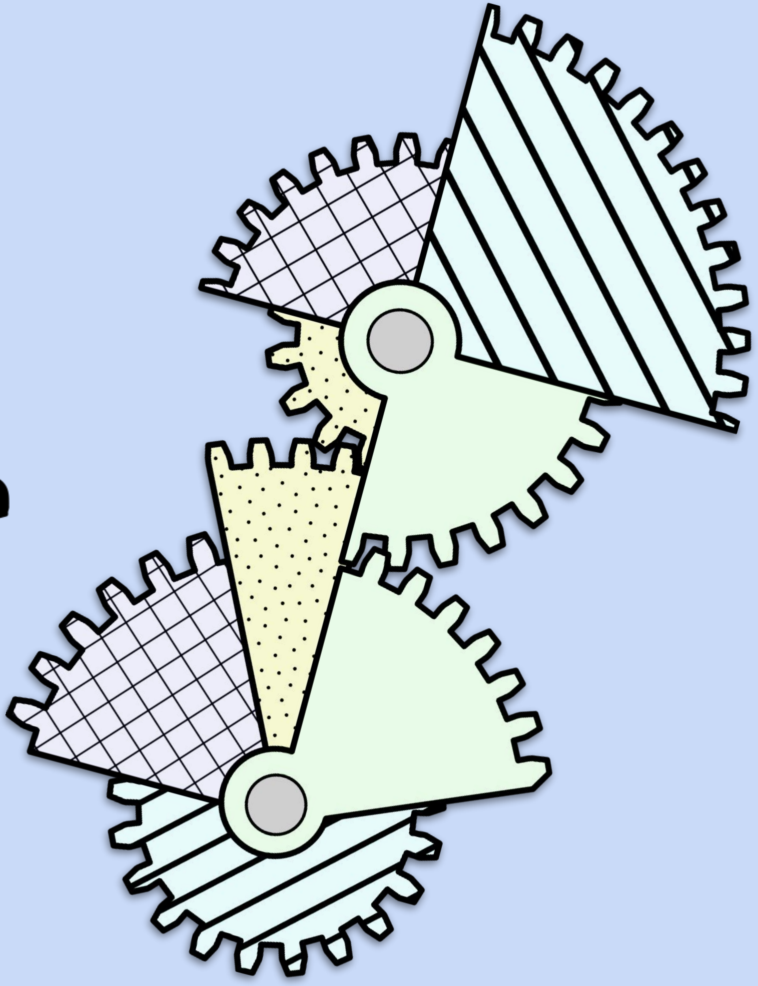
# An Overview of File Type Identifiers

Ange Albertini

10/2024



# An overview of file type identifiers

LibMagic, Yara, TrID, Magika…

Ange Albertini
Google

10/2024
HackLu

## About the author

- Reverse engineer, staring at files for 3 decades.
- Malware analyst for 2 decades: Symantec, Avira, Google.
- Known for: CPS2Shock, Corkami, PoC||GTFO*, Shattered…

*https://github.com/angea/pocorgtfo/blob/master/README.md

## Honest trailer

1. Interests in filtering files *quickly* & "reliably".
2. Build KB and corpus.
3. Classify & validate files, resolve existing conflicts.

... How are existing engines doing? Any caveats ?

*The file format landscape is a mess of messes.*

THE CURRENT SLIDE IS AN

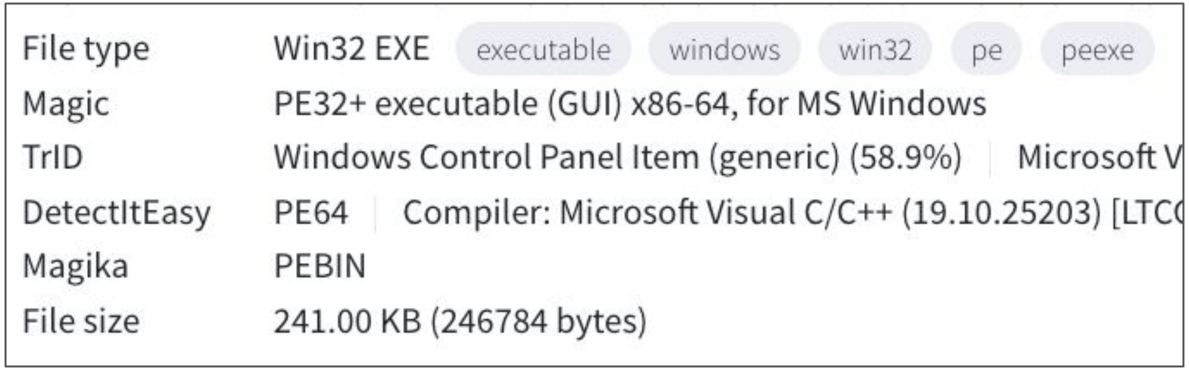
**HONEST TALK TRAILER**

A CORKAMI ORIGINAL PRODUCTION

## Side questions

- Why is TrID standing out?
- How are filetypes mapped on linux ?
  (-> is `ShareMime` equivalent to `file` ?)

What does that imply? ->



|  |  |
|---|---|
| File type | Win32 EXE executable windows win32 pe peexe |
| Magic | PE32+ executable (GUI) x86-64, for MS Windows |
| TrID | Windows Control Panel Item (generic) (58.9%) \| Microsoft V |
| DetectItEasy | PE64 \| Compiler: Microsoft Visual C/C++ (19.10.25203) [LTC |
| Magika | PEBIN |
| File size | 241.00 KB (246784 bytes) |

# Engines

PeID, PRONOM, FDD, TrID

LibMagic (file/BinWalk$_2$) Share-Mime Yara

DiE, BinWalk$_3$

Magika

## Features

- Fixed logic (data-only) or code?
- Specific syntax (limited) or standard language (heuristics)?
- Relative offsets, pointers, conditions, multiplication, variables, functions...
- Automated signatures generation
- Bytes signatures / Heuristics / ML?

## Expectations

- Extendable. Speed. Simplicity.
- Only infosec-stuff for scanning or "everything" ?

Reliability (FPs, Adversarial...):

- Is "MZPE" a valid executable?
- is `<!-- --><html></html>` a webpage?

Spoiler: they all have their pros and cons.

# File / [Lib]Magic

Tool: linux.die.net/man/1/file / Format: linux.die.net/man/5/magic

+ Multiple outputs
+ "Functions"
+ Pointers, relative offsets
- Peculiar syntax
- Old (v4.1 in 1973)

LibMagic-based: BinWalk v2, Polyfile.

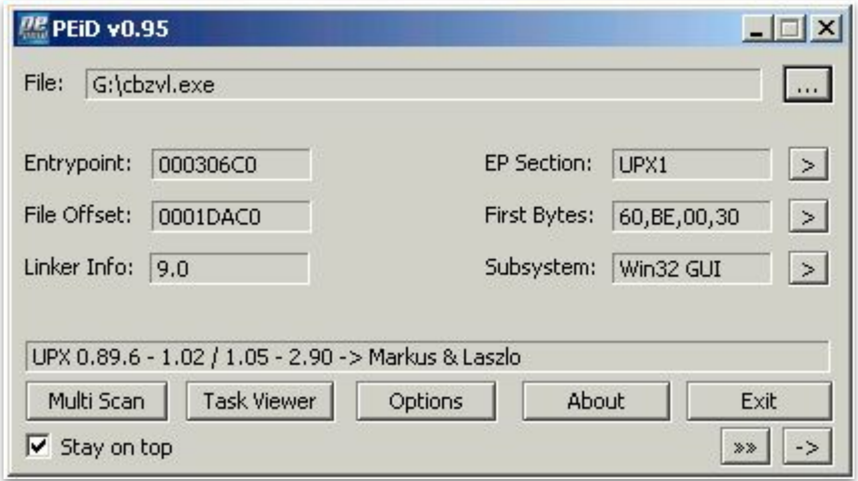
## TrID

Binary magic signatures at specific offsets:

with optional ASCII/Wide string signatures.

And no extra logic!

+ Generation can be automated (!) Non-ML learning:
  + Common bytes in the first 2Kb, strings in the first/last 5Mb.
  - It's clever and it works, but FPs can easily happen.


PEiD v0.95

File: G:\cbzvl.exe

Entrypoint: 000306C0
File Offset: 0001DAC0
Linker Info: 9.0
EP Section: UPX1
First Bytes: 60,BE,00,30
Subsystem: Win32 GUI

UPX 0.89.6 - 1.02 / 1.05 - 2.90 -> Markus & Laszlo

## UserDB.TXT

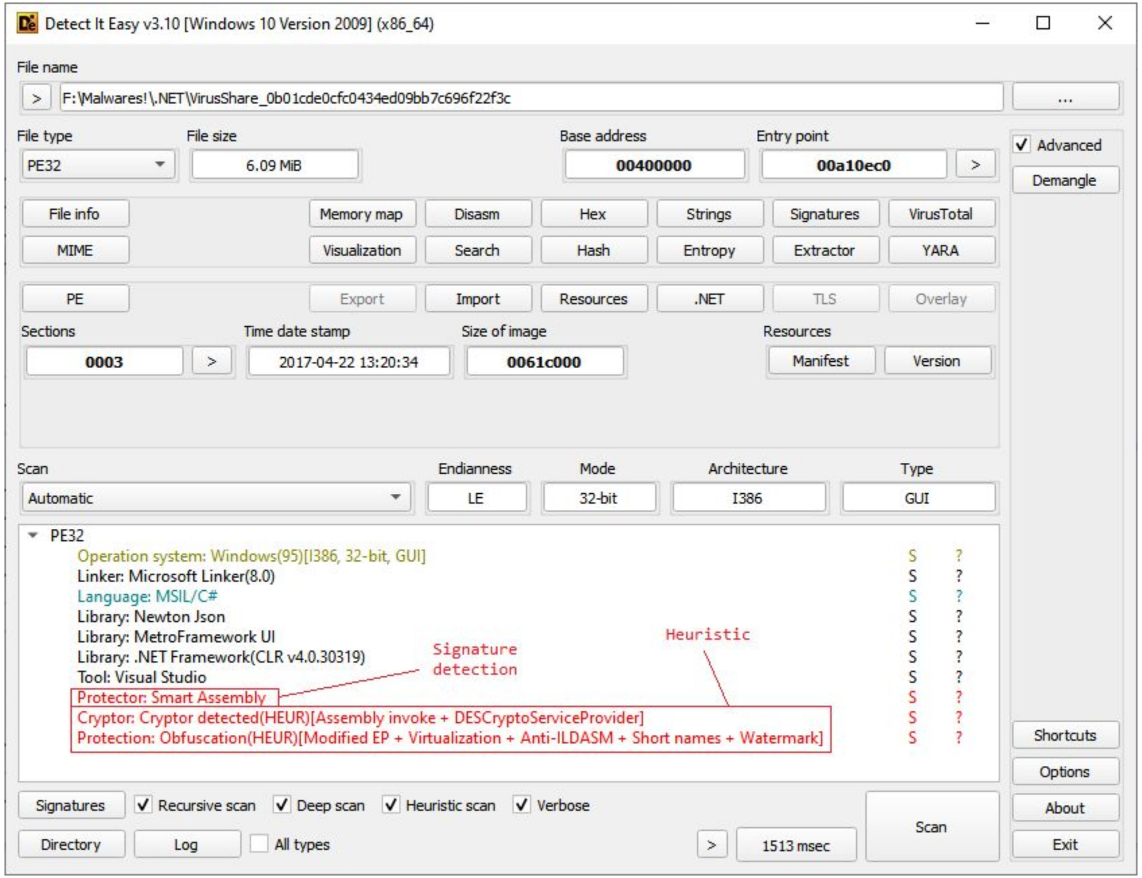
```
[UPX 2.00-3.0X -> Markus Oberhumer & Laszlo Molnar & John Reiser]
signature = 5E 89 F7 B9 ?? ?? ?? ?? 8A 07 47 2C E8 3C 01 77 F7 80 ...
ep_only = false
```

