Fearsome File Formats

3803

ILLEGAL INSTRUCTIONS

ANGE ALBERTINI

Fearsome File Formats

Ange Albertini

38C3
28/12/2024

About the author

My own views and opinions.

- Looking at hex editors for 35 years.
- Malware analyst for 20 years: Symantec, Avira, Google.
- Corkami: posters, PoCs, tools, tutorials (15k ⭐).
- CPS2Shock, PoC||GTFO…

Pixel art by Squiblydoo (2023)

CPS2 SHOCK
CAPCOM CPS II SYSTEM

cps2shock.emu-france.info

PoC||GTFO
Proof of Concept or Get the Fuck Out

github / angea / pocorgtfo

Let's look at a file...

...what do you think?

(YES, IT'S EMPTY)...

Can an empty file be useful?

Can one find purpose in emptiness ? 🤔

Besides:
- crashing code in production,
- stopping malware installation,
- shutting down botnets,

...

"THERE'S **NO** WAY
THE EMPTY FILE
COULD BE USED
IN STANDARD !"

/bin/true

USED TO BE EMPTY.

An empty shell script. Standard in every system.
It always works, and saves space.
It even became copyrighted despite its empty payload.

Nowadays, `/bin/true` is an ELF binary.

```
$ touch test
$ chmod +x test
$ ./test
$ echo $?
0
```

In Doom WADs,
empty files are used
as map index
in the archive table:
E1M1, …

VOID IS HERE

MS-Dos 1.25 added a size check in 1982.

```
The IBM Personal Computer DOS
Version 1.00 (C)Copyright IBM Corp 1981

A>DEBUG EMPTY.COM
File not found
-w
-q

A>DIR EMPTY.COM
EMPTY    COM          0  01-01-80

A>_
```

```
A>DIR TIME.COM
TIME     COM        250  08-04-81

A>TIME
Current time is 18:24:41.81
Enter new time:
A>DIR EMPTY.COM
EMPTY    COM          0  01-01-80

A>EMPTY
Current time is 18:24:53.27
Enter new time:
A>_
```


```
CP/M 2.2 - Amstrad Consumer Electronics
plc

A>ED EMPTY.COM

NEW FILE
     : *e

A>STAT EMPTY.COM

 Recs  Bytes  Ext Acc
    0     0k    1 R/W A:EMPTY.COM
Bytes Remaining On A: 5k

A>EMPTY EMPTY.COM

 Recs  Bytes  Ext Acc
    0     0k    1 R/W A:EMPTY.COM
Bytes Remaining On A: 5k

A>
```

Under IBM PC-DOS 1.0 and CP/M,

launching an empty file will just re-run the last one:

the memory wasn't cleared between executions.

So the empty file is... 

- A standard system shell script
  that always executes successfully.
- An index in Doom archives.
- A commercial (!) DOS executable
  that repeats the last command.

(among possibly many other things)

No content, and yet...

THE TYPE, CONTEXT AND PURPOSE ARE ALREADY UNCLEAR.

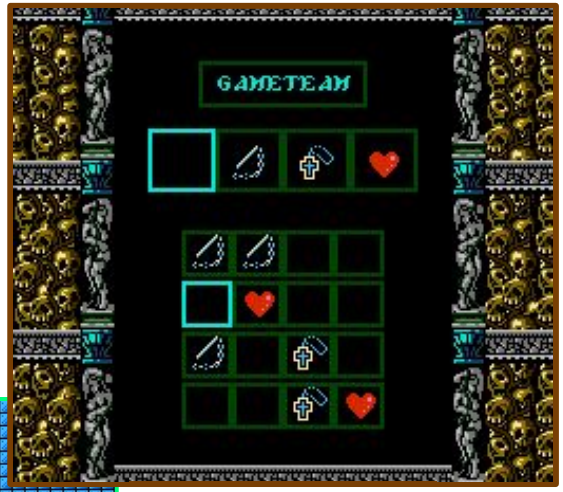
A file is more than its content.
Context and metadata are important.

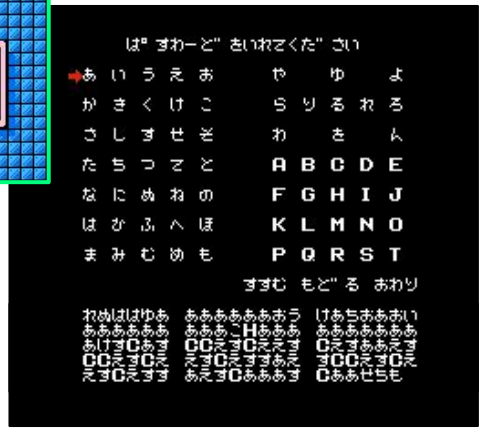
Let's look at something else...

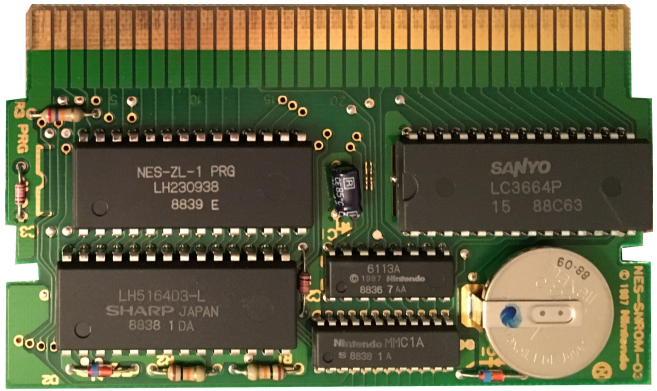
WHAT ARE 'FILES'?

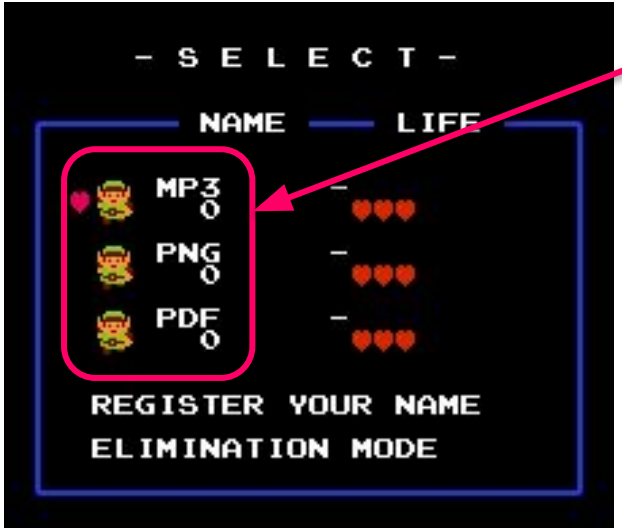
When storage was too expensive, games used to rely on long passwords to save your data!

Some aren't even in text!









00-09: 0-9
10-24: A-Z

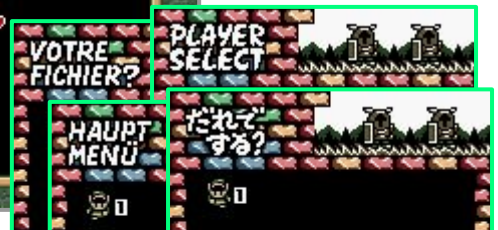
```
00:  16 19 03  .$ .$ .$ .$ .$
08:  19 17 10  .$ .$ .$ .$ .$
10:  19 0D 0F  .$ .$ .$ .$ .$
18:  00 00 00 00 00 00 00 00
20:  00 00 00 00 00 00 00 00
28:  00 00 00 00 00 00 00 00
30:  22 FF 00 00 00 00 00 00
38:  00 00 00 00 00 08 00 00
40:  00 00 00 00 00 00 00 00
48:  00 00 00 00 00 00 00 00
50:  00 00 00 00 00 00 00 00
58:  22 FF 00 00 00 00 00 00
60:  00 00 00 00 00 08 00 00
68:  00 00 00 00 00 00 00 00
70:  00 00 00 00 00 00 00 00
78:  00 00 00 00 00 00 00 00
80:  22 FF 00 00 00 00 00 00
88:  00 00 00 00 00 08 00 00
```

Saving games in 1986: hardcoded offsets in SRAM.

1986
the Legend of Zelda

1987
the Adventure of Link

1991
A Link to the Past



1993
Link's Awakening.

1998: Ocarina of Time

2001: Oracles of...

