# Formats de fichiers
## Décisions & Conséquences

*Ange Albertini*

*crimes et châtiments*
🔢📜: 🔪&👩

**G**roupe de **T**ravail
**S**écurité des **S**ystèmes, des **L**ogiciels et des **R**éseaux

27 Nov 2019
ESIEA Paris

# About the author

Opinions are my own and not the views of my employer

- Reversing since the late 80's
- Author of Corkami
- 6 years at PoC or GTFO*
- occasional drawer, singer
- Passionate about file formats

Professionally

- 13 years of malware analysis
- 1 year of Information Security Engineer


My license plate is a CPU,
My phone case is a PDF doc,
My resume is a PDF/SNES/Megadrive polyglot.

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

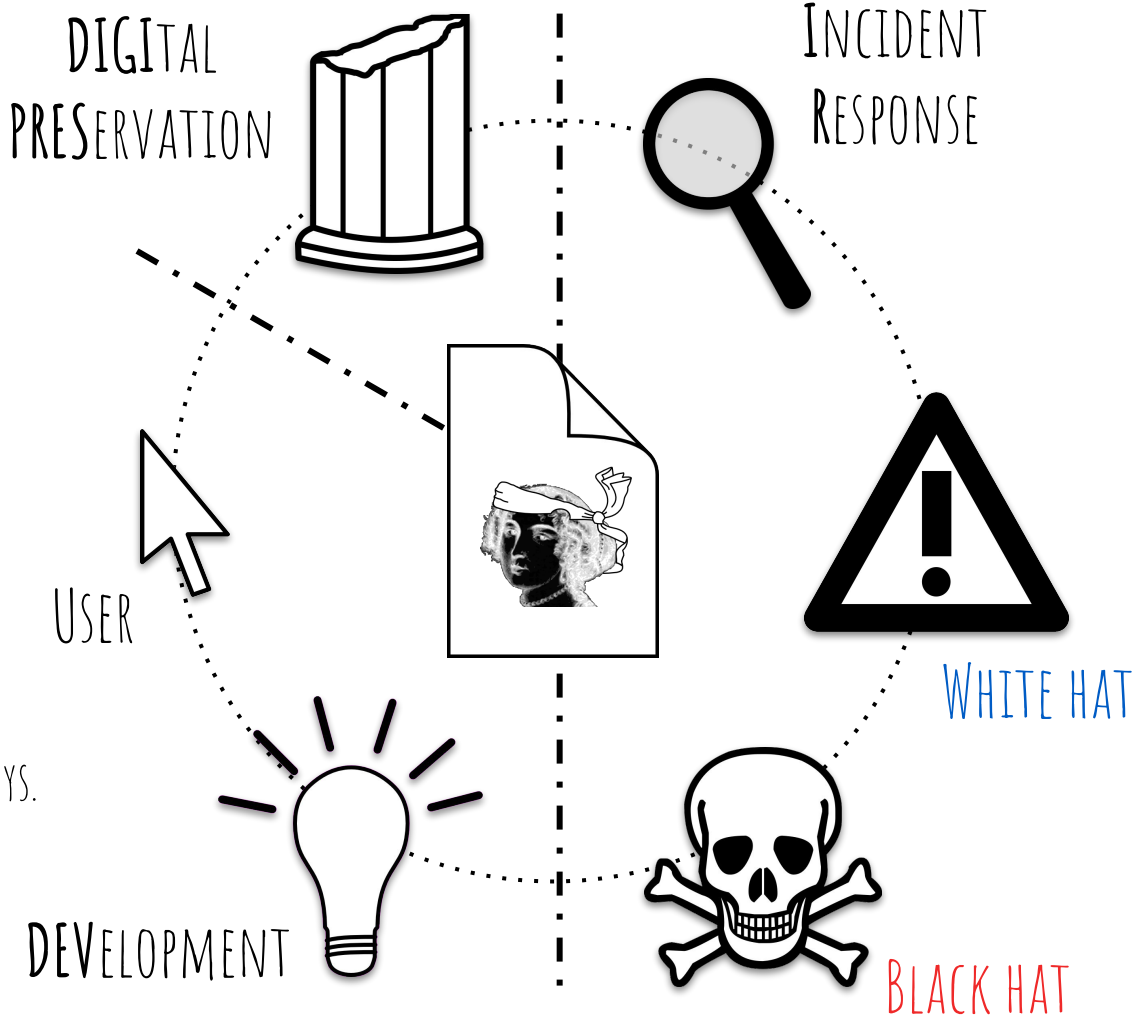
# There are various

(with a few things in common)

# communities around file formats

...and I'm interested in **all** of them.

My life is about file formats - they're my toys.



# This is **not** an advanced talk:


## more like a high-level presentation to address upstream problems regarding file formats.

And hopefully you can use them to convince others.

In 1989...

# our computer

(10 MHz CPU, 20 Mb HDD)

# was infected by a virus...

Thankfully,
a french magazine explained
how to remove it...

# ...BY YOURSELF, WITH A HEX EDITOR!

"...At the end of the first file allocation table of the hard disk, replace the last 3 bytes FF 7F FF by FF 0F 00. Then find the code of the virus itself which starts with FF 06 F3 7D 8B 1E and overwrite it (including all following bytes, until 55 AA) by F6..."

# THIS WAS MY INTRODUCTION TO HEX EDITORS AND MALWARE!

30 YEARS AGO! 🎂🥳🍻

comme endommagés dans la table d'alloca-
tion des fichiers. Par chance, il n'attaque que les IBM PC-XT. Pour s'en débarrasser, il faut rétablir les pistes de démarrage dans leur état d'origine. Avec un éditeur d'octets du type PC-Tools, vérifiez la présence des octets 33 C0 dans les zones 30 et 31 du secteur d'amorçage du disque dur ; s'ils sont bien présents, mieux vaut exécuter la commande SYS depuis une disquette Système saine ; à la fin de la première table d'allocation des fichiers du disque dur, remplacez les trois derniers octets (FF 7F FF) par FF 0F 00. Puis localisez le code du virus lui-même, qui commence par FF 06 F3 7D 8B 1E, et remplacez-le (ainsi que tous les octets qui suivent, jusqu'à 55 AA) par F6 si le formatage est dû à la commande FORMAT du système, ou par 00 s'il provient de PC-Tools. Si l'opération vous semble trop com-

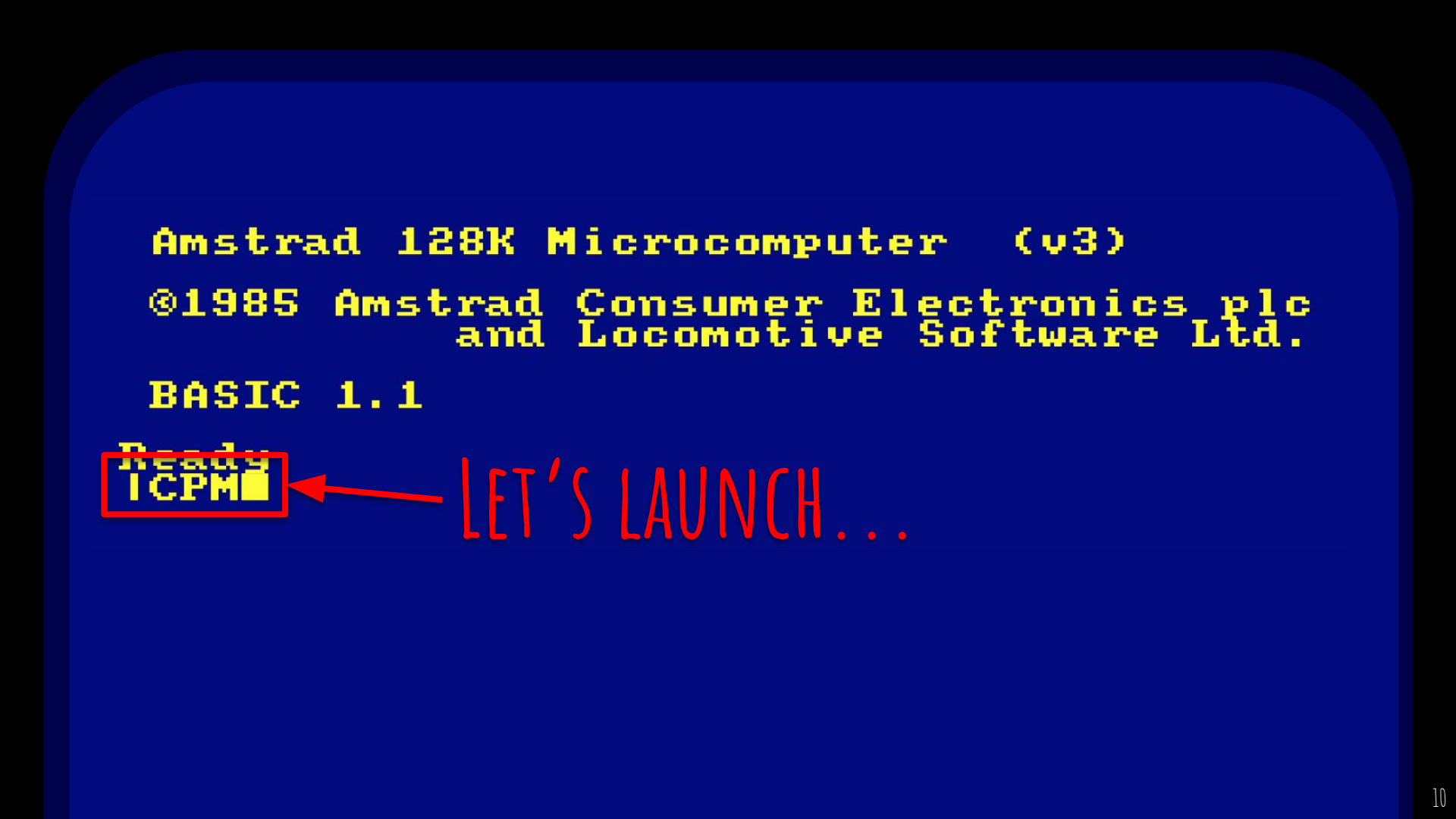
As a starter...

# Let's craft a valid file from scratch...

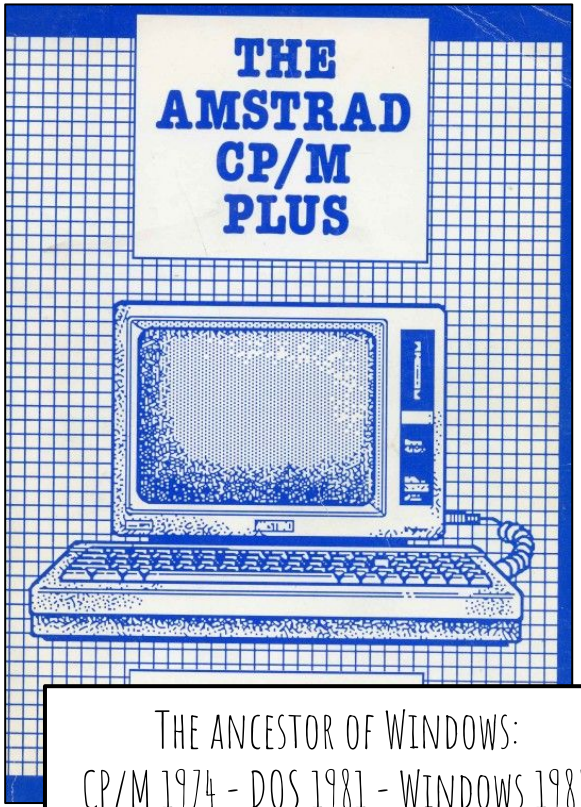
(a commercial **and** successful software!)