## PE identifier

## PEiD [github wolfram77web/app-peid]

- PE-only, pure byte sequences, at EntryPoint or not (boolean).
- UserDB.TXT (.INI format)

Useful for non-polymorphic binary packer identification.
(i.e. too many strings sequence for VmProtect)



File name: F:\Malwares!\.NET\VirusShare_0b01cde0cfc0434ed09bb7c696f22f3c

File type: PE32
File size: 6.09 MiB
Base address: 00400000
Entry point: 00a10ec0
[✓] Advanced

Sections: 0003
Time date stamp: 2017-04-22 13:20:34
Size of image: 0061c000

Scan: Automatic
Endianness: LE
Mode: 32-bit
Architecture: I386
Type: GUI

```
PE32
    Operation system: Windows(95)[I386, 32-bit, GUI]                                S  ?
    Linker: Microsoft Linker(8.0)                                                   S  ?
    Language: MSIL/C#                                                               S  ?
    Library: Newton Json                                                            S  ?
    Library: MetroFramework UI                                                      S  ?
    Library: .NET Framework(CLR v4.0.30319)                                         S  ?
    Tool: Visual Studio                                                             S  ?
    Protector: Smart Assembly                                                       S  ?
    Cryptor: Cryptor detected(HEUR)[Assembly invoke + DESCryptoServiceProvider]     S  ?
    Protection: Obfuscation(HEUR)[Modified EP + Virtualization + Anti-ILDASM + Short names + Watermark]  S  ?
```

[✓] Recursive scan [✓] Deep scan [✓] Heuristic scan [✓] Verbose
[ ] All types
1513 msec

# Die: Detect-It-Easy

## Detect-It-Easy github horsicq Detect-It-Easy (MIT)

- Code driven (Javascript)
- Signatures + heuristics

Unbalanced signature variety:

- 100s of DOS detections: Microsoft C, PKLite, LZExe, WatCom…
- 2 kinds of CFB files: MSI or Office97.

## Format Description Documents Library of Congress (loc.gov)

A knowledge base: ~600 entries

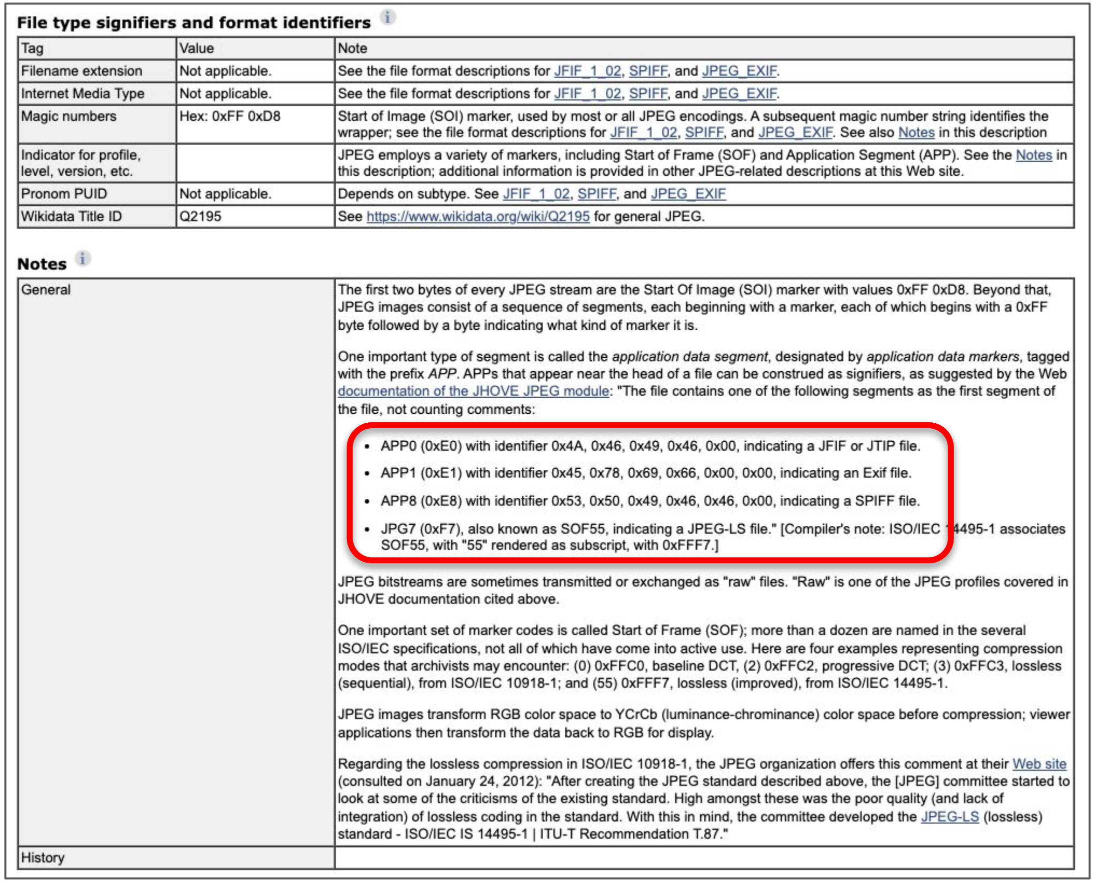
A lot of non-infosec stuff (ex: no executable ***at all***)

Examples:

- JPG: JPEG Image Encoding Family
- No ELF, no PE...

Looking for "Portable" ?

- PNG, Portable Network Graphics
- PEF: Portable Embosser Format (Braille)



**File type signifiers and format identifiers**

| Tag | Value | Note |
|---|---|---|
| Filename extension | Not applicable. | See the file format descriptions for JFIF_1_02, SPIFF, and JPEG_EXIF. |
| Internet Media Type | Not applicable. | See the file format descriptions for JFIF_1_02, SPIFF, and JPEG_EXIF. |
| Magic numbers | Hex: 0xFF 0xD8 | Start of Image (SOI) marker, used by most or all JPEG encodings. A subsequent magic number string identifies the wrapper; see the file format descriptions for JFIF_1_02, SPIFF, and JPEG_EXIF. See also Notes in this description |
| Indicator for profile, level, version, etc. | | JPEG employs a variety of markers, including Start of Frame (SOF) and Application Segment (APP). See the Notes in this description; additional information is provided in other JPEG-related descriptions at this Web site. |
| Pronom PUID | Not applicable. | Depends on subtype. See JFIF_1_02, SPIFF, and JPEG_EXIF |
| Wikidata Title ID | Q2195 | See https://www.wikidata.org/wiki/Q2195 for general JPEG. |

**Notes**

General

The first two bytes of every JPEG stream are the Start Of Image (SOI) marker with values 0xFF 0xD8. Beyond that, JPEG images consist of a sequence of segments, each beginning with a marker, each of which begins with a 0xFF byte followed by a byte indicating what kind of marker it is.

One important type of segment is called the *application data segment*, designated by *application data markers*, tagged with the prefix *APP*. APPs that appear near the head of a file can be construed as signifiers, as suggested by the Web documentation of the JHOVE JPEG module: "The file contains one of the following segments as the first segment of the file, not counting comments:

- APP0 (0xE0) with identifier 0x4A, 0x46, 0x49, 0x46, 0x00, indicating a JFIF or JTIP file.
- APP1 (0xE1) with identifier 0x45, 0x78, 0x69, 0x66, 0x00, 0x00, indicating an Exif file.
- APP8 (0xE8) with identifier 0x53, 0x50, 0x49, 0x46, 0x46, 0x00, indicating a SPIFF file.
- JPG7 (0xF7), also known as SOF55, indicating a JPEG-LS file." [Compiler's note: ISO/IEC 14495-1 associates SOF55, with "55" rendered as subscript, with 0xFFF7.]

JPEG bitstreams are sometimes transmitted or exchanged as "raw" files. "Raw" is one of the JPEG profiles covered in JHOVE documentation cited above.

One important set of marker codes is called Start of Frame (SOF); more than a dozen are named in the several ISO/IEC specifications, not all of which have come into active use. Here are four examples representing compression modes that archivists may encounter: (0) 0xFFC0, baseline DCT, (2) 0xFFC2, progressive DCT; (3) 0xFFC3, lossless (sequential), from ISO/IEC 10918-1; and (55) 0xFFF7, lossless (improved), from ISO/IEC 14495-1.

JPEG images transform RGB color space to YCrCb (luminance-chrominance) color space before compression; viewer applications then transform the data back to RGB for display.

Regarding the lossless compression in ISO/IEC 10918-1, the JPEG organization offers this comment at their Web site (consulted on January 24, 2012): "After creating the JPEG standard described above, the [JPEG] committee started to look at some of the criticisms of the existing standard. High amongst these was the poor quality (and lack of integration) of lossless coding in the standard. With this in mind, the committee developed the JPEG-LS (lossless) standard - ISO/IEC IS 14495-1 | ITU-T Recommendation T.87."