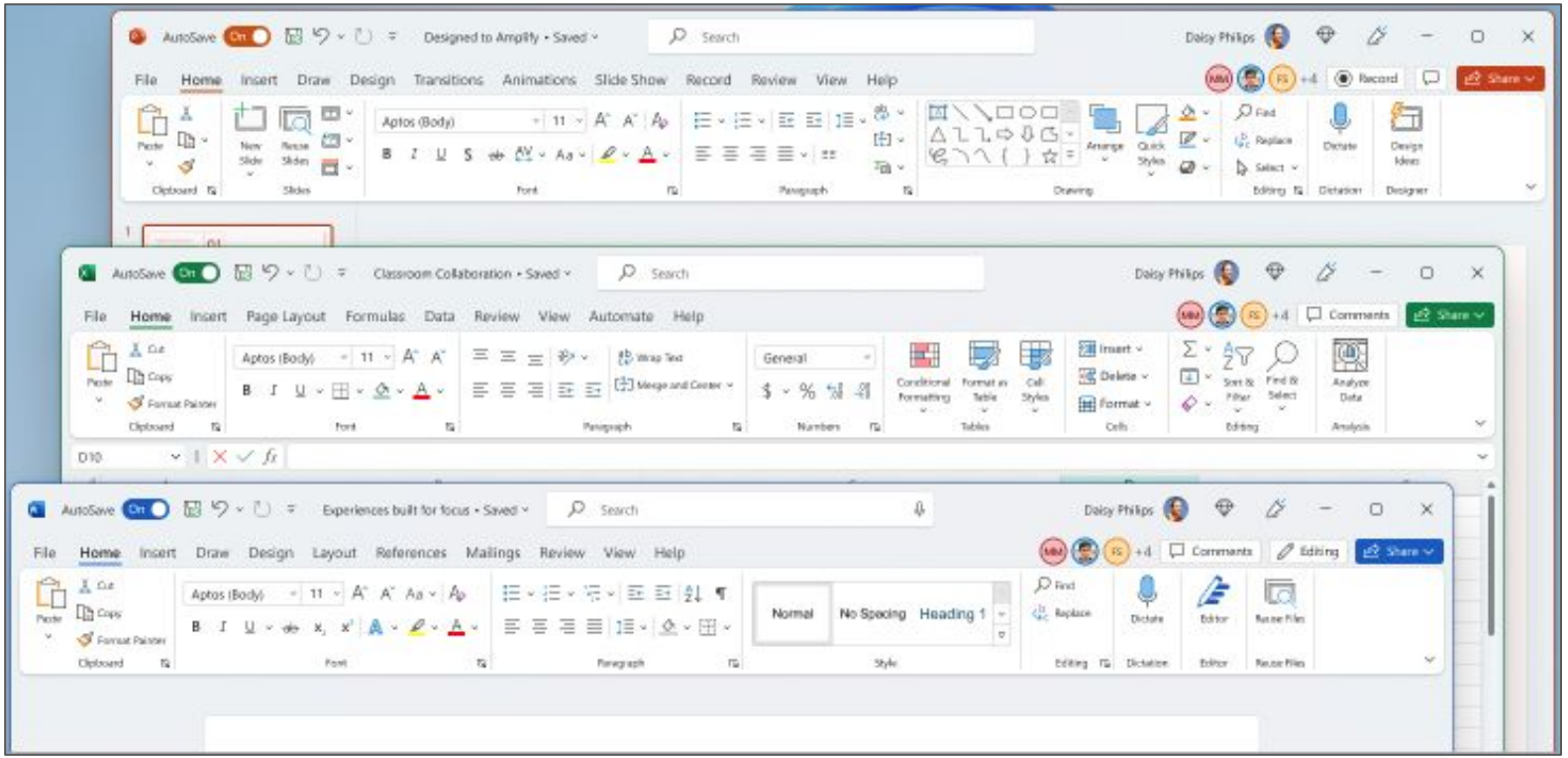
2004
The Minish Cap

From "player" to "file".

What's a file **without** a format?

What does that even mean?

Why would you do that?



Microsoft Office...

Microsoft

The logo since 2012.

(1982-1987)

Microsoft ~~Office~~ Word, in 1984...

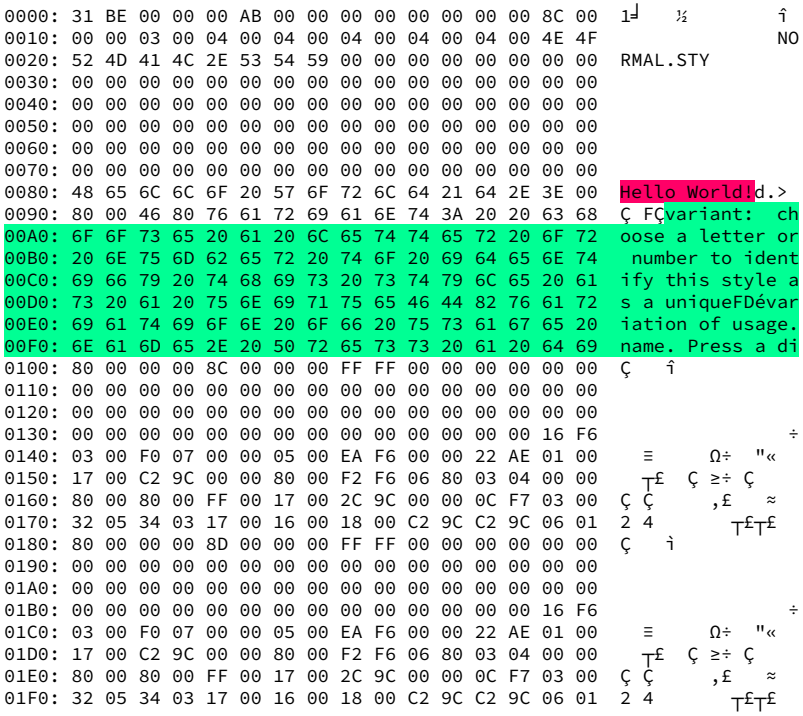
MICROSOFT

...DIDN'T USE A FILE FORMAT!

A sign of the times!

The whole memory page
was saved as a file...
...with whatever else in memory!

Who needs standardization
when you're just on your own?
It was just faster to snapshot
the memory range.



HELLOW.DOC

(512 BYTES FOR 12 BYTES OF TEXT!)

How do you reliably handle such files?

...

"What a mess!"

mustangstrombonehead**line**feed**back**hand**rail**road**side**show**down**turn**over**book**case**work**shop**

File format landscape 101:

Full of nasty surprises, exceptions, oddities, for historical or technical reasons.

We need to preserve file formats in a better way...

What's the  *of file formats?* 🤔

A decade later...

Some things haven't changed...

Ambiguous files (aka **werewolves** aka **parser differentials** aka **schizophrenic** files) are still there.

No reference parser, no test corpus.

Expensive specifications? -> devs don't pay for them!

And often, no real/serious specifications.

A simple example...

"Ange"

How do you pronounce this name ?

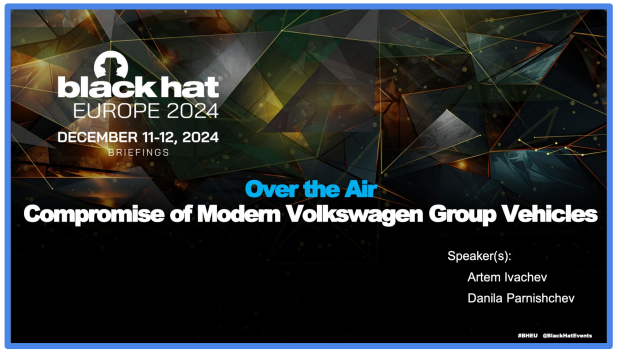
Anje, Enn-ji, An-gé, Anzu (杏), Enn-ré, Ąż...

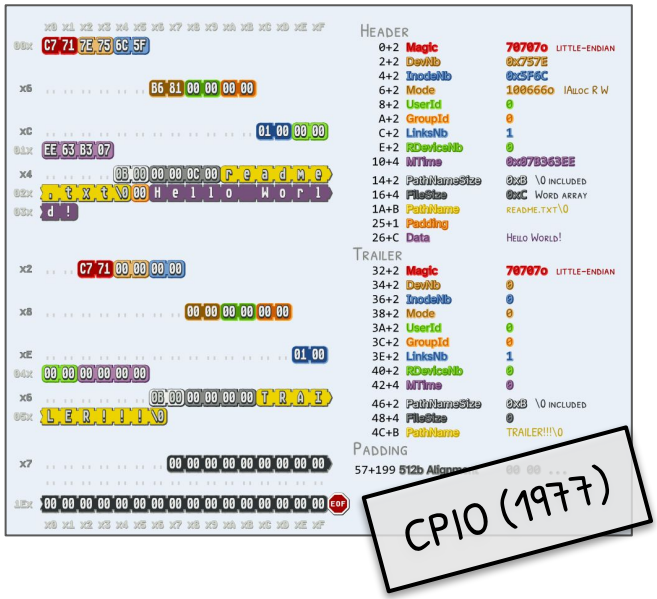
Male or female?

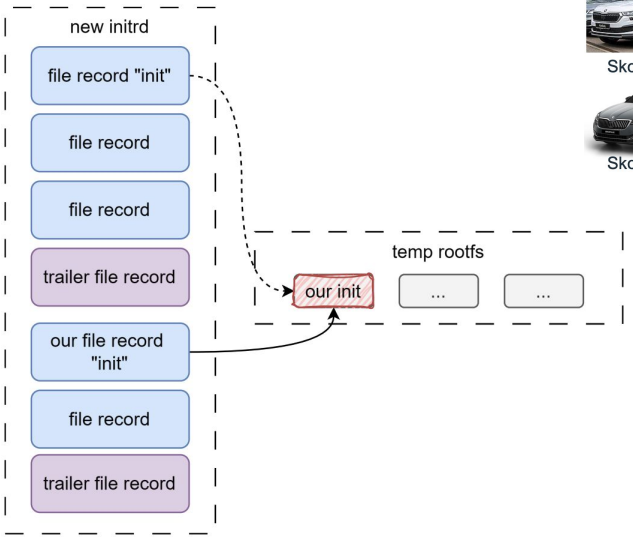
How many names are 'unpronunceable' ?
Without references, things quickly get messy.

Concatenation still works!

Duplicate file entry in a CPIO archive used to hack cars over the air, in 2024.



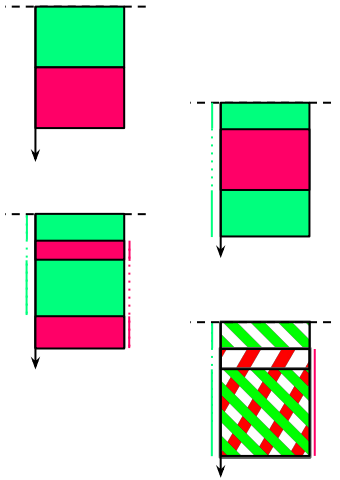


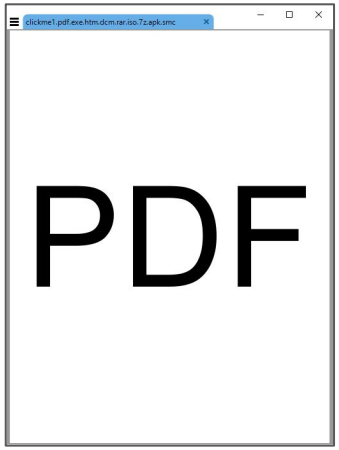


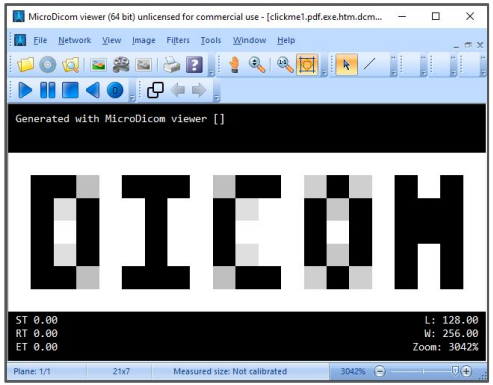
Multi-type / chameleon files, a.k.a.