....Yes, really!

# On this computer...


Amstrad 128K Microcomputer (v3)
©1985 Amstrad Consumer Electronics plc
and Locomotive Software Ltd.
BASIC 1.1
Ready
|CPM
LET'S LAUNCH...

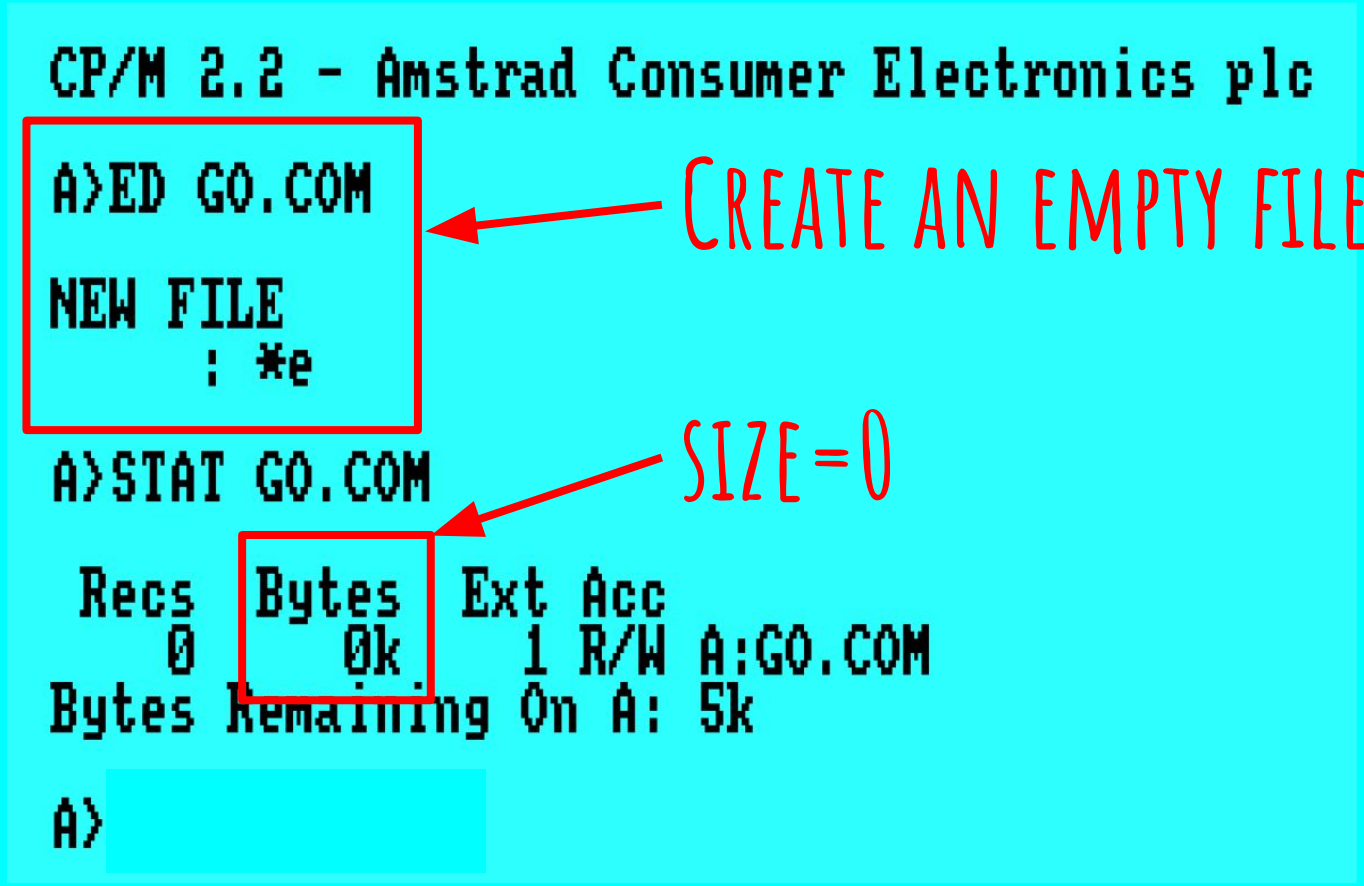
# ...this OS:




The ancestor of Windows:
CP/M 1974 - DOS 1981 - Windows 1985


3" Compact Floppy 2
180 Kb / side

# Let's create... an EMPTY executable!



# Is it even valid?

That's how executables were called on CP/M.

Yes: **Transient Commands** are blindly loaded and execution is started at offset zero.

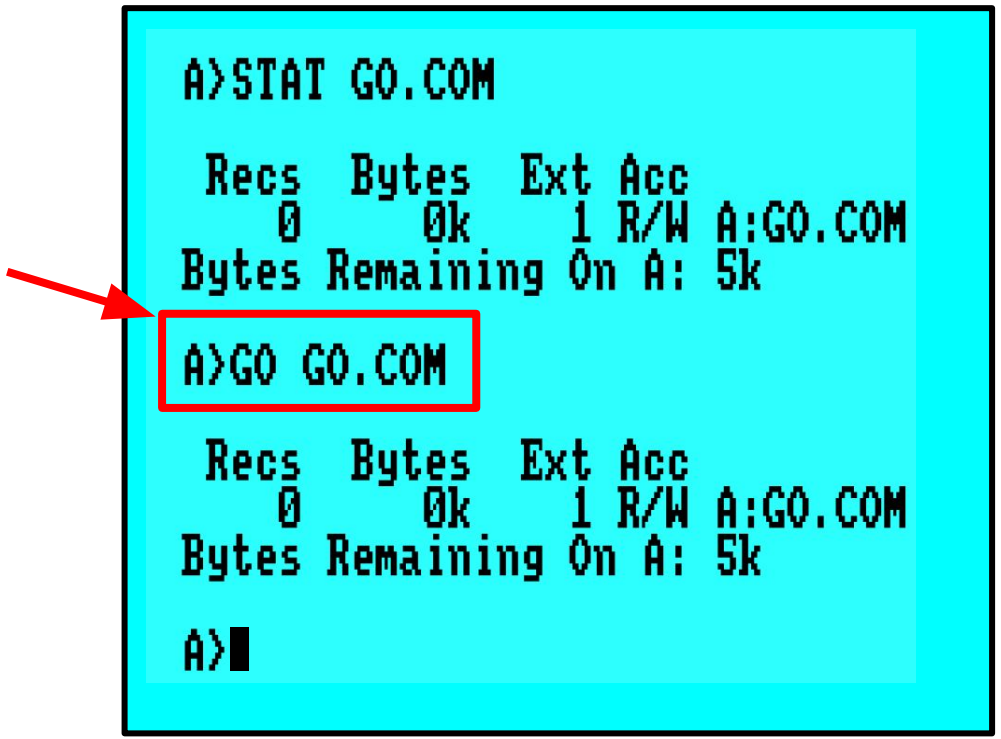
Only the **.com** filename extensions matters.

# Does it do anything?

The Transient Memory Area is **not**
cleared between executions,
so the previous command is re-executed.

# It works as intended!

(it repeats the previous command)



```
      R 03 02
     COMMODORE 64  44k CP/M vers 2.2
     ================================
     Copyright (C) 1988, ROSSMOELLER
A>ED GO.COM
NEW FILE
      : *e

A>DUMP GO.COM

A>DUMP
NO INPUT FILE PRESENT ON DISK
A>GO DUMP.TXT
0000 0D 0A 4D 49 54 20 44 45
0008 4D 20 42 45 46 45 48 4C

A>
```

COMMODORE 64

# Reliable & multi-platform!

Under a commercial OS (in the 80s),
the **empty** file is valid, useful and reliable.

It was even sold as a commercial program for ~5 EUR.

# Lessons learned

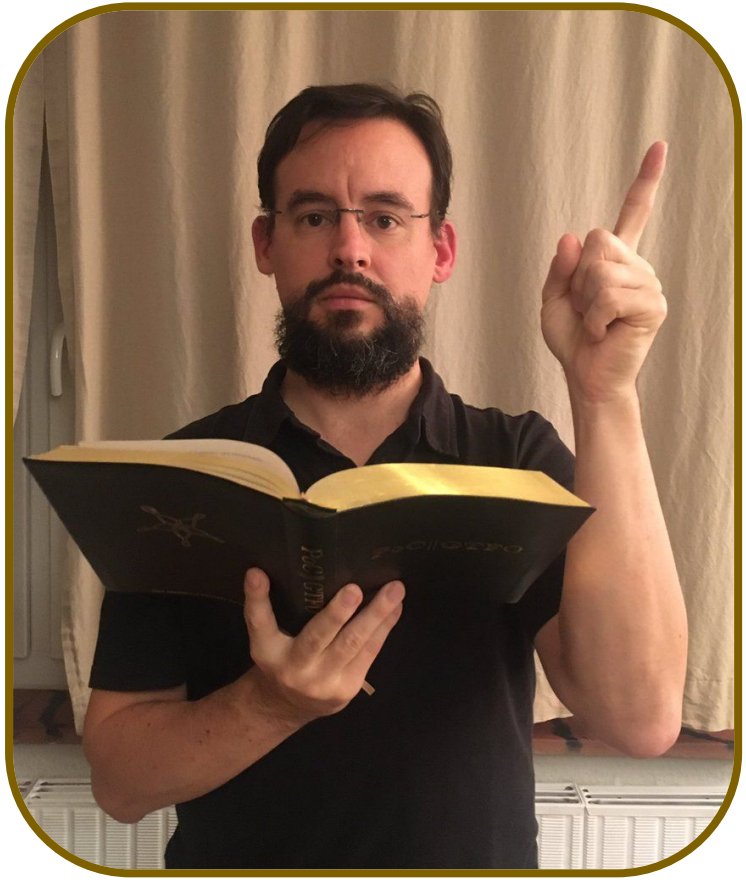
Many things have changed since the 80s, but...

- Weird files are nothing new.
- Software always defined the rules.
  - Specifications are entirely optional.
  - There's no "that's not how it works".

