Carriage return	46	2E	94	^	141	93	187	BA	259	103	142	94	188	BB	260	104	143	95	189	BC	261	105
14	Shift out	47	2F	95	_	142	94	188	BB	262	106	143	95	189	BC	263	107	144	96	190	BD	264	108
15	Shift in	48	30	96	`	143	95	189	BC	265	109	144	96	190	BD	266	110	145	97	191	BE	267	111
16	Data link escape	49	31	97	a	144	96	190	BD	268	112	145	97	191	BE	269	113	146	98	192	BF	270	114
17	Device control 1	50	32	98	b	145	97	191	BE	271	115	146	98	192	BF	272	116	147	99	193	C0	273	117
18	Device control 2	51	33	99	c	146	98	192	BF	274	118	147	99	193	C1	275	119	148	100	194	C2	276	120
19	Device control 3	52	34	100	d	147	99	193	C1	277	121	148	100	194	C2	278	122	149	101	195	C3	279	123
20	Device control 4	53	35	101	e	148	100	194	C2	280	124	149	101	195	C3	281	125	150	102	196	C4	282	126
21	Neg. acknowledge	54	36	102	f	149	101	195	C3	283	127	150	102	196	C4	284	128	151	103	197	C5	285	129
22	Synchronous idle	55	37	103	g	150	102	196	C4	286	130	151	103	197	C5	287	131	152	104	198	C6	288	132
23	End trans. block	56	38	104	h	151	103	197	C5	289	133	152	104	198	C6	290	134	153	105	199	C7	291	135
24	Cancel	57	39	105	i	152	104	198	C6	292	136	153	105	199	C7	293	137	154	106	200	C8	294	138
25	End of medium	58	3A	106	j	153	105	199	C7	295	139	154	106	200	C8	296	140	155	107	201	C9	297	141
26	Substitution	59	3B	107	k	154	106	200	C8	298	142	155	107	201	C9	299	143	156	108	202	CA	300	144
27	Escape	60	3C	108	l	155	107	201	C9	301	145	156	108	202	CA	302	146	157	109	203	CB	303	147
28	File separator	61	3D	109	m	156	108	202	CA	304	148	157	109	203	CB	305	149	158	110	204	CC	306	150
29	Group separator	62	3E	110	n	157	109	203	CB	307	151	158	110	204	CC	308	152	159	111	205	CD	309	153
30	Record separator	63	3F	111	o	158	110	204	CC	310	154	159	111	205	CD	311	155	160	112	206	CE	312	156
31	Unit separator	64	40	112	p	159	111	205	CD	313	156	160	112	206	CE	314	157	161	113	207	CF	315	158
32	Blank space	65	41	113	q	160	112	206	CE	316	158	161	113	207	CF	317	159	162	114	208	CG	318	159
		66	42	114	r	161	113	207	CF	319	160	162	114	208	CG	320	161	163	115	209	CH	321	162
		67	43	115	s	162	114	208	CG	322	162	163	115	209	CH	323	163	164	116	210	CI	324	164
		68	44	116	t	163	115	209	CH	325	164	164	116	210	CI	326	165	165	117	211	CJ	327	165
		69	45	117	u	164	116	210	CI	328	166	165	117	211	CJ	329	166	166	118	212	CK	330	166
		70	46	118	v	165	117	211	CJ	331	167	166	118	212	CK	332	167	167	119	213	CL	333	167
		71	47	119	w	166	118	212	CK	334	168	167	119	213	CL	335	168	168	120	214	CM	336	168
		72	48	120	x	167	119	213	CL	337	169	168	120	214	CM	338	169	169	121	215	CN	339	169
		73	49	121	y	168	120	214	CM	340	170	169	121	215	CN	341	170	170	122	216	CO	342	170
		74	4A	122	z	169	121	215	CN	343	171	170	122	216	CO	344	171	171	123	217	CP	345	171
		75	4B	123	[170	122	216	CO	346	172	171	123	217	CP	347	172	172	124	218	CQ	348	172
		76	4C	124]	171	123	217	CP	349	173	172	124	218	CQ	350	173	173	125	219	CR	351	173
		77	4D	125	^	172	124	218	CQ	352	174	173	125	219	CR	353	174	174	126	220	CS	354	174
		78	4E	126	_	173	125	219	CR	355	175	174	126	220	CS	356	175	175	127	221	CT	357	175
		79	4F	127	`	174	126	220	CS	358	176	175	127	221	CT	359	176	176	128	222	CU	360	176
		80	50	8F	F																		

- ASCII is nice and simple if you use American English, but it is quite limited for the rest of the world
 - Their native symbols cannot be represented
- Unicode helps solve this problem by using more than 1 byte to store the numerical version of a symbol
- The version 4.0 Unicode standard supports over 96,000 characters, which requires 4-bytes per character instead of the 1 byte that ASCII requires

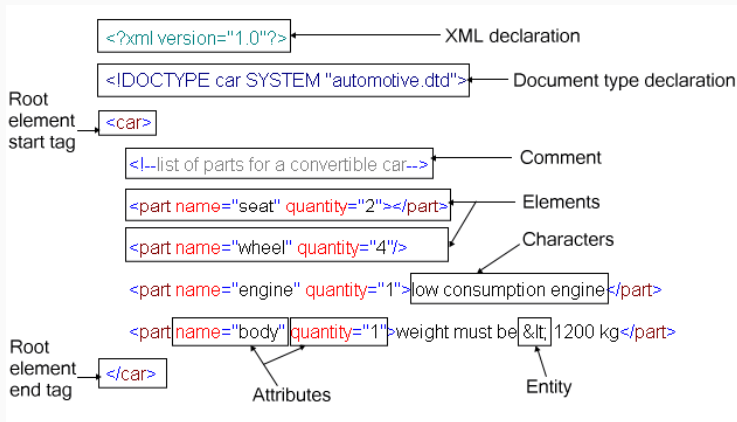
Unicode encodings – UTF-32, UTF-16, UTF-8