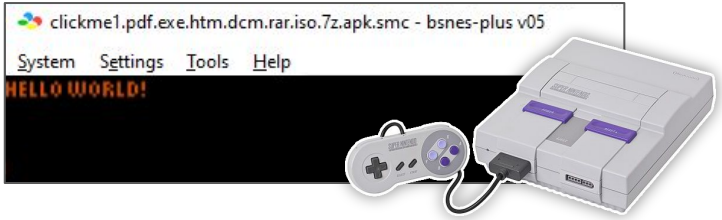
Polyglots

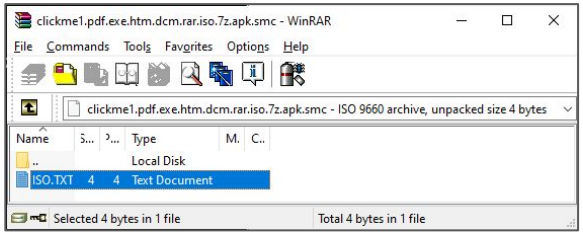
1. Concatenation (appended data)
2. Parasite (comment)
3. Zipper (mutual comments)
4. Chimera (shared data)



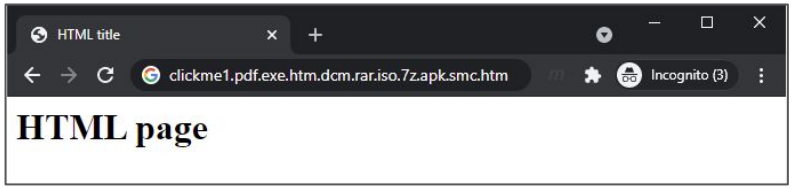


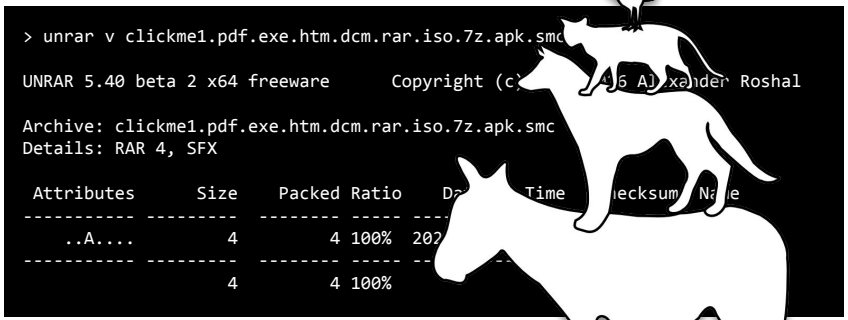







```
>clickme1.pdf.exe.htm.dcm.rar.iso.7z.apk.smc.exe
32-bit PE
```



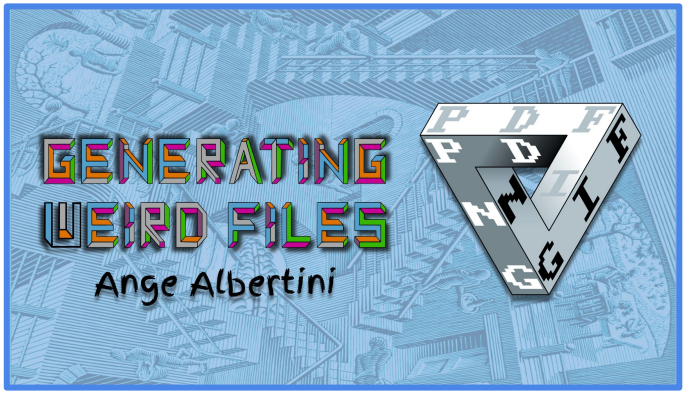


ClickMe (.PDF.EXE.HTM.DCM.RAR.ISO.7Z.APK.SMC)

Mitra

https://github.com/corkami/mitra

☆ Star 1.1k

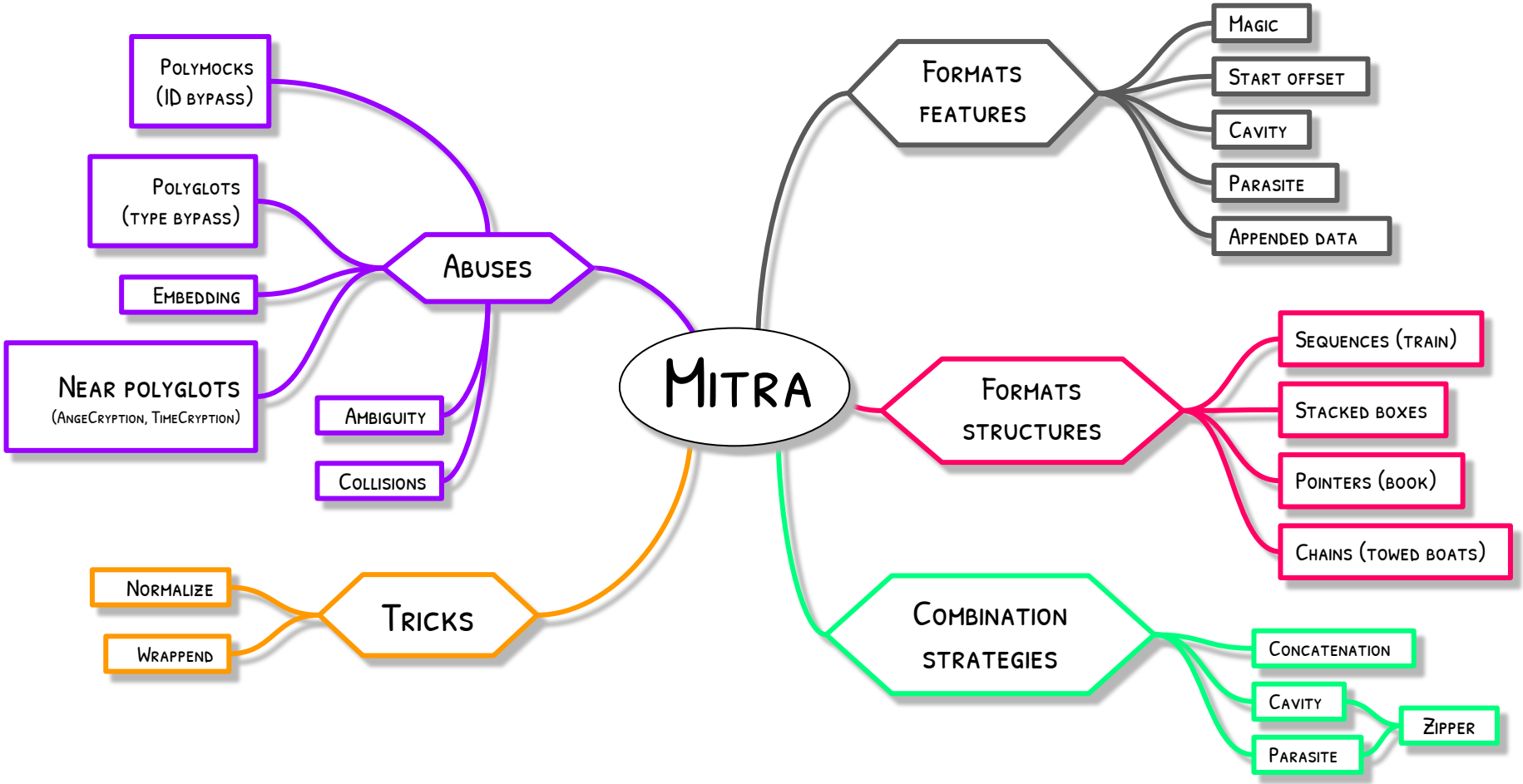


Identify file types, make space,
combine and adjust data.
It **should** keep the files valid:
no deep parsing, just the minimum.

```
$ mitra.py dicom.dcm png.png
dicom.dcm
File 1: DICOM / Digital Imaging and Communications in Medicine
png.png
File 2: PNG / Portable Network Graphics

Zipper Success!
Zipper: interleaving of File1 (type DCM) and File2 (type PNG)
```

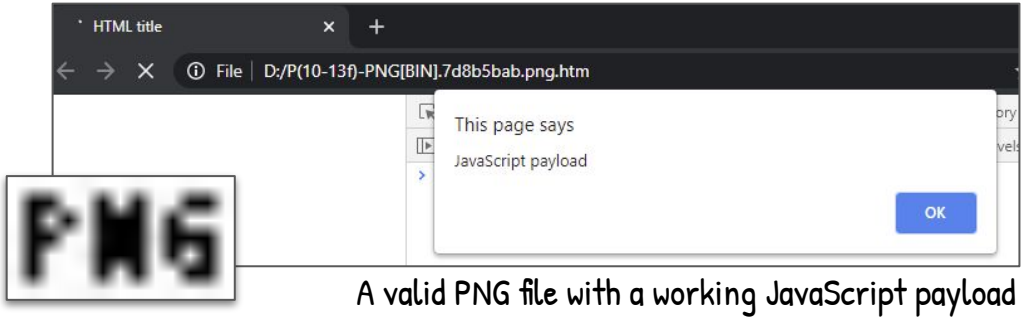
Named after Mithridates
(a famous polyglot)


Mitra
Abuses
Polymocks (ID bypass)
Polyglots (type bypass)
Embedding
Near polyglots (AngeCryption, TimeCryption)
Ambiguity
Collisions
Tricks
Normalize
Wrappend
Formats features
Magic
Start offset
Cavity
Parasite
Appended data
Formats structures
Sequences (train)
Stacked boxes
Pointers (book)
Chains (towed boats)
Combination strategies
Concatenation
Cavity
Parasite
Zipper

Embedding payloads

```
$ mitra.py png.png script.js -f
png.png
File 1: PNG / Portable Network Graphics
script.js
File 2: binary blob

Stack: concatenation of File1 (type PNG) and File2 (type BIN)
Parasite: hosting of File2 (type BIN) in File1 (type PNG)
```



A valid PNG file with a working JavaScript payload

```
000: 89 P N G \r \n ^Z \n 00 00 01 38 c O M M
010: - - > \r \n < d i v __ i d = ' m y
020: p a g e ' > \r \n < h 1 > H T M L
030: __ p a g e < / h 1 > \r \n < s c r
040: i p t __ l a n g u a g e = j a v
050: a s c r i p t __ t y p e = " t e
060: x t / j a v a s c r i p t " > __
070: \r \n d o c u m e n t . d o c u m
080: e n t E l e m e n t . i n n e r
090: H T M L __ = __ d o c u m e n t .
0A0: g e t E l e m e n t B y I d ( '
0B0: m y p a g e ' ) . i n n e r H T
0C0: M L ; \r \n d o c u m e n t . t i
0D0: t l e __ = __ ' H T M L __ t i t l
0E0: e ' ; \r \n a l e r t ( " J a v a
0F0: S c r i p t __ p a y l o a d " )
100: ; \r \n c o n s o l e . l o g ( "
110: J a v a S c r i p t __ p a y l o
120: a d " ) ; \r \n < / s c r i p t >
130: \r \n < / d i v > \r \n < ! - - __ 2E
140: DA DC 65 00 00 00 0D I H D R 00 00 00 0D 00
150: 00 00 07 01 03 00 00 00 E9 BE 55 59 00 00 00 06
160: P L T E FF FF FF 00 00 00 55 C2 D3 7E 00 00
170: 00 1B I D A T 08 1D 63 00 82 54 03 86 70 07
180: 86 F4 02 06 F7 00 06 57 03 06 06 06 00 21 1A 03
190: 10 32 6A 0B 48 00 00 00 00 I E N D AE 42 60
1A0: 82
```

```
-->
<div id='mypage'>
<h1>HTML page</h1>
<script language=javascript type="text/javascript">
document.documentElement.innerHTML =
document.getElementById('mypage').innerHTML;
document.title = 'HTML title';
alert("JavaScript payload");
console.log("JavaScript payload");
</script>
</div>
<!--
```