History

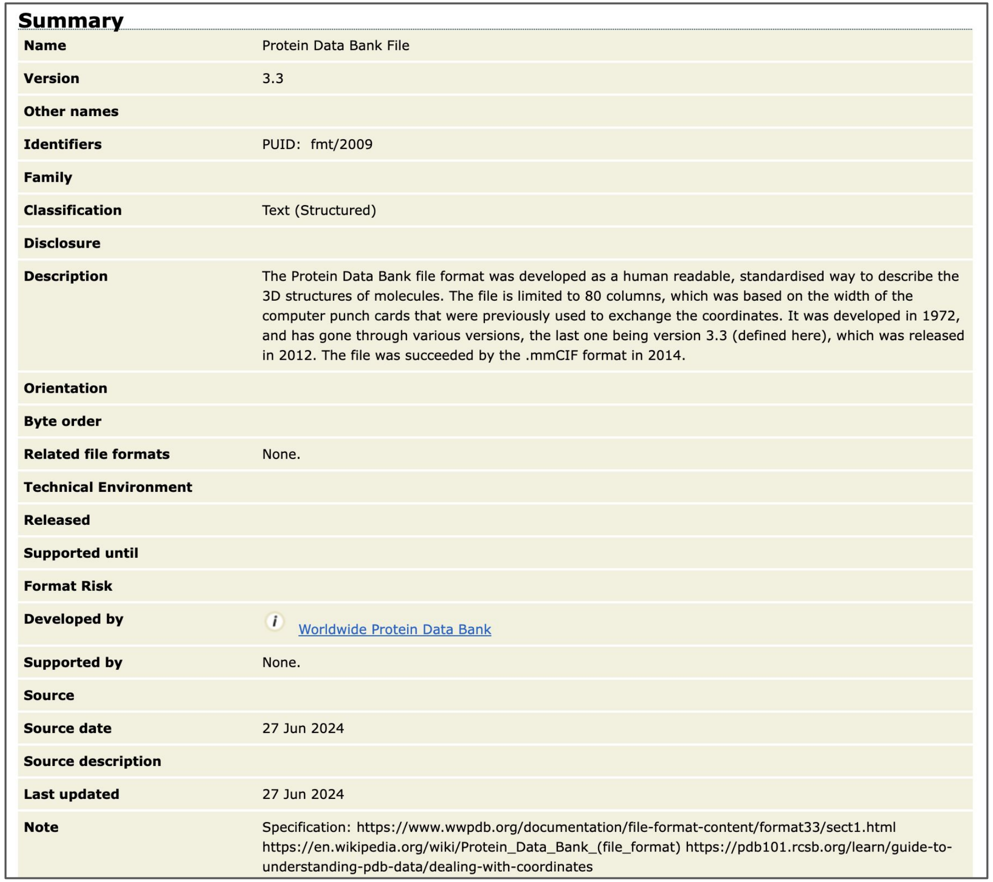
LoC's FDD about JPEG: JPEG Image Encoding Family (fdd000017)

## PRONOM & DROID (TOOL+SIGS)

[NATIONAL ARCHIVES .GOV.UK / PRONOM]

PRONOM: Technical format registry

DROID (**D**igital **R**ecord **O**bject **I**dentification):
tool + XML signatures



## Summary

| | |
|---|---|
| **Name** | Protein Data Bank File |
| **Version** | 3.3 |
| **Other names** | |
| **Identifiers** | PUID: fmt/2009 |
| **Family** | |
| **Classification** | Text (Structured) |
| **Disclosure** | |
| **Description** | The Protein Data Bank file format was developed as a human readable, standardised way to describe the 3D structures of molecules. The file is limited to 80 columns, which was based on the width of the computer punch cards that were previously used to exchange the coordinates. It was developed in 1972, and has gone through various versions, the last one being version 3.3 (defined here), which was released in 2012. The file was succeeded by the .mmCIF format in 2014. |
| **Orientation** | |
| **Byte order** | |
| **Related file formats** | None. |
| **Technical Environment** | |
| **Released** | |
| **Supported until** | |
| **Format Risk** | |
| **Developed by** | Worldwide Protein Data Bank |
| **Supported by** | None. |
| **Source** | |
| **Source date** | 27 Jun 2024 |
| **Source description** | |
| **Last updated** | 27 Jun 2024 |
| **Note** | Specification: https://www.wwpdb.org/documentation/file-format-content/format33/sect1.html https://en.wikipedia.org/wiki/Protein_Data_Bank_(file_format) https://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/dealing-with-coordinates |

# Protein Data Bank page on PRONOM: fmt/2009

```xml
<InternalSignature ID="69" Specificity="Specific">
   <ByteSequence Reference="BOFoffset">
       <SubSequence MinFragLength="0" Position="1"
           SubSeqMaxOffset="0" SubSeqMinOffset="0">
           <Sequence>FFD8FF</Sequence>
           <DefaultShift>4</DefaultShift>
           <Shift Byte="D8">2</Shift>
           <Shift Byte="FF">1</Shift>
       </SubSequence>
   </ByteSequence>
   <ByteSequence Reference="EOFoffset">
       <SubSequence MinFragLength="0" Position="1"
           SubSeqMaxOffset="65536" SubSeqMinOffset="0">
           <Sequence>FFD9</Sequence>
           <DefaultShift>-3</DefaultShift>
           <Shift Byte="D9">-2</Shift>
           <Shift Byte="FF">-1</Shift>
       </SubSequence>
   </ByteSequence>
</InternalSignature>
```

Beginning of File

Sequence of bytes

Bytes again…

## A fragment of a DROID signature for JPG files

```xml
<FileFormat ID="776"
   MIMEType="application/vnd.microsoft.portable-executable"
   Name="Windows Portable Executable" PUID="x-fmt/411">
   <InternalSignatureID>198</InternalSignatureID>
   <Extension>dll</Extension>
   <Extension>exe</Extension>
   <Extension>sys</Extension>
   <HasPriorityOverFileFormatID>774</HasPriorityOverFileFormatID>
   <HasPriorityOverFileFormatID>775</HasPriorityOverFileFormatID>
</FileFormat>
```

```xml
<InternalSignature ID="198" Specificity="Specific">
   <ByteSequence Reference="BOFoffset">
       <SubSequence MinFragLength="0" Position="1"
           SubSeqMaxOffset="0" SubSeqMinOffset="0">
           <Sequence>4D5A</Sequence>
           <DefaultShift>3</DefaultShift>
           <Shift Byte="4D">2</Shift>
           <Shift Byte="5A">1</Shift>
       </SubSequence>
       <SubSequence MinFragLength="0" Position="2" SubSeqMinOffset="0">
           <Sequence>50450000</Sequence>
           <DefaultShift>5</DefaultShift>
           <Shift Byte="00">1</Shift>
           <Shift Byte="45">3</Shift>
           <Shift Byte="50">4</Shift>
       </SubSequence>
   </ByteSequence>
</InternalSignature>
```

## A fragment of a DROID signature for PE files

## Shared-MIME-Info

https://specifications.freedesktop.org/shared-mime-info-spec/0.21/ar01s02.html

- Standard GNOME/KDE/ROX system
- File in `/usr/share/mime/magic`
- Maps file contents to Mime types.
- LibMagic-like, but more limited:
  - No relative offsets, no functions, no pointers
  - Just offsets, optional range scanning and bitmask

Very limited!

```
00  4d 49 4d 45 2d 4d 61 67 69 63 00 0a 5b 35 30 3a  MIME-Magic..[50:
10  74 65 78 74 2f 78 2d 64 69 66 66 5d 0a 3e 30 3d  text/x-diff].>0=
20  00 05 64 69 66 66 09 0a 3e 30 3d 00 04 2a 2a 2a  ..diff..>0=..***
30  09 0a 3e 30 3d 00 17 43 6f 6d 6d 6f 6e 20 73 75  ..>0=..Common.su
40  62 64 69 72 65 63 74 6f 72 69 65 73 3a 20 0a     bdirectories:..
```

MAGIC SIGNATURE

```
MIME-Magic\x00\n
```

PRIORITY MIME

```
[50:text/x-diff]\n
```

INDENT LENGTH VALUE BIG ENDIAN VALUE

```
>0=\x00\x05diff\x09\n
```

Is `diff\t` AT OFFSET 0 ?

```
>0=\x00\x04***\x09\n
```

Is `***\t` AT OFFSET 0 ?

```
>0=\x00\x17Common subdirectories:\x20\n
```

NO ESCAPED CHARACTERS:
A TEXT FILE WITH PURE BINARY!

## THE SHARED-MIME-INFO MAGIC FILE:
## INI-LIKE, LIBMAGIC-LIKE, AND NON-ASCII BYTES

```
1>100=\x00\x03ABC+100\n
```

```
[indent] ">" start-offset "=" value ["&" mask] ["~" word-size] ["+" range-length] "\n"
```

A Share-Mime-Info magic rule:

one-liners like LibMagic, but fewer possible operators.

# Magika

A new ML-based identifier (a "non-generative AI").

Returns several file types with percentage.

Handles all formats at once - text and binary formats.

Src (python, rust, go): github google/magika, Paper: arxiv 2409.13768

Fast: 6ms per file (only `file` is faster), Tiny model: 1Mb in memory.

Scans *start* and *end* buffers + specific offsets

-> not depending on file sizes, most of the file's content is ignored.

## Magika: pros and cons

v2 released in 08/2024: as many formats as possible*

Used in production.

No validation, no information extractions.

It can't be updated for now.

For adversarial files:
  trick: wipe the first X bytes, then re-scan it.