- There are three ways of storing a Unicode character:
 - UTF-32: uses a 4-byte value for each character
 - UTF-16: most used chars. in 2-byte value, lesser-used 4-bytes
 - UTF-8: uses 1, 2, or 4 bytes (most frequently used in 1 byte)
- Tradeoff between number of characters that can be represented, and space and processing efficiency
- UTF-8 is frequently used because it has the least amount of wasted space and because ASCII is a subset of UTF-8

Character	UTF-16	UTF-8
A	0041	41
c	0063	63
Ö	00F6	C3 B6
☞	4E9C	E4 BA 9C
♫	D834 DD1E	F0 9D 84 9E

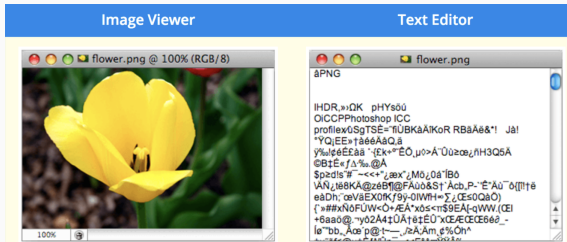
Text files can have structure

- Example: XML file



Binary files

- In binary files, bytes represent **custom data**
 - Binary files may contain both textual and custom binary data
- Binary file formats may include **multiple** types of data in the same file, such as image, video, and audio data
 - This data can be **interpreted** by supporting programs, but will show up as garbled text in a text editor



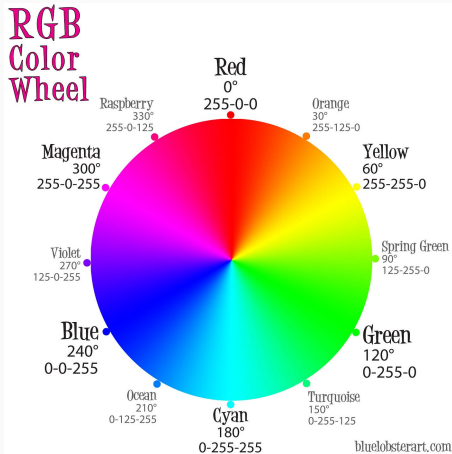
BMP image files

- BMP image: array of pixels, each encodes a specific color
 - Binary files may contain both textual and custom binary data
- The resolution indicates width and height of image
 - Ex: 750×491



Pixel color encoding

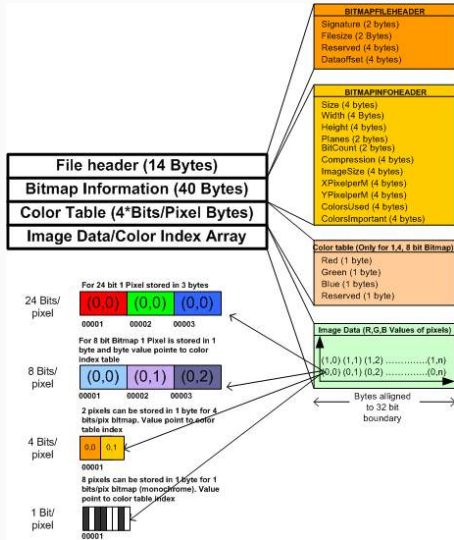
- 24-bit RGB image files
 - Each pixel encoded by 3 byte values for red, green, and blue



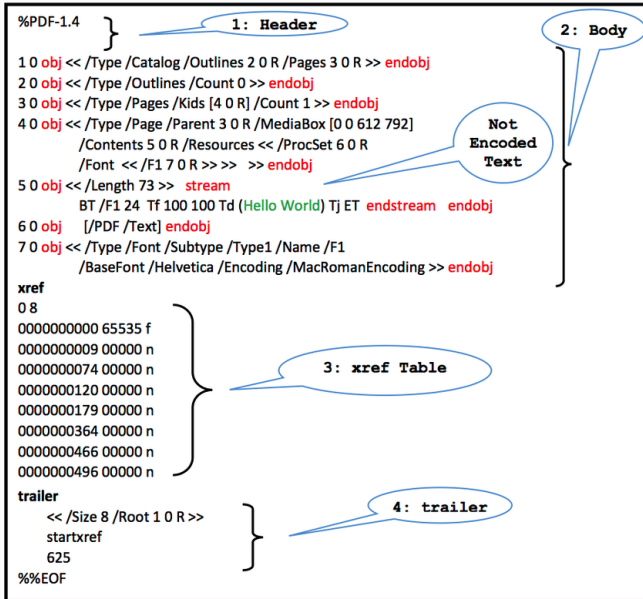
BMP header format

- Signature: 0x42 0x4D – “BM”

offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ASCII
00000000	42	4D	F6	04	00	00	00	00	00	76	00	00	00	28	00		BMP.....v..(.
00000010	00	00	30	00	00	00	00	00	00	01	00	04	00	00	00		..0...0.....
00000020	00	00	80	04						00	00	00	00	00	00		..b.....
00000030	00	00	00	00	00	00	00	00	00	00	00	80	00	00	80		...b..b...b..b
00000040	00	00	00	80	80	00	80	00	00	00	80	00	80	00	80		...b..b...b..b
00000050	00	00	C0	C0	C0	00	80	80	80	00	00	00	FF	00	00	FF	...AAA..b..b...A..A
00000060	00	00	00	FF	FF	00	FF	00	00	00	FF	00	FF	00	FF	FF	...AA..A...A..A..A
00000070	00	00	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	...AAA.AAAAAAAAAA