Parasite code

```
$ mocky.py --combined input/jpg.jpg
Filetype: JFIF / JPEG File Interchange Format
Parasite-combined sig(s): unicos / Symbian / snd / wdk / SoundFont / icc / VICAR / netbsd_ktraceS / SoundFX /
VirtualBox / ScreamTracker / Plot84 / ezd / dicom / Tar(checksum) / ds / CCP4 / DRDOS / pif / mbr
25676
> Combined Mock: mA-jpg.jpg
```

POLYMOCKS (ID BYPASS)

Add any possible signature with Mocky

```
$ file mA-jpg.jpg
mA-jpg.jpg: tar archive
```

<- FILE sees it as a TAR file!
(valid TAR signature + checksum)

Many detected file types

```
$ file mA-jpg.jpg --keep-going --raw
mA-jpg.jpg: tar archive
- DR-DOS executable (COM)
- JPEG image data, baseline, precision 8, 104x56, components 1
- Windows Program Information File for acsp`
- VICAR label file
- DOS/MBR boot sector
- Nintendo DS ROM image: "������" (SNDH, Rev.107) (homebrew)
- Plot84 plotting file
-  DOS/MBR boot sector
- sfArk compressed Soundfont
- Old EZD Electron Density Map
- Symbian installation file
- Scream Tracker Sample mono 8bit
- SNDH Atari ST music
- SoundFX Module sound file
- DICOM medical imaging data
- CCP4 Electron Density Map
- VirtualBox Disk Image (������), 5715999566798081280 bytes
- unicos (cray) executable
- data
```

Still a perfectly valid JPEG!
(with an extra COMment segment stuffed with signatures)

```
$ identify -verbose ./mA-jpg.jpg
Image:
  Filename: ./mA-jpg.jpg
  Format: JPEG (Joint Photographic Experts Group JFIF format)
  Mime type: image/jpeg
  Class: PseudoClass
  Geometry: 104x56+0+0
  Resolution: 36x36
  Print size: 2.88889x1.55556
  Units: PixelsPerCentimeter
  Colorspace: Gray
[...]
```

USING MOCKY TO BYPASS file IDENTIFICATION

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

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
…	…															

Many magics are
at the start of the file.

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

…

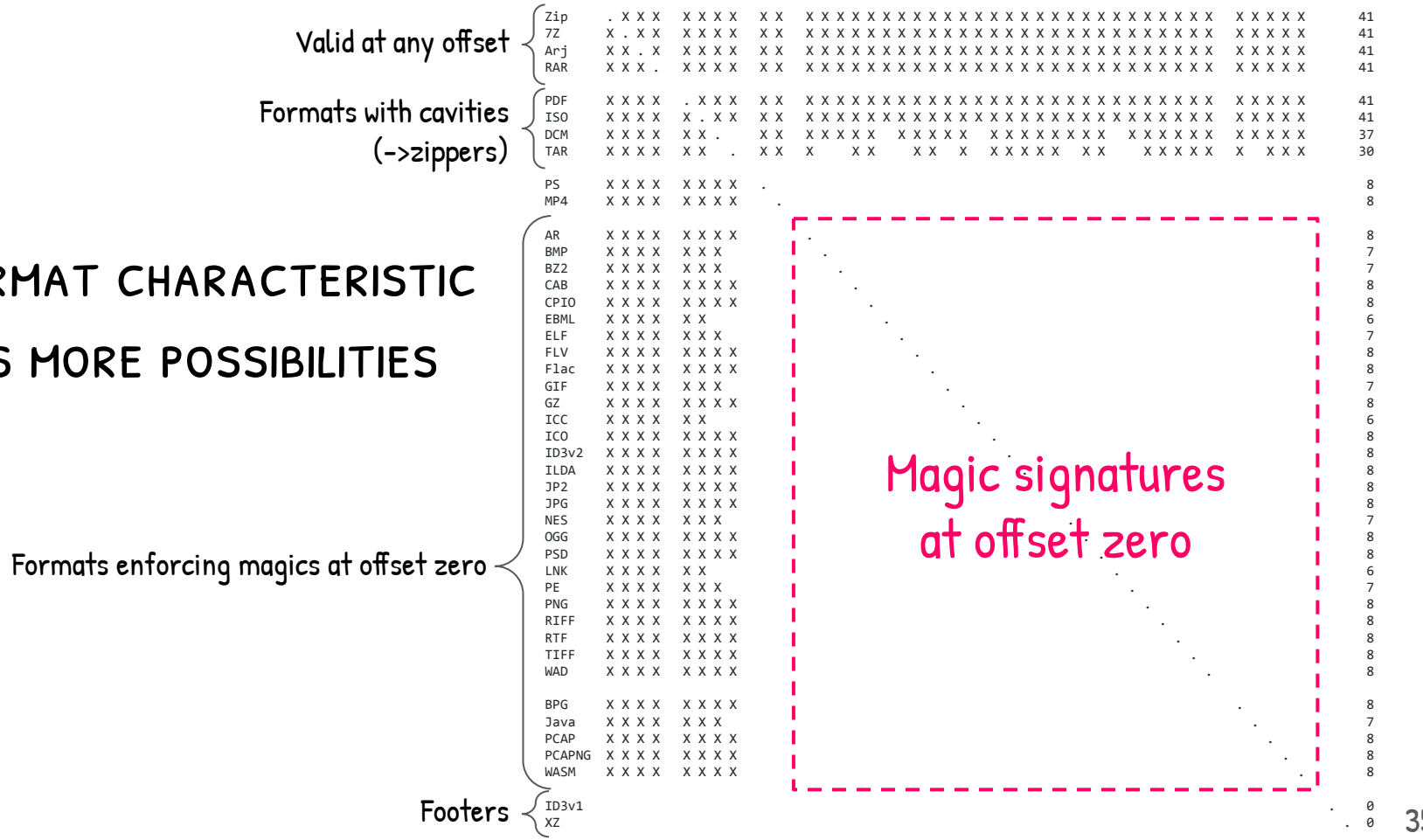
output from
`file --keep-going`

```
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\00
- Nintendo 3DS File Archive (CFA)
- Unix Fast File system [v1] (litt
- Unix Fast File system [v2] (litt
- Unix Fast File system [v2] (littl
- ISO 9660 CD-ROM filesystem data (
- 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
```

```
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 - versi
114     0x72    xz compressed data
120     0x78    LZ4 compressed data
...
```

output (150 sigs) from
`Binwalk`

Each format characteristic enables more possibilities



```
000: 2031 3233 3435 3637 3839 3031 3233 3435   123456789012345
010: 0a25 5044 462d 312e 3425 2020 2020 2020  .%PDF-1.4%
020: 3031 3233 3435 3637 3839 3031 3233 3435  0123456789012345
030: 0a31 2030 206f 626a 3c3c 2520 2020 2020  .1 0 obj<<%
040: 3031 3233 3435 3637 3839 3031 3233 3435  0123456789012345
050: 0a2f 5479 7065 2f43 6174 616c 6f67 2520  ./Type/Catalog%
060: 3031 3233 3435 3637 3839 3031 3233 3435  0123456789012345
070: 0a2f 5061 6765 7320 3220 3020 5225 2020  ./Pages 2 0 R%
080: 3031 3233 3435 3637 3839 3031 3233 3435  0123456789012345
090: 0a3e 3e65 6e64 6f62 6a0a 2520 2020 2020  .>>endobj.%
...
640: 3031 3233 3435 3637 3839 3031 3233 3435  0123456789012345
650: 0a74 7261 696c 6572 203c 3c25 2020 2020  .trailer <<%
660: 3031 3233 3435 3637 3839 3031 3233 3435  0123456789012345
670: 0a2f 526f 6f74 2031 2030 2052 3e3e 2520  ./Root 1 0 R>>%
680: 3031 3233 3435 3637 3839 3031 3233 3435  0123456789012345
```

A CUSTOM BINARY LASAGNA:

ABUSING LINE COMMENTS AND INTERLEAVE PDF STATEMENTS W/ ARBITRARY DATA.

```
00: 50 4B 03 04 00 00 00 08 00 00 00 00 00 00 95 19  PK
10: 85 1B 0C 00 00 00 0C 00 00 00 08 00 2E 00 4C 46              . LF
20: 48 20 4E 61 6D 65 75 70 11 00 01 BE A1 2C A5 55  H Nameup     , U
30: 6E 69 63 6F 64 65 20 4E 61 6D 65 05 26 15 00 5A  nicode Name &  Z
40: 50 49 54 08 4D 61 63 20 4E 61 6D 65 5A 49 50 20  PIT Mac NameZIP
50: 53 49 54 78 48 65 6C 6C 6F 20 77 6F 72 6C 64 21  SITxHello world!
60: 50 4B 01 02 00 00 00 00 00 00 00 00 00 00 00 00  PK
70: 95 19 85 1B 0C 00 00 00 0C 00 00 00 09 00 00 00
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 43 44                CD
90: 46 48 20 4E 61 6D 65 50 4B 05 06 00 00 00 00 01  FH NamePK
a0: 00 01 00 37 00 00 00 60 00 00 00 00 00              7   `
```

FILECRAFT

FileCraft

Duplicity in ZIPs:

4 names for the same archived file via older structures.

Near polyglots

Non-working parasites with data to be replaced.

