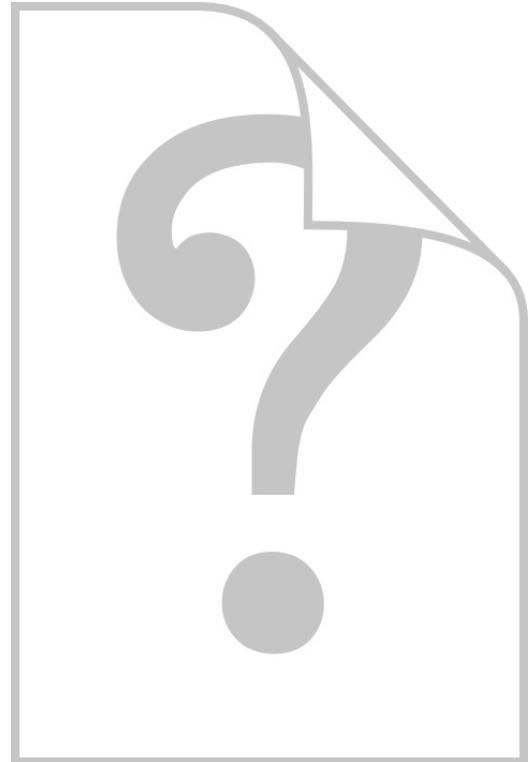


# THE CHALLENGES

OF

# FILE FORMATS

SHARING MY PERSPECTIVES  
WITH THE DIGIPRES COMMUNITY



ANGE ALBERTINI

# MY PERSPECTIVES

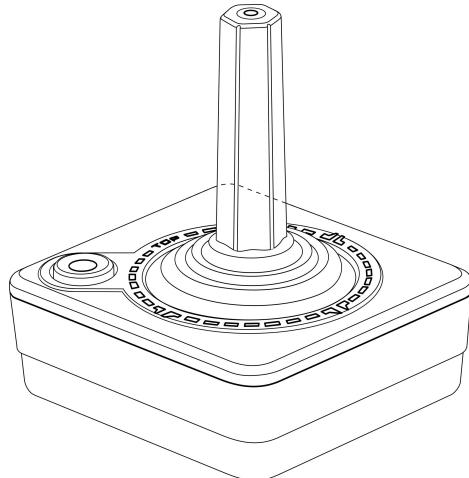
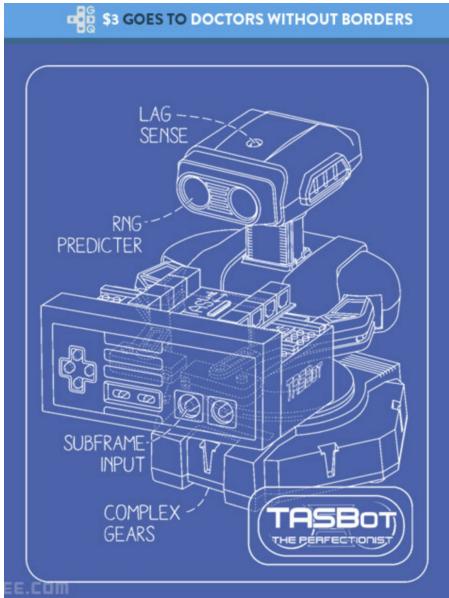
USE  
LOAD/SAVE  
SHARE

PRESERVE  
DOCUMENTS  
VIDEO GAMES

DISSECT  
MALICIOUS FILES

CRAFT  
EXTREME FILES

# TINKERING WITH COMPUTERS FOR 30+ YEARS...



I DRAW MANY KINDS OF THINGS...

...AND DOCUMENT FILE FORMATS.

# WAV<sup>101</sup> an audio file walkthrough

**PE<sup>101</sup>** a windows executable walkthrough

## Dissected PE

## Loading process

**① Headers**  
Headers like the DOS header, PE header, and optional header are loaded into memory. The optional header is parsed to determine the entry point.

**② Sections table**  
Sections like .text, .data, and .rsrc are loaded into memory. If it contains relocation elements or imports, the loader performs relocations and imports in the AMF entry.

**③ Mapping**  
VirtualAlloc() is called to map memory according to the section headers. This maps the .text section to the executable file's memory and the .rsrc section to the file's resources.

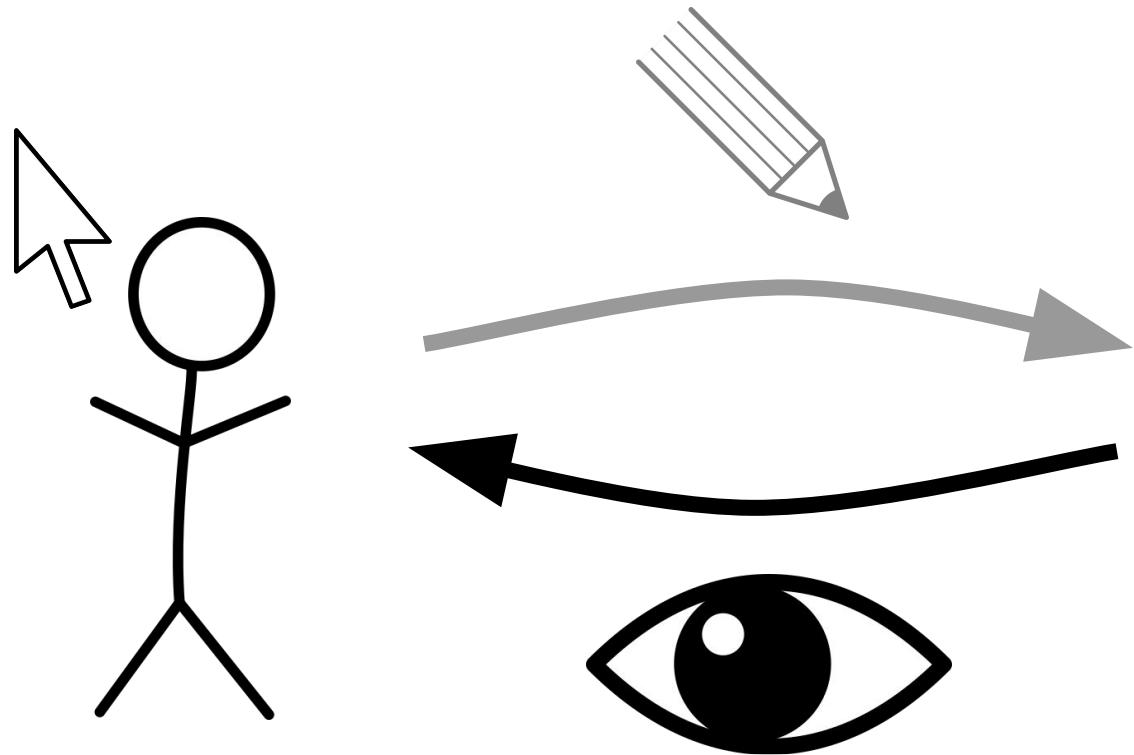
**④ Imports**  
Imports are resolved. They use the Ordinal resolver to find the ordinal number of the imported symbol. They also use the Name resolver to find the name of the imported symbol. This is done in memory and the loader performs relocations for each entry in the AMF entry.

**⑤ Execution**  
The entry point is determined. The code of the entry point is loaded into the AMF entry.

**Notes**

- PE Headers**  
Starts with NT 'P' header. Contains the COFF header. Contains the optional header which contains the entry point, code alignment, and other fields.
- Sections table**  
Tables like .text, .data, and .rsrc. It contains relocation elements and imports. It contains relocation elements and imports. Addressing is done using segments.
- Mapping**  
VirtualAlloc() is called to map memory according to the section headers. This maps the .text section to the executable file's memory and the .rsrc section to the file's resources.
- Imports**  
Imports are resolved. They use the Ordinal resolver to find the ordinal number of the imported symbol. They also use the Name resolver to find the name of the imported symbol. This is done in memory and the loader performs relocations for each entry in the AMF entry.
- Execution**  
The entry point is determined. The code of the entry point is loaded into the AMF entry.
- Notes**
- IMP Import Table**  
Imports are resolved by the loader. File names, base addresses, and imports are resolved.
- IMF Import Address Table**  
Imports are resolved by the loader. On file is the offset of the IMP table.
- AMF Address Map Table**  
Imports are resolved by the loader. It contains the exports table of a DLL to be imported.
- HENT**  
Imports the exports table of a DLL to be imported. Not required but provides a speed-up by reducing read-up.

[github.com/corkami/docs](https://github.com/corkami/docs)

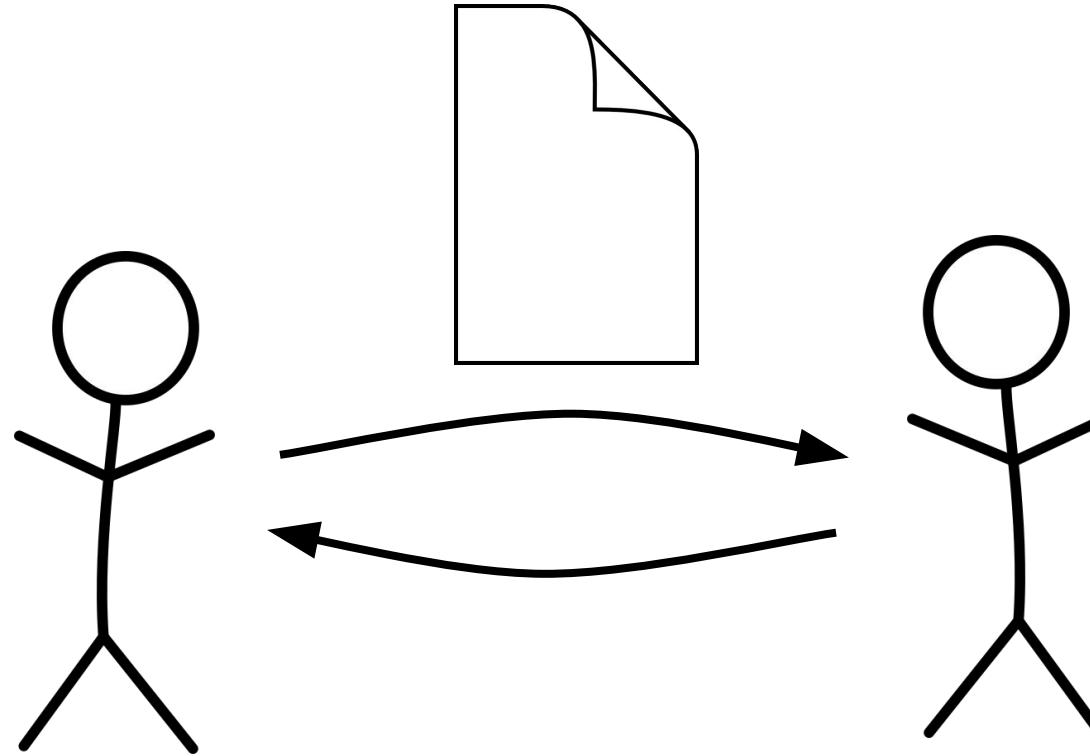


THE FIRST USE OF FILE FORMATS:  
"SAVE YOUR PROGRESS"

EVER PRINTED SOMETHING  
THAT LOOKED *NOTHING* LIKE  
WHAT YOU HAVE ON SCREEN?

*Story time:*

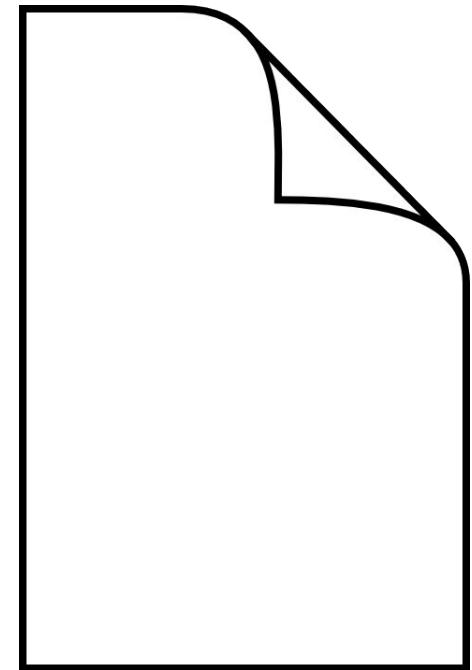
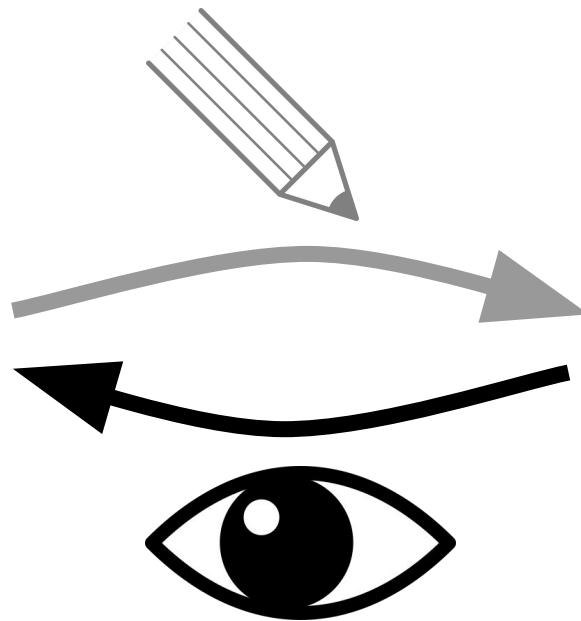
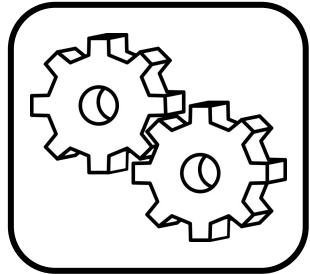
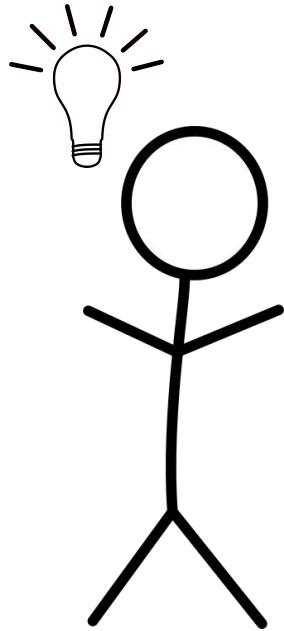
*A document without vowels*



ENABLING PEOPLE TO SHARE INFORMATION:  
A WONDERFUL DIGITAL LANGUAGE.

EVER TRIED TO IMPORT  
SOMEONE ELSE'S  
WORD DOCUMENT ?

"OUCH !"

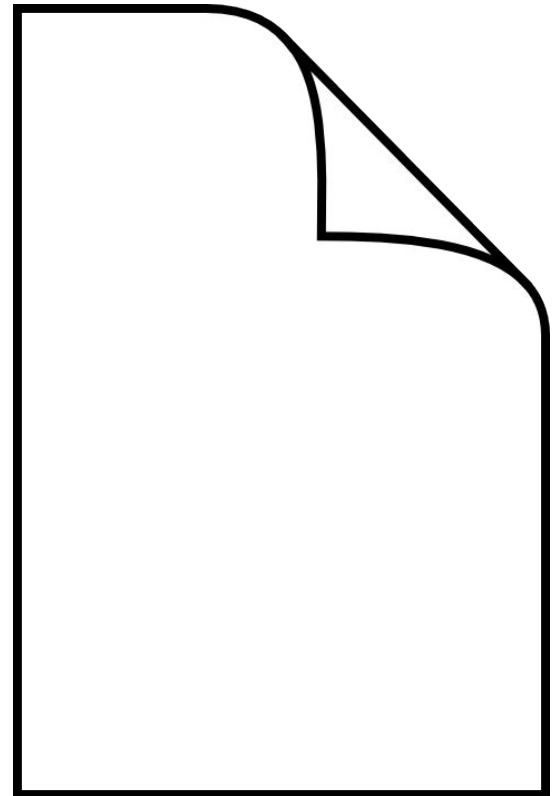
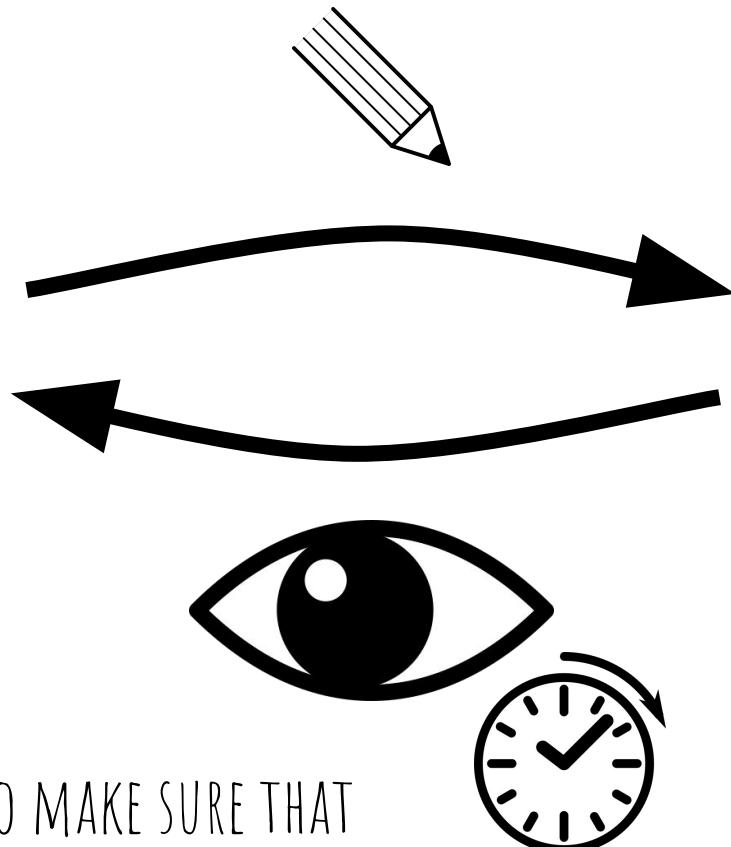
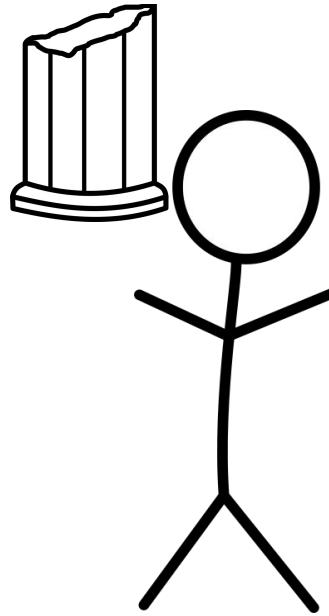


THE DEVELOPER RELIES ON THE SPECIFICATIONS  
TO ADD SUPPORT IN THEIR LIBRARY.