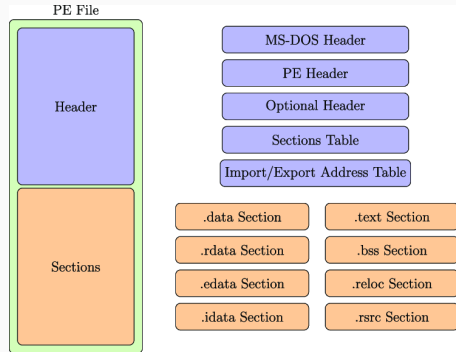


PDF header format



Executable file format

- **Portable Executable (PE)** format used to encode executable files on Windows (`.exe`, `.dll`)
- Common sections are:
 - `.text` (for code)
 - `.data` (read/write data)
 - `.rdata` (read-only data)
 - `.reloc` (relocation data)

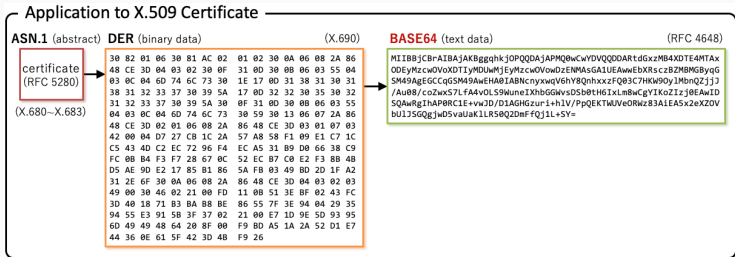


Common binary file formats

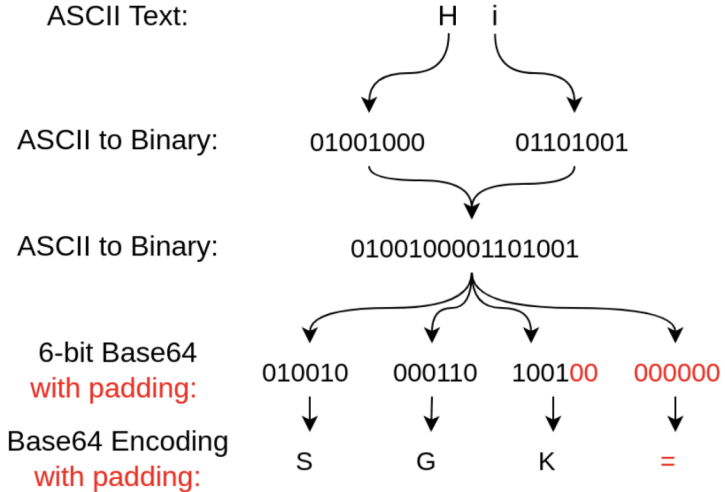
- Image file formats (JPG, GIF, BMP, PNG)
- Archive files (ZIP, TGZ)
- Filesystem images (EXT4)
- Packet captures (PCAP, PCAPNG)
- Memory dumps
- PDF
- Video (MKV, MP4) or Audio (WAV, MP3, FLAC)
- Microsoft Office formats (RTF, OLE, OOXML)

Binary to text encoding schemes

- Sometimes it is necessary to encode binary objects into text
 - E.g., when shipping a binary file across the network, some protocols could **interpret byte sequences as control characters**
- **Base64** is a popular encoding scheme (but there are more...)
 - 64 characters are present in most character sets
 - Used to encode email attachments and certificates



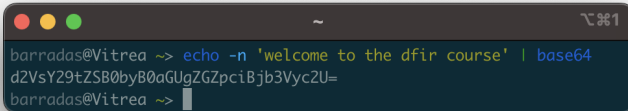
Base64 encoding mechanism



(Anthony Critelli, CC BY-SA 4.0)

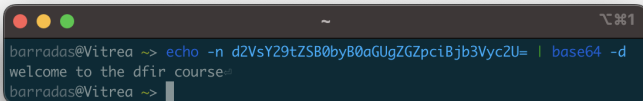
Base64 encoding mechanism

- To encode text to base64:



```
barradas@Vitrea ~> echo -n 'welcome to the dfir course' | base64
d2VsY29tZSB0byB0aGUgZGZpciBjb3Vyc2U=
barradas@Vitrea ~>
```

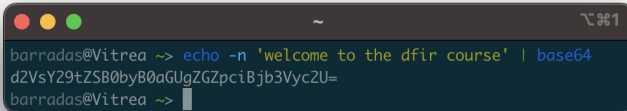
- To decode base64:



```
barradas@Vitrea ~> echo -n d2VsY29tZSB0byB0aGUgZGZpciBjb3Vyc2U= | base64 -d
welcome to the dfir course
barradas@Vitrea ~>
```

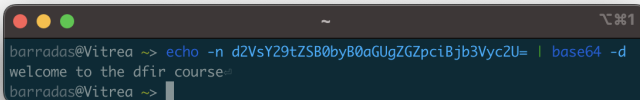
Base64 encoding mechanism

- To encode text to base64:

A terminal window with a dark background and a title bar containing three colored circles (red, yellow, green) and a tilde symbol. The prompt is 'barradas@Vitrea ~'. The command 'echo -n 'welcome to the dfir course' | base64' is entered, and the output 'd2VsY29tZSB0byB0aGUgZGZpciBjb3Vyc2U=' is displayed. The prompt 'barradas@Vitrea ~>' is shown again with a cursor.

```
barradas@Vitrea ~> echo -n 'welcome to the dfir course' | base64
d2VsY29tZSB0byB0aGUgZGZpciBjb3Vyc2U=
barradas@Vitrea ~>
```

- To decode base64:

A terminal window with a dark background and a title bar containing three colored circles (red, yellow, green) and a tilde symbol. The prompt is 'barradas@Vitrea ~'. The command 'echo -n d2VsY29tZSB0byB0aGUgZGZpciBjb3Vyc2U= | base64 -d' is entered, and the output 'welcome to the dfir course' is displayed. The prompt 'barradas@Vitrea ~>' is shown again with a cursor.

```
barradas@Vitrea ~> echo -n d2VsY29tZSB0byB0aGUgZGZpciBjb3Vyc2U= | base64 -d
welcome to the dfir course
barradas@Vitrea ~>
```

Recall that base64 is not an encryption algorithm!

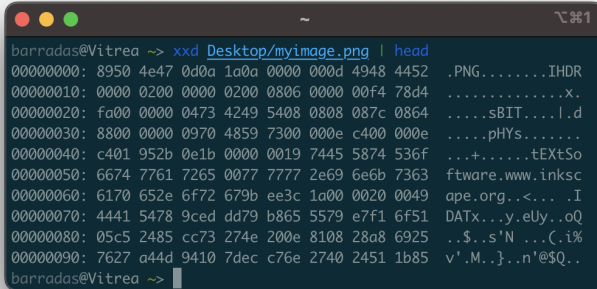
Syntactic analysis of files

Syntactic analysis

- The ability to parse the format of a file so as to identify its internal structure and components
- Techniques:
 - Raw file inspection
 - File format discovery
 - String extraction
 - Encrypted file cracking
 - File header repairing

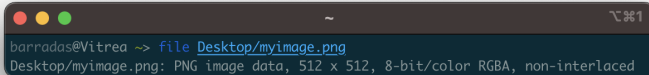
Inspection of a file's raw bytes

- Use an hex editor to read file contents, e.g., WinHex, xxd



```
barradas@Vitrea ~> xxd Desktop/myimage.png | head
00000000: 8950 4e47 0d0a 1a0a 0000 000d 4948 4452 .PNG.....IHDR
00000010: 0000 0200 0000 0200 0806 0000 00f4 78d4 .....x.
00000020: fa00 0000 0473 4249 5408 0808 087c 0864 .....sBIT....l.d
00000030: 8800 0000 0970 4859 7300 000e c400 000e .....pHYs.....
00000040: c401 952b 0e1b 0000 0019 7445 5874 536f ...+.....tExtSo
00000050: 6674 7761 7265 0077 7777 2e69 6e6b 7363 ftware.www.inksc
00000060: 6170 652e 6f72 679b ee3c 1a00 0020 0049 ape.org..<... .I
00000070: 4441 5478 9ced dd79 b865 5579 e7f1 6f51 DATx...y.eUy..oQ
00000080: 05c5 2485 cc73 274e 200e 8108 28a8 6925 ..$.s'N ...(.i%
00000090: 7627 a44d 9410 7dec c76e 2740 2451 1b85 v'.M..}.n'@$Q..
barradas@Vitrea ~>
```

- Use the file utility to match a file's signature



```
barradas@Vitrea ~> file Desktop/myimage.png
Desktop/myimage.png: PNG image data, 512 x 512, 8-bit/color RGBA, non-interlaced
```

Remarks about the encoding scheme

- ASCII encoded file

```
barradas@Vitrea -> xxd Desktop/sw-ascii.txt
00000000: 5374 6172 2057 6172 7320 6973 2061 6e20  Star Wars is an
00000010: 416d 6572 6963 616e 2065 7069 6320 7370  American epic sp
00000020: 6163 6520 6f70 6572 615b 315d 206d 756c  ace opera[!] mul
00000030: 7469 6d65 6469 6120 6672 616e 6368 6973  timedia franchis
00000040: 6520 6372 6561 7465 6420 6279 2047 656f  e created by Geo
00000050: 7267 6520 4c75 6361 732c 2077 6869 6368  rge Lucas, which
00000060: 2062 6567 616e 2077 6974 6820 7468 6520  began with the
00000070: 6570 6f6e 796d 6f75 7320 3139 3737 2066  eponymous 1977 f
00000080: 696c 6d5b 615d 2061 6e64 2071 7569 636b  ilm[a] and quick
00000090: 6c79 2062 6563 616d 6520 6120 776f 726c  ly became a worl
000000a0: 6477 6964 6520 706f 7020 6375 6274 7572  dvide pop cultur
000000b0: 6520 7068 656e 6f6d 656e 6f6e 206a 0d54  e phenomenon.
000000c0: 6865 2066 7261 6e63 6869 7365 2068 6173  he franchise has
000000d0: 2062 6565 6e20 6578 7061 6e64 6564 2069  been expanded i
000000e0: 6e74 6f20 7661 7269 6f75 7320 6669 6c6d  nto various film
```