The smaller that data, the better. (ex: overlapping magics)

An external operation will swap the overlapping data.

```
A: automated     ?: likely possible
M: manual        !: unknown
```

```
                                                               Variable  Unsupported
                                                                offset     parasite
Minimal start offset
         1 2 4 8     9  16  20  23  28  34  40  64  94  132    12  28
                      12          26  32  36      68  112 226    16

         P P J F M T F W G P R I R B C I P C J P E A P I I J    W B O      B E G L N
         S E P l P I L A Z N I D T M P L S A P C L R C C C a    A P G      Z B I N E
             G a 4 F V D   G F 3 F P I D D B 2 A F   A O C v    S G G      2 M F K S
               c   F         F v     O A       P     P     a    M            L
                               2               N
                                               G

1* PS    . M A ? ? ? ? ? ? A ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?    ? ? ?      ? ? ? ? ?
2^ PE    M . A A A A A A A A A A A A A A A A A A ! ! ! ! ! !    M M M      ! ! ! ! !
4+ JPG   A A . M M M M M M M M M M M M M M M M M M M M M M M    M M M      M M M M M
```

From near-polyglots to crypto-polyglots

Swap the overlap via [cryptographic] operations

En-/de-cryption with specific parameters (IV, Nonce):

Bruteforcing may be required.

Each payload is [partially] hidden
when the other is in clear.

```
    B  M 3C 00 00 00 00 00 00 00 20 00 00 00 0C 00
00: 89  P  N  G \r \n ^Z \r 00 00 00 2C  c  O  M  M
10: 00 00 0D 00 07 00 01 00 01 00 FF FF FF 00 00 00
20: 00 00 00 00 65 40 00 00 55 40 00 00 67 60 00 00
30: 57 50 00 00 65 60 00 00 00 00 00 00 00 00 00 00
40: 1D 44 05 DC 00 00 00 0D  I  H  D  R 00 00 00 0D
50: 00 00 00 07 01 03 00 00 00 E9 BE 55 59 00 00 00
60: 06  P  L  T  E FF FF FF 00 00 00 55 C2 D3 7E 00
70: 00 00 1B  I  D  A  T 08 1D 63 00 82 54 03 86 70
80: 07 86 F4 02 06 F7 00 06 57 03 06 06 06 00 21 1A
90: 03 10 32 6A 0B 48 00 00 00 00  I  E  N  D AE 42
A0: 60 82
```

```
B  M 3C 00 00 00 00 00 00 00 20 00 00 00 0C 00
```


```
89  P  N  G \r \n ^Z \n 00 00 00 2C  c  O  M  M
```


```
mitra.py bmp.bmp png.png --overlap
```

Generates `O(10-40)-PNG[BMP]{424D3C0000000000000020000000C00}.1965e270.png.bmp`

A BMP/PNG near polyglot, with 16 bytes of OVERLAP.

A valid BMP is AES-CBC encrypted as a PNG with a special IV to encrypt the first block as expected (AngeCryption).

AngeCryption works with
ECB, CBC, CFB, OFB

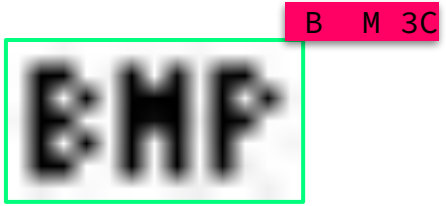
```
00:  B  M 3C 00 00 00 00 00 00 00 20 00 00 00 0C 00
10: 00 00 0D 00 07 00 01 00 01 00 FF FF FF 00 00 00
20: 00 00 00 00 65 40 00 00 55 40 00 00 67 60 00 00
30: 57 50 00 00 65 60 00 00 00 00 00 00 00 00 00 00
40: 00 A1 3B E2 E0 64 F0 A7 AE 5E 21 64 BC 44 5F 09
50: E3 67 D3 10 19 AF 09 F1 99 1A 33 B3 BF 28 EF 9E
60: 71 3D 87 79 EC 73 A9 60 82 74 1B EB 08 B4 4E B7
70: E5 9E 16 A9 CE BC 1B 71 99 E7 F8 E8 FA 8C C0 6C
80: 6B 85 4B 56 73 7D 22 BD 46 DE AC 3F BF EE 8B 96
90: AB 74 55 5F 21 B7 10 1B D6 96 18 45 6E E5 B0 3C
A0: 7C 22 99 87 EA FE 1F 4D FF C8 52 C0 24 C7 AD A8
```

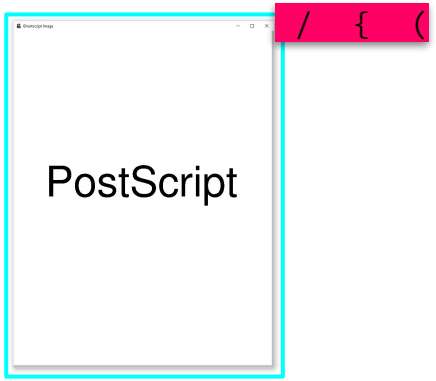
AES-CBC →

```
89  P  N  G \r \n ^Z \n 00 00 00 30  c  O  M  M
71 2F D8 C7 79 C1 EB CF 63 B0 22 2B 0A 6D E3 2D
24 49 57 B1 9B BB C2 FA 94 8A 8C 53 9E A1 30 63
30 C9 41 75 EA AF 75 EE 95 7C 57 E9 16 4F F7 3B
1D 44 05 DC 00 00 00 0D  I  H  D  R 00 00 00 0D
00 00 00 07 01 03 00 00 00 E9 BE 55 59 00 00 00
06  P  L  T  E FF FF FF 00 00 00 55 C2 D3 7E 00
00 00 1B  I  D  A  T 08 1D 63 00 82 54 03 86 70
07 86 F4 02 06 F7 00 06 57 03 06 06 06 00 21 1A
03 10 32 6A 0B 48 00 00 00 00  I  E  N  D AE 42
60 82 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```


```
mitra/utils/cbc$ angecrypt.py "O(10-40)-PNG[BMP]{424D3C000000000000000200000000C00}.1965e270.png.bmp" bmp-png.cbc
```

```
     B  M 3C
00:  /  {  ( 00 00 00 00 00 00 00 20 00 00 00 0C 00
10: 00 00 0D 00 07 00 01 00 01 00 FF FF FF 00 00 00
20: 00 00 00 00 65 40 00 00 55 40 00 00 67 60 00 00
30: 57 50 00 00 65 60 00 00 00 00 00 00 )  }  %  !
40:  P  S \r \n  /  N  i  m  b  u  s  S  a  n  s  -
50:  R  e  g  u  l  a  r     1  0  0     s  e  l  e
60:  c  t  f  o  n  t \r \n  7  5     4  0  0     m
70:  o  v  e  t  o \r \n  (  P  o  s  t  S  c  r  i
80:  p  t  )     s  h  o  w \r \n  s  h  o  w  p  a
90:  g  e \r \n  s  t  o  p \r \n 00 00 00 00 00 00
```





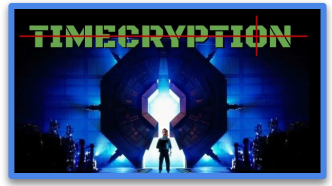

```
mitra.py postscript.ps bmp.bmp --overlap
```

Generates O(3-3c)-PS[BMP]{424D3C}.209881aa.ps.bmp

A BMP/PS NEAR POLYGLOT WITH 3 BYTES OF OVERLAP.

Both files are decrypted via GCM from the same ciphertext but via different keys.

The nonce is bruteforced to generate the right overlap with either key.



TimeCryption works with
CTR, OFB, GCM, GCM-SIV, OCB3

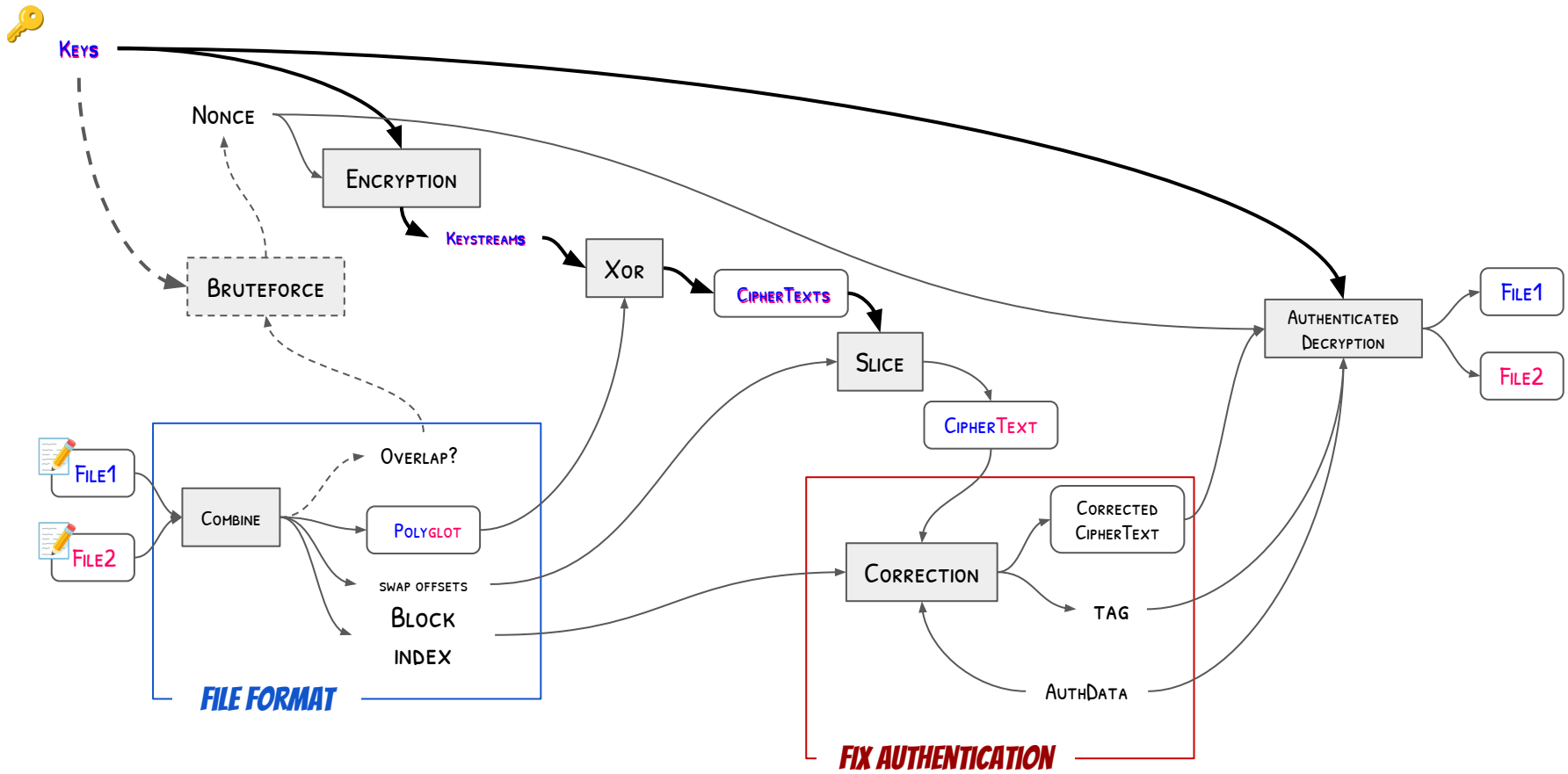
ciphertext

Key$_1$ Key$_2$

00:	B	M	3C	00	00	00	00	00	00	00	20	00	00	00	0C	00
10:	00	00	0D	00	07	00	01	00	01	00	FF	FF	FF	00	00	00
20:	00	00	00	00	65	40	00	00	55	40	00	00	67	60	00	00
30:	57	50	00	00	65	60	00	00	00	00	00	00	B7	EB	32	E8
40:	16	D6	9E	76	AC	20	9C	8C	9F	06	6F	55	3F	96	0E	09
50:	04	24	41	5D	22	7C	A6	E5	0E	AC	ED	1C	04	65	BE	E6
60:	E8	AB	E4	D2	C6	B6	CD	9F	AB	85	E1	CE	03	C5	A5	85
70:	70	B5	09	EB	EB	CB	D1	2F	7C	4D	B0	09	35	38	D9	B7
80:	82	31	BB	87	96	22	C8	4E	C0	EC	89	C3	CB	97	63	D3
90:	A0	28	47	5B	71	C2	95	EC	12	E2	52	B0	6F	B1	EE	61

/	{	(	07	3A	14	40	E5	3E	EC	AE	A2	AD	87	AA	38
11	C4	5D	5A	35	2D	EB	EC	47	CC	A7	B5	63	22	90	B7
5F	D7	41	7B	FD	6D	53	DB	78	9F	AA	A6	2B	22	61	AD
BB	38	48	4A	5C	A7	D5	E4	63	4F	4D	7B	)	}	%	!
P	S	\r	\n	/	N	i	m	b	u	s	S	a	n	s	-
R	e	g	u	l	a	r		1	0	0		s	e	l	e
c	t	f	o	n	t	\r	\n	7	5		4	0	0		m
o	v	e	t	o	\r	\n	(	P	o	s	t	S	c	r	i
p	t	)		s	h	o	w	\r	\n	s	h	o	w	p	a
g	e	\r	\n	s	t	o	p	\r	\n	00	00	00	00	00	00