PRESERVE . . .

- INSERT RANT HERE -

WEB PAGES ANYONE ? ;)



THE ARCHIVIST WANTS TO MAKE SURE THAT  
THEIR DATA WILL BE RE-USABLE MUCH LATER.

EVER TRIED TO RE-IMPORT  
AN *OLD* WORD DOCUMENT ?



# I'M ALSO A REVERSE ENGINEER

- INTERESTED SINCE 1989
- VIDEO GAMES PRESERVATION IN 1999



comme endommages dans la table d'allocation 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-

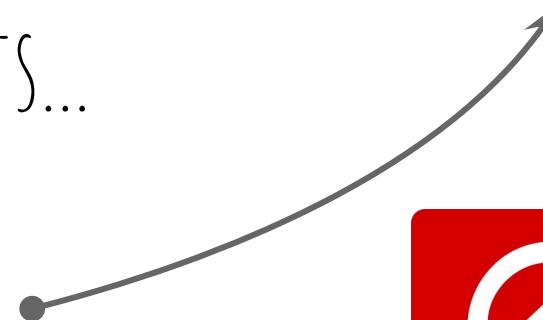
SCIENCE & VIE MICRO, NOVEMBER 1989  
INSTRUCTIONS TO MANUALLY REMOVE A BOOT SECTOR VIRUS

BUT...

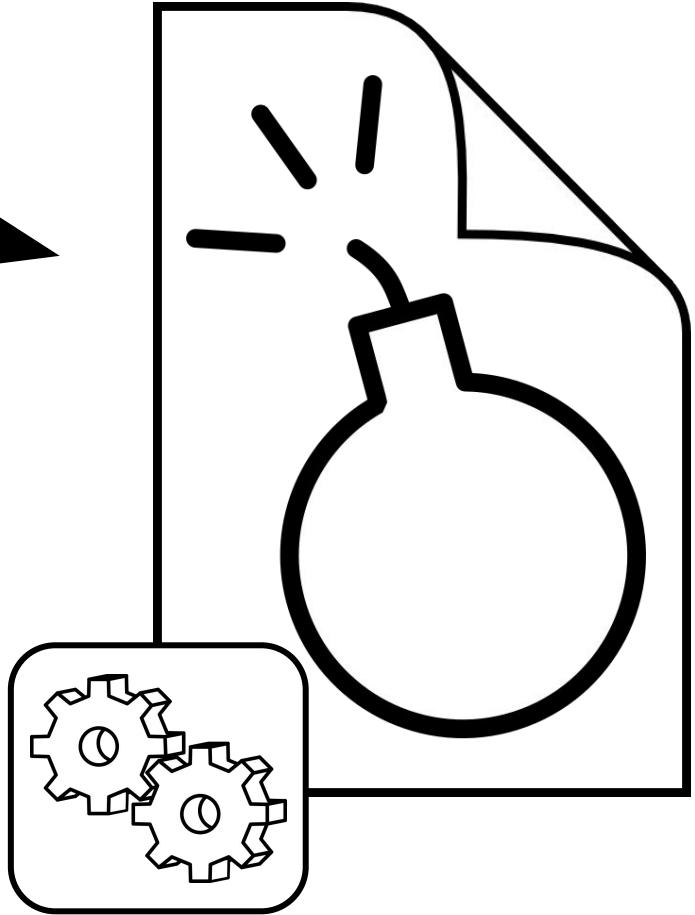
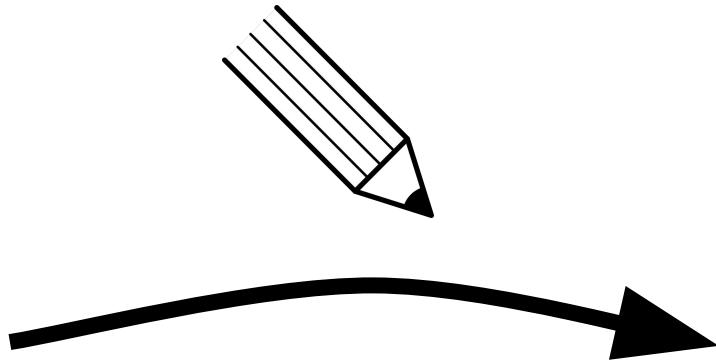
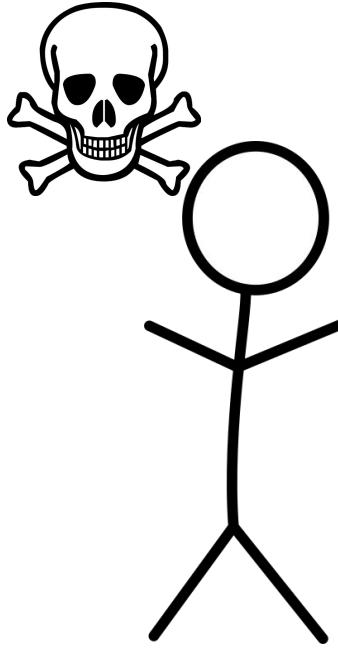
WE LIVE IN DARK TIMES!

# PROFESSIONALLY

- 12 YEARS OF MALWARE ANALYSIS
- EXECUTABLES, DOCUMENTS...



NOTE: THIS TALK REFLECTS MY OWN OPINION, NOT MY EMPLOYER.



ATTACKERS WILL TRY TO TAKE  
ADVANTAGES OF WEAKNESSES...

# BLACK HATS

FIND (AND SELL) VULNERABILITIES

IN SOFTWARE THAT CAN BE EXPLOITED

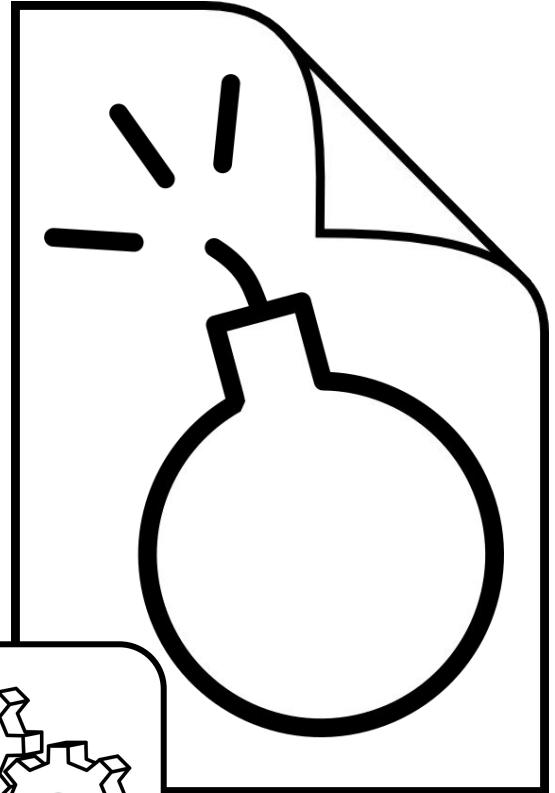
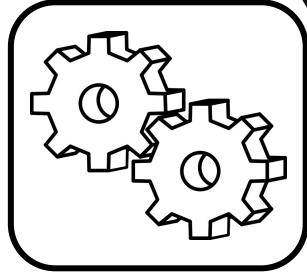
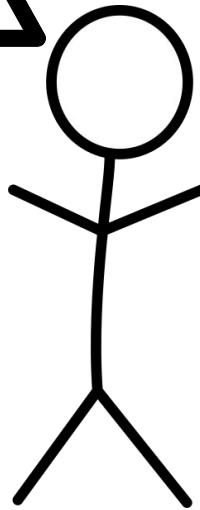
TO STEAL INFORMATION OR MONEY,

OR SPY ON PEOPLE (AND GET THEM ARRESTED...).

GOALS: BYPASS SECURITY, HACK INTO SYSTEMS.

# FIND BUGS TO HACK YOUR TARGET.

- ANALYZE CODE (SOURCE)  
OR DISASSEMBLY (THE BINARY ITSELF)
- BLIND FUZZING (FLIP BITS RANDOMLY AND SEE WHAT HAPPENS)
- GENERATIONAL FUZZING (ACCORDING TO THE STRUCTURE OF THE FILE FORMAT)
- SMART FUZZING (MODIFY/INSTRUMENT/ANALYSE CODE)



...WHILE DEFENDERS WILL TRY  
TO PREVENT THIS FROM HAPPENING.

DO THE SAME AS THE BAD GUYS

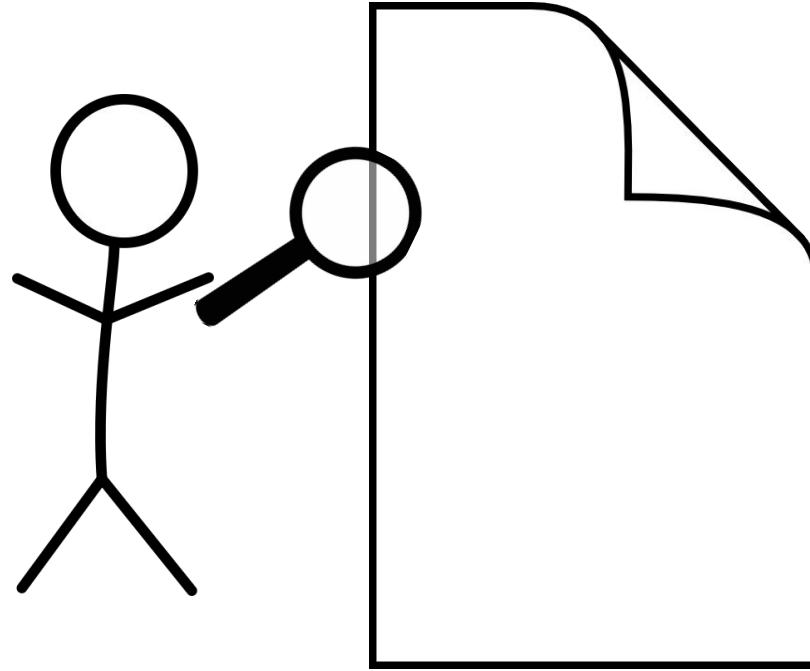
BUT DISCLOSE AND FIX VULNERABILITIES (SEE BUG BOUNTIES)

- REMOVE THE LOW-HANGING FRUITS FROM THE TREE.
  - CONTROLLED BURN TO PREVENT HUGE FIRES.
- => IMPROVE PRACTICES

IMHO FILE FORMATS SHOULD RECEIVE THIS TREATMENT TOO.

# ANTI-MALWARE INDUSTRY

- 1- ANALYZE FILES
- 2- DETERMINE IF THEY'RE CORRUPTED, OR MALICIOUS
- 3- COME UP WITH WAYS TO DETECT THEM
- 4- IMPROVE DEFENSES



MALWARE ANALYST OR DFIR ARE LOOKING FOR CLUES.  
(TYPICALLY, I OPEN FILES WITH A HEX EDITOR FIRST)

# DIGITAL FORENSICS - INCIDENT RESPONSE

DETERMINE IF AND HOW A FILE OR SYSTEM WAS:

- HACKED
- TAMPERED WITH (CASINO MACHINES)
- IF DATA WAS STOLEN (COPYRIGHT INFRINGEMENT)

# ONE MORE THING...

SO FAR, JUST INFOSEC THINGS:  
NOTHING FOR THE DIGIPRES WORLD.



# PROFESSIONALLY (NEXT)

- I WAS FED UP OF BEING ONLY RETROACTIVE TO ATTACKERS' "INNOVATION"
- I STARTED TO EXPERIMENT AND CREATE MY OWN FILES, FROM SCRATCH.  
AND SHARE THEM OPENLY AND FREELY.

# POLYGLOTS

A SINGLE FILE WITH MULTIPLE TYPES:

IT'S NOT A GADGET,

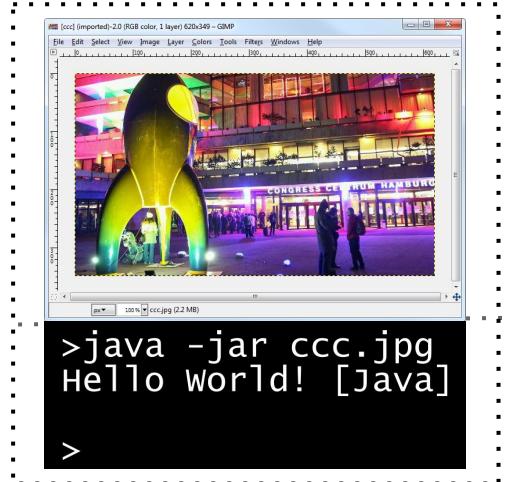
IT'S ACTUALLY USEFUL TO HACK PEOPLE!

USED IN THE WILD, IN 2008!

[HTTPS://EN.WIKIPEDIA.ORG/WIKI/GIFAR](https://en.wikipedia.org/wiki/Gifar)

IMAGE

JAVA

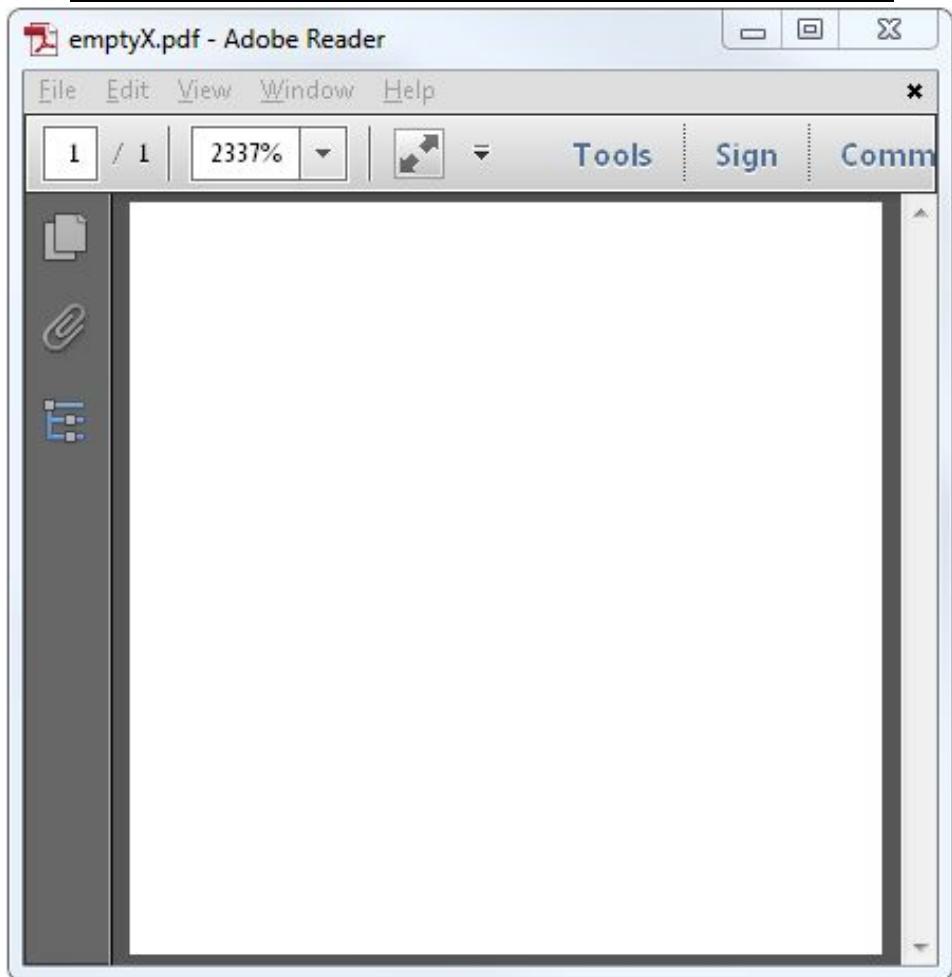


%PDF-\0trailer<</Root<</Pages<<>>>>

"TOO SMALL TO BE  
SUSPICIOUS!"

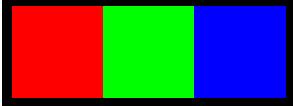
(OR EVEN CONSIDERED VALID)

A MINI PDF (ADOBESTRONG-ONLY, 36 BYTES)



# HOW DO I DO IT ?

1. I STUDY A SIMPLE FILE WITH THE SPECS
2. I CREATE MY OWN SCRIPT TO REPRODUCE IT  
(STANDARD TOOLS USUALLY DON'T GIVE YOU TOTAL CONTROL)
3. I CAN NOW CREATE MY OWN FILES, WITH FULL CONTROL
4. I CAN THEN EXPERIMENT,  
AND OPTIONALY, DOCUMENT AND VISUALIZE.



## Portable Network Graphics (PNG) Specification (Second Edition)

### Information technology — Computer graphics and image processing — Portable Network Graphics (PNG) Specification (Second Edition)

W3C Recommendation 10 November 2003

This version:  
<http://www.w3.org/TR/2003/REC-PNG-20031110>

Latest version:  
<http://www.w3.org/TR/PNG>

Previous version:  
<http://www.w3.org/TR/2003/PR-PNG-20030520>

Editor:  
David Duce, Oxford Brookes University (Second Edition)

Authors:  
See author list

Please refer to the [errata](#) for this document, which may include some normative corrections.

See also the [translations](#) of this document.