# *The Meaning of a File*

*First, you must realize that*
*a file has no intrinsic meaning.*
*The **meaning** of a file*
*- its type, its validity, its contents -*
*can be different for each parser or interpreter.*

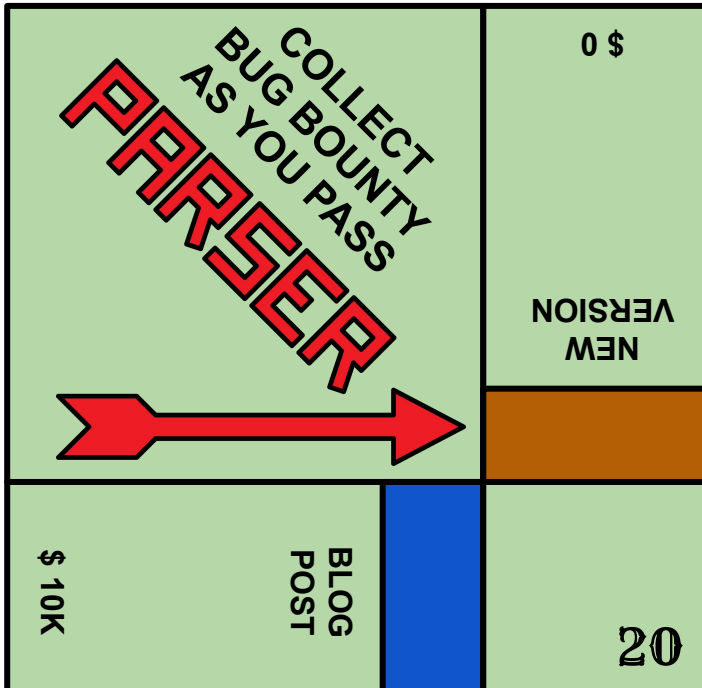
*Ange Albertini ;)*
https://archive.org/details/pocorgtfo07/page/n17



# Parser security so far? Fuzz/Fail/Fix !

Fuzz. Get bug fixed. Collect pride & glory.
Rinse. Repeat.

```
10 FUZZ
20 FAIL
30 FIX
40 GOTO 10
```



# The original sin

A misunderstood field: "specs are enough"

-> received less attention

-> least rigorous field of computing.

Not enough pre-natal checks.

Lacking growth control.

The next file format will likely suck.

**Crypto = Sparta**

**File formats: The Jungle Book**

# A typical file format timeline

Good (naive?) intentions:

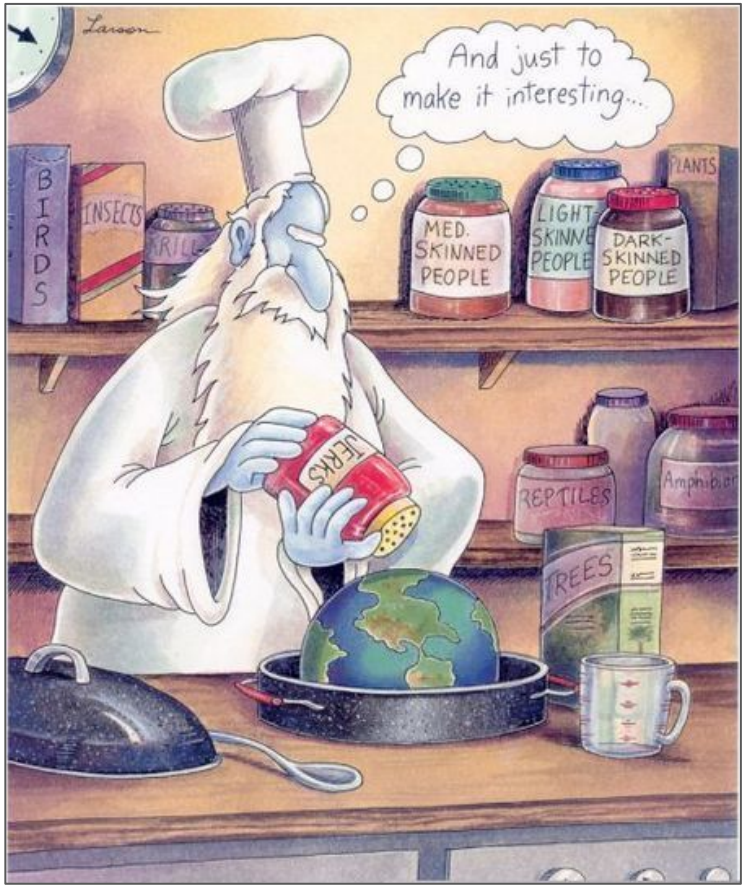
proper planning. Official specs. Set in stone.

Bad things happen:

Interpretation blur, unofficial extensions.

Format is now used everywhere:

Misunderstood. Unmovable.



# Common misconceptions

Some might be obvious to **you**.

They aren't to **everyone**.

Many developers don't have security in mind.

"I'll just use the security tools afterwards to make it secure".

# 'Solving' the file formats problems

Code review. Fuzzing.
Test benches. Hardening.
Normalizing. Yara.

It's not solving:

it's fixing - but too late?

# Common misconceptions

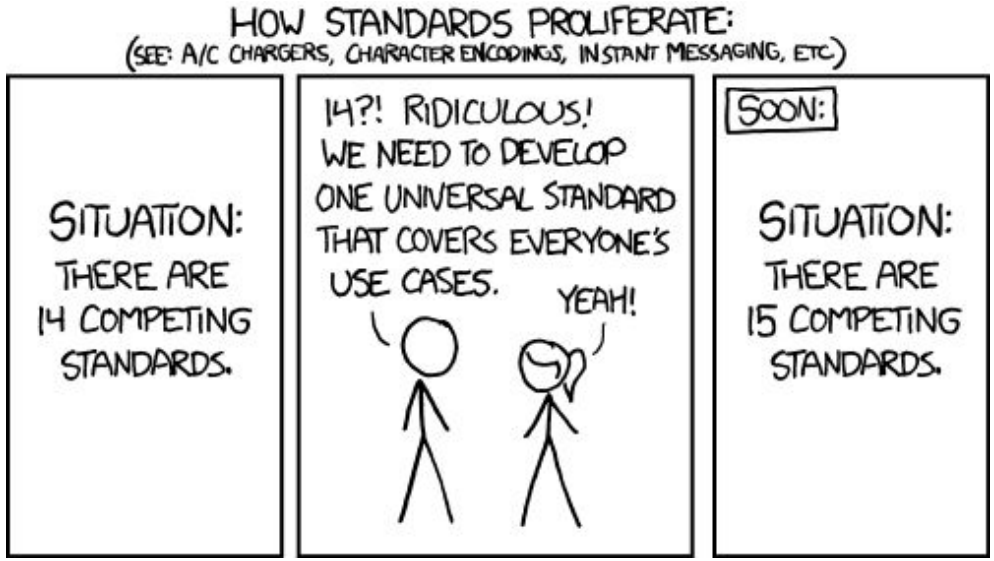
New formats are only created and new parsers are only written when strictly required.

Specs are available, they're clear, complete. The overall complexity is clear.

People read them thoroughly before starting coding, take sane decisions.

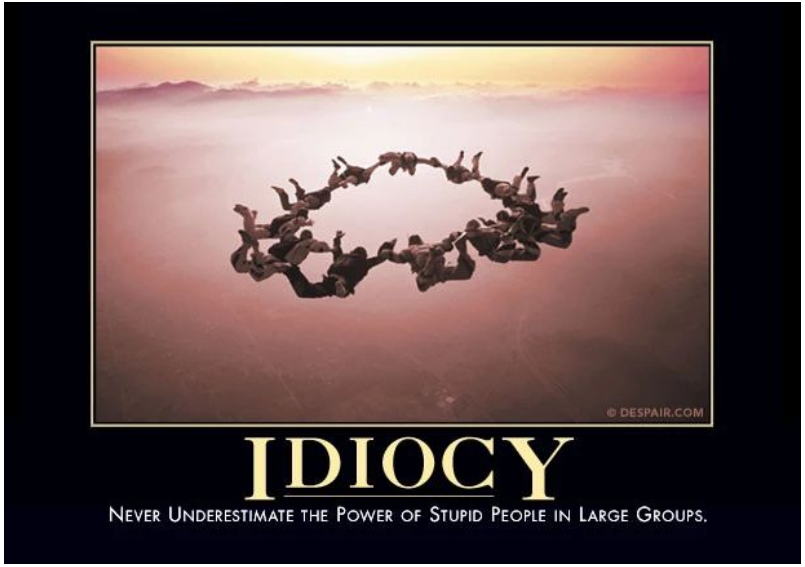
Crazy formats are discarded. Unsecure code is removed.

All formats need a magic at offset zero.




We need
a new parser

We need
a new format



# "There's already a..." ?

License? Language? Threading? Weight?
Robustness? Optimisation? Compatibilty?
...reinvent the wheel?

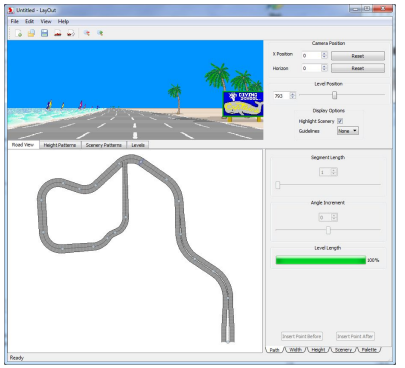
*Telling a programmer there's already a library to do X*
*is like telling a songwriter there's already a song about love.*

*~ Pete Cordell*

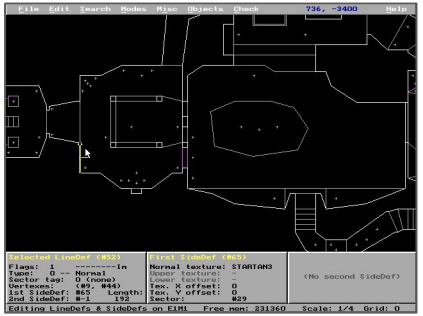
# 50 shades of specifications

| | |
|---|---|
| No files | - ROM / bootable floppy |
| No doc | - obfuscated reader (video games) |
| | - game w/ editors (Doom) |
| | - standard implementations: blah2XML + XML2Blah |
| No source | Binary + .h |
| Inaccessible specs | Price, NDA |
| Incomplete specs | No implementation |
| Blurry specs | Corner cases |
| Misleading specs | People take the wrong shortcuts. |

LayOut (OutRun)