Standard

```
    +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
0x  89 .P .N .G 0D 0A 1A 0A 00 00 00 0D .I .H .D .R
1x  00 00 09 54 00 00 02 C0 08 06 00 00 00 76 4E 6B
2x  38 00 00 20 00 .I .D .A .T 78 9C 9C FD 0B 96 EC
    .. ..
```

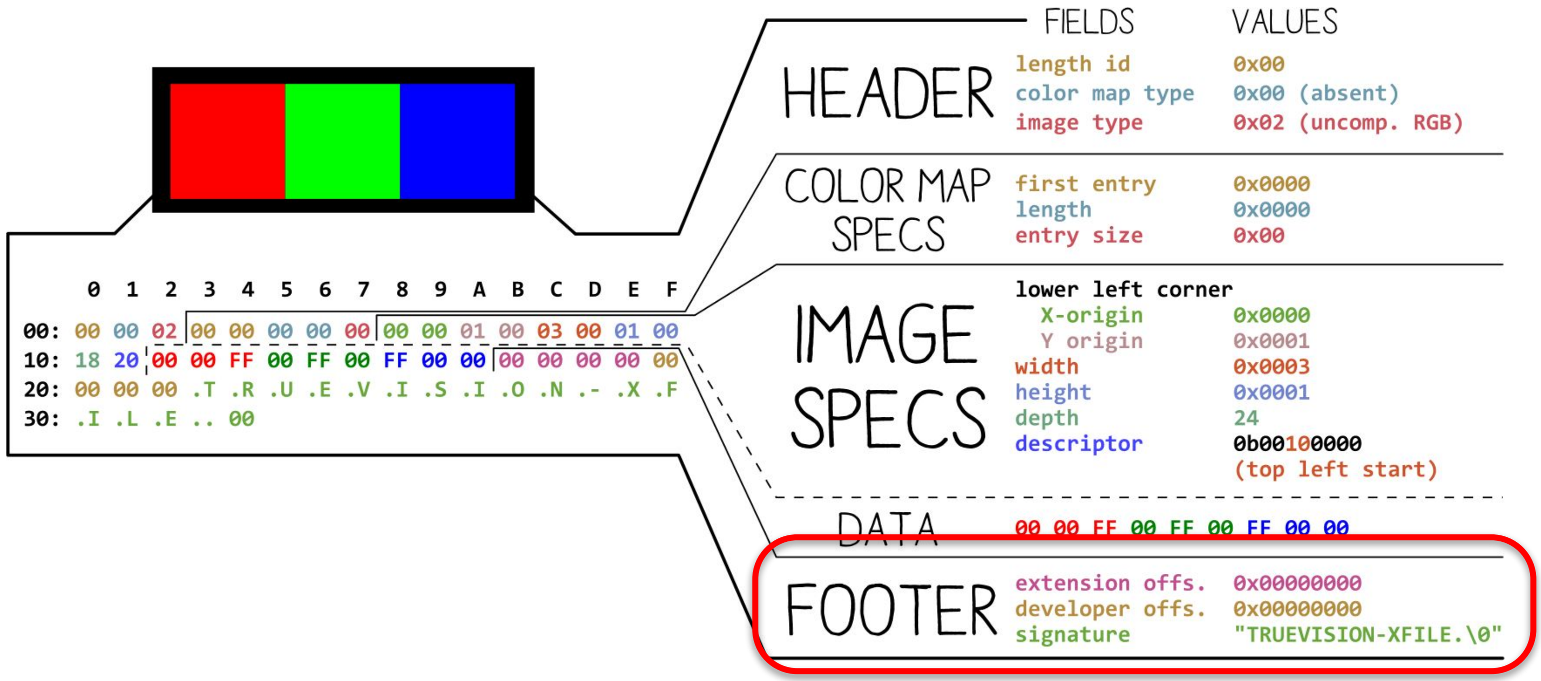
iOS

```
    +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
0x  89 .P .N .G 0D 0A 1A 0A 00 00 00 04 .C .g .B .I
1x  50 00 20 02 2B D5 B3 7F 00 00 00 0D .I .H .D .R
2x  .. ..
```

~~Weird~~

```
    +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
0x  89 .P .N .G 0D 0A 1A 0A 80 00 13 37 .d .u .m .b
1x  ./ ./ .  .  .p .a .y .l .o .a .d .  .  .  ./ \n
2x  .. ..
    +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
```

Magika is only trained on standard files.



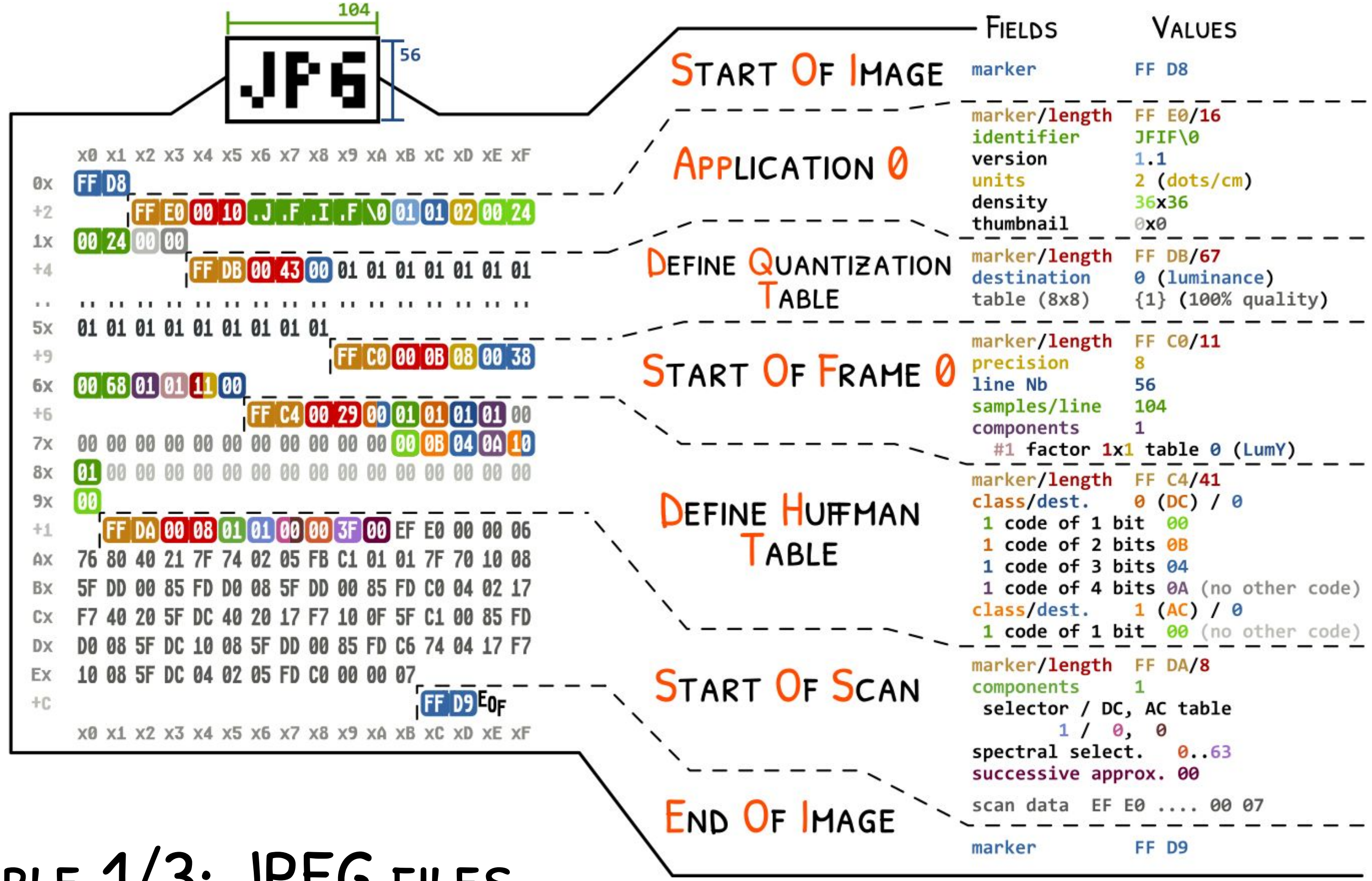
```
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00: 00 00 02 00 00 00 00 00 00 00 01 00 03 00 01 00
10: 18 20 00 00 FF 00 FF 00 FF 00 00 00 00 00 00 00
20: 00 00 00 .T .R .U .E .V .I .S .I .O .N .- .X .F
30: .I .L .E .. 00
```

|  | FIELDS | VALUES |
|---|---|---|
| HEADER | length id | 0x00 |
|  | color map type | 0x00 (absent) |
|  | image type | 0x02 (uncomp. RGB) |
| COLOR MAP SPECS | first entry | 0x0000 |
|  | length | 0x0000 |
|  | entry size | 0x00 |
| IMAGE SPECS | lower left corner |  |
|  | X-origin | 0x0000 |
|  | Y origin | 0x0001 |
|  | width | 0x0003 |
|  | height | 0x0001 |
|  | depth | 24 |
|  | descriptor | 0b00100000 (top left start) |
| DATA | 00 00 FF 00 FF 00 FF 00 00 |  |
| FOOTER | extension offs. | 0x00000000 |
|  | developer offs. | 0x00000000 |
|  | signature | "TRUEVISION-XFILE.\0" |

## ImagikaTragika (HitCon CTF 24, 1 solve): bypassing Magika by appending a TGA footer.

## Examples

How can file identifiers handle common cases?



```
    x0 x1 x2 x3 x4 x5 x6 x7 x8 x9 xA xB xC xD xE xF
0x  FF D8
+2     FF E0 00 10 .J .F .I .F \0 01 01 02 00 24
1x  00 24 00 00
+4           FF DB 00 43 00 01 01 01 01 01 01 01
..  .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
5x  01 01 01 01 01 01 01 01 01
+9                             FF C0 00 0B 08 00 38
6x  00 68 01 01 11 00
+6                    FF C4 00 29 00 01 01 01 01 00
7x  00 00 00 00 00 00 00 00 00 00 00 00 0B 04 0A 10
8x  01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
9x  00
+1     FF DA 00 08 01 01 00 00 3F 00 EF E0 00 00 06
Ax  76 80 40 21 7F 74 02 05 FB C1 01 01 7F 70 10 08
Bx  5F DD 00 85 FD D0 08 5F DD 00 85 FD C0 04 02 17
Cx  F7 40 20 5F DC 40 20 17 F7 10 0F 5F C1 00 85 FD
Dx  D0 08 5F DC 10 08 5F DD 00 85 FD C6 74 04 17 F7
Ex  10 08 5F DC 04 02 05 FD C0 00 00 07
+C                                      FF D9 EOF
    x0 x1 x2 x3 x4 x5 x6 x7 x8 x9 xA xB xC xD xE xF
```

| | Fields | Values |
|---|---|---|
| Start Of Image | marker | FF D8 |
| Application 0 | marker/length | FF E0/16 |
| | identifier | JFIF\0 |
| | version | 1.1 |
| | units | 2 (dots/cm) |
| | density | 36x36 |
| | thumbnail | 0x0 |
| Define Quantization Table | marker/length | FF DB/67 |
| | destination | 0 (luminance) |
| | table (8x8) | {1} (100% quality) |
| Start Of Frame 0 | marker/length | FF C0/11 |
| | precision | 8 |
| | line Nb | 56 |
| | samples/line | 104 |
| | components | 1 |
| | #1 factor 1x1 table 0 (LumY) | |
| Define Huffman Table | marker/length | FF C4/41 |
| | class/dest. | 0 (DC) / 0 |
| | 1 code of 1 bit | 00 |
| | 1 code of 2 bits | 0B |
| | 1 code of 3 bits | 04 |
| | 1 code of 4 bits | 0A (no other code) |
| | class/dest. | 1 (AC) / 0 |
| | 1 code of 1 bit | 00 (no other code) |
| Start Of Scan | marker/length | FF DA/8 |
| | components | 1 |
| | selector / DC, AC table | 1 / 0, 0 |
| | spectral select. | 0..63 |
| | successive approx. | 00 |
| | scan data | EF E0 .... 00 07 |
| End Of Image | marker | FF D9 |