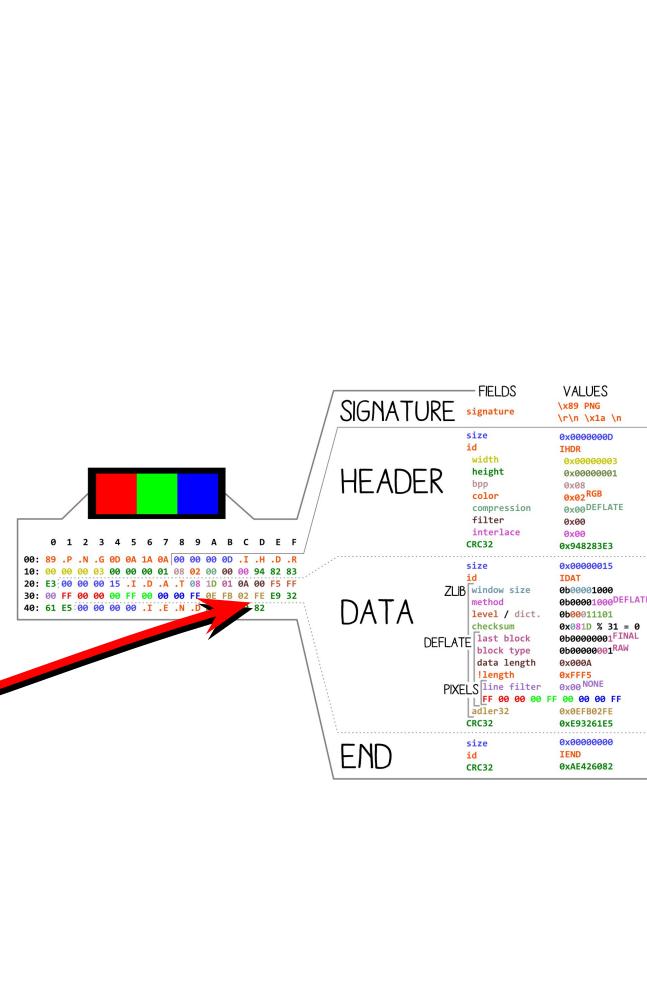
Copyright © 2003 W3C® (MIT, ERCIM, Keio). All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.

rgb.png/header					
address	name	type	size	data	description
89 50 4e 47 0d 0a 1a 0a 00 00 00 0d 49 48 44 52 00 00 00 03 00 00 00 01 08 02 00 00					
00 94 82 83 00 00 00 15 49 44 41 54 08 1d 01 0a 00 f5 ff ff 00 00 00 ff 00 00					
00 ff 0e fb 02 fe e9 32 61 e5 00 00 00 00 49 45 4e 44 ae 42 60 ff					
address	name	type	size	data	description
0000000.0 size	UInt32	00000004.0	13	Size	
000000c.0 tag	FixedSize<ASCII>	00000004.0	"IHDR"	Tag	
0000010.0 width	UInt32	00000004.0	3	Width (pixels)	
0000014.0 height	UInt32	00000004.0	1	Height (pixels)	
0000018.0 bit_depth	UInt8	00000001.0	8	Bit depth	
0000019.0 reserved	NullBits	00000000.5	<null>		
0000019.5 has_alpha	Bit	00000000.1	False	Has alpha channel?	
0000019.6 color	Bit	00000000.1	True	Color used?	
0000019.7 has_palette	Bit	00000000.1	False	Has a color palette?	
000001a.0 compression	UInt8	00000001.0	deflate	Compression method	
000001b.0 filter	UInt8	00000001.0	0	Filter method	
000001c.0 interlace	UInt8	00000001.0	0	Interlace method	
000001d.0 crc32	UInt32	00000004.0	0x948283e3	CRC32	

```
import struct
import binascii
import zlib

def make(chunks):
    s = ["\x89PNG\x0d\x0a\x1a\x0a"]
    for type_, data in chunks:
        s += [
            struct.pack(">I", len(data)), # length
            type_, # type
            data, # data
            struct.pack(">I",
                       binascii.crc32(type_ + data)
                       % 0x100000000)
        ]
    return "".join(s)

with open("rgb.png", "wb") as f:
    f.write(make([
        ["IHDR", # Image Header
         struct.pack(">I" * 6,
                     3, # Width
                     1, # Height
                     8, # Depth: 8
                     2, # Colour type: RGB
                     0, # Compression: Deflate
                     0, # Filter: adaptive filtering
                     0 # Interlace: no
                 ),
        ["IDAT", # Image Data
         zlib.compress(
             "\0" + # no interlacing
             "\xFF\x00\x00" + # Red pixel
             "\x00\xFF\x00" + # Green pixel
             "\x00\x00\xFF" # Blue pixel
         , 0)], # no compression
        ["IEND", # Image End
         []] # Empty chunk
    ]))
```

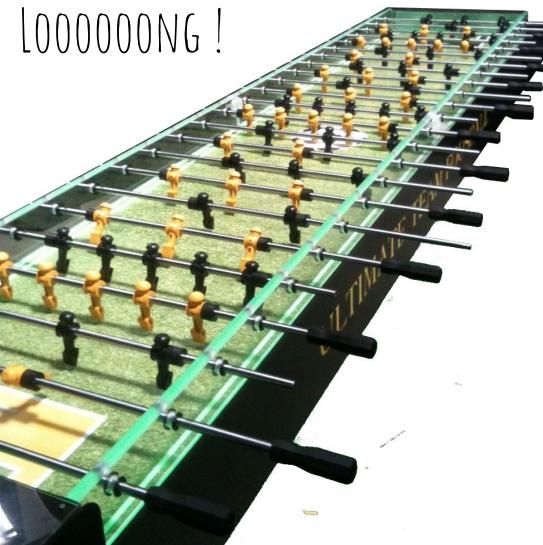


96emptysections	96workingsections	65535sects	appendddata	appenddr	appenddsecttbl	apphdrW7	appsectableW7	bigalign	bigib	bigsec	bigSoRD	bottomsecttbl	compiled	ctxt.dll	cbtx-ld	d_nonnull.dll
dllnonnull-ld	d_resource.dll	d_resource-ld	d_tiny.dll	d_tiny-ld	debug	delaycorrupt	delayfake	delayimports	dll.dll	dllbound.dll	dllbound-ld	dllbound-redirld	dllbound-redirldXP	dll-dynld	dll-dynamicld	dll-dynunicld
dllemptryxp.dll	dllemptryxp-ld	dllextep.dll	dllextep-ld	dlfw.dll	dlfw-ld	dlfwloop.dll	dlfwloop-ld	dll-ld	dlnegep.dll	dlnoexp.dll	dlnoexp-dynld	dlnomain.dll	dlnomain2.dll	dlnoain2-dynld	dlnoain2-dynld	dlnoaind
dllnoreloc.dll	dllnoreloc-ld	dllnullep.dll	dllnullep-dynld	dllnullep-ld	dllord.dll	dllord-ld	dllwebdavd	dllweirdexp.dll	dosZMXP	dotnet20	driver.sys	duphead	dupsec	exceptions	exe2pe	
exportobf	exports_doc	exportsdata	fakereg	fakeregslib.dll	fakerelocs	foldedhdr	foldedhdrW7	footer	gui	hdrcode	hiddenappdata1	hiddenappdata2	ibkernel	ibkmanual	ibknoreloc64	ibnullXP
ibreloc	ibrelocW7	impbyord	imports	imports_api_msW7	imports_baderm	imports_jatindesc	imports_mixed	imports_multidesc	imports_noext	imports_noint	imports_nothunk	imports_relocW7	imports_tinyW7	imports_tinyXP	imports_virtdesc	imports_vterm
importshint	manifest	manifest_broken	manifest_bsod	manyimportsW7	maxsec_lowaligW7	maxsecW7	maxsecXP	maxvals	mini	multiss	multiss_conn	multiss_drv.sys	multiss_gui	mz	namedresource	no_dd
no_dd64	no0code	normal	normal64	nosection	nosectionX	nothing.dll	nothing-ld	nullEP	nullISOH-X	ownexports	ownexports	pdf	pdf_zip_pe	quine	relocrypt	relocryptXP

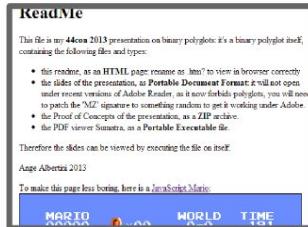
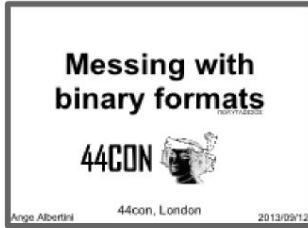
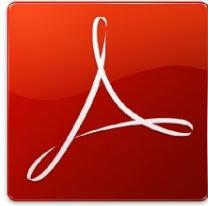
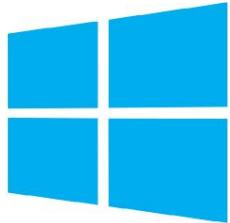
MY COLLECTION OF HAND-MADE EXECUTABLES AND "DOCUMENTATION" (COMPLETELY FREE).

# VICTOR FRANKENSTEIN

INITIALLY, I STARTED WITH SIMPLER STUFF  
BUT... IT EXCALATED QUICKLY :)



DUAL HEADED COW



A PRESENTATION SLIDE DECK VIEWING ITSELF  
(PDF VIEWER AND PDF DOCUMENT)

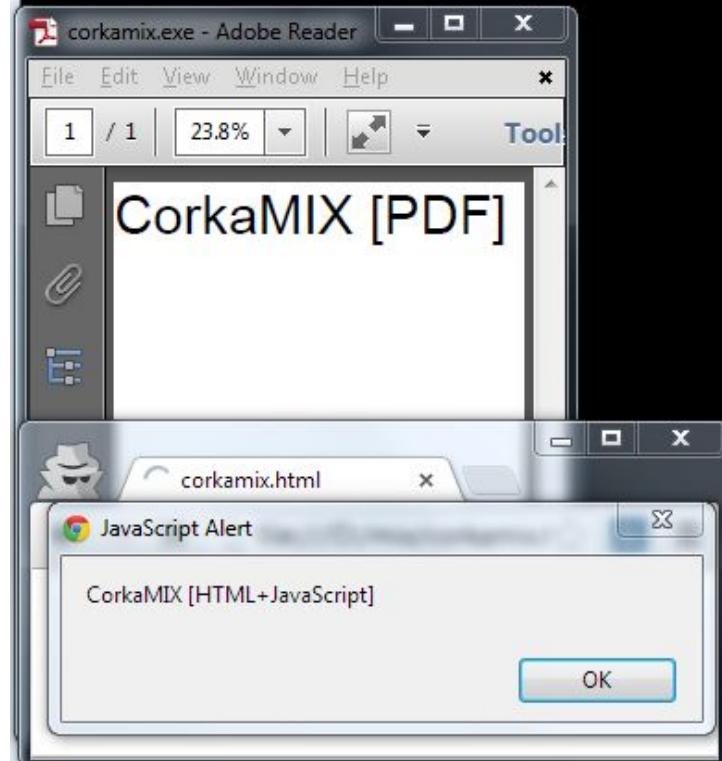
# HTML JAVASCRIPT JAVA

## WINDOWS EXECUTABLE

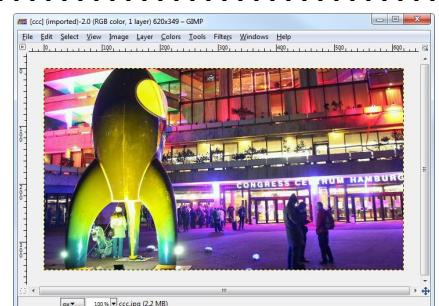
### PDF

## 2 STANDARD INFECTION CHAINS IN A SINGLE FILE

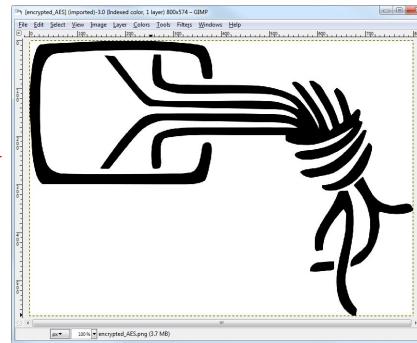
```
>corkamix.exe  
CorkaMIX [PE]  
>java -jar corkamix.exe  
CorkaMIX [Java CLASS in JAR]  
  
>cmp -b corkamix.exe corkamix_1b.exe  
cmp: EOF on corkamix.exe  
  
>python corkamix_1b.exe  
CorkaMIX [python]  
  
>copy corkamix.exe corkamix.html  
1 file(s) copied.
```



JPG



$AES_{K_1}$



PNG

JAR  
(ZIP + CLASS)

```
>java -jar ccc.jpg  
Hello World! [Java]
```

$AES_{K_2}$

3DES

PDF



FLV

MIXING BINARY AND CRYPTOGRAPHY

## unicode //

```
\u002f\u002f <html>
\u002f\u002f   <body>
\u002f\u002f     <script>
\u002f\u002f       alert('Hello World! [Javascript]');
\u002f\u002f     </script>
\u002f\u002f   </body>
\u002f\u002f </html>
```

```
public class HW
{
    public static void main(String[] args)
    {
        System.out.println("Hello World! [Java]");
    }
}
```

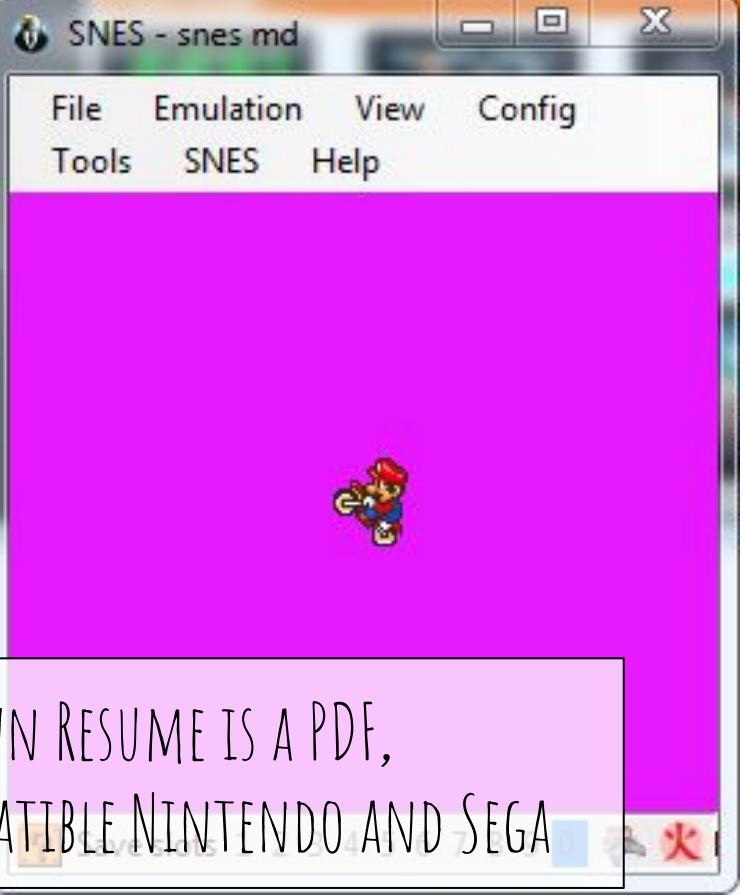
3C	68	74	6D	6C	3E	3C	62	6F	64	79	3E	3C	73	63	72
69	70	74	3E	61	6C	65	72	74	28	27	48	65	6C	6C	6F
20	57	6F	72	6C	64	21	20	5B	4A	61	76	61	73	63	72
69	70	74	5D	27	29	3B	3C	2F	73	63	72	69	70	74	3E
3C	2F	62	6F	64	79	3E	3C	2F	68	74	6D	6C	3E	50	4B
03	04	0A	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	09	00	00	00	4D	45	54	41
2D	49	4E	46	2F	50	4B	03	04	0A	00	00	00	00	00	00
00	00	00	00	00	00	00	1F	00	00	00	1F	00	00	00	14
00	00	00	4D	45	54	41	2D	49	4E	46	2F	4D	41	4E	49
46	45	53	54	2E	4D	46	43	72	65	61	74	65	64	2D	42
79	3A	20	31	0D	0A	4D	61	69	6E	2D	43	6C	61	73	73
3A	20	48	57	0D	0A	50	4B	03	04	0A	00	00	00	00	00
00	00	00	00	00	00	00	00	1C	01	00	00	1C	01	00	00
00	00	00	00	CA	FE	BA	BE	00	03	00	2D	00	16	07	00
02	01	00	02	48	57	07	00	04	01	00	10	6A	61	76	61
2F	6C	61	6E	67	2F	4F	62	6A	65	63	74	01	00	04	6D
61	69	6E	01	00	16	28	5B	4C	6A	61	76	61	2F	6C	61
6E	67	2F	53	74	72	69	6E	67	3B	29	56	01	00	04	43
6F	64	65	09	00	09	00	0B	07	00	0A	01	00	10	6A	61
76	61	2F	6C	61	6E	67	2F	53	79	73	74	65	6D	0C	00
0C	00	0D	01	00	03	6F	75	74	01	00	15	4C	6A	61	76
61	2F	69	6F	2F	50	72	69	6E	74	53	74	72	65	61	6D
3B	08	00	0F	01	00	13	48	65	6C	6C	6F	20	57	6F	72
6C	64	20	21	5B	4A	61	76	61	5D	0A	00	11	00	13	07

```
<html><body><script>alert('Hello  
.World!..[Javascr  
ipt]');</script>  
</body></html>PK  
.....  
.....META  
-INF/PK.....  
.....  
...META-INF/MANI  
FEST.MFCreated-B  
y..1..Main-Class  
:.HW..PK.....  
.....  
.....-....  
....HW.....java  
/lang/Object...m  
ain...([Ljava/la  
ng/String;)V...C  
ode.....ja  
va/lang/System..  
.....out...Ljav  
a/io/PrintStream  
;.....Hello.Wor  
ld.! [Java].....
```

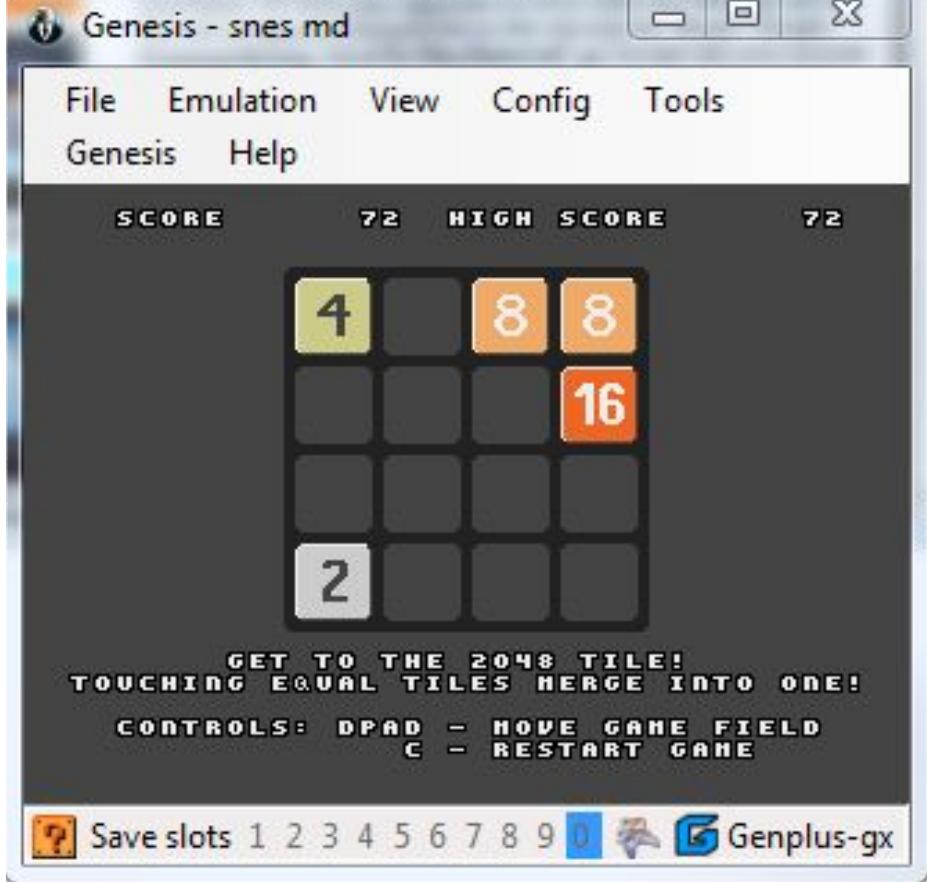
A JAVA & JAVASCRIPT POLYGLOT - AT BINARY LEVEL