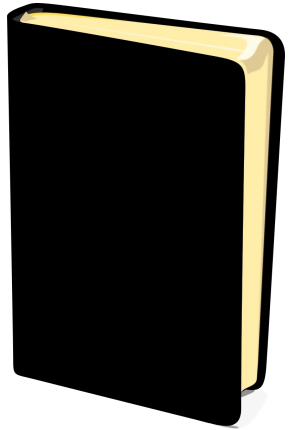


Doom Editing Utilities

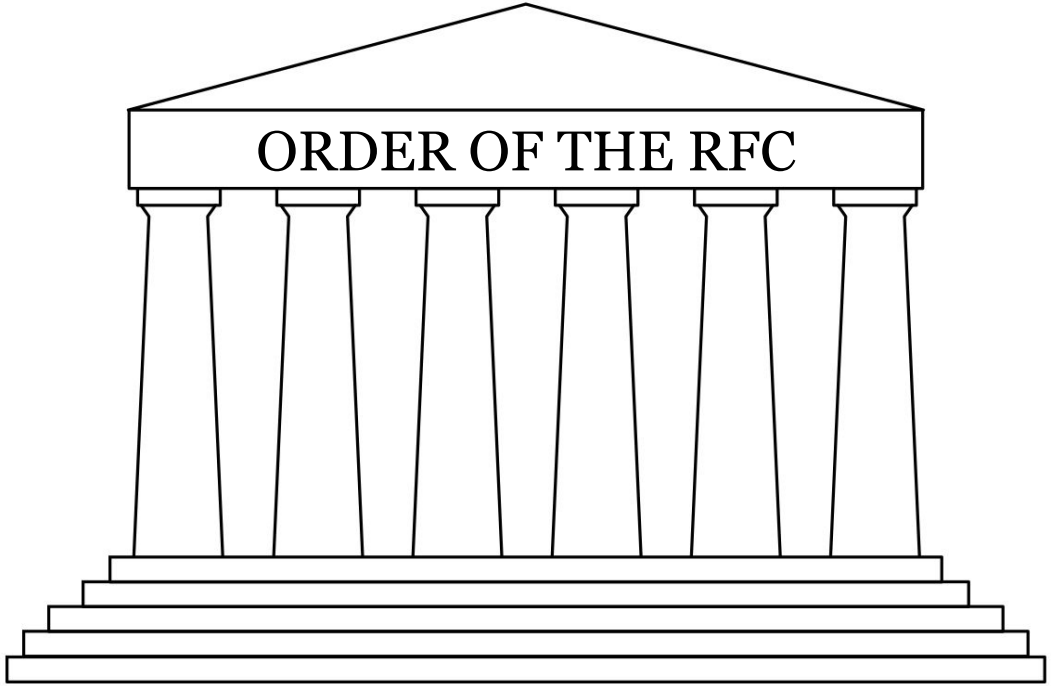


# How we perceive file formats:

## A holy text and its cult.



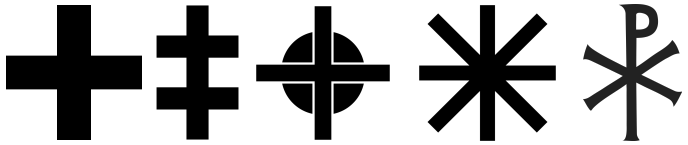
*"Specs are all you need"*

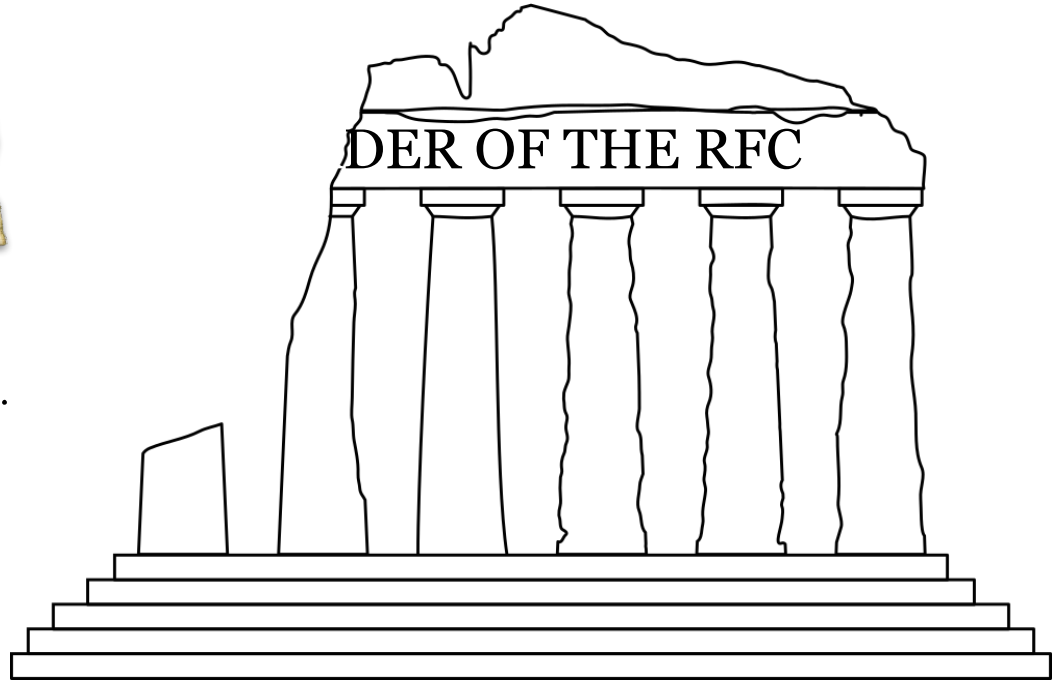


# More like...

## outdated and irrelevant practices.


## ...and a complex landscape.





# Specifications

Some were written years/decades ago.

Originally made for **80x25** screens :)

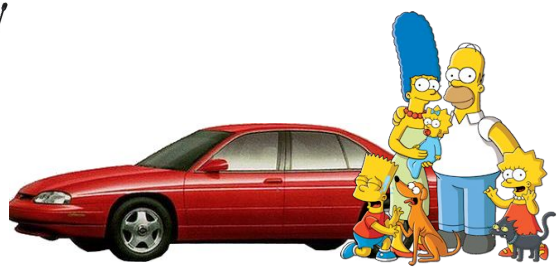
Never updated.

Some features are lost

or never implemented.

For reference,

novelties from 1989





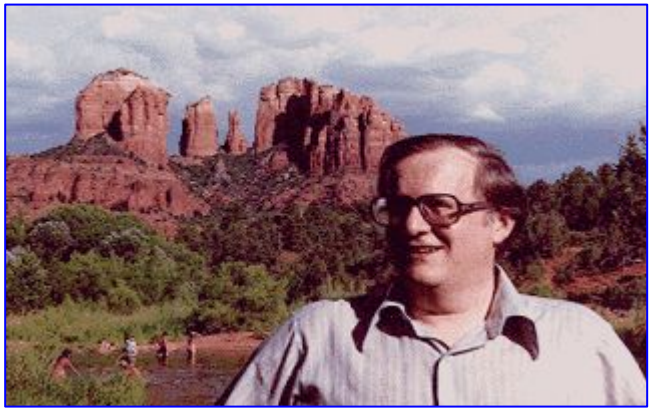
Graphics Interchange Format Data Definition

GENERAL FILE FORMAT

# GIF Plain Text Extension

https://github.com/corkami/formats/blob/WIP/image/gif89a.md#plain-text-extension

## A long forgotten (yet official) way for GIF to display text (they're not comments)



BOB_89A.GIF

```
--------: Introducing GIF89a :--------

When you finish reading this, press
any key to continue. If you just sit
back and watch, we'll continue when
the built-in delay runs out.

GIF89a provides for "disposing of"
an image or text. All the text in
this GIF is "restore to previous",
so that the underlying image is
restored when you press a key or
the delay runs out.

"Transparent" images
or text can be written
over an underlying
image so that parts of
the old image "show
through" the new one.

Oh, incidentally, it's
pronounced "JIF"
```

This image contains these text frames

I don't know **any** software supporting GIF Plain Text Extension!
LMK if you know any!

# Sh*tMySpecsSays (outdated/irrelevant)

[GIF]
The Plain Text Extension contains textual data and the parameters necessary to render that data as a graphic, in a simple form.

[ZIP]
Spanning is the process of segmenting a ZIP file across multiple removable media. This support has typically only been provided for DOS formatted floppy diskettes.

[GIF]
The following GIF Capabilities Response message describes three standard IBM PC Enhanced Graphics Adapter configurations with no printer; the GIF data stream can be processed within an error correcting protocol:

[PNG]
For colour types 2 and 6 (truecolour and truecolour with alpha), the PLTE chunk is optional. If present, it provides a suggested set of from 1 to 256 colors to which the truecolor image can be quantized if the viewer cannot display truecolor directly.
...
A CRC should be checked before processing the chunk data.

[JPEG]
The APP0 marker is used to identify a JPEG FIF file.
The JPEG FIF APP0 marker is mandatory right after the SOI marker.

# How bad parsers are born

Check all the files you have.
Make (wrong) assumptions.
Wrongly confirm with blurry specs.
-> a very bad parser is born

Now we will fuzz it, patch it...
It should just be deleted.

```
for i in *jpg; do xxd "$i" | head -1; done | sort -u
```

```
00000000: ffd8 ffe0 0010 4a46 4946 0001 0100 0001  ......JFIF......
00000000: ffd8 ffe0 0010 4a46 4946 0001 0100 0048  ......JFIF.....H
00000000: ffd8 ffe0 0010 4a46 4946 0001 0101 0048  ......JFIF.....H
00000000: ffd8 ffe0 0010 4a46 4946 0001 0101 0060  ......JFIF.....`
00000000: ffd8 ffe0 0010 4a46 4946 0001 0101 0064  ......JFIF.....d
00000000: ffd8 ffe0 0010 4a46 4946 0001 0101 006b  ......JFIF.....k
00000000: ffd8 ffe0 0010 4a46 4946 0001 0101 0078  ......JFIF.....x
00000000: ffd8 ffe0 0010 4a46 4946 0001 0101 0096  ......JFIF......
00000000: ffd8 ffe0 0010 4a46 4946 0001 0101 00c8  ......JFIF......
00000000: ffd8 ffe0 0010 4a46 4946 0001 0101 00f0  ......JFIF......
00000000: ffd8 ffe0 0010 4a46 4946 0001 0101 012c  ......JFIF.....,
00000000: ffd8 ffe0 0010 4a46 4946 0001 0101 0258  ......JFIF.....X
00000000: ffd8 ffe0 0010 4a46 4946 0001 0200 0001  ......JFIF......
00000000: ffd8 ffe0 0010 4a46 4946 0001 0200 0064  ......JFIF.....d
00000000: ffd8 ffe0 0010 4a46 4946 0001 0201 0048  ......JFIF.....H
00000000: ffd8 ffe0 0010 4a46 4946 0001 0201 012c  ......JFIF.....,
00000000: ffd8 ffe0 2f2a 4a46 4946 0001 0100 0001  ..../*JFIF......
00000000: ffd8 ffe1 0018 4578 6966 0000 4949 2a00  ......Exif..II*.
00000000: ffd8 ffe1 01d7 4578 6966 0000 4949 2a00  ......Exif..II*.
00000000: ffd8 ffe1 1100 4578 6966 0000 4d4d 002a  ......Exif..MM.*
00000000: ffd8 ffe1 181a 4578 6966 0000 4d4d 002a  ......Exif..MM.*
00000000: ffd8 ffe1 28bb 4578 6966 0000 4d4d 002a  ....(.Exif..MM.*
00000000: ffd8 ffe1 2a7a 4578 6966 0000 4d4d 002a  ....*zExif..MM.*
00000000: ffd8 ffe1 2f52 4578 6966 0000 4d4d 002a  ..../RExif..MM.*
00000000: ffd8 ffe1 333f 4578 6966 0000 4949 2a00  ....3?Exif..II*.
00000000: ffd8 ffe1 3e54 4578 6966 0000 4d4d 002a  ....>TExif..MM.*
```

In practice, **JFIF** and **Exif** are NOT required at offset 6.

*Story time*


Specifications
PARSER

# Most files are perfectly structured

They were generated
by one of the standard libraries,
in normal conditions,
and with typical requirements.