# Example 1/3: JPEG files

## Example: JPEG files

```
FF D8 FF E0 00 10 .J .F .I .F 00 01 ?? ?? ?? ??

FF D8 FF E1 ?? ?? .E .x .i .f 00 00 .I .I 2A 00

FF D8 FF E1 ?? ?? .E .x .i .f 00 00 .M .M 00 2A
```

Standard JPEG Headers:

- Starts with FF D8 signature.
  - Always starts with "FF D8 FF"
- "JFIF" or "Exif" at offset 6.
  - In this case, "II" or "MM" at offset 14 (TIFF-like Exif)

Common contents:

- FF D8 FF at 0 (always, correct)
- JFIF or Exif strings usually at 6 (but not necessarily).

-> Multiple patterns are required + potentially "confusing" strings.

Very basic & prone to FPs!

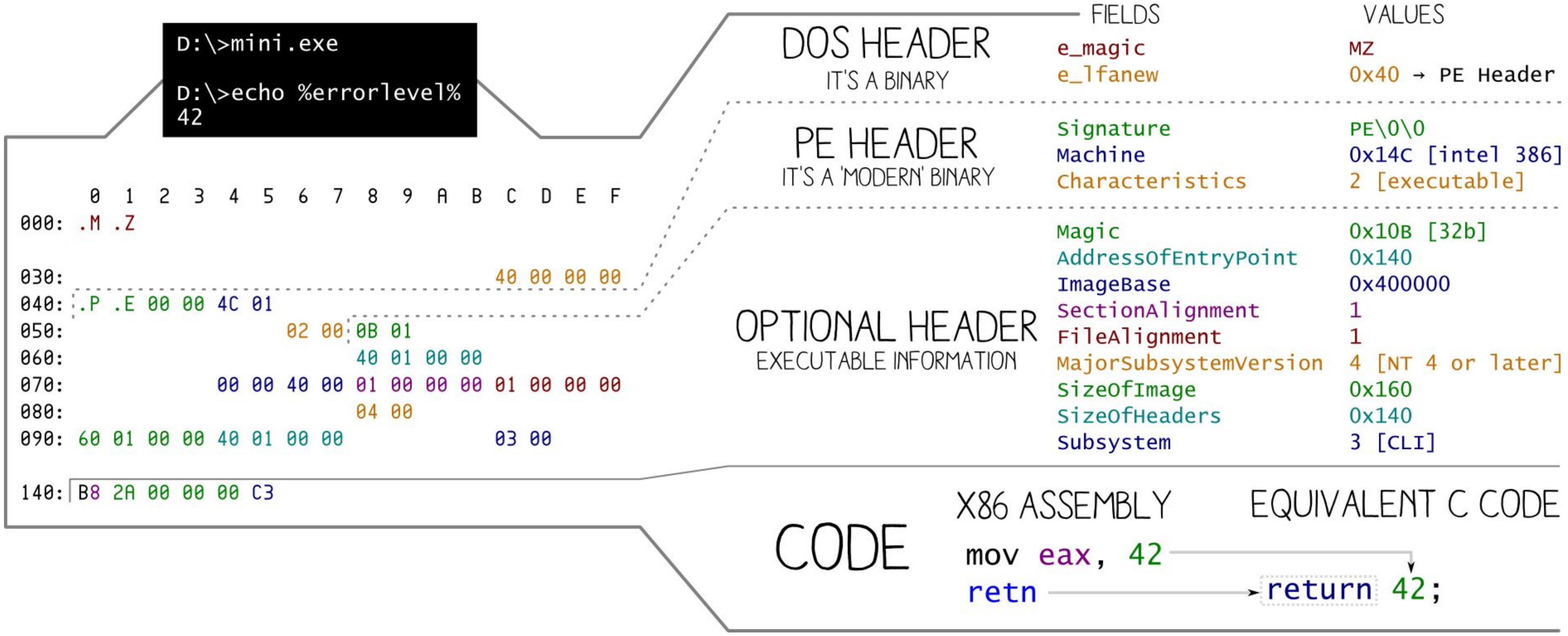
```
[50:image/jpeg]
>0=\x00\x03\xFF\xD8\xFF
>0=\x00\x02\xFF\xD8
```

## Parse JPEG files w/ Share-Mime

```
<FrontBlock>
   <Pattern>
      <Bytes>FFD8FF</Bytes>
      <Pos>0</Pos>
   </Pattern>
</FrontBlock>
<GlobalStrings>
   <String>EXIF''II*'</String>
   <String>EXIF''MM'*</String>
   <String>JFIF</String>
</GlobalStrings>
```

The strings could be anywhere!

## Parse JPEG files w/ TrID



```
D:\>mini.exe

D:\>echo %errorlevel%
42
```

```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000: .M .Z

030:                                     40 00 00 00
040: .P .E 00 00 4C 01
050:                   02 00 0B 01
060:                         40 01 00 00
070:             00 00 40 00 01 00 00 00 01 00 00 00
080:                         04 00
090: 60 01 00 00 40 01 00 00             03 00

140: B8 2A 00 00 00 C3
```

| | FIELDS | VALUES |
|---|---|---|
| DOS HEADER<br>IT'S A BINARY | e_magic | MZ |
| | e_lfanew | 0x40 → PE Header |
| PE HEADER<br>IT'S A 'MODERN' BINARY | Signature | PE\0\0 |
| | Machine | 0x14C [intel 386] |
| | Characteristics | 2 [executable] |
| OPTIONAL HEADER<br>EXECUTABLE INFORMATION | Magic | 0x10B [32b] |
| | AddressOfEntryPoint | 0x140 |
| | ImageBase | 0x400000 |
| | SectionAlignment | 1 |
| | FileAlignment | 1 |
| | MajorSubsystemVersion | 4 [NT 4 or later] |
| | SizeOfImage | 0x160 |
| | SizeOfHeaders | 0x140 |
| | Subsystem | 3 [CLI] |

CODE

| X86 ASSEMBLY | EQUIVALENT C CODE |
|---|---|
| mov eax, 42<br>retn | return 42; |

## Example 2/3: Microsoft executables

## Example 2/3: Microsoft executables

- "MZ" signature at offset 0
- 32b **pointer** at offset 0x3C
  - Points to a signature:
    - NE\0\0: Windows Bitmap Font (*.FON)
    - PE\0\0: Executables
    - Also, LE\0\0, LX\0\0, W3, W4

Signature at variable offsets:
-> needs a pointer operator + range scanning might fail

Ex: Windows 95's regedit.exe: the PE signature at offset 0x9548 (!)

```
[80:application/vnd.microsoft.portable-executable]
>0=\x00\x02MZ
1>64=\x00\x04PE\x00\x00+193
```

No pointers, only scanning.

## Parsing PE w/ Share-Mime

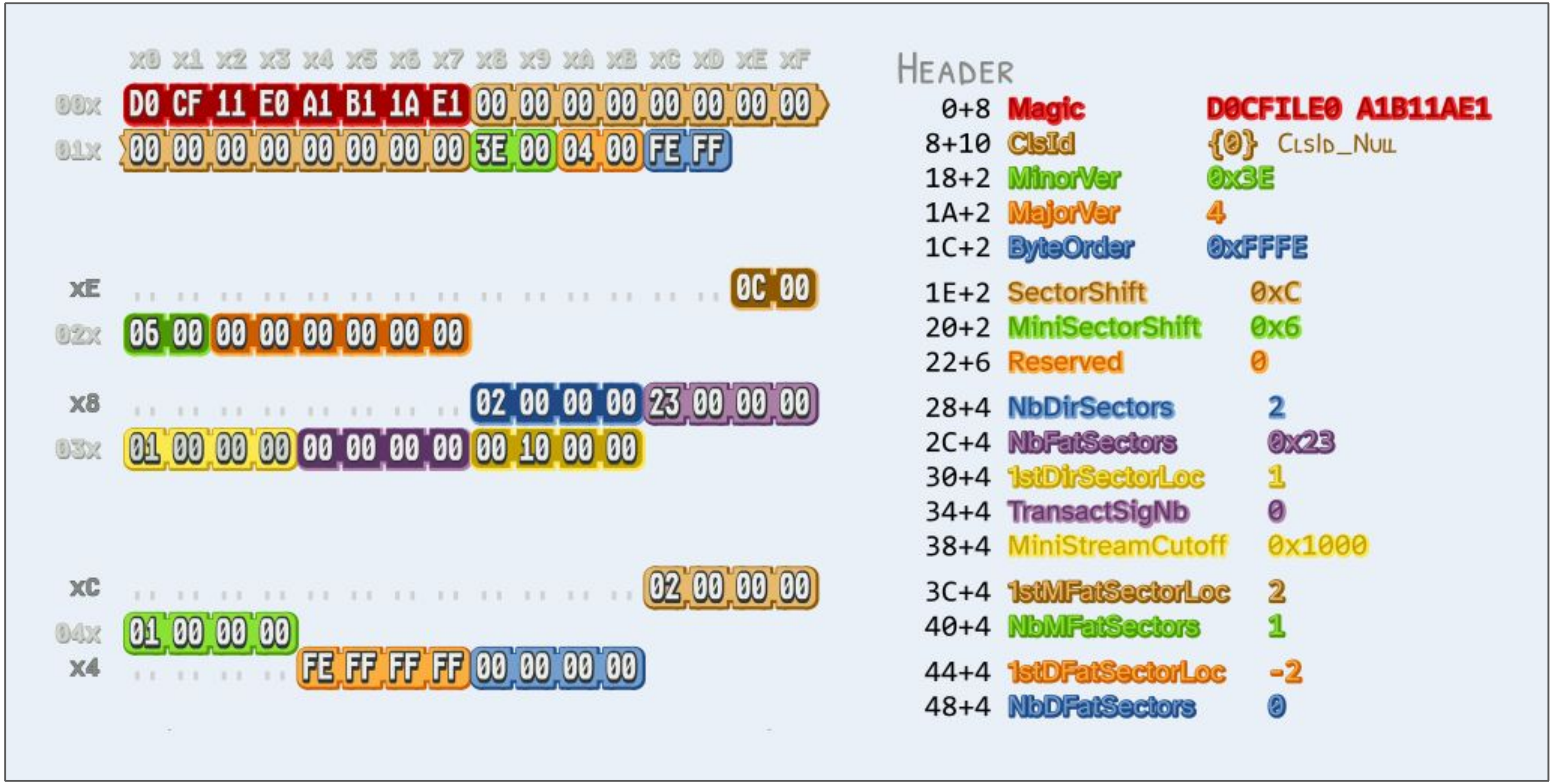
```
<FrontBlock>
   <Pattern>
      <Bytes>4D5A</Bytes>
      <ASCII> M Z</ASCII>
      <Pos>0</Pos>
   </Pattern>
</FrontBlock>
<GlobalStrings>
   <String>PE''</String>
   <String>THIS PROGRAM CANNOT BE RUN IN DOS MODE.</String>
</GlobalStrings>
```