=> JAVA = JAVASCRIPT

YES, YOUR MANAGEMENT WAS RIGHT ALL ALONG ;)



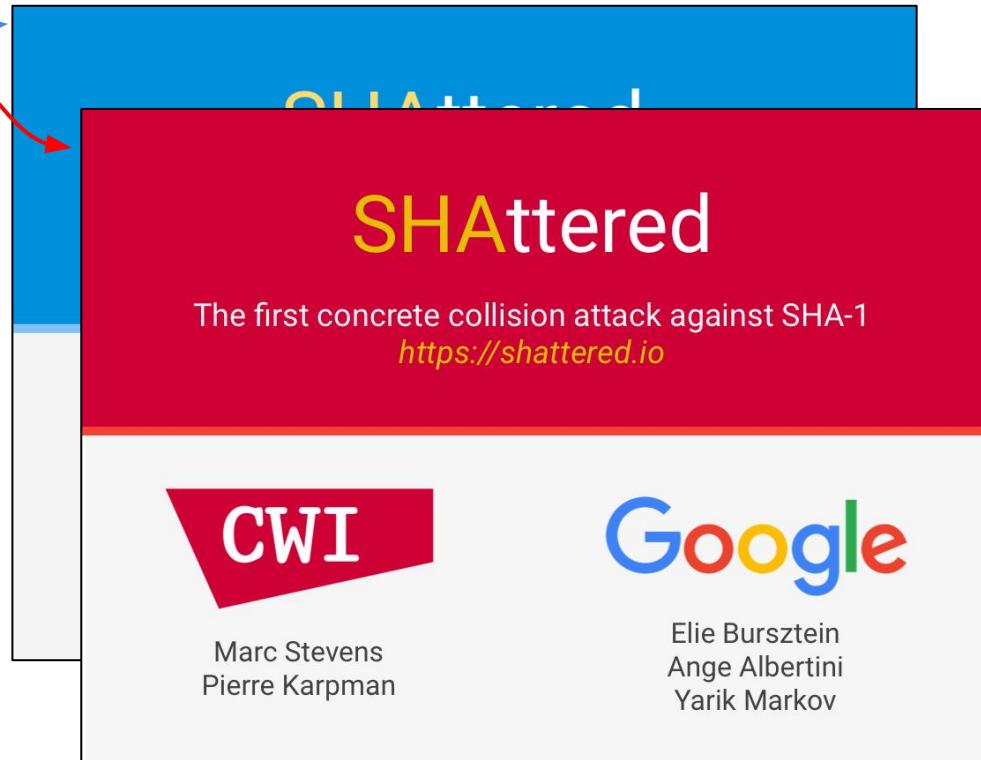
NINTENDO



Sega®

I CRAFTED THE BINARY STRUCTURE OF THE FILES  
WITH THE FIRST SHA1 COLLISION.

THESE FILES VIOLATE THE SPECIFICATIONS!  
AND YET THEY WORK EVERYWHERE.



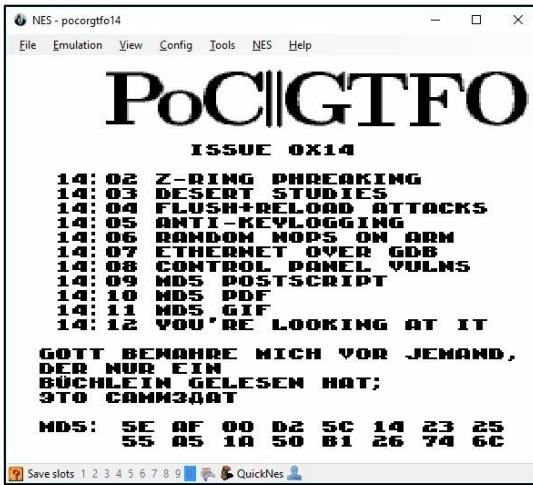
The JPEG File Interchange Format is entirely compatible with the standard JPEG interchange format; the only additional requirement is the mandatory presence of the APP0 marker right after the SOI marker. Note that JPEG interchange format requires (as does JFIF) that all table specifications used in the encoding process be coded in the bitstream prior to their use.

<https://www.w3.org/Graphics/JPEG/jfif3.pdf>

# THESE FILES DISPLAY THEIR OWN MD5!

(NOT BY ME)

NINTENDO ROM



PDF

```
$ pdftotext -q md5text.pdf -
66DA5E07C0FD4C921679A65931FF8393
```

```
$ md5sum md5text.pdf
66da5e07c0fd4c921679a65931ff8393 md5text.pdf
```

GIF

The image shows a digital display with two rows of red seven-segment digits on a black background. The top row displays the sequence: F5 CA 4F 93 5d 44 b8 5C. The bottom row displays: 43 1A 8b F7 88 C0 EAC8.

	Offset	0 1 2 3 4 5 6 7	8 9 A B C D E F	Ascii
	00000000	47 49 46 38 39 61 2F 2A	0A 00 00 FF 00 2C 00 00	GIF89a/*..... <-Format data
	00000010	00 00 2F 2A 0A 00 00 02	00 3B 2A 2F 3D 31 3B 61	.../*....;*/=1;a <-Format data - For...
	00000020	6C 65 72 74 28 22 48 65	6C 6C 6F 20 57 6F 72 6C	lert("Hello.Worl <-Foreign data
	00000030	64 5C 6E 28 66 72 6F 6D	20 61 20 47 49 46 20 66	d\n(from.a.GIF.f
	00000040	69 6C 65 29 22 29 3B		ile");

JAVASCRIPT

IMAGE

The screenshot shows a debugger interface with a hex dump of a file. The ASCII column reveals a multi-part payload: a GIF header ('GIF89a/\*.....'), a JavaScript exploit ('.../\*....;\*/=1;a'), and a foreign data block ('lert("Hello.Worl'). Below the debugger, two browser windows are displayed. The left window shows the source code of 'gifjs.html' which includes an image tag pointing to 'gifjs.gif' and a script tag also pointing to 'gifjs.gif'. The right window shows a browser alert box with the message 'Hello World (from a GIF file)'. A red box highlights the exploit code in the hex dump, and another red box highlights the image reference in the source code. Red arrows point from these highlighted areas to their respective counterparts in the browser windows.

A JAVASCRIPT // GIF POLYGLOT (USEFUL TO EMBED PAYLOAD - ALSO WORKS WITH JPG OR BMP)

ISN'T THIS ALL  
"USELESS?"

I REMOVE POTENTIAL TRAPS BY RESEARCHING,  
AND I TRAIN MYSELF WITH EXTREME FILES.

ALSO, SOME OF THESE WERE USED IN THE WILD TO ATTACK PEOPLE,  
AND I CAN REPRODUCE KEY FEATURES INTO SHAREABLE FILES.

# MY PERSPECTIVE:

1 - WHAT IS A FILE?

A SEQUENCE OF BYTE(S)

ANY PARSER CAN GIVE AN INCOMPLETE PERSPECTIVE.

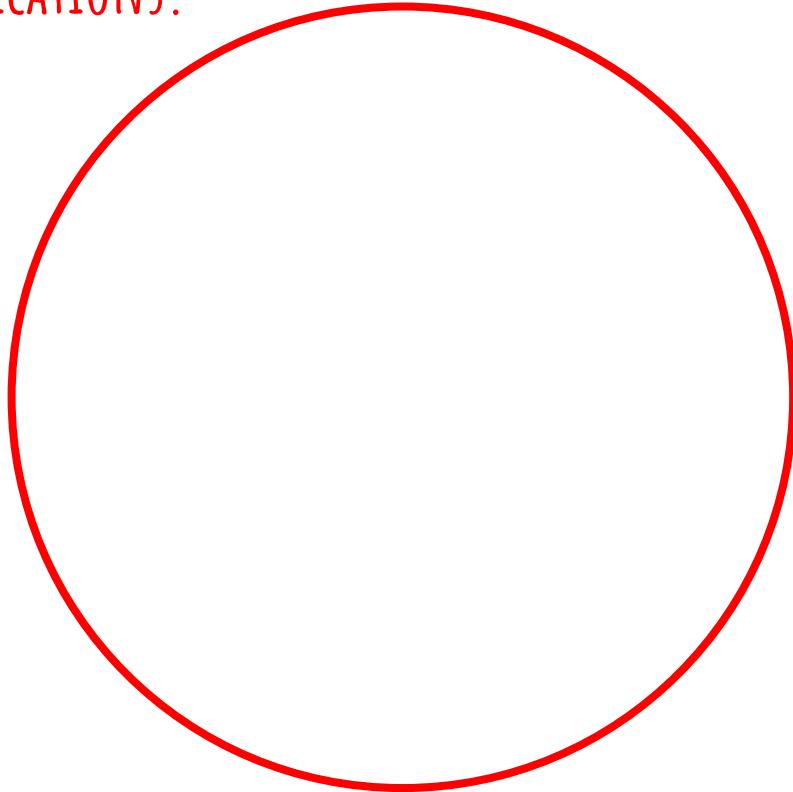
I OPEN MOST FILES FIRST WITH A HEX EDITOR (OUT OF CURIOSITY, AT LEAST)

YES, I'M A HEX-ADDICT :)

# 2- WHAT IS A VALID FILE?

A FILE LOADED SUCCESSFULLY  
BY A PARSER/LOADER/PROCESSOR.  
A FILE IN ITSELF IS NOTHING.

SPECIFICATIONS.



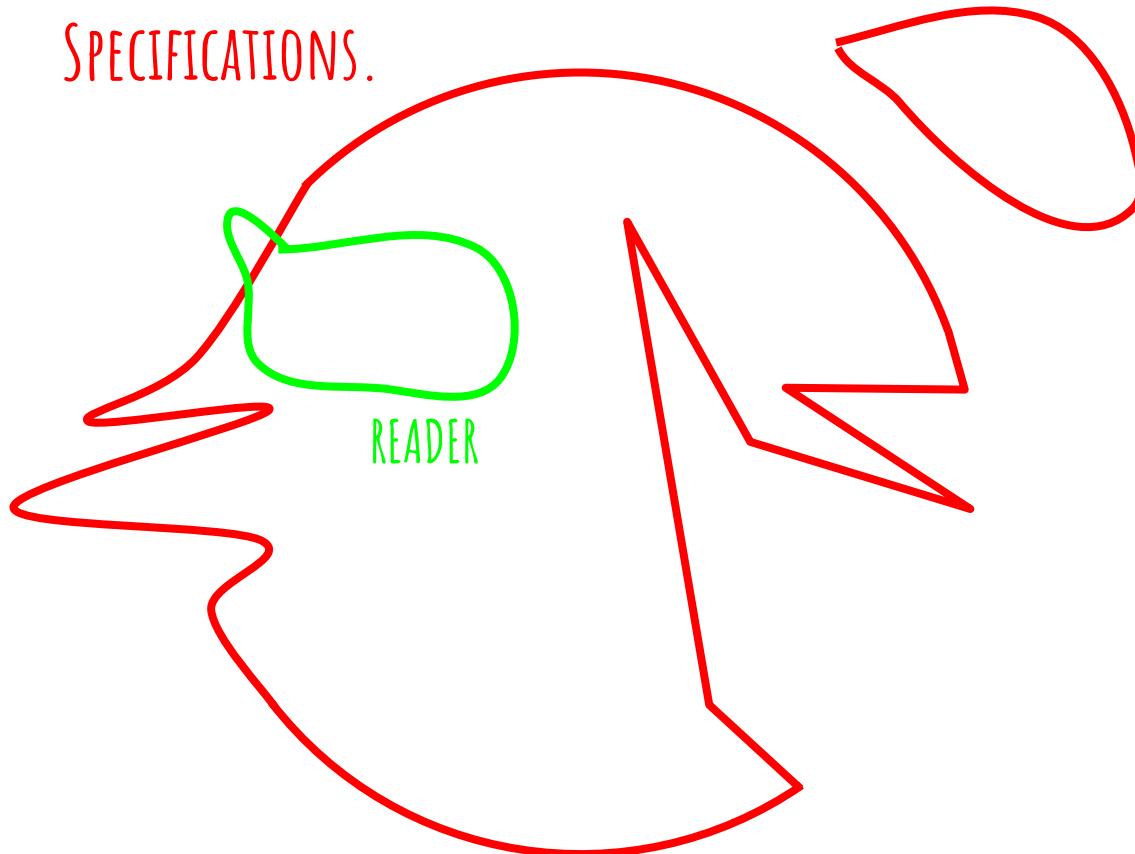
WHAT WE WISHED...

SPECIFICATIONS.



IN REALITY, THEY ARE MORE COMPLEX,  
OFTEN FOR NO PARTICULAR REASON (SEE DESIGN BY COMMITTEE)

SPECIFICATIONS.



NOW COMES A SOFTWARE THAT TAKES THESE FILES AS INPUT.  
THIS SOFTWARE DEFINES VALIDITY (THE PARSER/LOADER), NOT THE SPECIFICATIONS.

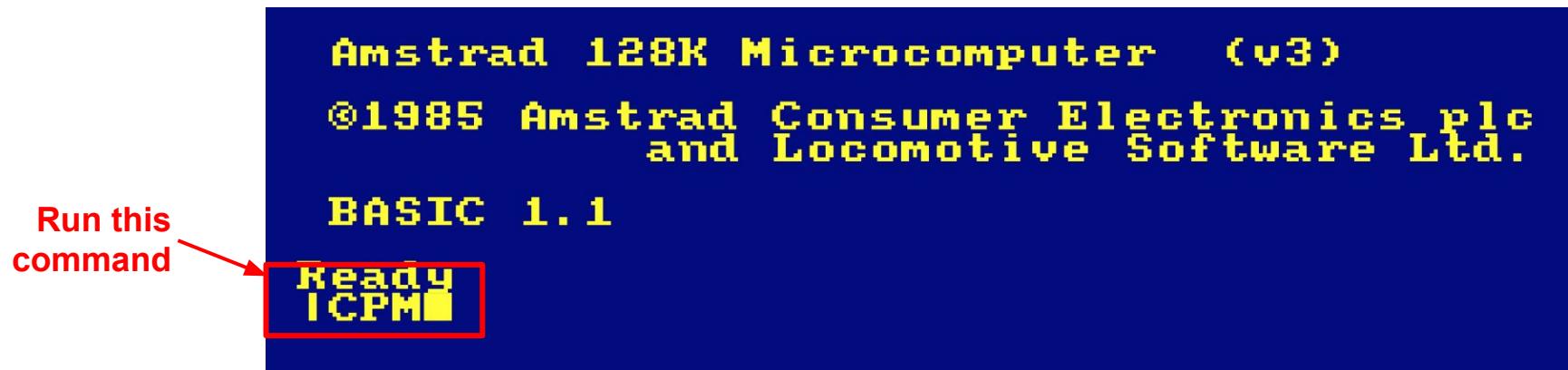
# SPECIFICATIONS ARE IRRELEVANT!

AS LONG AS THE FILE 'WORKS AS INTENDED'.

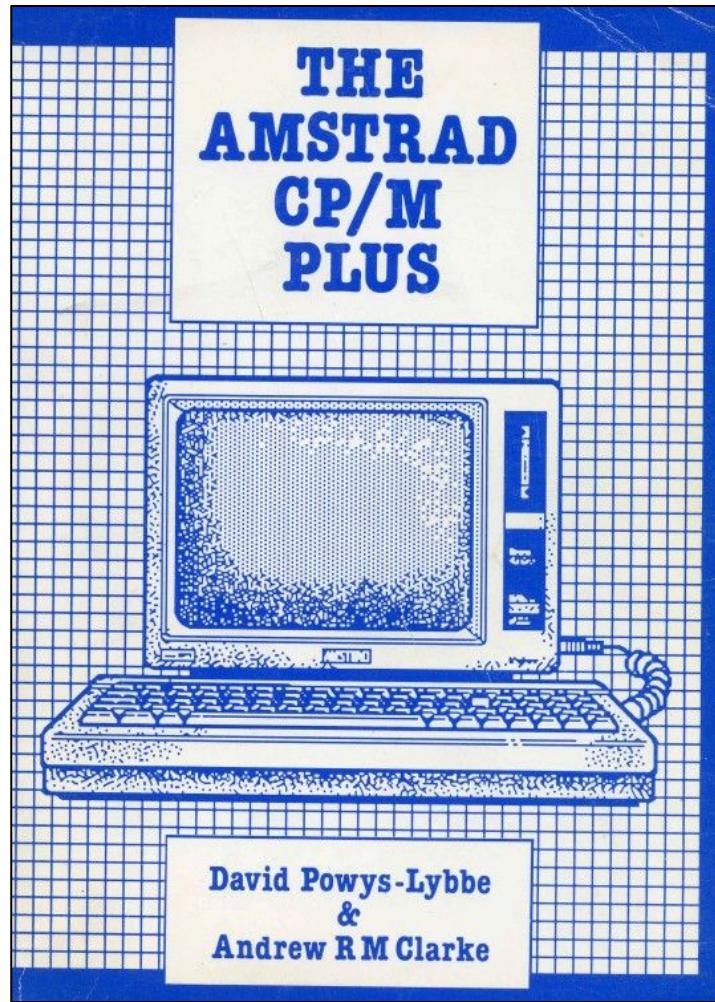
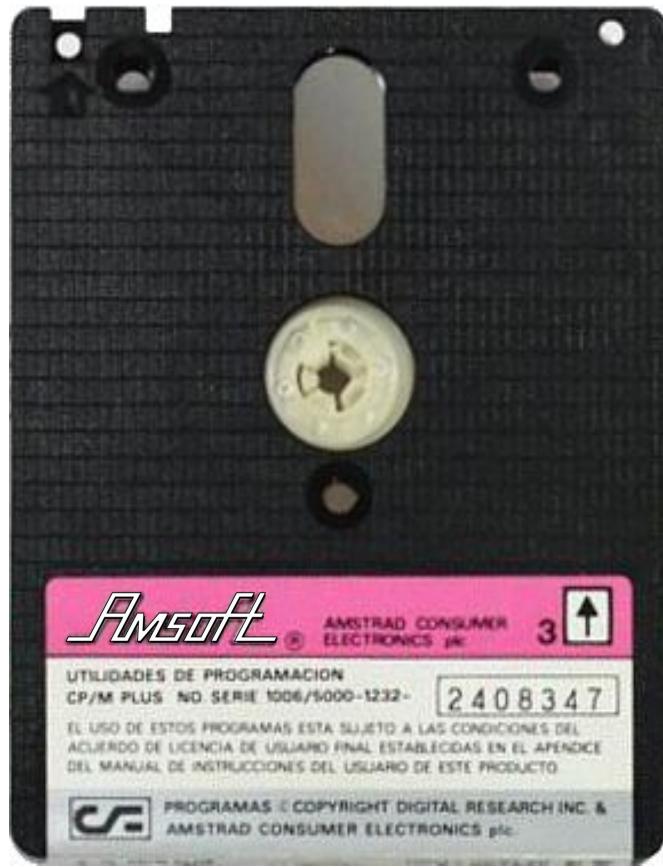
ON THIS COMPUTER...