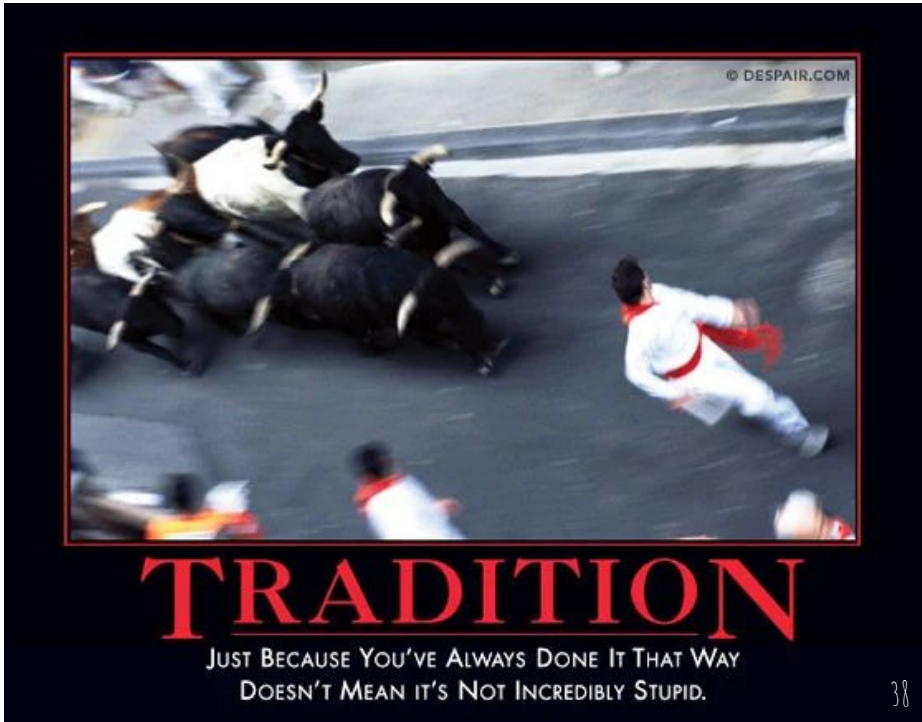
~~Corner cases~~

# Robust parsers act like detectives:
gathering clues,
then reaching a conclusion.

# Magic signatures at offset zero

I can't believe that

I **still** have to say that in <year> !



# Magic signatures differentiate file types.

Easy, quick, reliable filtering.

```
$ cat test1
alert("Hello World");
$ file test1
test1: ASCII text
```

Some JavaScript text
(not identified as JavaScript)

```
$ cat test2
<script>alert("Hello World");</script>
$ file test2
test2: HTML document, ASCII text
```

Add HTML tags
It's detected as expected.

```
$ xxd test3
00000000: 7f3c 7363 7269 7074 3e61 6c65 7274 2822  .<script>alert("
00000010: 4865 6c6c 6f20 576f 726c 6422 293b 3c2f  Hello World");</
00000020: 7363 7269 7074 3e                        script>
$ file test3
test3: data
```

Add a single NON-ASCII character.
It's now considered binary.
It still works as HTML.

```
$ xxd test4
00000000: 4d5a 7f3c 7363 7269 7074 3e61 6c65 7274  MZ.<script>alert
00000010: 2822 4865 6c6c 6f20 576f 726c 6422 293b  ("Hello World");
00000020: 3c2f 7363 7269 7074 3e                   </script>
$ file test4
test4: MS-DOS executable
```

Prepend a fake signature:
It's now identified as an executable.
It **still** works as HTML.

# A fake Windows executable

```
$ ./hexii.py testPE
00: .M .Z .< .s .c .r .i .p .t .> .a .l .e .r .t .(
10: ." .H .e .l .l .o .  .W .o .r .l .d ." .) .; .<
20: ./ .s .c .r .i .p .t .> .P .E \0 \0
30:                                     28 00 00 00
$ file testPE
testPE: PE Unknown PE signature, for MS Windows
```

MZ at 0

PE\0\0 at 0x28

Pointer to 0x28 at 0x3C

Our JavaScript + a few signatures => fooled type finder (Anti-Virus bypass).
-> "corrupted executable"

libMagic definition

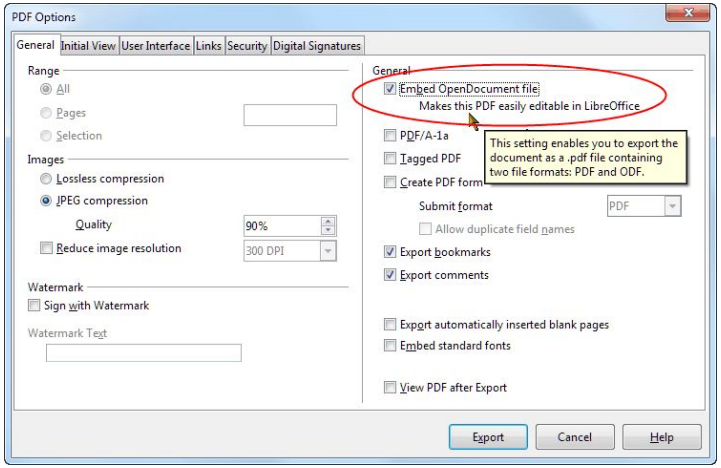
```
0       string/b        MZ
...
# Maybe it's a PE?
>>(0x3c.l) string PE\0\0 PE
!:mime application/x-dosexec
...
>>>(0x3c.l+24) default         x       Unknown PE signature
```

# Magic signatures at offset zero prevent multi-type files.

## Aka "binary polyglots":
## Easy security bypass.

*Story time: "stream formats traditionally don't have a header."*

# Polyglots in the wild




## Clean:

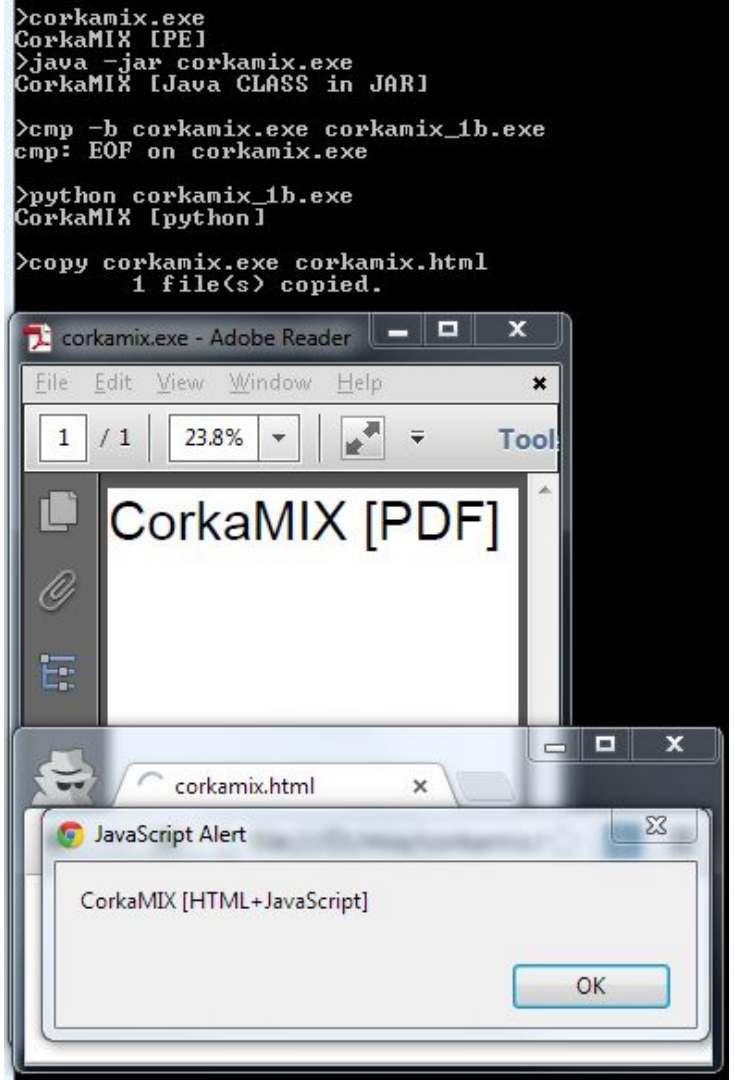
- hybrid ISOs : Iso + MBR
- self-extracting archives (executable+archive)
- hybrid PDFs: PDFs with embedded OpenOffice doc.

## Malicious:

- Gifar: avatar GIF with appended Java archive.
- CVE-2017-13156 Janus, DEX+APK

# HTML JavaScript Java Windows executable PDF

## 2 standard infection chains in a **single** file



# TYPE FINDER

1. Identify a type
2. Take a branch
3. End

*"In a perfect world,*
*There's no need to enforce*
*magic signatures at offset zero"*

Filtering can't take as long as parsing.

How many file types do we actually need to parse?

(hint: way too many)

*Story time*

Quizz Time !

# Which common file format usually starts with:

`00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00`

(a complete row of 16 zeroes)

[and actually more]

?

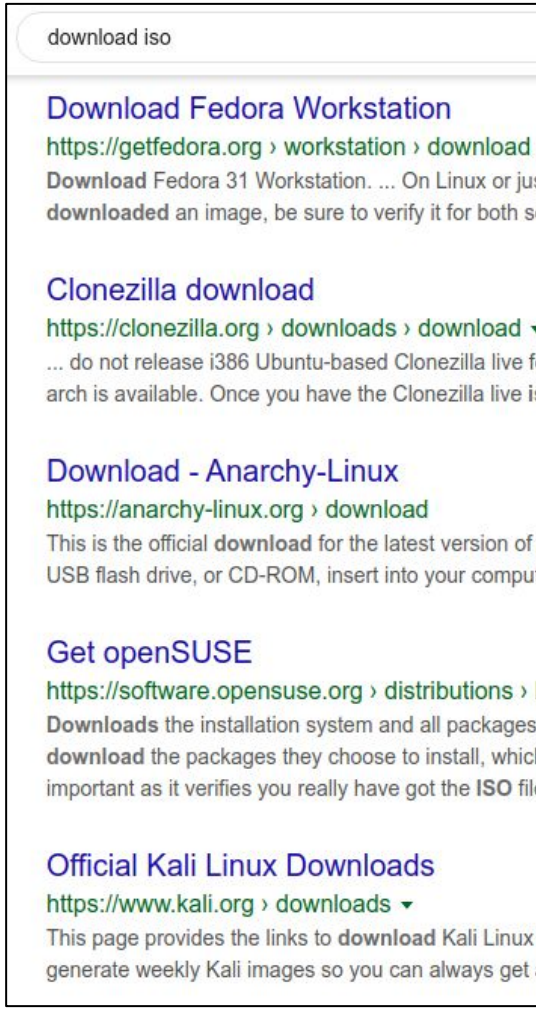
...which is not super useful for identification TBH.