## Parsing PE with TrID:

ONLY BYTE PATTERNS AT FIXED OFFSETS, AND STRINGS.

```
0            string  MZ            Executable
>(0x3C.l)    string  NE\x00\x00    NE
>(0x3C.l)    string  PE\x00\x00    PE
```

## Parsing Microsoft executables w/ LibMagic



```
     x0 x1 x2 x3 x4 x5 x6 x7 x8 x9 xA xB xC xD xE xF
00x  D0 CF 11 E0 A1 B1 1A E1 00 00 00 00 00 00 00 00
01x  00 00 00 00 00 00 00 00 3E 00 04 00 FE FF

xE                                             0C 00
02x  06 00 00 00 00 00 00 00

x8                           02 00 00 00 23 00 00 00
03x  01 00 00 00 00 00 00 00 00 10 00 00

xC                                       02 00 00 00
04x  01 00 00 00
x4               FE FF FF FF 00 00 00 00
```

Header

| | | |
|---|---|---|
| 0+8 | Magic | D0CF11E0 A1B11AE1 |
| 8+10 | ClsId | {0} ClsId_Null |
| 18+2 | MinorVer | 0x3E |
| 1A+2 | MajorVer | 4 |
| 1C+2 | ByteOrder | 0xFFFE |
| 1E+2 | SectorShift | 0xC |
| 20+2 | MiniSectorShift | 0x6 |
| 22+6 | Reserved | 0 |
| 28+4 | NbDirSectors | 2 |
| 2C+4 | NbFatSectors | 0x23 |
| 30+4 | 1stDirSectorLoc | 1 |
| 34+4 | TransactSigNb | 0 |
| 38+4 | MiniStreamCutoff | 0x1000 |
| 3C+4 | 1stMFatSectorLoc | 2 |
| 40+4 | NbMFatSectors | 1 |
| 44+4 | 1stDFatSectorLoc | -2 |
| 48+4 | NbDFatSectors | 0 |

## Example 3/3

# Example 3/3: Office CFB files

A.k.a. OLE or "Doc" File

Container's easy identification: `D0 CF 11 E0` at offset `0`

Distinction between subformats:

- 16bits at offset `26`: `Version` (3 or 4)
  - if `v3: SectorSize = 512`
  - if `v4: SectorSize = 4096`
- 32bits at offset `48`: Number of sectors
- CLSID at offset `80` of the first sector ([60+ possible values]())

-> conditional paths

-> relative offsets, multiplication

-> many checks

| Software | CLSID |
|---|---|
| MSI | `{000c1084-0000-0000-c000-000000000046}` |
| Excel 5-95 | `{00020810-0000-0000-c000-000000000046}` |
| Autodesk Inventor | `{4D29B490-49B2-11D0-93C3-7E0706000000}` |

```
<FrontBlock>
   <Pattern>
      <Bytes>D0CF11E0A1B11AE1</Bytes>
      <Pos>0</Pos>
   </Pattern>
</FrontBlock>
```

Parse CFB files w/ TrID:

A complex format w/ no common patterns

No sub-formats differentiaton!

```
[50:application/x-ole-storage]
>0=\x00\x08\xD0\xCF\x11\xE0\xA1\xB1\x1A\xE1
>0=\x00\x04\xD0\xCF\x11\xE0
```

## Parse CFB files w/ Share-Mime (standard sigs)

One rule per signature.

```
rule cfb
{
 strings:
   $_docfile = { d0 cf 11 e0 a1 b1 1a e1 }
   $clsidMSI = { 84 10 0C 00 00 00 00 00 c0 00 00 00 00 00 00 46 }
   $clsidXLS = { 10 08 02 00 00 00 00 00 c0 00 00 00 00 00 00 46 }
 condition:
   $_docfile at 0
   and (
   (uint8(26) == 3 and any of ($clsid*) at ((uint32(48) + 1) * 512 + 80)) or
   (uint8(26) == 4 and any of ($clsid*) at ((uint32(48) + 1) * 4096 + 80))
   )
}
```

## Parse CFB files w/ Yara:

## a rule can only return true/false.

```
0               bequad  0xd0cf11e0a1b11ae1   CFB

>26             leshort 0x03                 v%d
>>(48.l*512)    default x
>>>&512         use     clsid-check

>26             leshort 0x04                 v%d
>>(48.l*4096)   default x
>>>&4096        use     clsid-check


0       name    clsid-check
>&80    string  \x84\x10\x0c\x00\x00\x00\x00\x00\xc0\x00\x00\x00\x00\x00\x00\x46   MSI
>&80    string  \x10\x08\x02\x00\x00\x00\x00\x00\xc0\x00\x00\x00\x00\x00\x00\x46   XLS
```

Intermediary information

Always true

## Parse CFB files w/ LibMagic: information extraction, 'functions'

## Failing detections

Quick & fast scanning
leads to easy abuse!

## Strategies

1. Avoid detection:
   - corner case
   - abuse specifications
   - extreme case: put signature out of scanning range.

2. Force misdetection: insert contents to influence the result.
   Insert signature or just fuzz until the detection verdict has changed.
   Scanning order of engine is important.

## Keep functionality and insert dummy space

Some formats give you full control over the first X bytes.
Some make it possible to insert exploitable contents early.

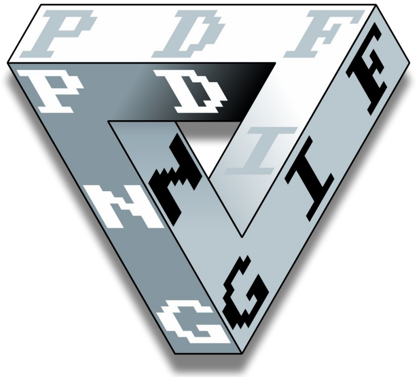
Use Mitra to insert 1 kb of free space in your file:

```
mitra.py <inputfile> /dev/null --pad 1 -f
```

Use Mocky to insert dummy signatures:

```
mocky.py <inputfile> --combined
```

Mocky & Mitra @ [Github corkami/mitra]



# A POLYMOCK – A 190-IN-1 YET EMPTY FILE

The file is mostly empty!
It only contains magics to fake file types.

https://github.com/corkami/pocs/tree/master/polymocks

```
multi: Windows Program Information File for \030(o\001
- MAR Area Detector Image,
- Linux kernel x86 boot executable RW-rootFS,
- ReiserFS V3.6
- Files-11 On-Disk Structure (ODS-52); volume label is '            '
- DOS/MBR boot sector
- Game Boy ROM image (Rev.00) [ROM ONLY], ROM: 256Kbit
- Plot84 plotting file
-  DOS/MBR boot sector
- DOSFONT2 encrypted font data
- Kodak Photo CD image pack file , landscape mode
- SymbOS executable v., name: HNRO0\334\247\304\375]\034\236\243
- ISO 9660 CD-ROM filesystem data (raw 2352 byte sectors)
- Nero CD image at 0x4B000 ISO 9660 CD-ROM filesystem data
- High Sierra CD-ROM filesystem data
- Old EZD Electron Density Map
- Apple File System (APFS), blocksize 24061976
- Zoo archive data, modify: v78.88+
- Symbian installation file
- 4-channel Fasttracker module sound data Title: "MZ`\352\210\360'\315!"
- Scream Tracker Sample adlib drum mono 8bit unpacked
- Poly Tracker PTM Module Title: "MZ`\352\210\360'\315!"
- SNDH Atari ST music
- SoundFX Module sound file
- D64 Image
- Nintendo Wii disc image: "NXSB\030(o\001
- Nintendo 3DS File Archive (CFA) (
- Unix Fast File system [v1] (litt                          ...
- Unix Fast File system [v2] (litt                     on , ...
- Unix Fast File system [v2] (littl              mounted on , …
- ISO 9660 CD-ROM filesystem data (           boot sector)
- F2FS filesystem, UUID=00000000-0000-0000-0000-000000000000, volume name ""
- DICOM medical imaging data
- Linux kernel ARM boot executable zImage (little-endian)
- CCP4 Electron Density Map
- Ultrix core file from 'X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVI...
- VirtualBox Disk Image (MZ`\352\210\360'\315!), 5715999566798081280 bytes
- MS Compress archive data
- AMUSIC Adlib Tracker MS-DOS executable, MZ for MS-DOS COM executable for DOS
- JPEG 2000 image
- ARJ archive data
- unicos (cray) executable
- IBM OS/400 save file data
- data
```

output from
file --keep-going

```
00 .M .Z 60 EA .j .P 01 07 19 04 00 10 .S .N .D .H
10 .N .R .O .0 DC A7 C4 FD 5D 1C 9E A3 .R .E .~ .^
20 .N .X .S .B 18 28 6F 01 .P .K 03 04 .P .T .M .F
30 .S .y .m .E .x .e .7 .z BC AF 27 1C .S .O .N .G
40 7F 10 DA BE 00 00 CD 21 .P .K 01 02 .S .C .R .S
50 .R .a .r .! ^Z 07 01 00 .L .R .Z .I .P .L .O .T
60 .% .% .8 .4 .R .a .r .! ^Z 07 00 00 00 .M .A .P
70 .  .( FD .7 .z .X .Z 00 04 22 4D 18 03 21 4C 18
80 .D .I .C .M .% .P .D .F .- .1 .. .4 .  .o .b .j
...  ...
```

This file is simultaneously detected as:

- DOS EXE, COM and MBR
- Zoo, ARJ, VirtualBox, MS Compress, 3DS
- ISO, RAW ISO, Nero, PhotoCD
- FastTracker, ScreamTracker, Adlib tracker, Polytracker, SoundFX
- Apple, IBM, HP, Linux, Ultrix, Raid, ODS, Nintendo, Kodak
- EZD, CCP4, Plot84, MAR, Dicom

...