WE'LL LAUNCH...



...THIS OS.



# LET'S CREATE... AN EMPTY FILE!

CP/M 2.2 - Amstrad Consumer Electronics plc

A>ED GO.COM

NEW FILE  
: \*e

A>STAT GO.COM

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

create empty file

size=0

# IS IT VALID?

YES: TRANSIENT COMMANDS ARE COPIED BLINDLY  
AND EXECUTION STARTED AT OFFSET ZERO.

# DOES IT DO ANYTHING?

TRANSIENT MEMORY AREA IS NOT  
CLEARED BETWEEN EXECUTIONS,  
SO THE PREVIOUS COMMAND IS RE-EXECUTED.

A>GO GO.COM

Recs Bytes Ext Acc  
0 0k 1 R/W A:GO.COM

Bytes Remaining On A: 5k

A>■

The terminal window has a red border. The command 'A>GO GO.COM' is highlighted with a red box and a red arrow points from the text 'works as intended' to it.

UNDER A COMMERCIAL OS FROM 1985,  
THE EMPTY FILE IS VALID, USEFUL AND RELIABLE.  
IT WAS EVEN SOLD AS A COMMERCIAL PROGRAM FOR £5.

# TAKEAWAY

- IT'S OLD-SCHOOL, OBSOLETE...
- AND YET THIS EXAMPLE IS PURE SIMPLICITY...  
TO PROVE THAT THE SOFTWARE DEFINES THE RULES!
- THE SPECIFICATIONS ARE VOLATILE....  
THE SOFTWARE IS THE GROUND TRUTH.

# TEXT FILES

WHAT COULD GO WRONG?

JUST A BAD ENCODING MAYBE?

# THIS IS A ... MALICIOUS FLASH FILE !!

NO, NOT BASE64 - IT'S DIRECTLY EXECUTABLE AS IS!

BUT, AREN'T FLASH FILES...  
PURE BINARY!?

.F .W .S 07 8C 00 00 00 78 00 05 5F 00 00 0F A0  
00 00 0C 01 00 3F 03 6D 00 00 00 88 2C 00 04 00  
.m .e .s .s .a .g .e 00 .c .r .e .a .t .e .T .e  
.x .t .F .i .e .l .d 00 .t .e .x .t 00 .H .e .l  
.l .o . .W .o .r .l .d .! 00 96 2A 00 07 32 00  
00 00 07 64 00 00 00 06 00 00 00 00 00 00 00 00 00  
06 00 00 00 00 00 00 00 00 07 01 00 00 00 08 00  
07 06 00 00 00 08 01 3D 17 96 02 00 08 00 1C 96  
04 00 08 02 08 03 4F 00 40 00 00 00

CWSMIKI0hCD0Up0IZUnnnnnnnnnnnnnnnUU5nnnnnn3Snn7iudIbEAAt333swW0ssG03  
sDDtDDDt0333333Gt333swv3wwwFPOHtoHHvwHHFhH3D0Up0IZUnnnnnnnnnnnnnnnnU  
U5nnnnnn3Snn7YNqdIbeUUUFv13333333333333s03sDTvqefXAooooD0CiudIbEAAt33  
swwEpt0GDG0GtDDDtwwGGGGGsGDt33333ww03333GfBDTHHHUhHHHeRjHHHHUccUSsg  
SkKoE5D0Up0IZUnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7YNqdIbe1333333333sUUe133  
333wf03sDTVqefXA8oT50CiudIbEAAtwEpDDG033sDGtwGDtwDwttDDDGtwG33wwGt0w33  
33sG03sDDdFPhHHhbWqHxHjHZNAqFzAHZYqqEHeYAH1qzfJzYyHqQdzEzHVMvnAEYzEVHMH  
bBRrHyVQfdQf1qzfHLTrHAqzfH1YqEqEmIVHaznQHzIIHDRVVEbYqItAzNyH7D0Up0IZUnn  
nnnnnnnnnnnnnnnUU5nnnnnn3Snn7CiudIbEAAt33swwEDt0GGDDDGPtDtwwG0GGptDtwwG0G  
DtDDDGDDGDDtDD3333s03GdFPXHLHAZZOXHrhwXHLhAwXHLhgBHHhHDEHXsSHoHwXHLXAw  
XHLxMZOXHWHwtHtHHHLDUGHxvwDHDxLdgbHHhHDEHXkKSHuHwXHLTMZOXHeHwtHt  
HHHLDUGHxvwTHDxLxDxmwTHLLdxLXAwXHLTMw1HtxHHHDxL1Cvm7D0Up0IZUnnnnnnnnn  
nnnnnnnnnUU5nnnnnn3Snn7CiudIbEAAtw3sG33ww0sDtDt0333Gdw0w3333www033GdFP  
DHTLxXtnohHTxgotHdXHHhxXTlwf7D0Up0IZUnnnnnnnnnnnnnnUU5nnnnnn3Snn7C  
iudIbEAAtwWtD333wwG03www0GDGpt03wDDDGDD3333s033GdFPhHHkoDHDHTLKwhHhzoD  
HDHT10LHHhHxeHXwqHZHoXHTHNo4D0Up0IZUnnnnnnnnnnnnnnUU5nnnnnn3Snn7Ciud  
IbEAAt33wwE03GDDGwGGDDGdwGtwDtwDDGGDDtGDwGw0GDDw0w3333www033GdFPLRDxt  
hHHHLqeeorHthHHHXDhtxHHHLravHQxQHHHOonHDHyMIuiCyIYEHWSSgHmHKcskHoXHLwhH  
HvoXHLhAoHthHHHLXaoXHLxUvH1D0Up0IZUnnnnnnnnnnnnnnnnUU5nnnnnn3SnnwwNq  
dIbe1333333333333333WF03sTeqefXA888oooooooooooooooooooooooooooooooooooo  
oo  
oo  
oo888888880Nj0h

< = "HELLO WORLD" IN FLASH:  
IT'S A LOT OF NON-ASCII!

FLASH CAN BE COMPRESSED WITH ZIP'S DEFLATE.  
WHICH CAN USE THE HUFFMAN ALGORITHM,  
IN WHICH CASE YOU SUPPLY A CODE DICTIONARY.  
YOU CAN CRAFT SUCH A DICTIONARY TO 'EXPAND' YOUR DATA,  
BUT IN RETURN IT'S ASCII-ONLY.



```
>>> zlib.decompress('xf3H\xcd\xc9\xc9W\x08\xcf/\xcaIQ\xe4\x02\x00 \x91\x04H', -8)
'Hello World!\n'
>>> zlib.decompress("D0Up0IZUnnnnnnnnnnnnnnnnnUU5nnnnnn3SUUnUUUwCiudIbEAtwwEt3
33wwG0swGpDDGpDDwDDDGTd3333s033333GdFPk\wwwOaG0owgQ4", -8)
'Hello World!\n'
```

<https://github.com/molnarg/ascii-zip>



<https://miki.it/blog/2014/7/8/abusing-jsonp-with-rosetta-flash/>

VALID FLASH FILES ENTIRELY IN ASCII:

- > BYPASSED ALL FILTERS
- > ABUSED MOST WEBSITES

## FLASH FILE TYPES

### TYPE FILE STRUCTURE

FLAT FWS <Version:1> <FileLength:4> <uncompressed data...>

=> ZLIB CWS <Version:1> <FileLength:4> <zlib data>

<CMF:1> <FLG:1> <dict>\* <deflate> <adler32:4>

LZMA ZWS <Version:1> <FileLength:4> <lzma data>

\*UNCOMPRESSED

Version AND FileLength ARE NOT CHECKED.

## ADLER32 CHECKSUM

FOR EACH BYTE OF THE UNCOMPRESSED STREAM:

... . . . . XX . . . . .

S1 += XX

S2 += S1

FINAL RESULT:

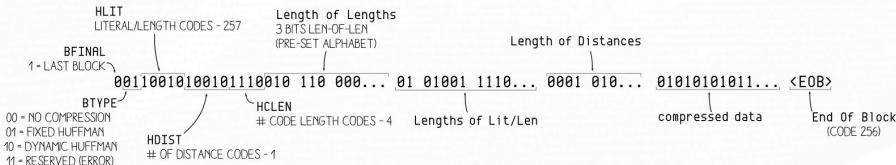
ADLER32 = S2 << 16 | S1

WITH BOTH S1 & S2 MODULO 65521 (LARGEST PRIME <2^16)

## ZLIB STREAM START



## STRUCTURE OF A DEFLATE BOCK



FLASH ALLOWS APPENDED DATA AFTER END MARKER:

1. ADJUST S1:

- APPEND 0xFE TO *uncompressed* DATA  
UNTIL S1 IS VALID ([0-9a-zA-Z./]\*)  
(0xFF DOESN'T WORK WELL FOR HUFFMAN MANIPULATION)

2. ADJUST S2:

- APPEND 0x00  
UNTIL S2 IS VALID  
(APPENDING 0x00 DOESN'T AFFECT S1)

SPECIFICATIONS ARE BLURRY.

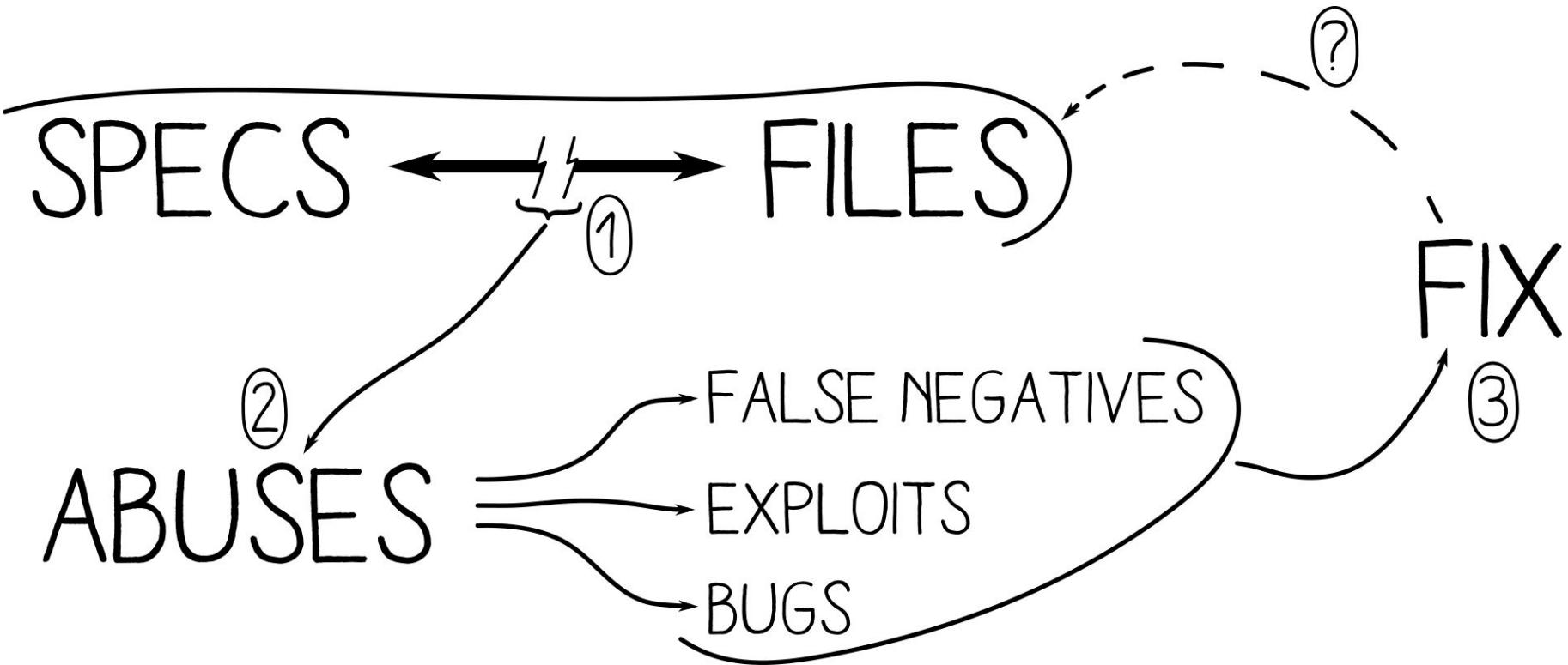
THERE'S ALWAYS A CORNER CASE THAT IS NOT CLARIFIED.

# SPECIFICATIONS ARE IMPERFECT.

I KNOW NO PERFECT SPECIFICATIONS.

EXCEPT FOR THE EMPTY FILE! :)

I COULD ONLY MAKE SPECIFICATIONS BETTER  
BY FINDING PROBLEMS BEFORE THEY WERE  
FINALIZED AND SET IN STONE.

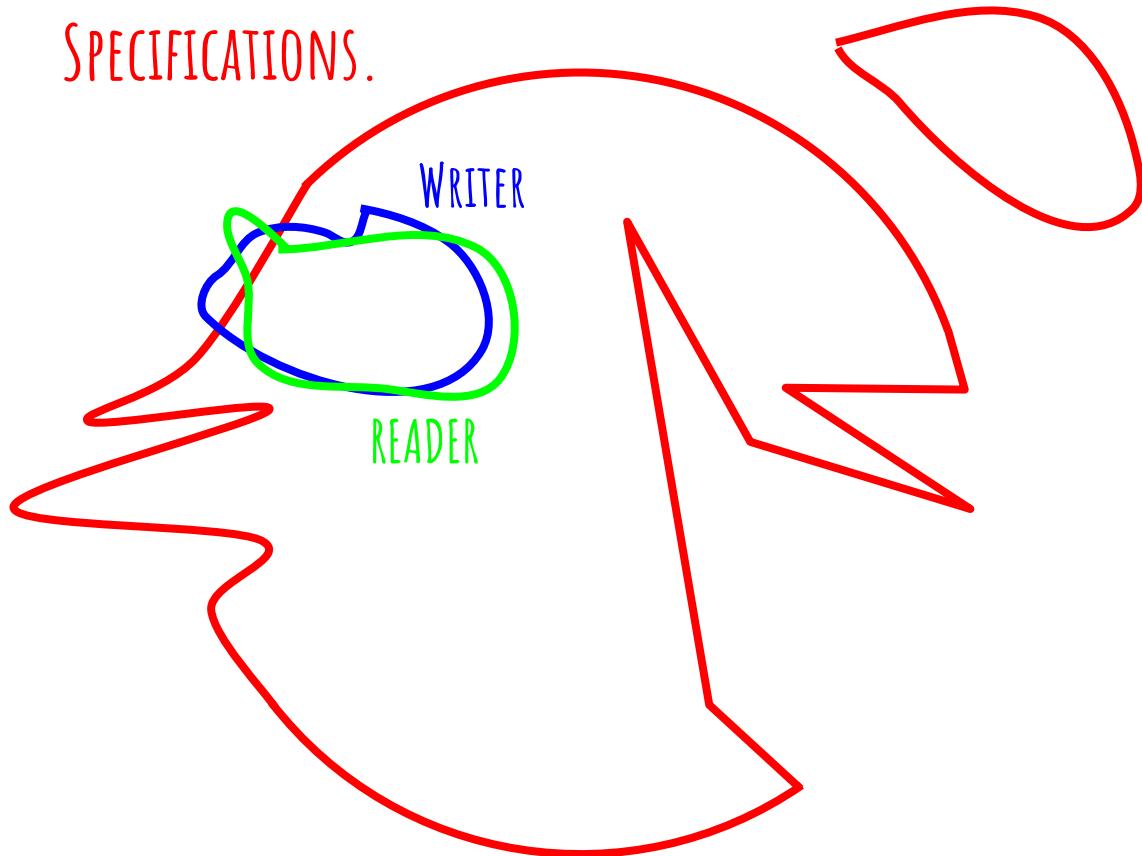


# THE REAL PROBLEM

- UNLIKE LAWS, SPECIFICATIONS ARE NOT ENFORCED.
- NOT UPDATED EITHER. SET IN STONE.
- THE PROBLEM REMAINS THE SAME - AND GROWS WITH TIME.
- SURVIVAL OF THE FITTEST SOFTWARE:  
THE WINNER DECIDES THE RULES.



SPECIFICATIONS.



NOW COMES A SOFTWARE THAT CREATES THESE FILES.

THEIR PLANNED FEATURES AND ACTUAL ABILITIES OF READERS AND WRITERS MAY NOT OVERLAP.

LARGE FORMAT SCANNERS:  
INFINITE "HEIGHT" SCANS  
-> IMAGE HEIGHT FIXED TO 65535!