- UTF-8 encoded file

```
barradas@Vitrea -> xxd Desktop/sw-utf
00000000: 5374 6172 2057 6172 7320 2842 7261 7369  Star Wars (Brasi
00000010: 6c3a 2047 7565 7272 6120 6e61 7320 4573  l: Guerra nas Es
00000020: 7472 656c 6173 202f 2050 6f72 7475 6761  trelas / Portuga
00000030: 6c3a 2047 7565 7272 6120 6461 7320 4573  l: Guerra das Es
00000040: 7472 656c 6173 2920 c3a9 2075 6d61 2066  trelas) .. uma f
00000050: 7261 6e71 7569 6120 646f 2074 6970 6f20  ranquia do tipo
00000060: 7370 6163 6520 6f70 6572 6120 6573 7461  space opera esta
00000070: 6475 6e69 6465 6e73 6520 6372 6961 6461  dunidense criada
00000080: 2070 656c 6f20 6369 6e65 6173 7461 2047  pelo cineasta G
00000090: 656f 7267 6520 4c75 6361 732c 2071 7565  eorge Lucas, que
000000a0: 2063 6f6e 7461 2063 6f6d 2075 6d61 2073  conta com uma s
000000b0: c3a9 7269 6520 6465 206e 6f76 6520 6669  ..rie de nove fi
000000c0: 6c6d 6573 2064 6520 6661 6e74 6173 6961  lmes de fantasia
```

- Special non-printable character

- Character “é” = 0xc3a9

Inspecting a binary file

- Manual inspection
 - Parse the file according to the file format specification

Image width and height.

The file size which is 176 bytes.

The position of the start of the image data which is 54.

The ASCII letters "B" and "M" identifying this as a .bmp file.

0: 42 4D B0 00 00 00 00 00 00 00 36 00 00 00 28 00 BM°.....6...(.
16: 00 00 06 00 00 00 06 00 00 00 01 00 18 00 00 00
32: 00 00 00 00 00 00 12 0B 00 00 12 0B 00 00 00 00
48: 00 00 00 00 00 00 51 52 60 00 00 EF 00 00 EF 00QR`..i..i.
64: 00 EF 00 00 EF 00 26 8A 00 00 F3 EF EF 51 52 60 ..i..i.Š..óiiQR`
80: 00 00 EF 00 00 EF 00 00 EF 2D 10 04 00 00 F3 EF ..i..i..i-...óyi
96: EF ED 8A 88 DF 5D 62 00 00 EF 00 26 8A 00 9D 25 iis`B]b..i.Š.□%
112: 00 00 ED 8A 88 F3 EF EF ED 8A 88 ED 8A 88 2D 10 ..iŠ`óiiiŠ`iŠ`-.
128: 04 00 AD 94 00 00 ED 8A 88 F3 EF EF ED 8A 88 F3 ..-~..iŠ`óiiiŠ`ó
144: EF EF ED 8A 88 01 52 35 00 00 ED 8A 88 F3 EF EF yiiŠ`.R5..iŠ`óyi
160: ED 8A 88 ED 8A 88 F3 EF EF ED 8A 88 00 00 00 00 iŠ`iŠ`óiiiŠ`....

136R 136G 237B

96R 82G 81B

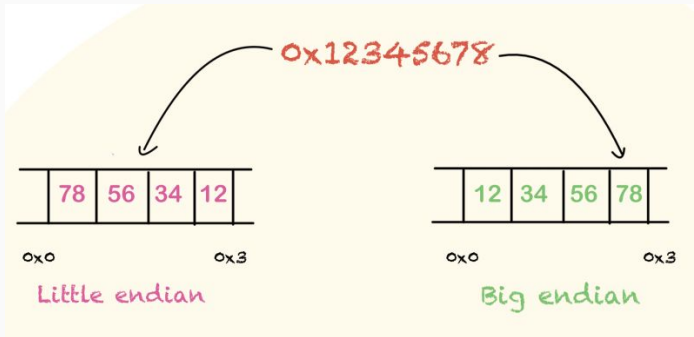
138R 38G 00B

end of row padding

end of file padding

Remarks about endianness

- Numbers can be stored as a **sequence** of one or more bytes
- Endianness deals with the order in which bytes are stored
 - We encounter two different approaches:



x86 and ARM make use of Little Endian.

Why do we care?

- In the sequence below, the two highlighted bytes represent a 16-bit integer (8 bit \times 2 = 16 bits or 2 bytes)

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	01	23	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

- In a big-endian system, the value would be calculated as:
 - Big-endian calculation: **0x0123 = 291**
- In a little-endian system, the value would be calculated as:
 - Little-endian calculation: **0x2301 = 8961**

How to decode a file not knowing its format?

- Consider a file with the following byte sequence (hex):

```
48 65 6c 6c 6f 20 57 6f 72 6c 64 0a
```

- Is it a text file? ASCII? Base64?

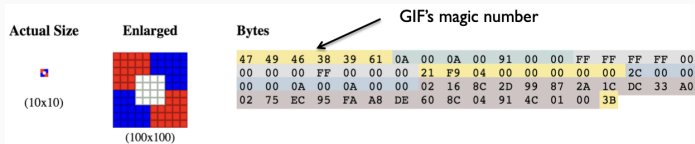
```
'H' 'e' 'l' 'l' 'o' ' ' 'W' 'o' 'r' 'l' 'd' '\n'
```

- Or a binary file? Unsigned ints? Other format?

```
'18533' '27756' '28448' '22383' '29292' '25610'
```

Magic numbers

- Look for **magic numbers**: consist of constant numerical or text value used to identify a file format or protocol
 - E.g., GIF files start with sequence: 0x47 49 46 38 39 61
- Remember the file utility?