# ISO 9660 - THE CD/DVD IMAGE DUMP FORMAT

MAGIC AT OFFSET 32KB (AFTER 16 SECTORS OF 2048 BYTES)

```
00000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
...
07000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
08000: 01 .C .D .0 .0 .1 01 00  .  .  .  .  .  .  .  .
...
```

CD001 AT 32KB+1



# DICOM

DIGITAL IMAGING AND COMMUNICATIONS IN MEDICINE

CONTENTS:
IMAGE, PATIENT INFORMATION,
ANNOTATIONS...

| | | | |
|---|---|---|---|
| (0002,0003) | UI | 1.2.826.0.1.3680043.8.1055.1.20111103112244831.30826609.78057758 | Media Storage SOP Instance UID |
| (0002,0010) | UI | 1.2.840.10008.1.2 | Transfer Syntax UID |
| (0002,0012) | UI | 1.2.826.0.1.3680043.8.1055.1 | Implementation Class UID |
| (0002,0013) | SH | dicomlibrary-100 | Implementation Version Name |
| (0002,0016) | AE | DICOMLIBRARY | Source Application Entity Title |
| (0008,0008) | CS | ORIGINAL,SECONDARY,OTHER,ARC,DICOM,VALIDATION | Image Type |
| (0008,0016) | UI | 1.2.840.10008.5.1.4.1.1.7 | SOP Class UID |
| (0008,0018) | UI | 1.2.826.0.1.3680043.8.1055.1.20111103112244831.30826609.78057758 | SOP Instance UID |
| (0008,0060) | CS | OT | Modality |
| (0008,0064) | CS | WSD | Conversion Type |
| (0010,0010) | PN | Anonymized | Patient's Name |
| (0010,0020) | LO | 0 | Patient ID |

## THE FORMAT YOUR DOCTOR USES...

```
000: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
...
070: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
080: .D .I .C .M-02 00 00 00-55 4C 04 00-D4 00 00 00
...
```

MAGIC AT `0x80`

DOCTORS:

NOT CONCERNED BY INFOSEC, CRITICAL,

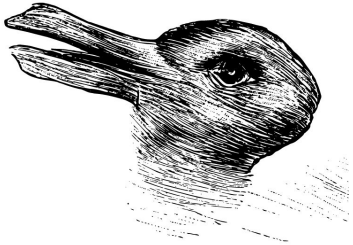
DEPENDING ON LESS SCRUTINIZED WEIRD FORMATS.

-> PERFECT TARGET.

# Magic signatures *could* differentiate file **intents**.

They should also be used to differentiate intents,
to compartimentalize security.
Same format but different use -> different magic please

# It's a DB dump... an archive... a file system!



Duck or Rabbit?

SQLite Archive: from DB to archive to fileSystem

Still the same thing: requires too much parsing to differentiate!

-> Please use a **different** magic instead!

```
00000  53 51 4c 69 74 65 20 66 6f 72 6d 61 74 20 33 00  SQLite format 3·
...
000A0  00 00 00 00 00 00 00 00 00 00 00 00 82 1e 01 07  ················
000B0  17 17 17 01 84 1b 74 61 62 6c 65 73 71 6c 61 72  ·····tablesqlar
000C0  73 71 6c 61 72 02 43 52 45 41 54 45 20 54 41 42  sqlar·CREATE TAB
000D0  4c 45 20 73 71 6c 61 72 28 0a 20 20 6e 61 6d 65  LE sqlar(·  name
...
```

eicar.sqlar

https://github.com/KyleBruene/sqlar/blob/master/sqlarfs.c

Open Suggestion

# Add a magic at offset 0 if there is none.

Just put a 4 letters filetype at the start.
Then a 4 letters subtype for intent if needed.
Then append the original file.
~~File confusion~~. ~~Intent confusion~~

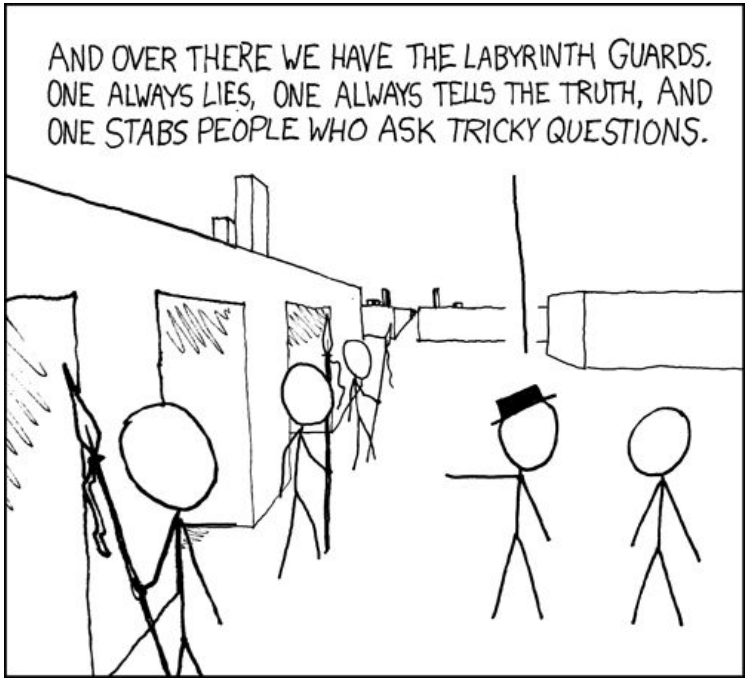
# FORMAT

# Duplicity -> discrepancy

The information is duplicated:
which source to rely?

In practice, rejecting 'incorrect' files is not tolerated.
See "spell-checking virus" myth.

CVE-2013-4787 Android master key:
1 files, 2 archived files: one verified, one executed.




https://xkcd.com/246/

# Confusion

*What may be so obvious to you now*
*may be seriously misleading to anyone else...*

← **LOOK**
← **RIGHT**

# Don't force 'traditions' into your file formats

Does your format make sense? Abstract it from the language of your current parsers.

Ex: Signed Int everywhere because the first parser was written in Java.

-> so `-32,767` is a valid version number...?

See also: bogus code with matching bogus tests.

Large Format Scanners:
Infinite "height" scans
-> image height fixed to 65535!

Tolerated by libJPEG,
So valid **everywhere**!

Detected by Anti-Virus, because it was used to exploit MS04-028.

# HEADER

```
%PDF-1.1
```

SIGNATURE & VERSION INFORMATION

# BODY

DICTIONARY << [ID VALUE]* >>

OBJECT REFERENCE: <OBJECT NUMBER> <REVISION NUMBER> R

IDENTIFIER (WITH / )

ARRAY

STREAM PARAMETERS: LENGTH, COMPRESSION.....

STRING

```
1 0 obj
<<
  /Pages 2 0 R
>>
endobj

2 0 obj
<<
  /Type /Pages
  /Count 1
  /Kids [3 0 R]
>>
endobj

3 0 obj
<<
  /Type /Page
  /Contents 4 0 R
  /Parent 2 0 R
  /Resources <<
    /Font <<
      /F1 <<
        /Type /Font
        /Subtype /Type1
        /BaseFont /Arial
      >>
    >>
  >>
>>
endobj

4 0 obj
<< /Length 50 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!)Tj
ET
endstream
endobj
```

BEGIN TEXT
FONT F1 (ARIAL) SET TO SIZE 110
MOVE TO COORDINATE 10, 400
OUTPUT TEXT "HELLO WORLD!"
END TEXT

# WHAT A NORMAL PDF USUALLY LOOKS LIKE.

(BUT DONE BY HAND, SO MUCH SMALLER THAN COMMON FILES)

# XREF TABLE

CROSS REFERENCE

```
xref                        CROSS REFERENCES
0 5                         5 OBJECTS, STARTING AT INDEX 0
0000000000 65535 f          (STANDARD FIRST EMPTY OBJECT 0
0000000010 00000 n          OFFSET TO OBJECT 1, REV 0
0000000047 00000 n          TO OBJECT 2...
0000000111 00000 n          3...
0000000313 00000 n          4
```

# TRAILER

```
trailer
<<
  /Root 1 0 R
>>

startxref
413
%%EOF
```

# What a **weird** PDF can look like.

This one works fine
with all readers
without any warning.

No XREF, no /Length, no /Size

```
%PDF-1.3
1 0 obj<</Type/Catalog/Pages 2 0 R>>endobj
2 0 obj<</Type/Pages/Count 1/Kids[3 0 R]>>endobj
3 0 obj<</Type/Page/Contents 4 0 R/Parent 2 0
R/Resources<</Font<</F<</Type/Font/Subtype/Type1/BaseFont/
Arial>>>>>>>>endobj
4 0 obj<<>>stream
BT/F 55 Tf 10 400 Td(http://www.corkami.com)' ET
endstream
endobj
trailer <</Root 1 0 R>>
```

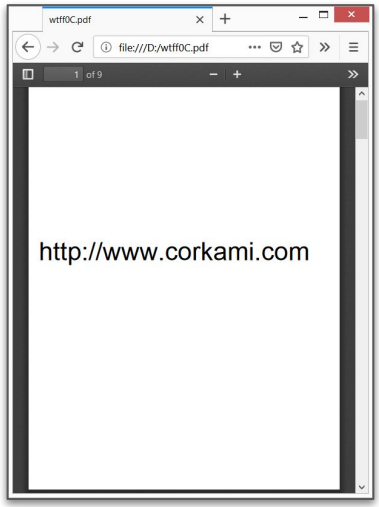
# What a **crazy** PDF can look like....

```
\t1\t0\tobj<</Resources<</Font<</<<BaseFont//Subtype/>>>>>>/Contents<<>>stream\n
/\t50Tf20\r450Td(http://www.corkami.com)Tjendstream>>endobj\x20
trailer<</Root<</Pages<</Kids[1\t0R]/Count\f9
```

# This is a valid PDF for firefox.

# It breaks so many rules, and yet...

# it works without any warning!



```
\t1\t0\tobj<</Resources<</Font<</<</BaseFont/Subtype/>>>>>>/Contents<<>>stream\n
/\t50Tf20\r450Td(http://www.corkami.com)Tjendstream>>endobj\x20
trailer<</Root<</Pages<</Kids[1\t0R]/Count\f9
```

No %PDF signature,no Type, no Parent...

Mixed whitespace. Empty font name, BaseFont, Subtype.

Recursive & inline stream object. Non-closed dictionaries.

No whitespace between keywords and numbers.

**9** pages counted but only **1** kid.

We really have a lot of cleaning to do...

```
$ mutool clean wtff0C.pdf
error: cannot recognize version marker
warning: trying to repair broken xref
error: invalid key in dict
error: cannot parse dict
error: invalid indirect reference in dict
error: cannot parse dict
error: cannot parse dict
error: cannot parse dict
error: invalid key in dict
error: cannot parse dict
error: cannot load object (1 0 R) into cache
warning: ignoring broken object (1 0 R)
error: invalid key in dict
error: cannot parse dict
error: cannot load object (1 0 R) into cache
warning: cannot load object (1 0 R) into cache

$ qpdf wtff0C.pdf repaired.pdf
WARNING: wtff0C.pdf: can't find PDF header
WARNING: wtff0C.pdf: file is damaged
WARNING: wtff0C.pdf: can't find startxref
WARNING: wtff0C.pdf: Attempting to reconstruct cross-reference table
wtff0C.pdf: unable to find trailer dictionary while recovering damaged file

$
```

This crazy PDF **can't be repaired** with standard tools.