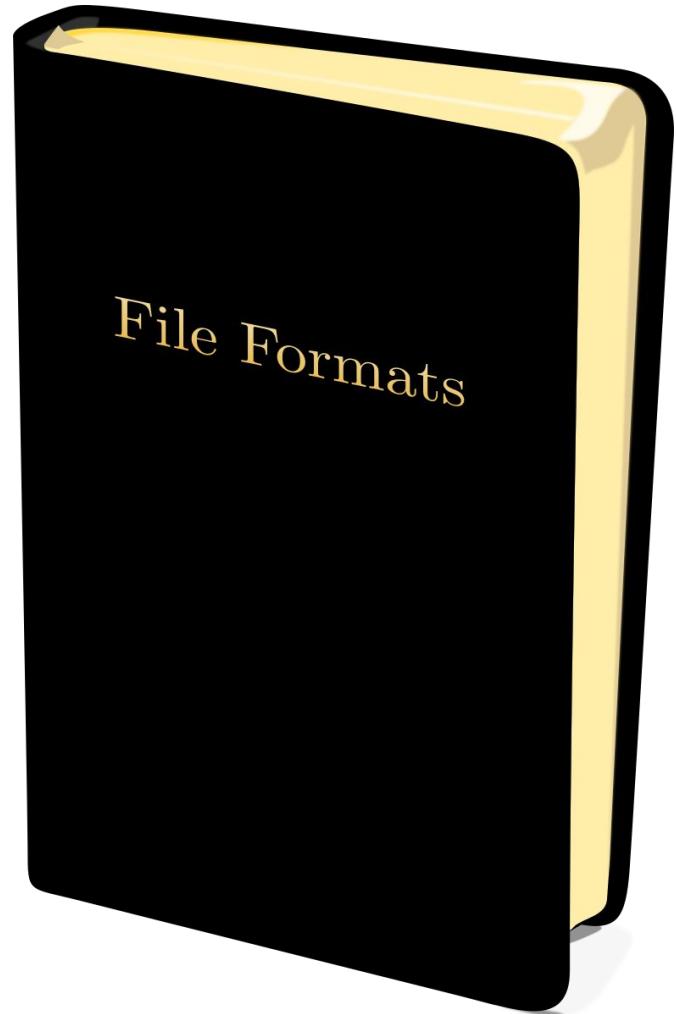
TOLERATED BY LIBJPEG,  
SO VALID EVERYWHERE!



DETECTED BY ANTI-VIRUS, BECAUSE IT WAS USED TO EXPLOIT MS04-028.

Thou shalt use  
Adobe PDF writer..

SPECIFICATIONS ARE NOT UPDATED:  
THEY BECOME OUTDATED AND IRRELEVANT.



SOME SPECIFICATIONS ARE EVEN WORSE:  
THEY'RE NOTHING BUT...  
A "GENTLE INTRODUCTION":  
ALMOST USELESS FROM THE START.



Data is  
in the .data  
section



# DIVERGENCES

AS SPECIFICATIONS ARE NOT PERFECT,  
THEY ARE INTERPRETED BY DIFFERENT PEOPLE IN DIFFERENT WAYS:  
SO ONE FILE MAY WORK ON ONE READER, NOT ON THE OTHER ONE.

# A NORMAL PDF

## HEADER

%PDF-1.1

SIGNATURE & VERSION INFORMATION

## 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
```

## BODY

DICTIONARY  
<< [ID VALUE]\* >>

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

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

IDENTIFIER (WITH /)

ARRAY

STREAM PARAMETERS:  
LENGTH, COMPRESSION....

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

```
%PDF-1.
```

```
1 0 obj
<< /Kids [<<
    /Parent 1 0 R
    /Resources <<>>
    /Contents 2 0 R
  >>]
>>

2 0 obj
<<>>
stream
BT
/F1 110 Tf
10 400 Td
(Adobe Reader) Tj
ET
endstream
endobj

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

```
%PDF
```

```
1 0 obj
<< /Pages
  << /Kids [
    << /Contents 2 0 R >>
  ] >>
>>

2 0 obj
<<>>
stream
95 Tf
20 400 Td
(Chrome) Tj
endstream

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

WORKING PDFS

%PDF-1. truncated signature

```
1 0 obj
<< /Kids [ direct /Kids
  /Parent 1 0 R No /Type
  /Resources <<>> No /Font
  /Contents 2 0 R
  No /Type
  No /Count ]>>
```

```
2 0 obj
<< No /Length
stream
```

BT

```
/F1 110 Tf
10 400 Td
(Adobe Reader) Tj
```

ET

endstream

endobj

No XREF

trailer <<

```
  Direct /Root
  No /Size
  /Pages 1 0 R
  No %%EOF
>> No /Type
No startxref
```

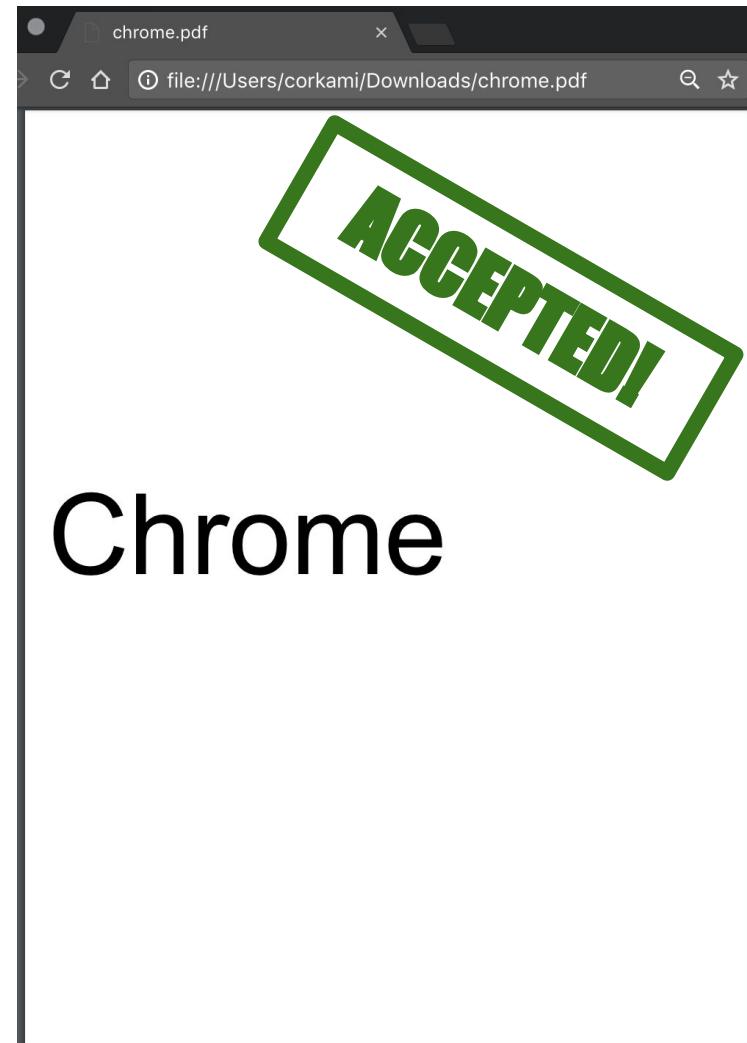
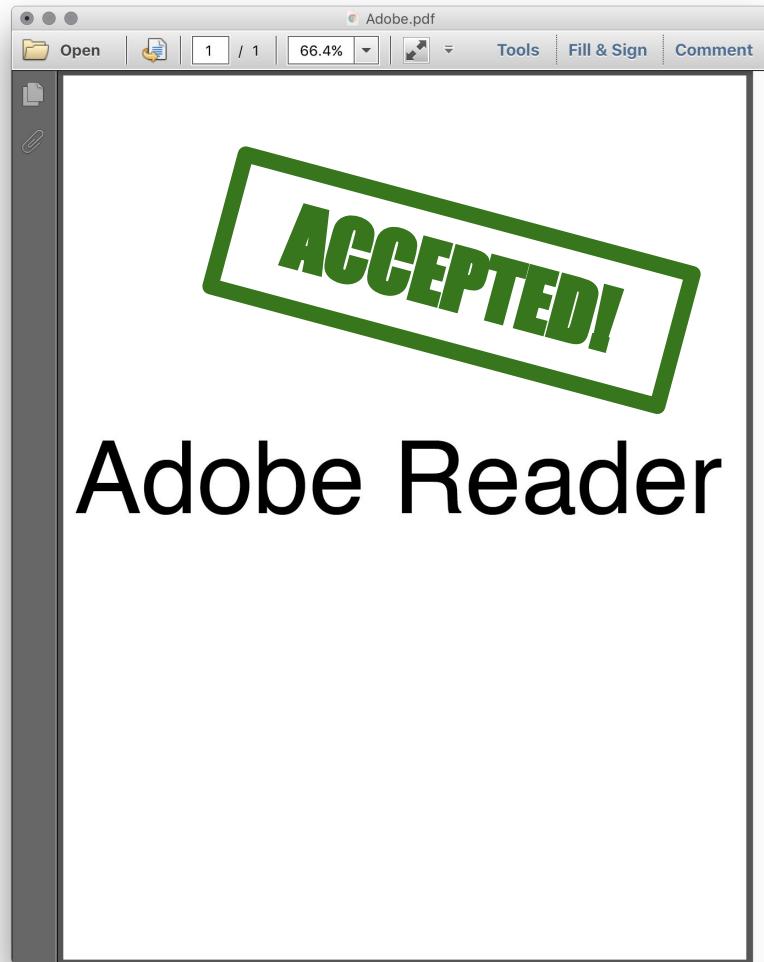
%PDF very truncated signature

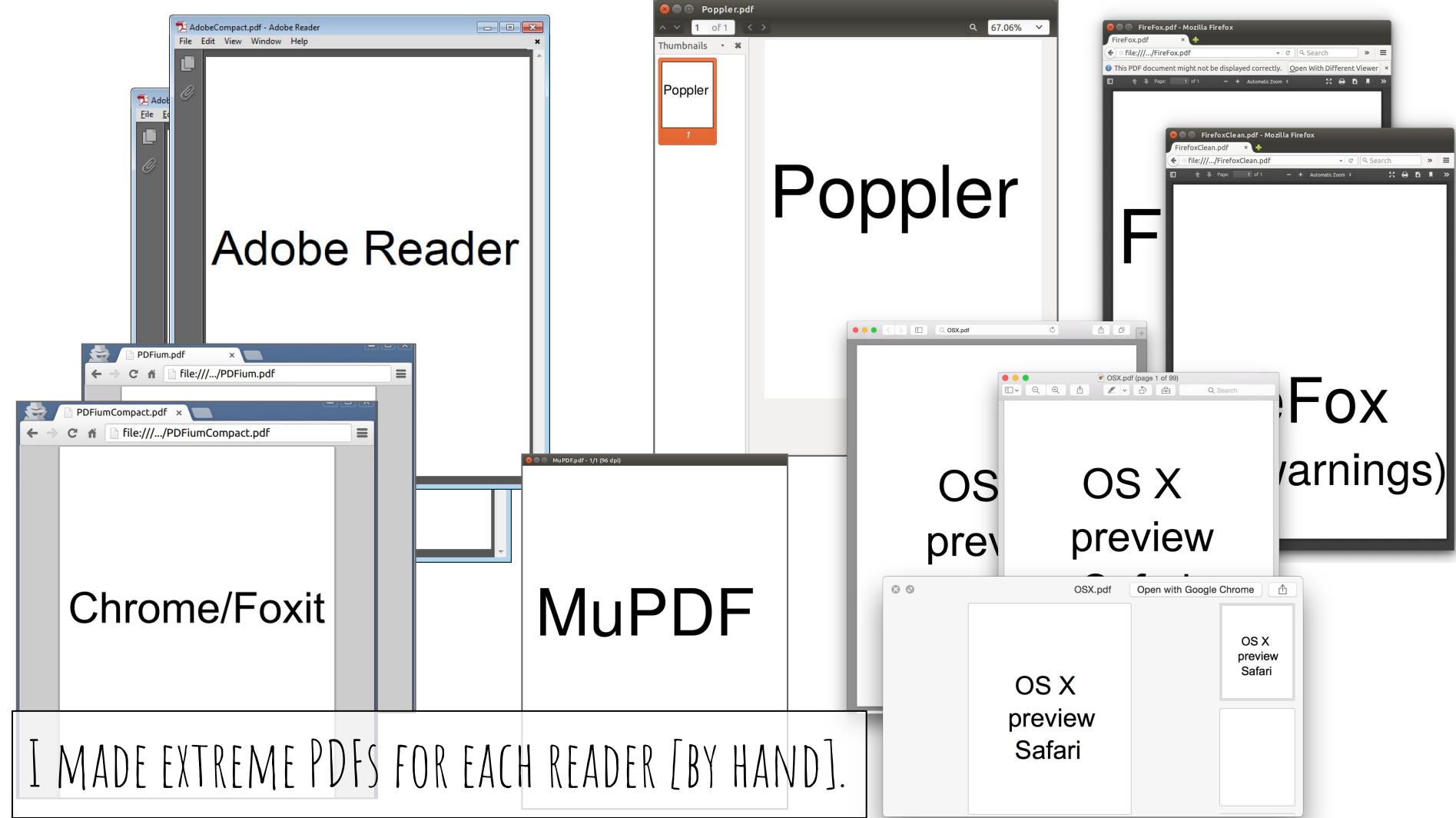
```
1 0 obj
<< /Pages [ direct /Kids
  /Parent 1 0 R No /Type
  /Resources <<>> No /Font
  /Contents 2 0 R
  No /Type
  No /Count ]>>
>> No endobj
```

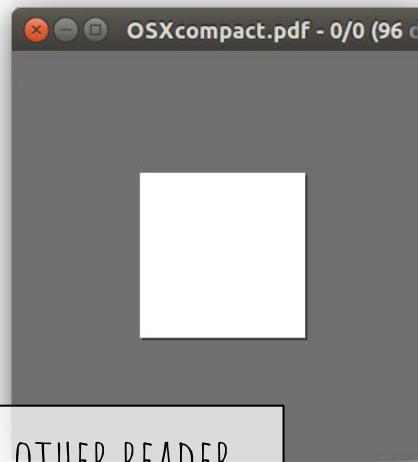
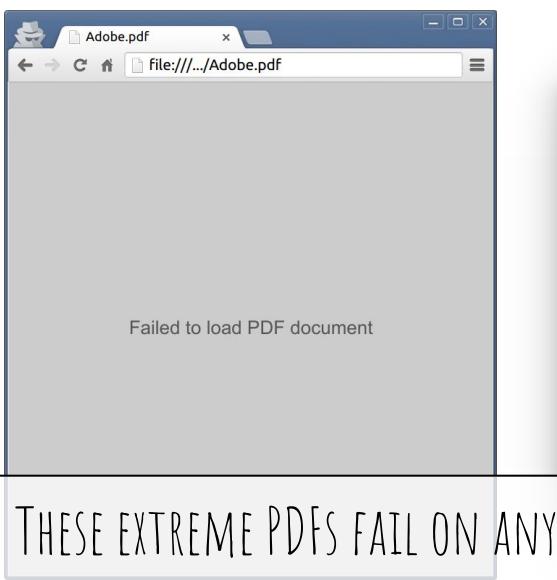
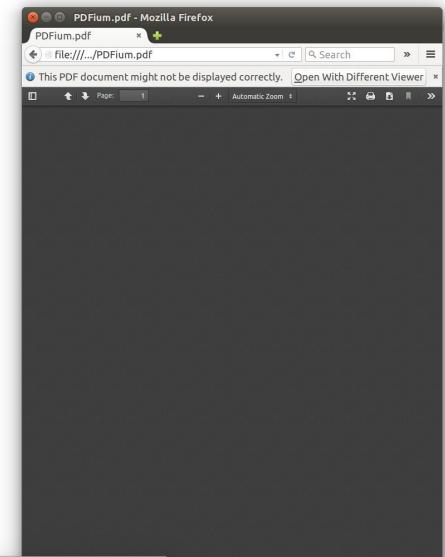
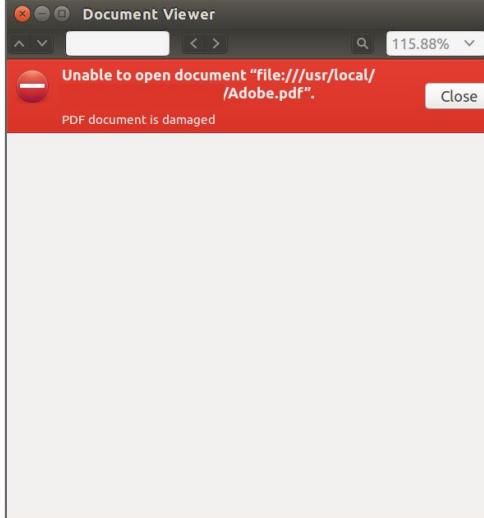
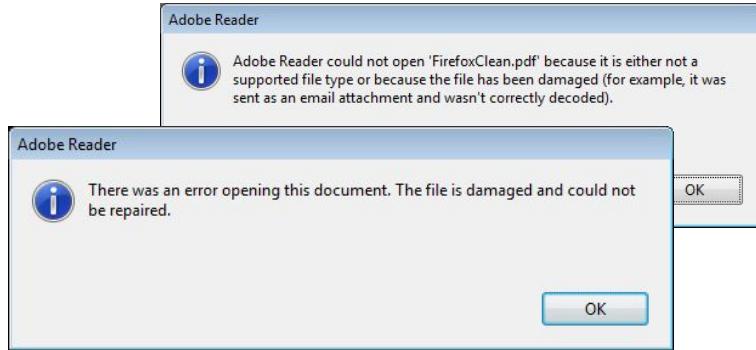
```
2 0 obj
<< No /Length
stream
No BT/ET Tj
20 400 Td
(Chrome) Tj
endstream
```