- Magic numbers of common file formats:
 - http://www.garykessler.net/library/file_sigs.html

Opening files with specialized programs

- After determining the file type, use a corresponding program to open it
 - E.g., examine file `c.dat`
- Remember the `file` utility?



`c.dat` when opened in
GIMP

```
barradas@Vitrea ~> file Desktop/c.dat
Desktop/c.dat: Adobe Photoshop Image, 404 x 226, grayscale, 3x 8-bit channels
barradas@Vitrea ~> 
```

String extraction from files

- The strings utility: prints a file's **readable characters**
 - Useful for looking at data fields without the originating program, searching executables for strings, etc.

```
barradas@Vitrea ~-> strings /usr/sbin/arp
andflsSi:x
-i not applicable to this operation
interface %s does not exist
if_nametoindex(%s)
%-23s %-17s %-9.9s %-9.9s %14.14s %4s %4s
Neighbor
Linklayer Address
Expire(0)
Expire(I)
Netif
Refs
Prbs
%-7.7s %-7.7s %-7.7s
RSSI
ifscope
ifscope has bad interface name: %s
cannot open %s
%49s %49s %49s %49s %49s %49s %49s
bad line: %s
```

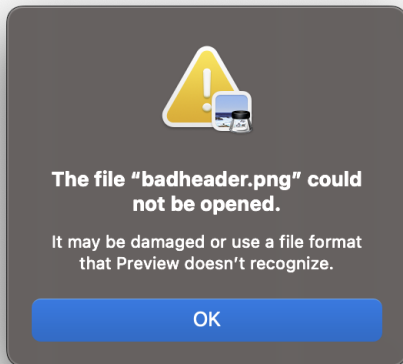
```
permanent
published (proxy only)
published
(weird)
[ethernet]
[fddi]
[atm]
[vlan]
[firewire]
[bridge]
%x%s
usage: arp [-n] [-i interface] hostname
        arp [-n] [-i interface] [-l] -a
        arp -d hostname [pub] [ifscope interface]
        arp -d [-i interface] -a
        arp -s hostname ether_addr [temp] [reject] [blackhole] [pub [only]] [ifscope interface]
        arp -S hostname ether_addr [temp] [reject] [blackhole] [pub [only]] [ifscope interface]
        arp -f filename
%-23s
%-17s
```

What if files are encrypted?

- Might be worth trying a password cracking software
 - Essentially, a password cracker works by trial and error
- Cracking approaches:
 - Brute force: try every possible key / password until succeeds
 - Dictionary attacks: attempt to reduce the number of trials required and will usually be attempted before brute force
- Examples of tools:
 - For encrypted zip files: `fcrackzip`
 - For encrypted pdf files: `pdfcrack`

What if files are corrupted?

- Specific metadata of a file may have been removed or tampered with thereby preventing its decoding
 - E.g., opening file `badheader.jpg` returns this error



Repairing a corrupted file

- Check if file metadata is consistent with the file format spec

```
89 50 4E 47 0D 0A 1A 0A %PNG....
```

PNG [Portable Network Graphics file](#)

Trailer: 49 45 4E 44 AE 42 60 82 (IEND®B` , ...)

Repairing a corrupted file

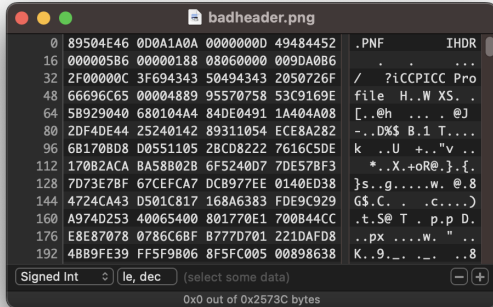
- Check if file metadata is consistent with the file format spec

```
89 50 4E 47 0D 0A 1A 0A %PNG....
```

PNG [Portable Network Graphics file](#)

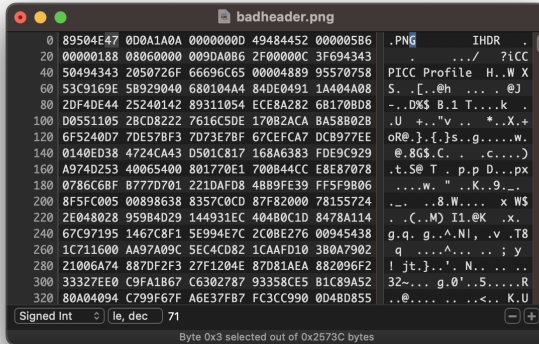
Trailer: 49 45 4E 44 AE 42 60 82 (IEND@B` , ...)

- Our header looks wrong: 89 50 4E 46 0D 0A 1A 0A ...