## Output from `mutool`: (it's empty)

```
%PDF-0.0
%%µÕ

1 0 obj
null
endobj
xref
0 2
0000000000 65536 f
0000000018 00000 n

trailer
<</Size 2>>

startxref
38
%%EOF
```

# Hash collisions

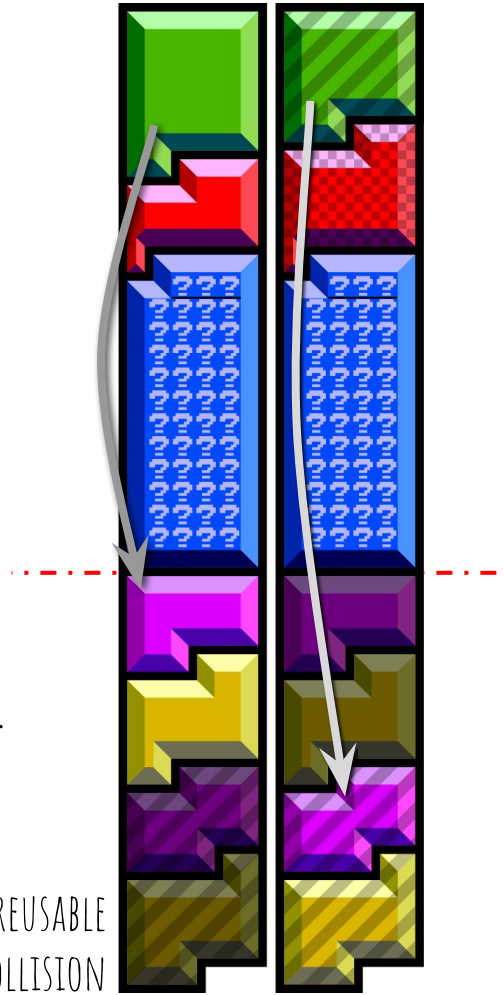
Normalize files. Filter out comments.

Check the end of the files.

# Hash collisions and file formats?

Hash collisions already exist for MD5 & SHA1.
They can be combined with file formats tricks for faster results.
-> instant collisions of arbitrary JPG, PNG, GIF / MP4 / PE / PDF....
They create valid, but very weird files structure-wise

If you can't use another hash algorithm, you can filter out files.
You can also define formats to make collision exploitation harder.



Layouts of a reusable chosen-prefix collision

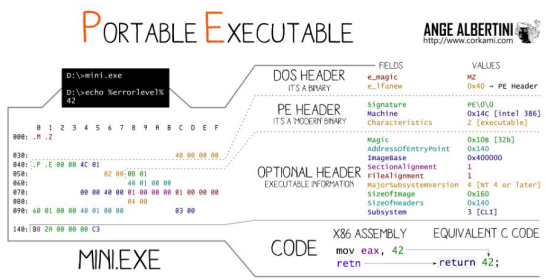
https://github.com/corkami/collisions

# More details in my repository

Docs:
- Attacks
- Tricks
- Strategies
- talk
- workshop

Files:
- Test PoCs
- Scripts

**Portable Executable**




The Portable Executable has a peculiar structure:

- the old DOS header is almost useless, and points to the next structure, the PE header. The DOS headers has no other role. DOS headers can be exchanged between executables.
- the DOS header has to be at offset 0, and has a fixed length of a full block, and the pointer is at the end of the structure, beyond UniColl's reach: so only chosen-prefix collision is useful to collide PE files this way.
- The PE header and what follows defines the whole file.

So the strategy is:

1. the PE header can be moved down to leave room for collision blocks after the DOS header.
2. The DOS header can be exploited (via chosen-prefix collisions) to point to two different offsets, where two different PE headers will be moved.
3. The sections can be put next to each other, after the `DOS/Collisions/Header1/Header2` structure. You just need to apply a delta to the offsets of the two section tables.

This means that it's possible to instantly collide any pair of PE executables. Even if they use different subsystems or architecture.

While executables collisions is usually trivial via any loader, this kind of exploitation here is transparent: the code is identical and loaded at the same address.

Examples: tweakPNG.exe (GUI) ↔ fastcoll.exe (CLI)

Here is a script to generate instant MD5 collisions of Windows Executables.

## Collisions examples

### MD5

#### FastColl

single frame GIF: collision1.gif / collision2.gif

#### UniColl

JPG: collision1.jpg / collision2.jpg - tldr-1.jpg / tldr-2.jpg

PDF: collision1.pdf / collision2.pdf

PNG:

- generic headers (not OS X compatible): collision1.png / collision2.png
- specific headers (same metadata): 0a959025-1.png / 0a959025-2.png - aac2423a-1.png / aac242

JP2: collision1.jp2 / collision2.jp2

MP4:

- generic header, 32b length (LTV): collision1.mp4 / collision2.mp4
- generic header, 64b length (TLV): collision1.mp4 / collision2.mp4
  - specific header: collisions1.mp4 / collisions2.mp4

Strategies:

- Good/bad contents (gotta catch 'em all): gcea1.png / gcea2.png
- Valid/Invalid: png-valid.png / png-invalid.png

#### Multiple UniColl

poeMD5 (not Adobe compatible by accident): poeMD5_A.pdf / poeMD5_B.pdf

ZIP: collision1.zip / collision2.zip

#### HashClash

PE: collision1.exe / collision2.exe

#### polycolls

- JPG / PE: jpg-pe.exe / jpg-pe.jpg
- PE / PDF: pepdf.exe / pepdf.pdf
- PNG / PDF: png-pdf.pdf / png-pdf.png

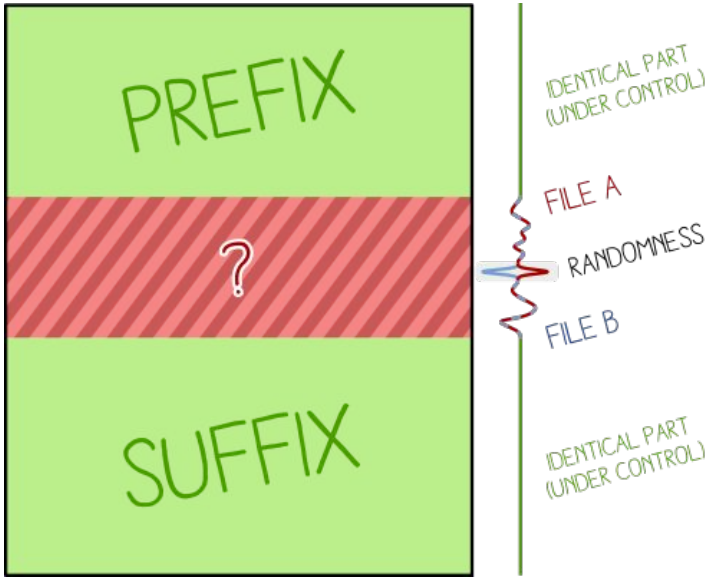
# Similarities of all current collision attacks

All current hash collisions attacks work with 64B alignment:
padding, then adding (at block boundaries) a number of blocks.

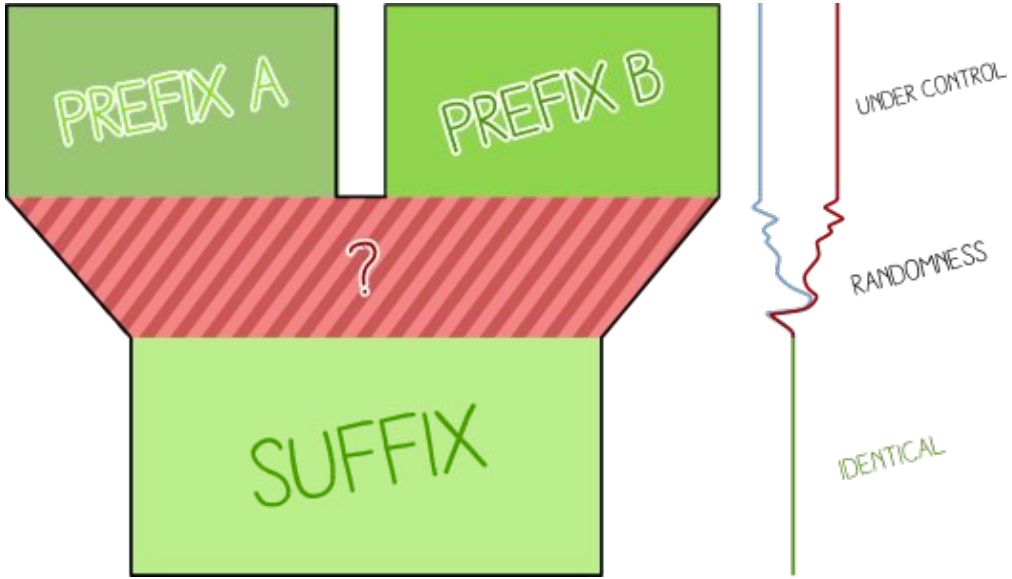
-> Via these attacks:

1- Every pair with the same hash will have the **same length**.

2- The end of the files is either **identical** (suffix),
Or high entropy, very similar and aligned to 64 bytes
(no suffix, just collision blocks).