```
trailer <<
  Direct /Root
  No /Size
  /Root 1 0 R
  No %%EOF
>> No /Type
No startxref
```

**INVALID?**

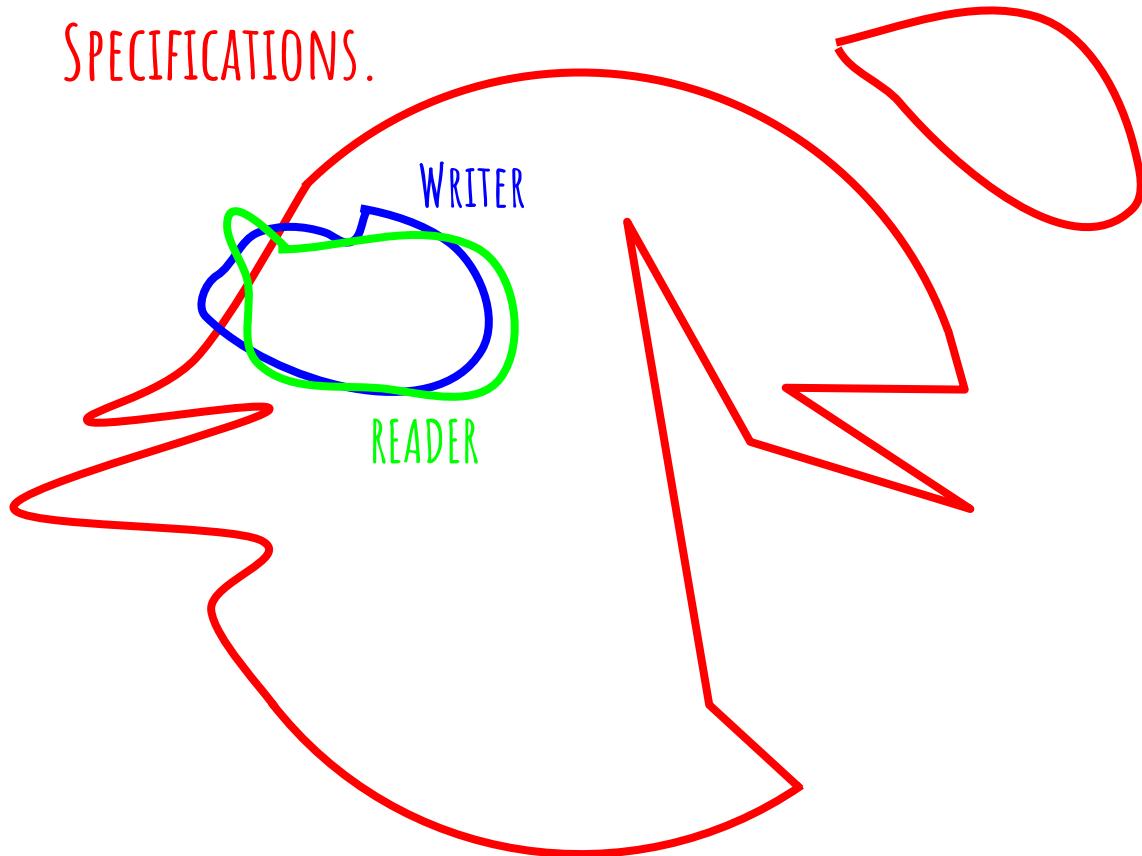






THESE EXTREME PDFS FAIL ON ANY OTHER READER.

SPECIFICATIONS.



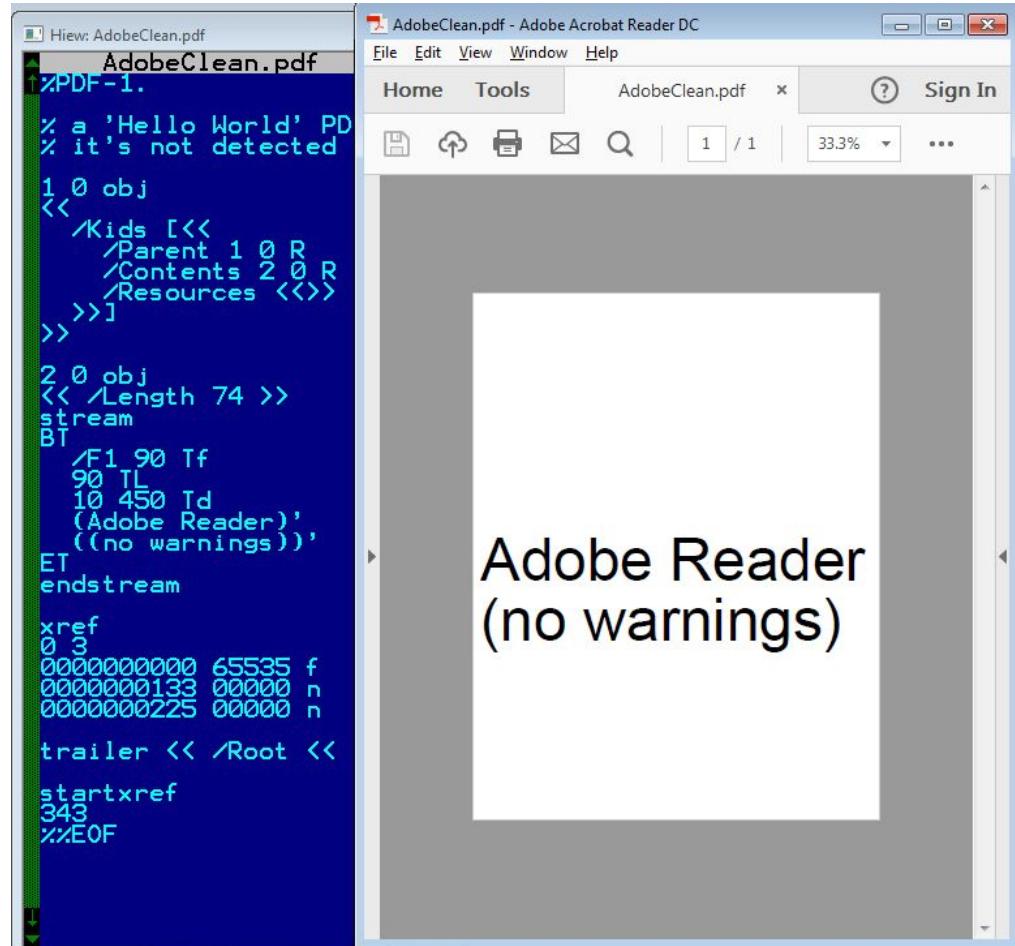
IF ONE OF THESE SOFTWARE BECOMES STANDARD, THE OTHER SOFTWARE WILL HAVE TO ADAPT TO IT.

# RECOVERY

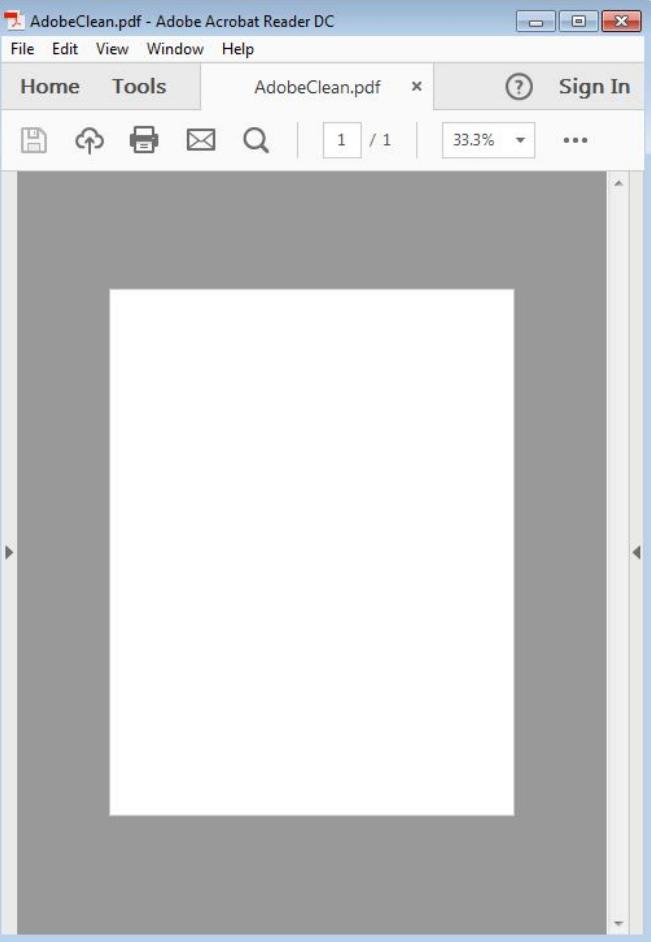
SINCE THERE'S NO OFFICIAL 'DIRECTION',  
OTHER SOFTWARES MAY HAVE  
TO BE TAKEN INTO CONSIDERATION.

# TAKE A STANDARD PDF.

(IT OPENS IN ADOBE  
WITH NO WARNINGS)



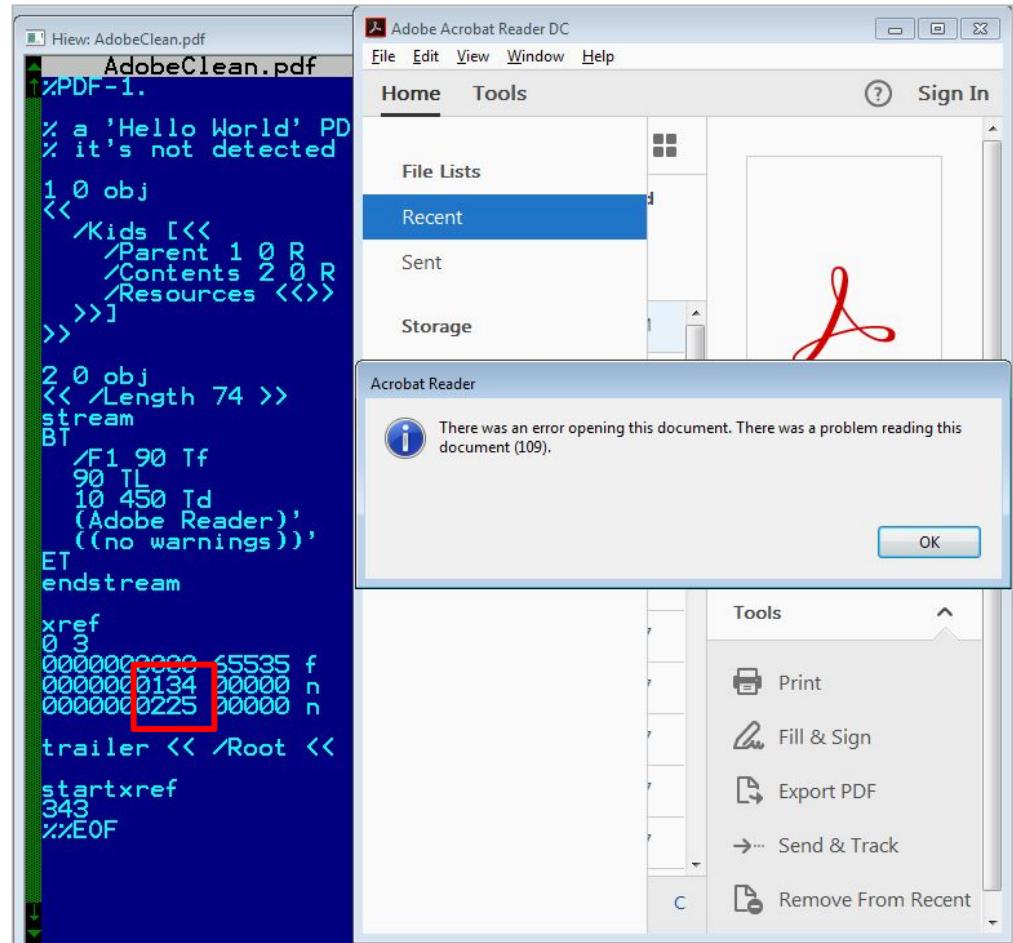
IF YOU MODIFY  
ITS XREF TABLE,  
IT WON'T WORK CORRECTLY...



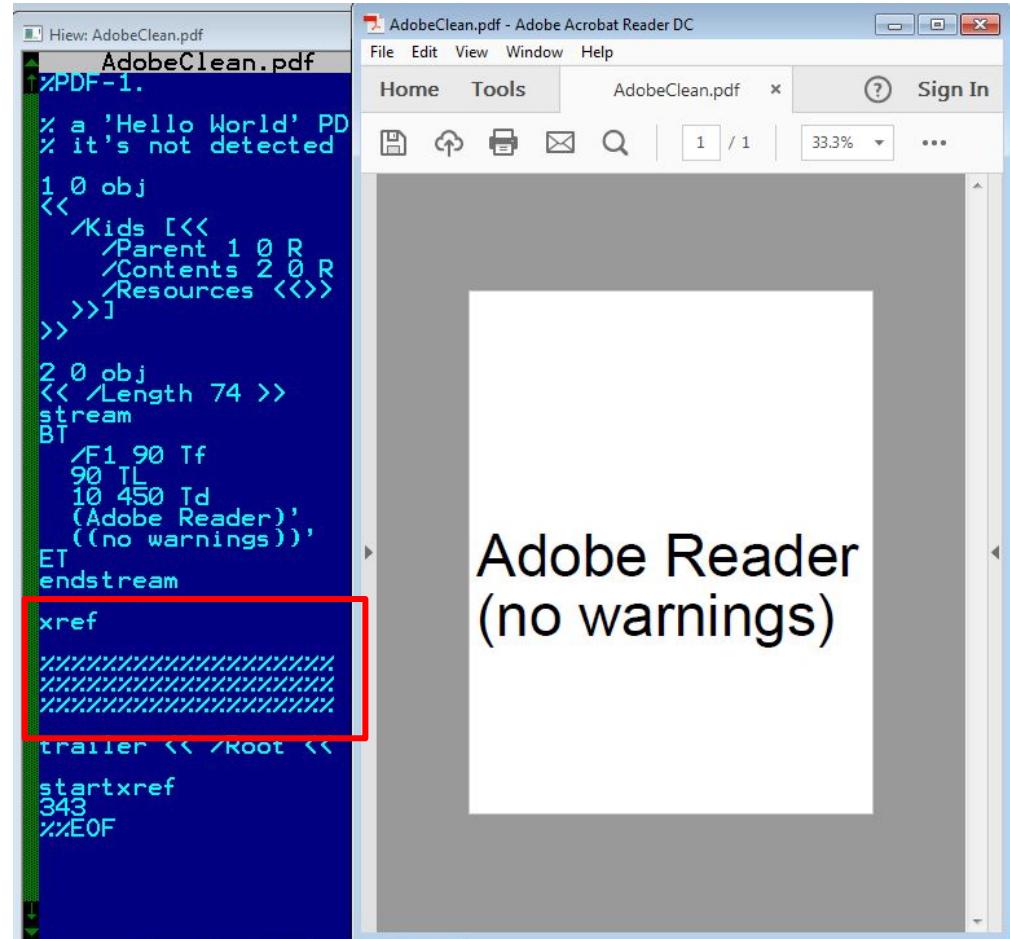
The image shows two windows side-by-side. On the left is a terminal window titled 'Hiew: AdobeClean.pdf' displaying the PDF file's raw code. On the right is the 'Adobe Acrobat Reader DC' application window titled 'AdobeClean.pdf - Adobe Acrobat Reader DC', which shows a completely blank white page.

```
%PDF-1.  
% a 'Hello World' PDF  
% it's not detected  
  
1 0 obj  
<<  
/Kids [<<  
/Parent 1 0 R  
/Contents 2 0 R  
/Resources <>>  
>>  
  
2 0 obj  
<< /Length 74 >>  
stream  
BT  
/F1 90 Tf  
90 TL  
10 450 Td  
(Adobe Reader),  
((no warnings)),  
ET  
endstream  
  
xref  
0 3  
0000000000 65535 f  
00000002122 20000 n  
00000002226 20000 n  
  
trailer << /Root <<  
  
startxref  
343  
%%EOF
```

...OR MAYBE EVEN  
NOT OPEN AT ALL!



BUT IF YOU ERASE  
THE XREF TABLE,  
ADOBEB WILL FALL BACK  
TO RECOVERY MODE,  
AND OPEN THE FILE  
WITHOUT ANY WARNING!



# SCHIZOPHRENIA

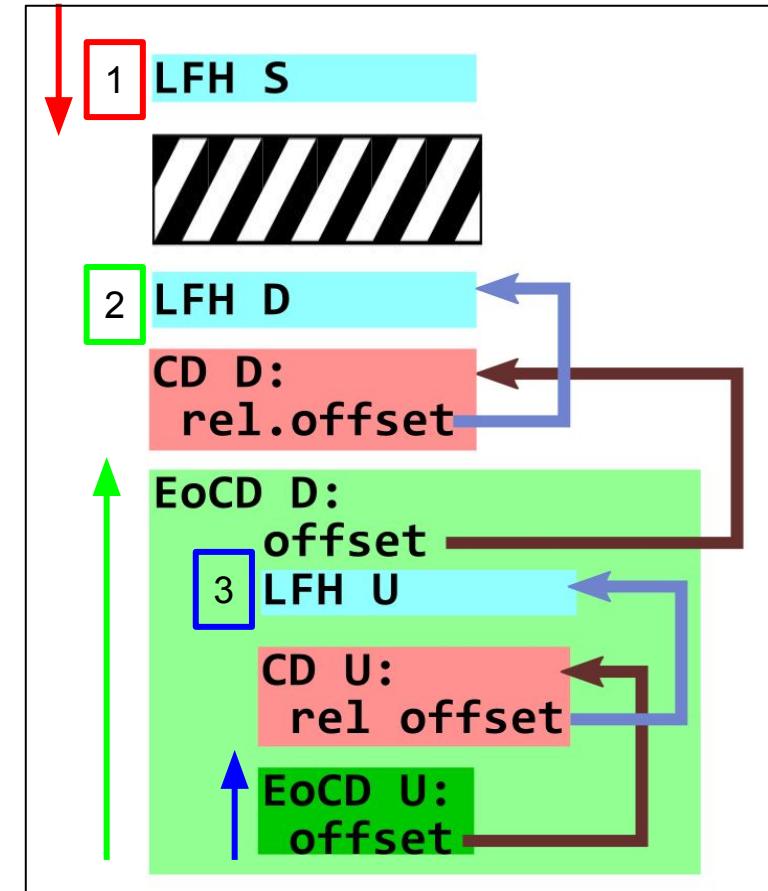
DIFFERENT CONTENTS (CLEAN & MALICIOUS) CAN BE COMBINED  
IN THE SAME FILE, TO BYPASS SECURITY OR FOOL SOFTWARES.

# ZIP ARCHIVES

3 DIFFERENT SOFTWARES WILL SEE  
3 DIFFERENT ARCHIVES  
FROM THE SAME FILE

THIS ENABLED A CRITICAL VULNERABILITY  
IN ALL ANDROID DEVICES IN 2013:

VALIDATE A CONTENT, EXECUTE A DIFFERENT ONE!

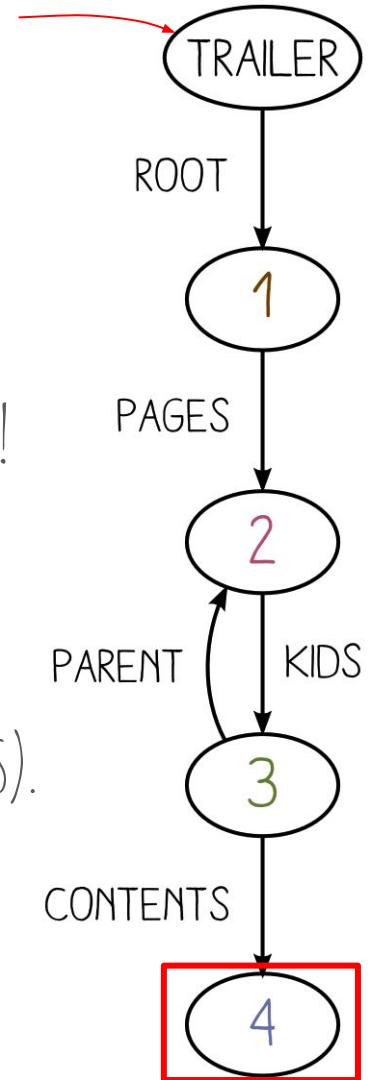


# PDF

THE TRAILER DEFINES THE START OF THE DOCUMENT TREE.

SEE A DIFFERENT TRAILER -> SEE A COMPLETELY DIFFERENT DOCUMENT!

SEVERAL TRAILERS CAN CO-EXIST IN THE SAME FILE, AND  
PARSERS TOLERATE 'UNUSED' OBJECTS (REFERENCED BY UNSEEN TRAILERS).



COMMENTED LINE - SEEN BY PDFIUM