```
mitra/utils/gcm$ meringue.py "O(3-3c)-PS[BMP]{424D3C}.209881aa.ps.bmp" bmp-ps.gcm
```



Under the hood - check Mitra & KeyCom.

```
$ wget https://eprint.iacr.org/2020/1456.pdf
[...]

$ openssl enc -in 1456.pdf -out crypted \
   -aes-128-ctr -iv 00000000000000000000e7c600000002 \
   -K 4e6f773f00000000000000000000000
$ openssl enc -in crypted  -out viewer.exe \
   -aes-128-ctr -iv 00000000000000000000e7c600000002 \
   -K 4c3474337221212100000000000000
$ wine viewer.exe 1456.pdf
```

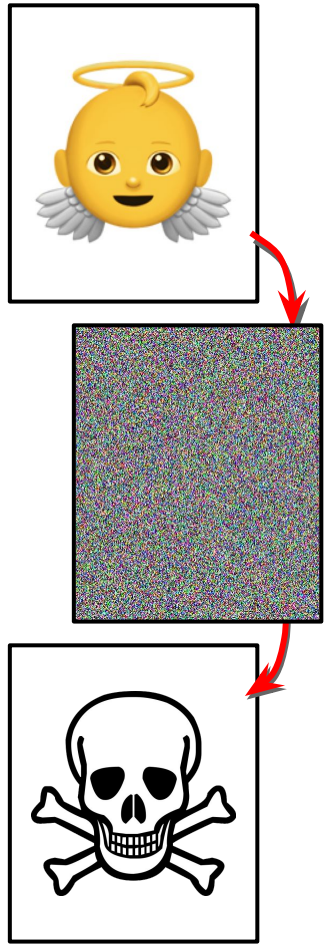
OUR PDF ARTICLE FILE IS ALSO A PDF VIEWER EXECUTABLE! VIA **AUTHENTICATED** ENCRYPTION.

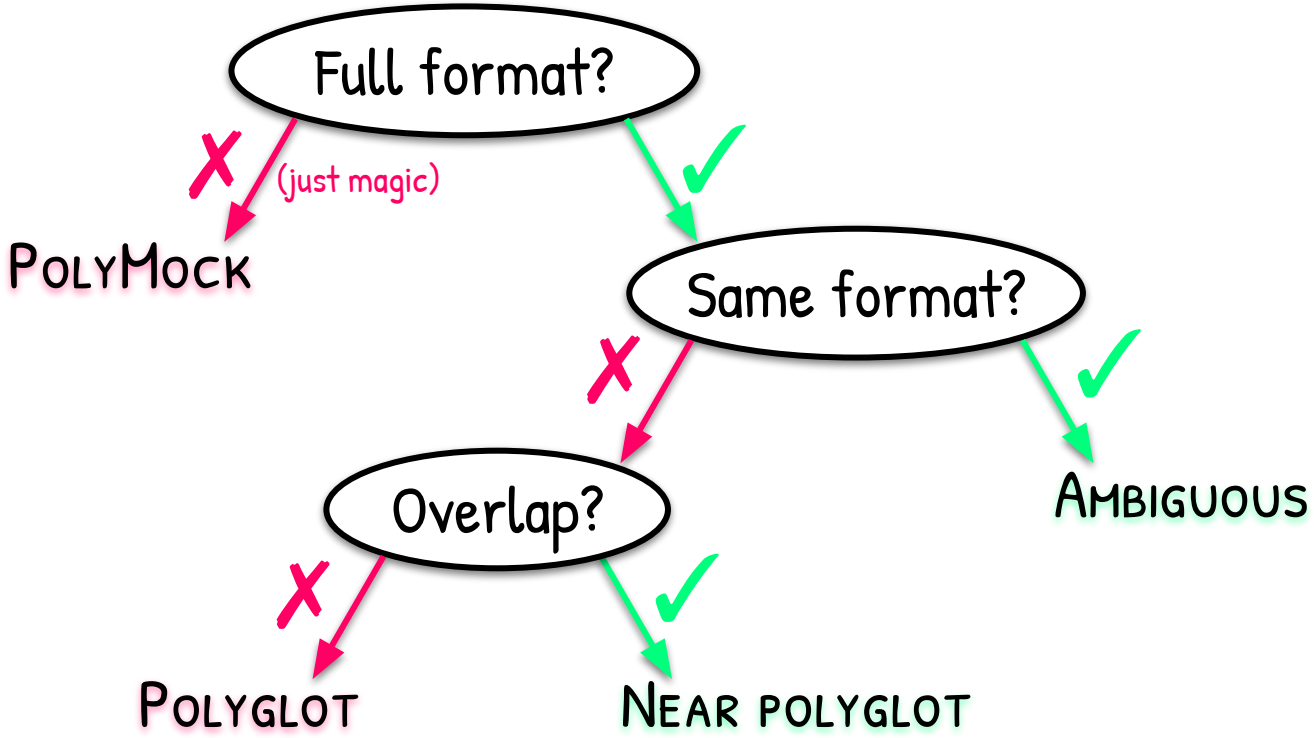
TIMECRYPTION

Without key commitment,
a ciphertext can be crafted to decrypt **with authentication**
to different payloads.

Vulnerabilities @ Facebook, Amazon, Google...

With key management: friendly today, evil tomorrow.





A hierarchy of weird files

File formats challenges in 2024

Private information can leak at cloud's scale.

Leaked credentials are abused within minutes.

Keys, login/passwords, cookies.

A single "minor" bug can affect billions of users!

ACROP ALYPSE

by Simon Aarons and David Buchanan.

Just standard PNG files.

Cropped by the user.

The smaller file is kept with trailing data leftovers.

-> major leak of information for users.

SQLite: data leaks in plain sight

Magic: "`SQLite format 3\0`" (16 bytes)

-> very strong identification.

No easy subtype-identification: the Application ID is rarely used.

Is it a standard assets index ? A mountable filesystem?
Cookies / web history / credit cards / bitcoin wallet ?

-> Identification tool: sqlbuddy.py

Hash collisions

Collisions in 2024?

Who needs cryptographic hashes for collisions?

Some AVs detect the EICAR file by CRC32!

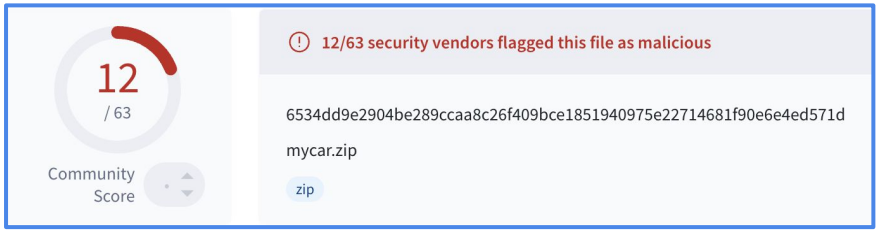
```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

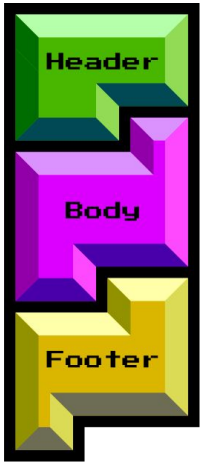
or

```
DpVRUX<=EICAR CRC collision? Use Shake128/Kangaroo12/Blake3 instead!
```

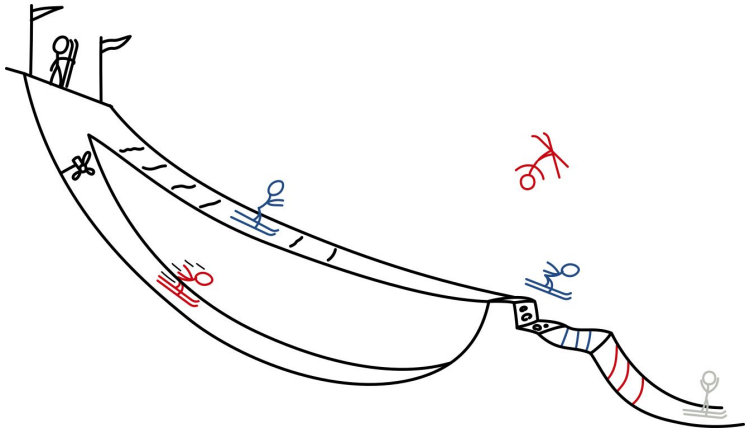
Same CRC32

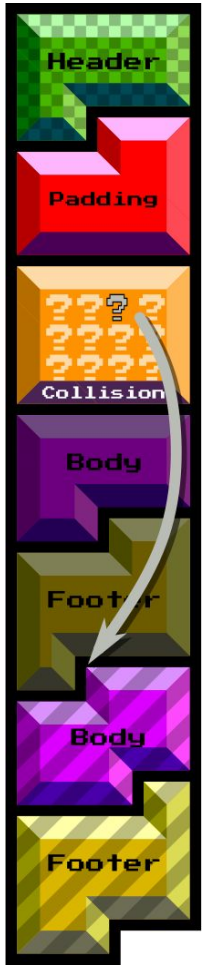
Script: [mycar.sh]





Cryptographic hash collisions require acrobatic constructs.