Identical




Chosen

# Collision types

# An MD5 collision of YES and NO.

```
0000:  .y .e .s 00-00 00 00 00-00 00 00 00-00 00 00 00
0010:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0020:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0030:  00 00 00 00-00 00 00 00-B7 46 38 09-8A 46 F1 7B
0040:  F3 45 26 13-66 60 C8 01-B9 2A 75 25-5A 67 23 A6
0050:  92 3D EB 8D-B0 B7 57 F1-45 9F 22 95-BE C0 43 75
0060:  91 98 A2 D3-E0 FD 59 ED-D1 C5 FA 0B-79 65 97 4D
0070:  B3 B3 E4 0C-11 0C 90 32-DE 4B A1 4B-B8 1B 5E C8
0080:  25 D3 8F 19-CD 10 43 07-D9 BB FF 8C-B7 5A 23 F9
0090:  4D D8 13 14-58 A3 35 97-C5 D1 D4 A9-9A E2 FD 1F
00A0:  BA 78 40 00-C3 7E 93 B2-31 A3 6E 2D-34 6A 4A C9
00B0:  53 4E C0 45-36 1E C8 6A-56 98 E6 F0-57 1D 61 98
00C0:  13 FC FF CD-4D 83 A2 D2-BB B8 DC 04-2B E2 B8 83
00D0:  DB 53 80 D7-3D E9 97 D3-23 5A 27 F9-98 9A E7 56
00E0:  7D 86 E4 35-1E B8 33 EE-EA 15 D1 81-BA 96 62 EC
00F0:  75 31 FB DA-4F AE 24 6F-67 D6 AF 10-96 29 FB C7
0100:  A3 32 BB A9-EA D5 E4 AE-1F C2 FB 23-41 22 B2 E0
0110:  69 1E 29 20-6F 5B 20 1E-5E 3D 11 2F-3E 4D 9F 39
0120:  8B C9 5C 93-A5 EF A4 22-7D 9A 66 51-6E ED AD 70
0130:  32 90 D4 BD-67 92 38 9B-DC 15 0D BF-DC 71 72 27
0140:  E0 5B 43 FA-44 59 E8 60-F7 63 7F F0-73 0A D4 BE
0150:  33 28 AA 99-2C 90 2D D0-01 58 E3 8F-58 50 30 99
0160:  E8 60 DB 91-00 13 C9 1D-7A 61 9B 9A-5D 5E BD 71
0170:  23 1A D2 BD-A6 E0 38 66-0B 8C F5 99-56 79 63 D6
0180:  6E 5E D7 7E-C3 4E 9D 5F-65 23 C0 38-C9 55 5A A1
0190:  E2 3C CA 78-58 4D B5 3B-04 45 C3 B4-44 C8 87 26
01A0:  02 60 F6 62-91 34 70 FE-C3 34 54 6D-76 07 7F 1A
01B0:  73 53 E6 0B-08 FB 82 80-AD 5F 22 15-18 69 B5 6E
01C0:  BB 06 C3 A7-FF 39 15 52-BE FE D4 5C-D2 55 5A 71
01D0:  EC E9 BC 1A-B7 BB 08 61-C5 3E E7 89-7C 93 03 FC
01E0:  1F 8A 9A D8-42 BF 6C 01-6A 39 26 84-74 58 E2 E4
01F0:  00 D4 67 7B-27 BD 93 6D-DF F0 10 4A-2B 00 7E 68
0200:  1D DE D5 8A-67 89 EA 52-0C 32 BD 30-A2 8C BE D0
0210:  A7 35 BA C6-BB 7D 07 80-49 22 EF E5-10 B2 83 6D
0220:  E6 18 6E E3-F0 52 E4 35-83 61 42 35-72 97 C5 8D
0230:  4F F7 93 68-5A 70 5F 5A-04 3A D5 42-C1 FA 0F E2
0240:  AE 57 DB AF-F1 51 B8 B7-38 18 EF 2E-B8 A6 A9 2C
0250:  81 87 FA FE-B2 C4 DC 45-A3 64 91 6D-B8 6E F5 D1
0260:  4F 9C FA 62-3D 42 46 59-67 32 EC 99-DA 89 7A 88
0270:  E7 AD E3 21-ED 3C 4B C0-4D 9F 83 3C-DC 7F B7 0A
```

Padding

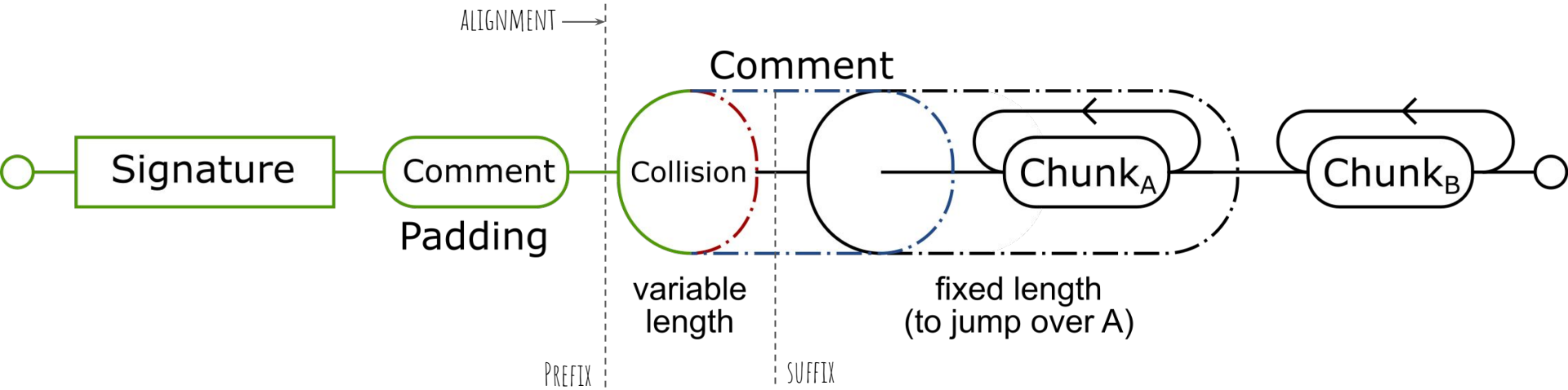
Random buffer
(partial birthday attack bits)

Collision blocks

```
0000:  .n .o 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0010:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0020:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0030:  00 00 00 00-00 00 00 00-19 71 E7 F7-09 72 FB 06
0040:  F3 45 26 13-66 60 C8 01-B9 2A 75 25-5A 67 23 A6
0050:  92 3D EB 8D-B0 B7 57 F1-45 9F 22 95-BE C0 43 75
0060:  91 98 A2 D3-E0 FD 59 ED-D1 C5 FA 0B-79 65 97 51
0070:  B3 B3 E4 0C-11 0C 90 32-DE 4B A1 4B-B8 1B 5E C8
0080:  25 D3 8F 19-CD 10 43 07-D9 BB FF 8C-B7 5A 23 F9
0090:  4D D8 13 14-58 A3 35 97-C5 D1 D4 A9-9A E2 FD 1F
00A0:  BA 78 40 00-C3 7E 93 B2-31 A3 6E 2D-34 72 4A C9
00B0:  53 4E C0 45-36 1E C8 6A-56 98 E6 F0-57 1D 61 98
00C0:  13 FC FF CD-4D 83 A2 D2-BB B8 DC 04-2B E2 B8 83
00D0:  DB 53 80 D7-3D E9 97 D3-23 5A 27 F9-98 9A E7 56
00E0:  7D 86 E4 35-1E B8 33 EE-EA 15 D1 81-FA 96 62 EC
00F0:  75 31 FB DA-4F AE 24 6F-67 D6 AF 10-96 29 FB C7
0100:  A3 32 BB A9-EA D5 E4 AE-1F C2 FB 23-41 22 B2 E0
0110:  69 1E 29 20-6F 5B 20 1E-5E 3D 11 2F-3E 4D 9F 39
0120:  8B C9 5C 93-A5 EF A4 22-7D 9A 66 51-6E ED AF 70
0130:  32 90 D4 BD-67 92 38 9B-DC 15 0D BF-DC 71 72 27
0140:  E0 5B 43 FA-44 59 E8 60-F7 63 7F F0-73 0A D4 BE
0150:  33 28 AA 99-2C 90 2D D0-01 58 E3 8F-58 50 30 99
0160:  E8 60 DB 91-00 13 C9 1D-7A 61 9B 9A-5D 60 BD 71
0170:  23 1A D2 BD-A6 E0 38 66-0B 8C F5 99-56 79 63 D6
0180:  6E 5E D7 7E-C3 4E 9D 5F-65 23 C0 38-C9 55 5A A1
0190:  E2 3C CA 78-58 4D B5 3B-04 45 C3 B4-44 C8 87 26
01A0:  02 60 F6 62-91 34 70 FE-C3 34 54 6D-76 07 FF 1A
01B0:  73 53 E6 0B-08 FB 82 80-AD 5F 22 15-18 69 B5 6E
01C0:  BB 06 C3 A7-FF 39 15 52-BE FE D4 5C-D2 55 5A 71
01D0:  EC E9 BC 1A-B7 BB 08 61-C5 3E E7 89-7C 93 03 FC
01E0:  1F 8A 9A D8-42 BF 6C 01-6A 39 26 84-6C 58 E2 E4
01F0:  00 D4 67 7B-27 BD 93 6D-DF F0 10 4A-2B 00 7E 68
0200:  1D DE D5 8A-67 89 EA 52-0C 32 BD 30-A2 8C BE D0
0210:  A7 35 BA C6-BB 7D 07 80-49 22 EF E5-10 B2 83 6D
0220:  E6 18 6E E3-F0 52 E4 35-83 61 42 35-72 97 CD 8D
0230:  4F F7 93 68-5A 70 5F 5A-04 3A D5 42-C1 FA 0F E2
0240:  AE 57 DB AF-F1 51 B8 B7-38 18 EF 2E-B8 A6 A9 2C
0250:  81 87 FA FE-B2 C4 DC 45-A3 64 91 6D-B8 6E F5 D1
0260:  4F 9C FA 62-3D 42 46 59-67 32 EC 99-DA 89 7A 08
0270:  E7 AD E3 21-ED 3C 4B C0-4D 9F 83 3C-DC 7F B7 0A
```

# Prevent hash collisions?

Reject: appended data. (a long-lasting tradition)
weird/multiple comments (we need 3 of them)



# Conclusion

*Never attribute to malice*
*that which can be adequately*
*explained by stupidity.*

*Robert J. Hanlon*

# In our security bubble,
# we easily forget that some people
# will still do things the possible worst way
# just because of some "traditions".

More preaching is needed.

Fuzzing/Failing/Fixing is not enough - on our side.

Sandboxing/hardening/normalizing is an after-fix.

Future plans?

# Magic at offset zero

## Yes, seriously!

Open suggestion:

- If there's none, define and prepend one - move the file by 4 bytes.
- Define a submagic at offset 4 if the intent is changed

  Ex w/ SQLAR: from DB dump to file system.

# Duplicity

Prevent any. If not, bad things **will** happen.

Mistakes -> tolerance -> over-tolerance -> discrepancy.

# Specs obsolescence

They don't explain the need for security.

Why aren't CVEs reflected back in the original document?

They don't prevent people to shoot themselves in the foot.

Too many formats/parsers to Fuzz/Fail/Fix.

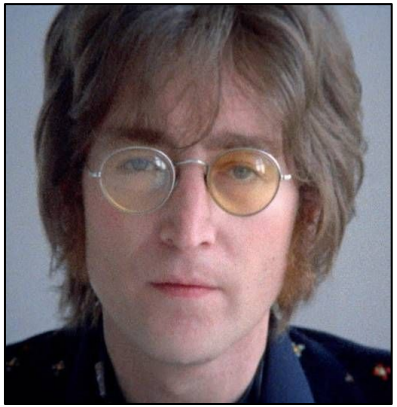
# Extras

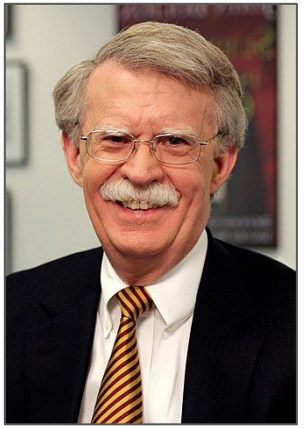
# Duplicity

*Let's ask John!*
*Well…which one?*

Cena / McEnroe
Wick / Travolta / Wayne / Cleese / Carpenter
Lennon / Bonham / Williams
Kennedy / Bolton / McCain / Kerry
Deere / Rockfeller
Stewart / Oliver

Elton / Jon St





# Thank you!
# Any feedback?

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

Acknowlegdments:
Philippe Teuwen

**Formats de fichiers**
Décisions et conséquences

*Ange Albertini*
🔢📜:🔪&👩