```
% trailer <</Root ...>>
```



STANDARD TRAILER - THE ONLY ONE SEEN BY ADOBE READER

```
trailer <</Root ...>>
```

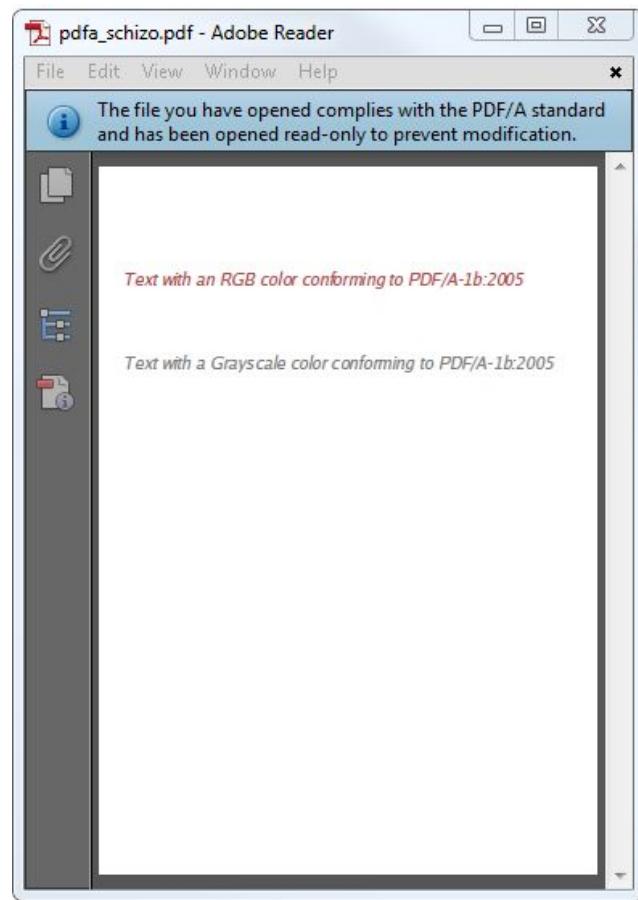


MISSING TRAILER KEYWORD, BUT SEEN BY POPPLER

```
<</Root ...>>
```



3 DIFFERENT DOCUMENTS AS SEEN BY 3 DIFFERENT READERS IN THE SAME FILE  
(SUCH THINGS EVEN HAPPEN ACCIDENTALLY IN THE WILD!)



IT USED TO WORK WITH PDF/A TOO  
(OK FOR ADOBE READER, BUT NOT FOR PREFLIGHT)

LAYERS PRESENT

File Edit View Window Help

Open Tools Fill &

Print

Printer: Microsoft XPS Document Writer Properties Advanced Help

Copies: 1 Print in grayscale (black and white)

Pages to Print: All

Comments & Forms: Document and Markups Summarize Comments

Scale: 166%

11.69 x 8.27 Inches

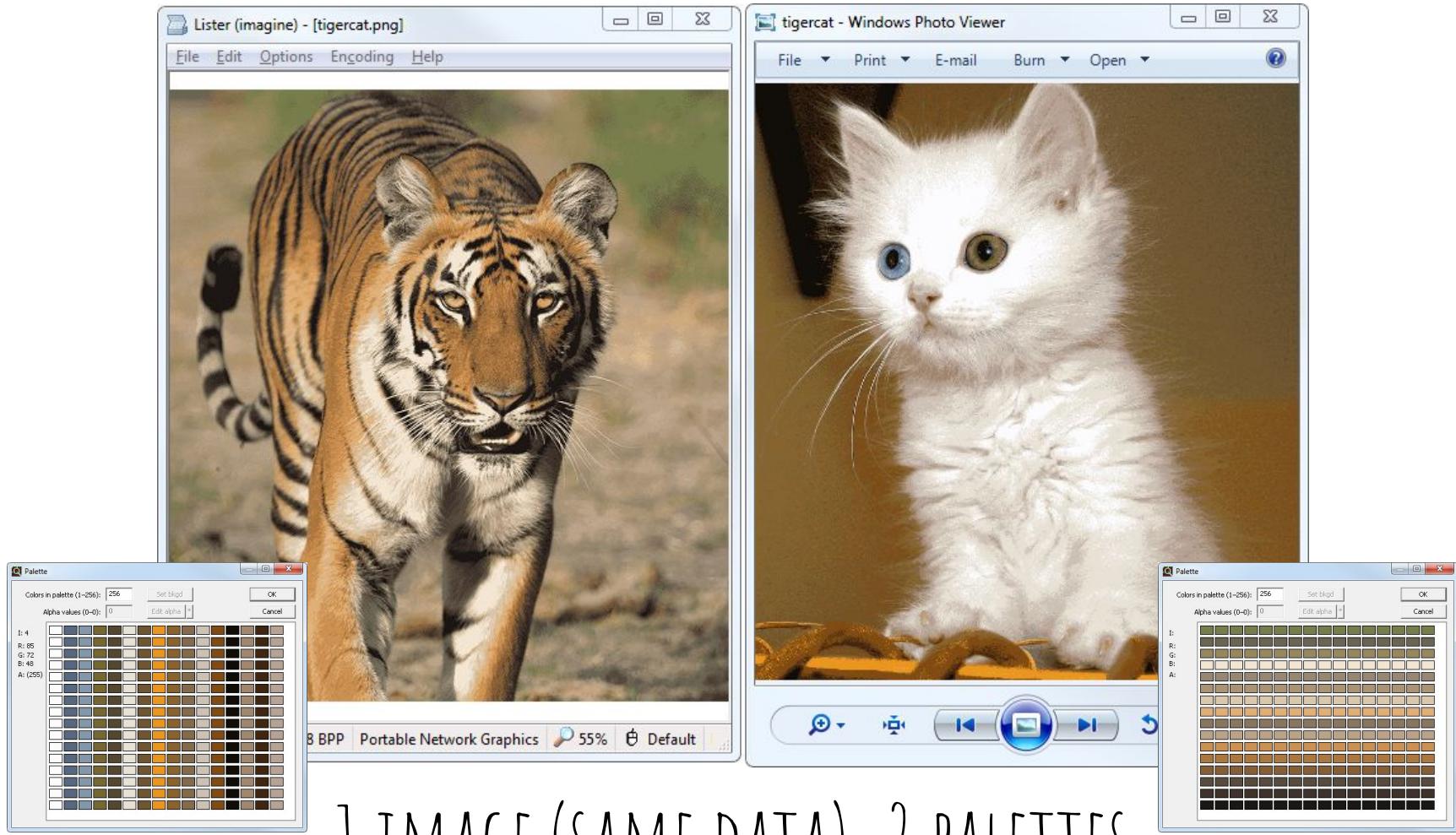
NATIONAL SECURITY AGENCY  
UNITED STATES OF AMERICA

Page 1 of 1

Print Cancel

WHAT YOU SEE IS NOT ALWAYS WHAT YOU PRINT - WHEN YOU USE LAYERS [OPTIONAL CONTENT GROUPS]!

FUN FACT: YOU CAN'T CHANGE THE PRINTING OUTPUT WITH ADOBE READER ;)



1 IMAGE (SAME DATA), 2 PALETTES

# IS THIS JUST AN INFINITE VICIOUS CIRCLE?

DO WE NEED A GREAT FIRE DESTROYING OUR KNOWLEDGE  
SO THAT WE BUILD FILE FORMATS IN A SMARTER WAY?

OR ULTIMATELY, WE'LL FORGET OUR ROOTS AND  
JUST MAINTAIN STACKS OF EMULATION LAYERS...?

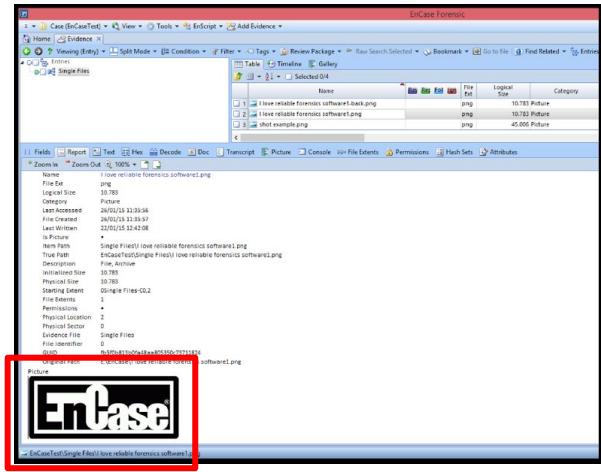


# LAW ENFORCEMENTS

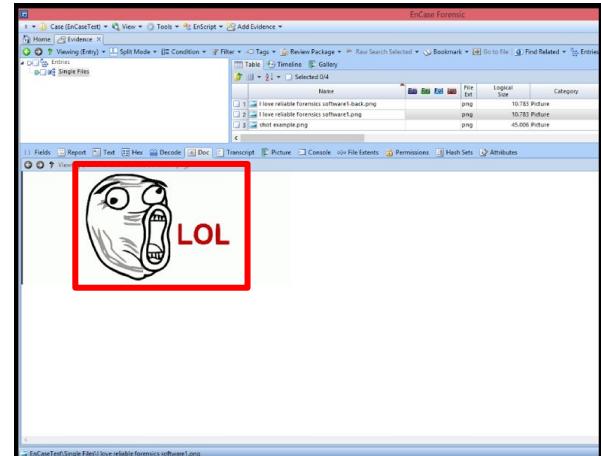
LIKE ARCHIVISTS, INVESTIGATORS RELY ON SOFTWARES  
TO DETERMINE IF SOMEONE IS GUILTY OR NOT.

THESE SOFTWARES RELY ON THE SAME BLURRY SPECS...

THEY'RE VULNERABLE TO THE SAME PROBLEMS.



SAME FILE, TWO DIFFERENT TABS



# What's unique about PDF?

And why PDF will live forever

PDF Association | October 6, 2015

<http://www.pdfa.org/2015/10/whats-unique-about-pdf/>

- FLASH IS NOW DYING, FOR SECURITY REASONS.
- ADOBE IS OUT OF THE GAME FOR PDF.
- LOOKING AT PDF 2.0, I'M VERY SKEPTICAL...  
(MANY EXTRA SECURITY RISKS)

The screenshot shows a web browser displaying the PDF Association website at [www.pdfa.org/2015/10/whats-unique-about-pdf/](http://www.pdfa.org/2015/10/whats-unique-about-pdf/). The page features a large red box highlighting the title "What's unique about PDF?" and the subtitle "And why PDF will live forever". Below this, a smaller red box highlights the author information "PDF Association | October 6, 2015". To the right of the main content is a stylized illustration of a superhero figure with a cape that has the letters "PDF" on it. The superhero is holding a pair of glasses. The website navigation bar includes links for Members' News, Products, Downloads, Videos, Discussion Forums, Login, and a language switcher for English. On the right side of the page, there is a sidebar with a "Tags" section containing various conference names and terms like "accessibility", "conversion", and "archiving".

What's unique about PDF?  
And why PDF will live forever  
PDF Association | October 6, 2015

The Portable Document Format possesses a variety of attributes that are, themselves, more subtle than what one normally thinks of as "features". Taken together, however, they describe a format of such flexibility and power that it will define the essential "electronic document" concept forever.

**It's a document**

Wikipedia, among others, defines "document" in terms of content (text and graphics, together in a layout) as it exists at a given moment in time. The need for a sharable electronic document drove the fundamental design of PDF. The format allows pages – that is, a fixed layout of text and graphics – to be shared with total fidelity to the author's intent. However they were made, PDF documents look the same way to everyone. This feature was critical to establishing PDF as a candidate for the standard electronic document format, but it was not enough.

**One format can hold it all**

Closely related to PDF's ability to reliably share fixed-layout content is the fact that a PDF document may include pages from many (any) different source. Users can (and often do) mix PDF pages produced from MS Word with PDF pages from scanned documents, screen-captures, CAD images and more. No electronic document format could replace PDF unless it was also able to allow users to mix pages together to form the documents (see above) they need.

**Totally unrestricted**

**Tags**

3rd International PDF/A Conference 4th PDF/A Conference 2012 PDF Technical Conference accessibility accessible accessible PDF AIM archive Archiving assistive technology barrierefrei compression conversion convert digital signature DMS DMS Expo document conversion ECM encryption ISO Standard JAVA long-term archiving metadata OCR

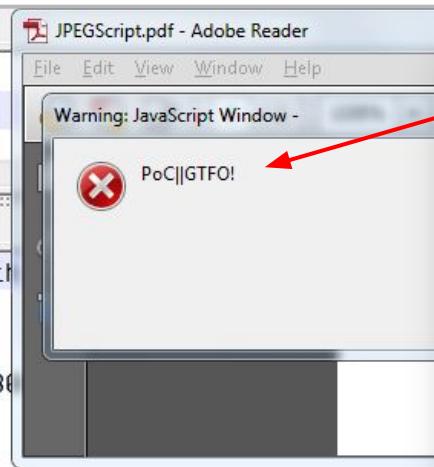
JPEGScript.pdf

```
trailer << /Size 2 /Root <</Pages <>>
/OpenAction << /S/JavaScript /JS 1 0 R
>>
```

JPEGScript.pdf

```
1 0 obj<</Filter[/ASCIIHexDecode/DCTDecode]/Width
/ColorSpace/DeviceGray >>
stream
ffd8ffe000104a46494600010100000100010000ffdb00430
endstream
endobj
```

- SPECS & SOFTWARE TOLERATED TO ENCODE A (MALICIOUS) JAVASCRIPT AS JPEG.
- THIS WAS USED TO BYPASS SECURITY SCANS.
- THIS 'FEATURE' WAS KILLED TO IMPROVE SECURITY
- > SPECIFICATIONS WERE NEVER UPDATED -> THEY'RE NOW CLEARLY OUTDATED.



# PDF IS GREAT, BUT . . .

NO MAJOR ACTOR (ADOBE) BEHIND IT ANYMORE?

PDF 2.0 IS TOO DIFFERENT?

TOO PERMISSIVE SECURITY-WISE?

(DON'T GET ME WRONG, I REALLY LIKE TO [AB]USE PDF)

# MY POINT OF VIEW:

## NEED TO CREATE BETTER CORPUSES

TO DEMONSTRATE THE CURRENT STATE OF THINGS.

TO DETERMINE THE ACTUAL LIMITS OF FILE FORMATS.

TO MAKE AUTOMATION EASIER FOR EVERYONE.

ATOMIC - COPYRIGHT-FREE - PII-FREE

# PRESERVATION

CONTENT, OR THE FILES? AND PRIVATE DATA?

PRESERVING USUALLY MEANS "SAVING TO A SAFER FORMAT"

THIS MEANS WE GAVE UP ON THE ORIGINAL FORMAT...

BUT WHICH ONE IS SAFE?

WE NEED TO PRESERVE INTERESTING FILES AS-IS TOO.

# WE DON'T NEED 'SAFER' FORMATS

(WHATEVER THAT MEANS)

YET ANOTHER FORMAT?  
BLURRY SPECS,  
DIVERGENCES,  
UNCLEAR CORNER CASES...

HOW STANDARDS PROLIFERATE:  
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)

SITUATION:  
THERE ARE  
14 COMPETING  
STANDARDS.

14?! RIDICULOUS!  
WE NEED TO DEVELOP  
ONE UNIVERSAL STANDARD  
THAT COVERS EVERYONE'S  
USE CASES.

YEAH!



SOON:

SITUATION:  
THERE ARE  
15 COMPETING  
STANDARDS.

# WE NEED A NEW PROCESS: FILE FORMATS SHOULD BE ALIVE!

LIKE CRYPTOGRAPHY: UPDATES, DEPRECATIONS...

-> EVENTUALLY UNIFORM STANDARDISATION

WITH A DATE, A VERSION NUMBER, A COMMIT.

UP TO DATE SPECIFICATIONS AND OPEN VALIDATOR.

ONCE THINGS ARE PROPERLY DOCUMENTED, WE COULD GO BACK EASILY.

# WHY DOESN'T IT HAPPEN?

BECAUSE WE HAVEN'T PROVED ENOUGH YET HOW BROKEN THEY ARE?

IN CRYPTOGRAPHY, THERE ARE OFFICIAL COMPETITIONS  
TO BREAK THE DRAFTS BEFORE CHOOSING THE STANDARD:  
SURVIVAL OF THE FITTEST BEFORE SETTING STANDARDS IN STONE.

# CONCLUSION

- FILES FORMATS ARE AWESOME (EVEN PDF!)
  - EXCEPT WHEN THEY DON'T WORK AS EXPECTED :)
- SPECIFICATIONS ARE NOT CHALLENGED ENOUGH.
- FORMATS AUTHORSHIP IS NOT A LIABILITY.

WE NEED TO DEVELOP OUR EXPERTISES  
AND SHARE OUR KNOWLEDGE.

# FEEDBACK?

THANK YOU FOR  
READING THAT FAR!

**ANGE ALBERTINI**  
reverse engineering

VISUAL DOCUMENTATIONS

[@angealbertini](https://twitter.com/angealbertini)  
ange@corkami.com  
<http://www.corkami.com>



# EXTRA RESOURCES

FUNKY FILES FORMATS:

[HTTPS://SPEAKERDECK.COM/ANGE/FUNKY-FILE-FORMATS-31C3](https://speakerdeck.com/ange/funky-file-formats-31c3)

SCHIZOPHRENIC FILES:

[HTTPS://SPEAKERDECK.COM/ANGE/SCHIZOPHRENIC-FILES-V2](https://speakerdeck.com/ange/schizophrenic-files-v2)

ABUSING FILE FORMATS:

[HTTPS://ARCHIVE.ORG/STREAM/POCORGTFO07#PAGE/N17/MODE/2UP](https://archive.org/stream/pocorgtf007#page/n17/mode/2up)