Trophy wall

2017

BlackHat, RWC, Crypto

2019

PTS, Hack.lu

2019 (workshop)

PTS, Hack.lu, BA...

https://github.com/corkami/collisions ☆ Star 3.1k

docs, precomputed prefixes, scripts, pocs... (MIT licence)

Detecting collisions w/ signatures

DetectColl can detect any MD5 or SHA1 hash collision.

Flame's unique collision.

```
$ detectcoll_unsafe flame.der | ./logparse.py
flame.der
block: 11, collision: Flame
```

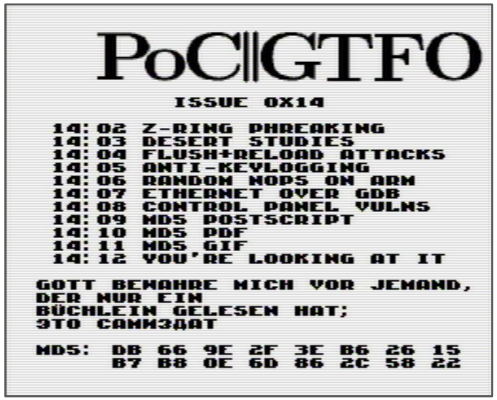
Newest SHA1's: Shambles

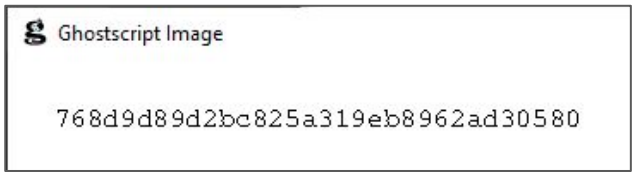
```
$ detectcoll 13-shambles1.bin | ./logparse.py
  13-shambles1.bin
  block: 9, collision: SHAttered/Shambles
```

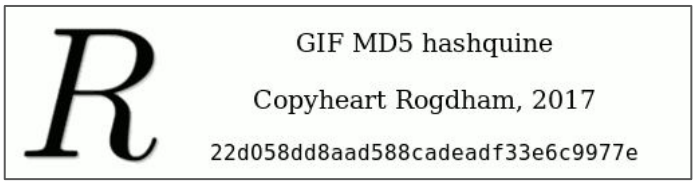
Github / corkami / collisions / README.md#signatures

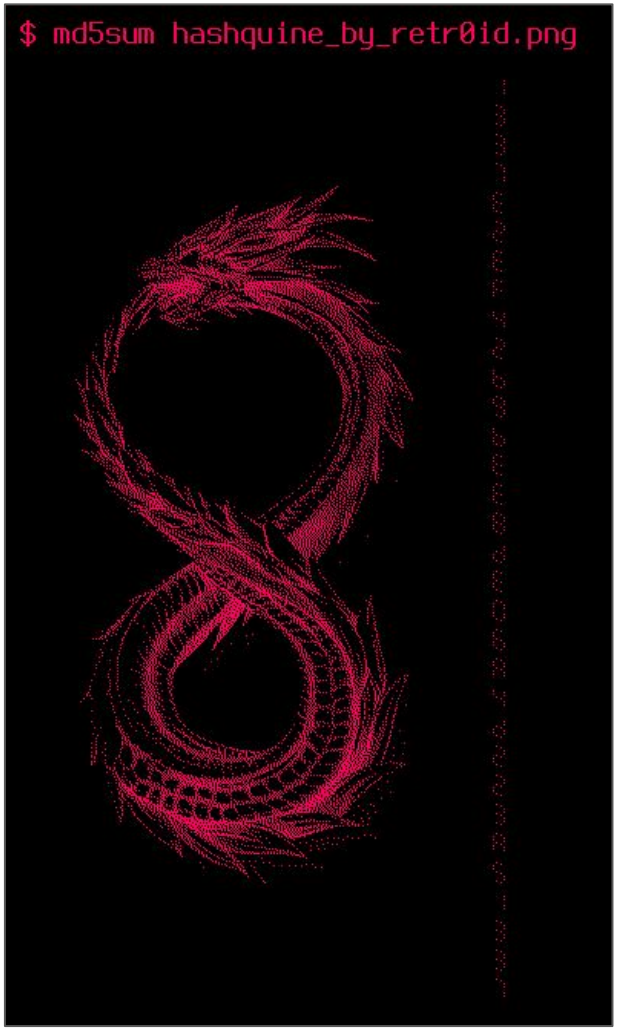
Hashquines

Chain 128-512 collisions
to change the displayed hash
but keep the file hash constant.









RetrOid's hashquine archive (2023)

A **generic** tar file contains a hash list
with the hash of the whole archive.

```
$ tar -xvf self.tar.zst
x hash.md5
x hello.txt

$ md5sum -c hash.md5
self.tar.zst: OK
hello.txt: OK
```

The file is "building" a Tar header
via 653 MD5 collisions abusing ZStandard frames.

Explanations on [github / corkami / collisions / hashquines]

And yet...

"If it's not broken in practice... ...it must be good enough!"

23 October 2023
Expires: 25 April 2024

Deprecating Insecure Practices in RADIUS

While MD5 has been broken, it is a testament to the design of RADIUS that there have been (as yet) no attacks on RADIUS Authenticator signatures which are stronger than brute-force.
https://www.ietf.org/archive/id/draft-dekok-radext-deprecating-radius-05.txt

New MD5 attack - June 2024

TextColl (2024)

```
TEXTCOLLBYfGiJUETHQ4hAcKSMd5zYpgqf1YRDhkmxHkhPWptrkoyz28wnI9V0aHeAuaKnak
TEXTCOLLBYfGiJUETHQ4hEcKSMd5zYpgqf1YRDhkmxHkhPWptrkoyz28wnI9V0aHeAuaKnak
```

```
TEXTCOLLBYfGiJUETHQ4hAcKSMd5zYpgqf1YRDhkmxHkhPWptrkoyz28wnI9V0aHmSZaAAAA()(()()(()(((((((()(((()(((()))))()(()))))))())(()))))))()(()
TEXTCOLLBYfGiJUETHQ4hEcKSMd5zYpgqf1YRDhkmxHkhPWptrkoyz28wnI9V0aHmSZaAAAA()(()()(()(((((((()(((()(((()))))()(()))))))())(()))))))()(()
```

```
BASEX64G5MM2g4CpNnoHBgERiMZ3J5P2YsP7wIlz4Kfh+JGOxOiptV+pvQZ0whAt1q3Jt+la2CKWqu5H9bzDIxBaNrzCkij91ZB9M5DlPne5sUir5TZ6yQGfGKtaX0BG
BASEX64G5MM2g4CpNnoHBiERiMZ3J5P2YsP7wIlz4Kfh+JGOxOiptV+pvQZ0whAt1q3Jt+la2CKWqu5H9bzDIxBaNrzCkij91ZB9M5DlPne5sUir5TZ6yQGfGKtaX0BG
```

- 1 bit-difference
- Not 64 bytes rounded!
- Custom alphabet: alphanum -> test your password!

Github / cr-marcstevens / hashclash / tree / textcoll

Formats are complex.
Files are layered.

Archive, stack, encapsulation, compressions...

One format may be robust against some attacks,
but its inner/outer format or side format
might make the whole system vulnerable.

Shattered

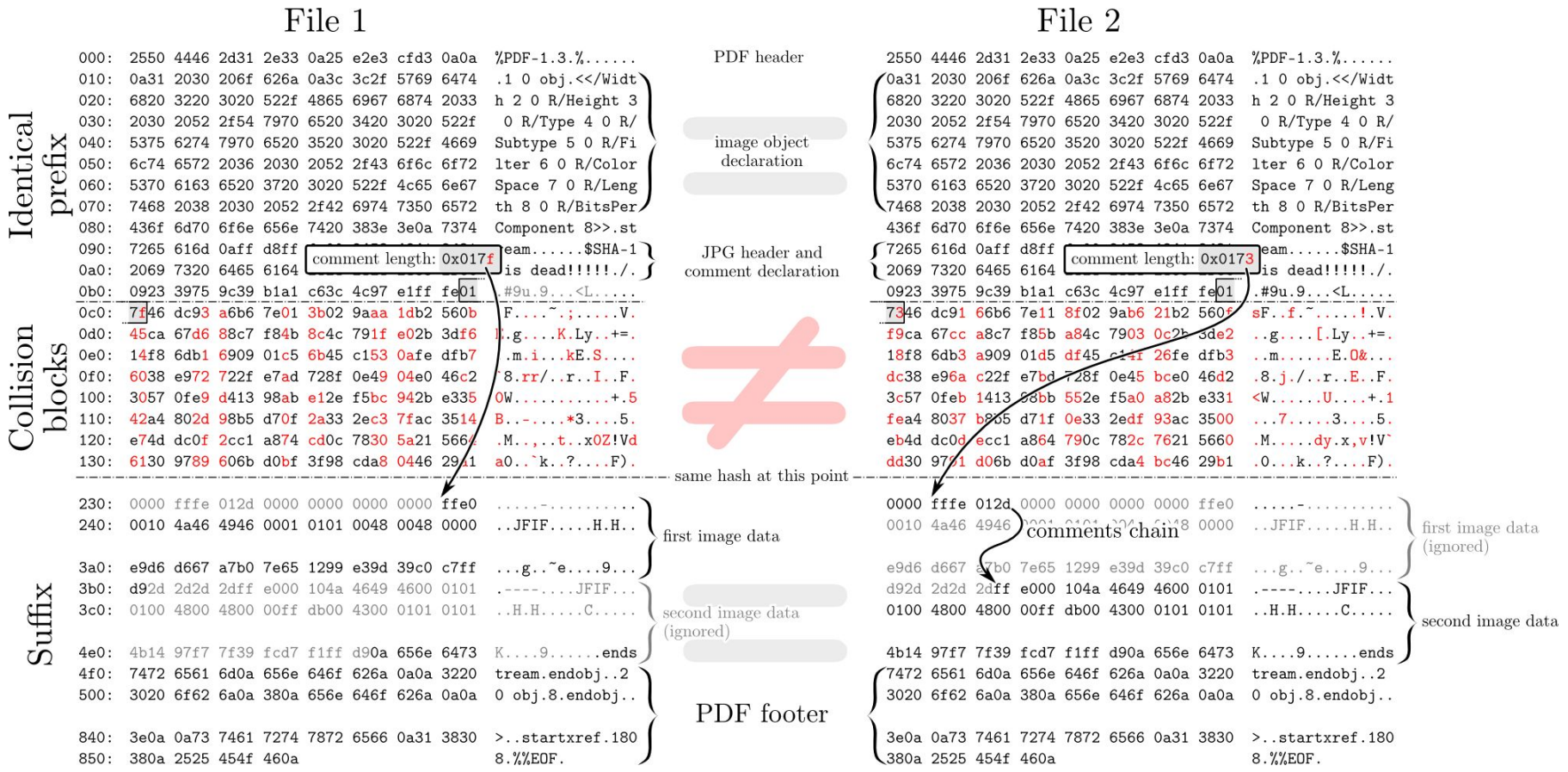
First SHA1 collision on PDF files.