**Many** magics are at the start of the file.

```
0       0x0      Gameboy ROM,, [ROM ONLY], ROM: 256Kbit
80      0x50     RAR archive data, version 5.x
88      0x58     lrzip compressed data
89      0x59     rzip compressed data - versio
114     0x72     xz compressed data
120     0x78     LZ4 compressed data
...
```

output (150 sigs) from
Binwalk

6385..3d4c

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00  FF 54 41 47 4C 5A 2A 3F 2A 00 2A 00 53 4E 44 48   .TAGLZ*?* * SNDH
10  11 00 00 EF DC A7 C4 FD 00 00 4D 2A 2A 2A 00 00      ....  M***
20  01 03 2A 50 52 45 53 2A 2A 2A 2A 2A 2A 2A 2A 2A     *PRES*********
30  27 18 28 18                                       ' (.
```

| | |
|---|---|
| File type | Unknown |
| Magic | RISC OS AIF Executable |
| TrID | MegaZeux game (99.6%)<br>ZOO compressed archive (strict) (0.1%)<br>RISC OS AIF executable (0.1%)<br>HandStory eBook (0.1%)<br>Animatic Film (0.1%) |

## It even works across engines!

## Impact of old format w/ bad signatures

## PrintFox

The past **HAUNTS** us

```
     +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
0x   .G
        9B 4F 00 FF
                    FE

                       9B 07 00 FF
                                   0F
                                      9B 8A 00 FF
                                                  F9
1x   .. ..
     +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
```

G = Gesamtbild

| | |
|---|---|
| .G | SIGNATURE |
| 9B | RLE MARKER (9B) |
| 4F 00 | 4F LENGTH |
| FF | FF REPEATED VALUE |
| 9B | RLE MARKER (9B) |
| 07 00 | 07 LENGTH |
| FF | FF REPEATED VALUE |
| 9B | RLE MARKER (9B) |
| 8A 00 | 8A LENGTH |
| FF | FF REPEATED VALUE |

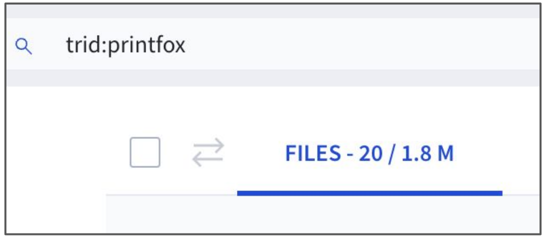
## A GENUINE PRINTFOX FILE: avanger.gb

## PrintFox FP via TrID

A C64 image format from the 1980s.

The file structure is just a single letter signature,
then pure RLE data. Cf C64-Wiki
A bad structure, but a sign of the times.

-> *many* FPs - 1.8 M files on VirusTotal.
Yet only a **handful** of actual PrintFox files.


trid:printfox

FILES - 20 / 1.8 M

## Conclusion

## Different engines & KB w/ different goals

All double-edged swords.

Fixed offsets / pointers / range scanning…

Extendable? Binary patterns or ML-powered.

Extract information? Quality of the signatures?

They can all be fooled to some extend.

KB and signatures of various quality and scope.

Pick one: Fast or in-depth scanning

## ABUSING FILE TYPES DETECTIONS CAN BE TRIVIAL.

1. Make free space (w/ Mitra)
2. Insert mock signatures (w/ Mocky) or fuzz

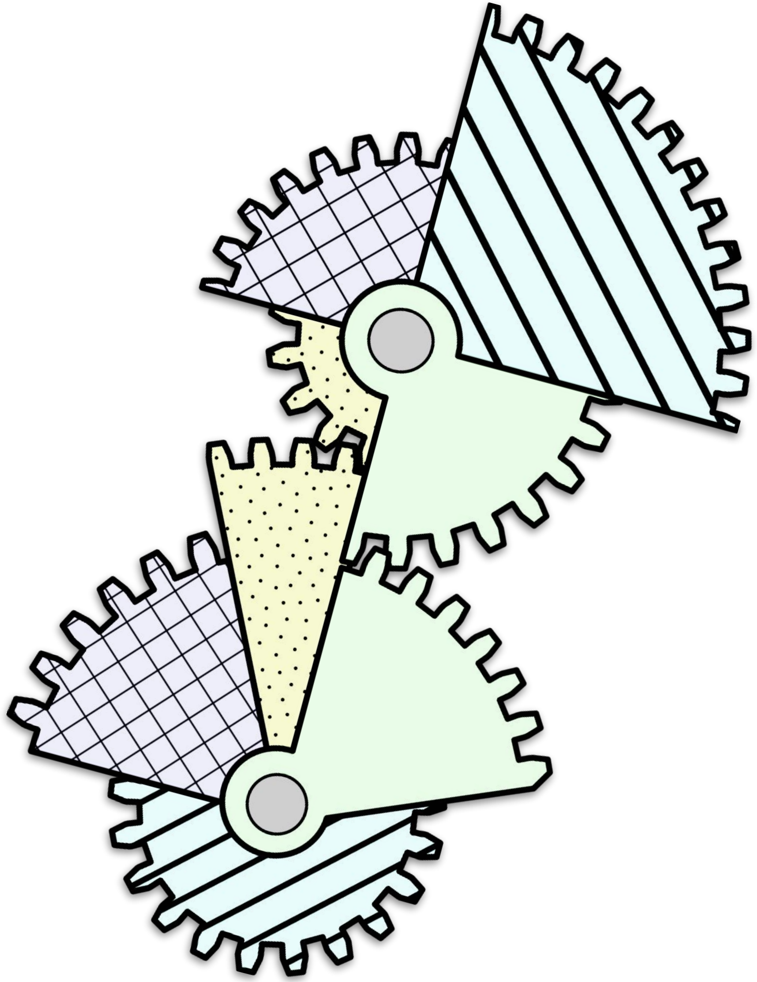
# ML CHANGES THE GAME IN FILE FORMAT FILTERING.

Outperforms existing solutions. Used in production.
Solves new formats overlap. Not a deep scanner.

Many new leads to explore!

# Thank you!

Any feedback is welcome!



## Extra slides

## Formats conflicts

Extensions:

- .s: assembly source
  .S: *preprocessed* assembly source
- .m: matlab or Objective-C ?
- .3ds: Nintendo 3ds or 3d Studio?
- .dm: DreamMaker or Dominion Mods?

## Troublesome formats

No magic:

- Pickle (ML models)
- Protobuf
- MP3 (frames-only), Minecraft, STL…

Tiny magic signature:

- PrintFox & many others…

Footer-only (such a bad idea!):

- TGA, QOP

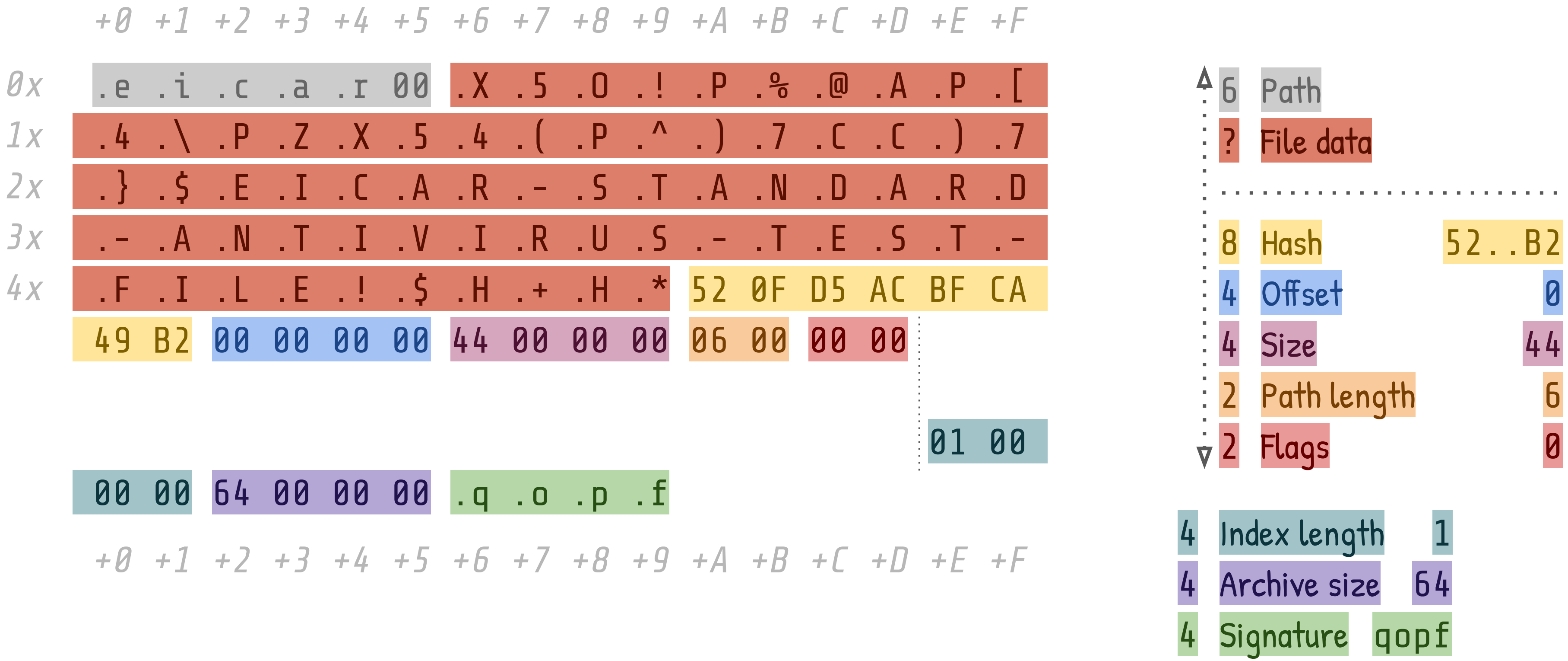
Hash: 028d33d7fd40eaa61d38bea93325a7e88f03e929c193f04c0cacddb3c0a15c2c