Apply the fix

- Fix the wrong header byte



```
0 89504E47 0D0A1A0A 0000000D 49484452 000005B6 .PNG IHDR .
20 00000188 08060000 009DA0B6 2F00000C 3F694343 . .../ ?iCC
40 50494343 2050726F 66696C65 00004889 95570758 PICC Profile H..W X
60 53C9169E 5B929040 680104A4 84DE0491 1A404A08 S. .[. @h ... . @J
80 2DF4DE44 25240142 89311054 ECE8A282 6B170BD8 -. .D%$ B.1 T...k .
100 D0551105 2BCD8222 7616C5DE 170B2ACA BA58B02B .U +.. "v .. *.X.+
120 6F5240D7 7DE57BF3 7D73E7BF 67CEFA7 DCB977EE oR@.}.{.}s..g.....w.
140 0140ED38 4724CA43 D501C817 168A6383 FDE9C929 @.8G$.C. . .c....)
160 A974D253 40065400 801770E1 700B44CC E8E87078 .t.S@ T . p.p D...px
180 0786C6BF 8777D701 221DAFD8 4BB9FE39 FF5F9B06 ....w. " ..K..9.._
200 8F5FC005 00898638 8357C0CD 87F82000 78155724 _.. ..8.W.... x W$
220 2E048028 95984D29 144931EC 404B0C1D 8478A114 . .(.M) I1..@K .x.
240 67C97195 1467C8F1 5E994E7C 2C0BE276 00945438 g.q. g..^..NI, .v .T8
260 1C711600 AA97A09C 5EC4CD82 1CAAFD10 3B0A7902 q ....^..... ; y
280 21006A74 887DF2F3 27F1204E 87D81AEA 882096F2 ! jt.}..' N. ....
300 33327EE0 C9FA1867 C6302787 93358CE5 B1C89A52 32~... g.0'..5....R
320 80A04094 C799F67F A6E37FB7 FC3CC990 0D48D855 ..@.... . .<.. K.U

Signed Int le, dec 71
Byte 0x3 selected out of 0x2573C bytes
```

- Verify the fix by reopening the file using the viewer
- This case was easy, but may require additional effort

Summary of tips for interpreting file content

- Extension not entirely reliable
- Open the file and check between (text / binary)
- Look for known header and footer information
 - Especially file format signatures (e.g., magic numbers)
- Use tools that know how to interpret specific file format
- If the file format is unknown, we analyze it manually

Semantic analysis of files

Semantic analysis

- The ability to interpret and acquire information from the data content of a given file
- Presupposes the ability to parse the file's internal structures
- Some examples:
 - Vulnerability analysis
 - Image processing
 - Provenance analysis

Image processing

- Forensic image processing involves the computer restoration and enhancement of imagery
- It aims to maximize information extraction from imagery that is noisy, incomplete, or over/under exposed
- It also involves measurement of objects pictured on images



Image processing

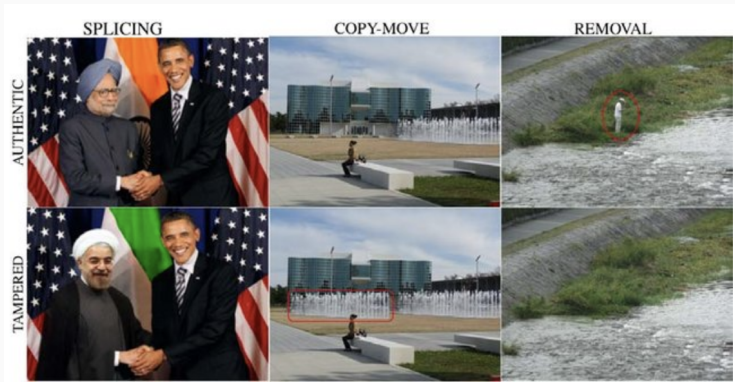
- Forensic image processing involves the computer restoration and enhancement of imagery
- It aims to maximize information extraction from imagery that is noisy, incomplete, or over/under exposed
- It also involves measurement of objects pictured on images



CSI-style “Enhance!”

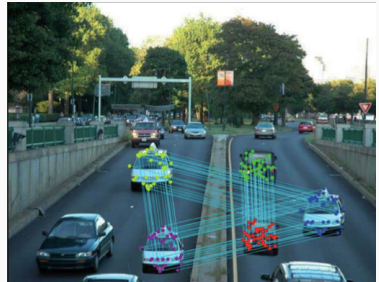
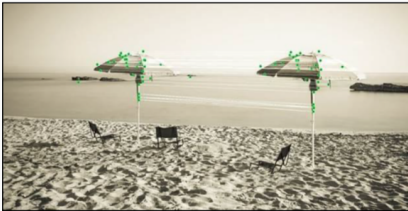
Tampering detection

- In the context of digital imaging, **tampering** recalls the intentional manipulation of images for malicious purposes



Example of tampering detection technique

- Structural analysis: helps detect copy-move of an image region
 - A structural analysis algorithm splits the image into segments
 - A histogram is built count the number of matching segments which are separated by the same distance
 - The higher the number of pairs located at the same distance, the higher is the probability that those pairs belong to copied regions



Deepfake detection

- Deepfakes make image tampering detection difficult

UK NEWS WEBSITE OF THE YEAR

The Telegraph News Sport Money Business Opinion Ukraine Royals Life Style Travel

UK news - Politics - World - Health - Defence - Science - Education - Environment - Investigations - Global Health Security

I was 'deepfaked' committing a crime – here's why you should be worried too

Deepfake technology, which can map one person's face onto another, is more sophisticated than ever – the implications are terrifying

By Michael Grothaus
30 October 2021 - 5:00pm

intel

FakeCatcher

For security, not just deepfake detection

Personalized Intel® FakeCatcher deepfake detection analyzes "Visual Flow" in video pixels to determine a video's authenticity, intelligently.