It wasn't a PDF collision

as PDF parsers can't be reliably collided with SHA1.

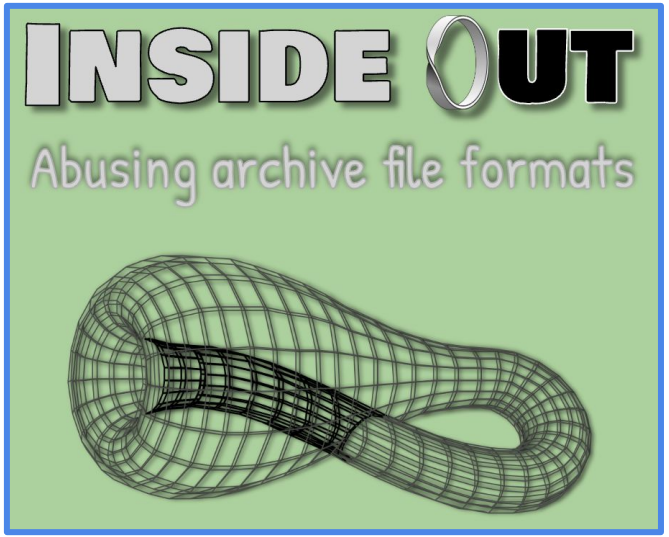
-> Abuse JPG in PDF as JPG can be collided reliably.

This would likely work in any format based on JPG.



Shattered files layout: a normal PDF with a funky JPG.

Inside Out



Abusing .docx:

- XML can't tolerate collision blocks.
- ZIPs can't be collided generically.

-> Abuse XML in Zips via Zip structures.

Abusing .tar.gz:

(Tar can't be collided generically)

-> Abuse Gzip structure to show different Tar contents.

Tar hashquines

Tar can't be collided generically:

-> Abuse GZip/L4/Zstandard archive structure
to present different archived file contents
to external parsers.

Let's talk **seriously** about...

AI

No hype, no fake,
no empty promises.

File formats... and AI ?

"Who needs AI to check a magic signature?"
"It won't catch polyglots anyway."

But...

What about source files?

Or ***any*** kind of attachments?

...

MAGIKA

200+ formats: text **and** binary.

Small model: runs on CPU, needs 1 Mb.

Fast: <5ms per file. Used in production on 100s of billion files weekly.
Used in 150+ projects.

Open-source: Python, Go, Rust, JavaScript.

https://github.com/google/magika ☆ Star 8.2k

Paper (ICSE 25) https://arxiv.org/abs/2409.13768

Non-generative AI: no copyright infringement, just a detection verdict.

The 'Magika in production' effect...

How many file formats overall? Who knows... 🤯🤷

Each community have its weirdnesses, overlaps, do's and don'ts.

"What a mess 👌"

No silver bullet

It doesn't scan the whole file (only the first & last 2Kbs).

Not enough samples to train on many formats.

Standard AI limitations: no editting / omitting.

May fail on weird files 😉

May catch corrupted/spoofed files:
-> useful for carving, recovery-abuse.

-> Remove the first 16 bytes, then re-scan.

Magika on corrupted files

A ZIP with invalid signatures:

An invalid file recovered by applications.

-> scanning bypass.

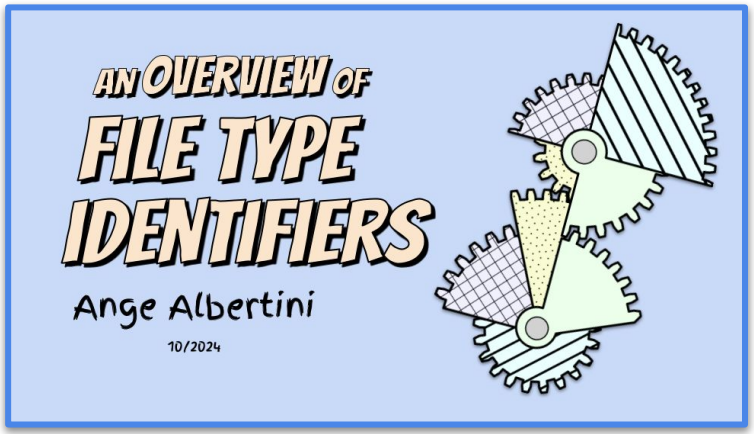
```
00  .B .K \3 \4 0a 00 00 00 00 00 00 00 00 00 23 8e
10  5a 6b 05 00 00 00 05 00 00 00 07 00 00 00 .z .i
20  .p .. .t .x .t .Z .I .P \r \n .B .K 01 02 1f 00
30  0a 00 00 00 00 00 00 00 00 00 23 8e 5a 6b 05 00
40  00 00 05 00 00 00 07 00 00 00 00 00 00 00 00 00
50  00 00 00 00 00 00 00 00 .z .i .p .. .t .x .t .B
60  .K 05 06 00 00 00 00 01 00 01 00 35 00 00 00 2a
70  00 00 00 00 00
```

```
$ file badsigs.zip
badsigs.zip:      data
$ magika badsigs.zip -s
badsigs.zip: Zip archive data (archive) 98%
```

Magika is new & different, and useful in its own way.

Planning to make a new engine?

-> Investigate all existing ones, then give a talk on the topic ->



Fool AI identification?

Some formats give you full control over the first X bytes.

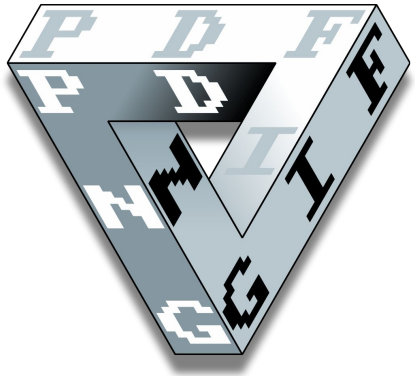
Most make it possible to insert exploitable contents early.

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

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

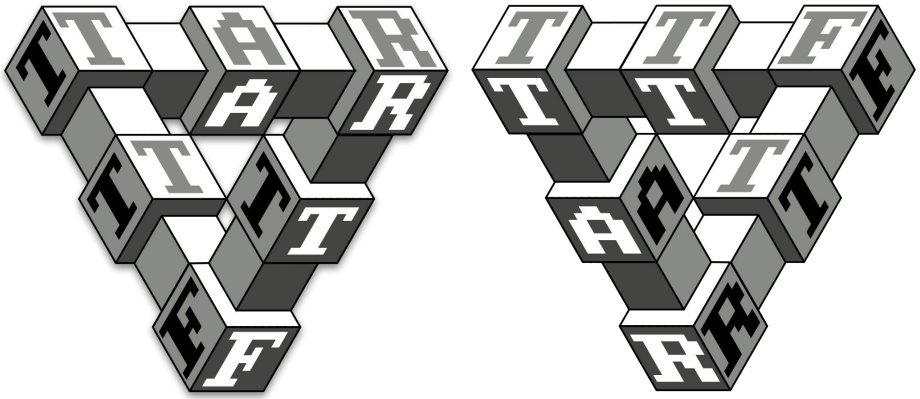
Use Mocky to insert dummy signatures:

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



Mocky & Mitra @ [Github corkami/mitra]

Conclusion



In 2024...

Many old tricks still work.

Specifications can still be naive or laughable.

No reference code, no test cases.

No incentive to fix anything if it's not a security bug.

-> back to the eternal: "let's check wikipedia..." ?

From funky PoCs to fearsome tools.

Working at scale with new tools:

- 100s of collisions possibilities
- 1000s of polyglot combinations
- 100s of billions of scanned files by AI.

AI & FILE FORMATS

- Many AI formats are vulnerable.
- Magika brings something new to file format processing.
- Mitra can be used to inject arbitrary data in formats (and fool AI).

Room for improvment 

- Specifications writing and updating.
- Sample crafting and sharing.
- Format identification and heuristics.
- Format classifying and rating.

Commandments of a good file format

Magic at offset zero
fast identification, no bypass

Clear chunk structure
forward compatibility, easy parsing/cleanup

Version number
Forward thinking

No duplicity
Duplicity → discrepency 

No "constant" variables
Ossification → hardcoding

Up-to-date specs
Reflect reality

Samples set
Theory isn't enough

Extensibility
*Your format **will** evolve in unknown ways*

Keep the spirit
Don't reuse formats for different intent without trivial distinction

Perfect is the enemy of good
Shortcuts will be taken to avoid over-complexity.

Thanks for your attention!

Acknowledgements:
Marc Stevens, Philippe Teuwen, Stefan Kölbl, Atul Luykx, Daniel Bleichenbacher, David Buchanan, Sophie Schmieg, Yanick Fratantonio, and the Fabianis.

ANGE ALBERTINI
reverse engineering
VISUAL DOCUMENTATIONS
@angealbertini
ange@corkami.com
http://www.corkami.com