```
     +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F

0x   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
..   .. ..
4x   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5x   0C 00 00 00
+4               00 00 00 00 00 00 00 00 00 00 80 3F
6x   FF FF DB C2 FE FF DB C2 C7 CC 4C 3E FF FF DB 42
7x   FE FF DB C2 C7 CC 4C 3E FF FF DB C2 04 00 DC 42
8x   C7 CC 4C 3E 00 00
                         .. ..
     +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
```

| | |
|---|---|
| 80 | Header |
| 4 | Number of triangles |
| 12 | Normal vector |
| 12 | Vertex 1 |
| 12 | Vertex 2 |
| 12 | Vertex 3 |
| 02 | Attribute byte count |

## A Binary STL file: no signature, just data.



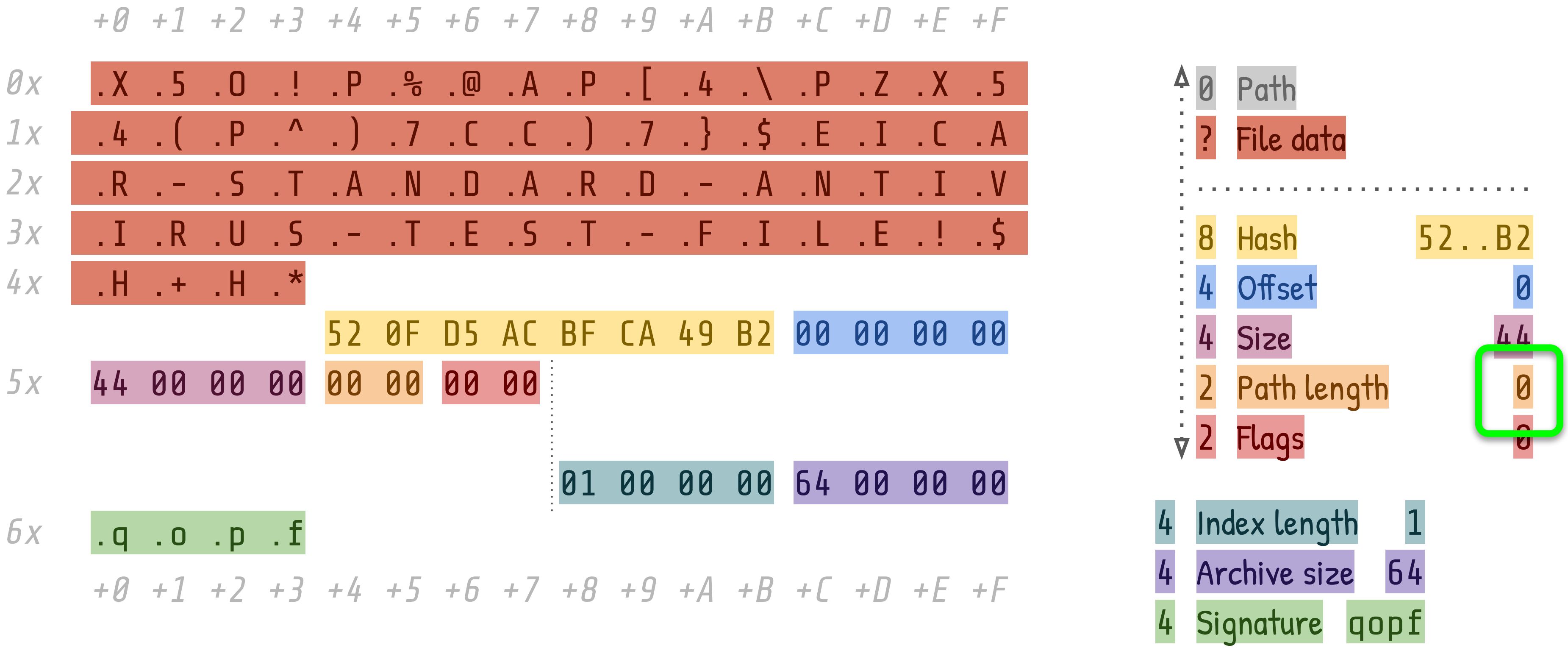
```
    +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
0x  .e .i .c .a .r 00 .X .5 .O .! .P .% .@ .A .P .[
1x  .4 .\ .P .Z .X .5 .4 .( .P .^ .) .7 .C .C .) .7
2x  .} .$ .E .I .C .A .R .- .S .T .A .N .D .A .R .D
3x  .- .A .N .T .I .V .I .R .U .S .- .T .E .S .T .-
4x  .F .I .L .E .! .$ .H .+ .H .* 52 0F D5 AC BF CA
    49 B2 00 00 00 00 44 00 00 00 06 00 00 00

                                          01 00
    00 00 64 00 00 00 .q .o .p .f
    +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
```

| | | |
|---|---|---|
| 6 | Path | |
| ? | File data | |
| 8 | Hash | 52..B2 |
| 4 | Offset | 0 |
| 4 | Size | 44 |
| 2 | Path length | 6 |
| 2 | Flags | 0 |

| | | |
|---|---|---|
| 4 | Index length | 1 |
| 4 | Archive size | 64 |
| 4 | Signature | qopf |

# A footer-based QOP archive: github phoboslab/qop



```
     +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
0x   .X .5 .O .! .P .% .@ .A .P .[ .4 .\ .P .Z .X .5
1x   .4 .( .P .^ .) .7 .C .C .) .7 .} .$ .E .I .C .A
2x   .R .- .S .T .A .N .D .A .R .D .- .A .N .T .I .V
3x   .I .R .U .S .- .T .E .S .T .- .F .I .L .E .! .$
4x   .H .+ .H .*
                 52 0F D5 AC BF CA 49 B2 00 00 00 00
5x   44 00 00 00 00 00 00 00
                             01 00 00 00 64 00 00 00
6x   .q .o .p .f
     +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
```

| | | |
|---|---|---|
| 0 | Path | |
| ? | File data | |
| 8 | Hash | 52..B2 |
| 4 | Offset | 0 |
| 4 | Size | 44 |
| 2 | Path length | 0 |
| 2 | Flags | 0 |
| 4 | Index length | 1 |
| 4 | Archive size | 64 |
| 4 | Signature | qopf |

## A path-less QOP archive:

## The beginning is undistinguishable from another file.

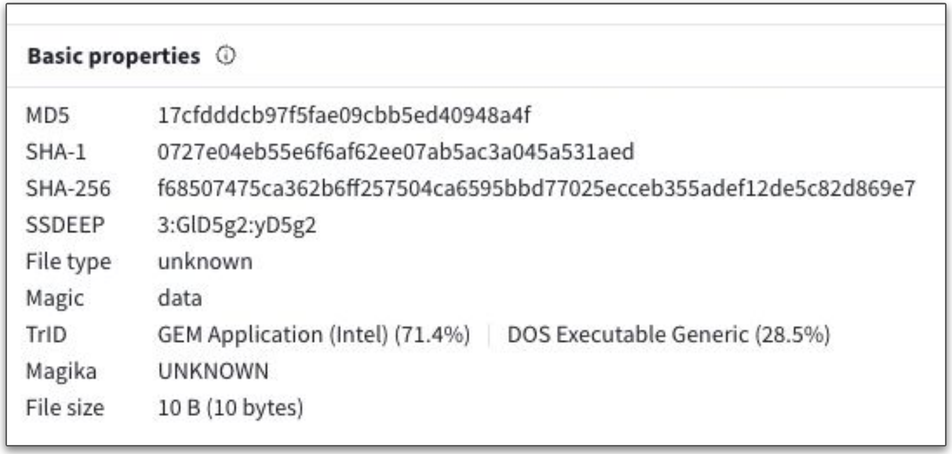
```
    +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
0x  .M .Z 00 00 .G .E .M .A .E .S
    +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
```

🟥 Fake DOS signature

🟨 Fake Binary file

🟦 GEM signature

File on VT



**Basic properties**

| | |
|---|---|
| MD5 | 17cfdddcb97f5fae09cbb5ed40948a4f |
| SHA-1 | 0727e04eb55e6f6af62ee07ab5ac3a045a531aed |
| SHA-256 | f68507475ca362b6ff257504ca6595bbd77025ecceb355adef12de5c82d869e7 |
| SSDEEP | 3:GlD5g2:yD5g2 |
| File type | unknown |
| Magic | data |
| TrID | GEM Application (Intel) (71.4%) \| DOS Executable Generic (28.5%) |
| Magika | UNKNOWN |
| File size | 10 B (10 bytes) |

## A Fake TrID detection

```
...
<InternalSignature ID="123" Specificity="Specific">
    <ByteSequence Reference="BOFoffset">
        <SubSequence MinFragLength="0" Position="1"
            SubSeqMaxOffset="0" SubSeqMinOffset="0">
            <Sequence>255044462D312E30</Sequence>
            <DefaultShift>9</DefaultShift>
            <Shift Byte="25">8</Shift>
            <Shift Byte="2D">4</Shift>
            <Shift Byte="2E">2</Shift>
            <Shift Byte="30">1</Shift>
            <Shift Byte="31">3</Shift>
            <Shift Byte="44">6</Shift>
            <Shift Byte="46">5</Shift>
            <Shift Byte="50">7</Shift>
        </SubSequence>
    </ByteSequence>
...
```

Beginning of File

"%PDF-1.0"

Each byte again...

## A fragment of a DROID signature for PDF files