What is a deepfake?
Deepfakes are synthetic videos, images, or audio clips where the actor or the action of the actor is faked.

72% improvement
Intel FakeCatcher can improve deepfake detection accuracy by 72% compared to baseline detection.

96%
Intel FakeCatcher can improve deepfake detection accuracy by 96% compared to baseline detection.

How could FakeCatcher be used?

- Content Creation Tools**
Deepfake detection is a useful tool for content creators to ensure their content is authentic.
- Media & Broadcasters**
Deepfake detection is a useful tool for media and broadcast companies to ensure their content is authentic.
- Security & Investigation**
Deepfake detection is a useful tool for security and investigation agencies to ensure their content is authentic.
- Artificial Social Media**
Deepfake detection is a useful tool for artificial social media platforms to ensure their content is authentic.



The Story Of Luka Magnotta

Don't [REDACTED] With Cats tells the gripping story of one of Canada's worst murderers, Luka Magnotta.

In the three-part docuseries, we are introduced to Baudi Moovan (Deanna Thompson) and John Green who recount everything that lead up to Luka's eventual demise.

You see, before Luka Magnotta was a murderer, he was an animal abuser. Back in 2010, a mystery man uploaded a video onto Youtube titled *1 boy 2 kittens*. The video featured a male in a hooded jacket, vacuum sealing two kittens until they suffocated and died. The clip sparked outrage online and lead to Baudi and John creating the 'Find the Kitten Vacuumer...for Great Justice' Facebook group.

Provenance analysis

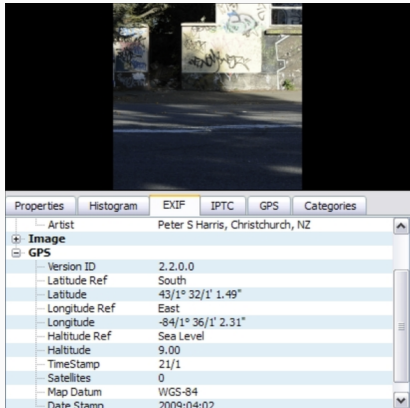
- Images & video frames: Tell one's location by looking at them

The Facebook group consisted of people from all over the world, determined to uncover who the animal abuser was.

Together they analysed Luka's video frame-by-frame. They looked at plug sockets on the wall to narrow down the killer's location. They listened to background sounds to determine the languages heard. The group even went so far as to find the online sellers of blankets and vacuums seen in shots to find some clue of who the cat killer was.

Metadata

- Image metadata can also give us a lot of information
 - (If available)



General		Exif	GPS	TIFF
Aperture Value	2.526			
Brightness Value	1.671			
Color Space	sRGB			
Components Configuration	1, 2, 3, 0			
Date Time Digitized	Dec 31, 2015, 3:10:13 PM			
Date Time Original	Dec 31, 2015, 3:10:13 PM			
Exif Version	2.2.1			
Exposure Bias Value	0			
Exposure Mode	Auto exposure			
Exposure Program	Normal program			
Exposure Time	1/30			
Flash	No flash function			
FlashPix Version	1.0			
FNumber	2.4			
Focal Length	1.85			
Focal Length In 35mm Film	35			
ISO Speed Ratings	250			
Lens Make	Apple			
Lens Model	iPhone 4S front camera 1.85mm f/2.4			

Source device identification

- In a court of law, the device used for acquisition of a particular image can represent crucial evidence
- Helpful clues on the source imaging device might be found in the **file's header** (EXIF), or by checking (if present) a **watermark**
- However, since this information can be easily modified or removed, we may need to employ **blind** techniques

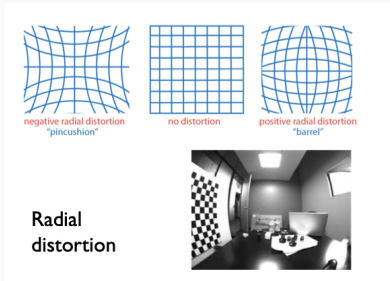


Blind image forensics techniques

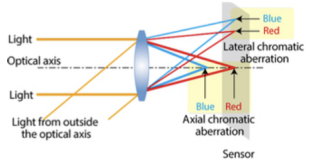
- Retrieve information on the source device at two levels:
 - Try to distinguish between different camera models
 - Try to distinguish between single devices, even different devices using the same camera model – harder
- Examine the traces left by the different processing steps in the image acquisition and storage phases

Acquisition of artifacts produced by lenses

- Dirt / deformations
- Lens distortion
- Chromatic aberration



Chromatic aberration



More relevant sources of info for file provenance

- Embedded cryptographic signatures
- References to the program that modified the file
- References about place and time of the file
- References about the author of the file
- Indirect references from other sources (e.g., logs)
- Watermarks
- ...

Takeaways

- Interpretation of file contents is one of the first steps in a typical forensic analysis procedure
- To interpret file contents it may be necessary to understand how the data is represent in its raw form
- In addition, many files are structured according to a particular layout, and thus it is necessary to learn its specific format in order to properly interpret them

- **Textbook:**
 - Carrier – Chapter 2.1
- **Other resources:**
 - A. Piva, “An Overview on Image Forensics”, ISRN Signal Processing, January 2013
 - Judith A. Redi et al. “Digital image forensics: a booklet for beginners”, Multimedia Tools and Applications, 2010
- **Acknowledgements:**
 - Slides adapted from Nuno Santos’s Forensics Cyber-Security course at Técnico Lisbon