Trusting files
(and their formats)

Ange Albertini - Hack.Lu 2015

ANGE ALBERTINI

reverse engineering

VISUAL DOCUMENTATION

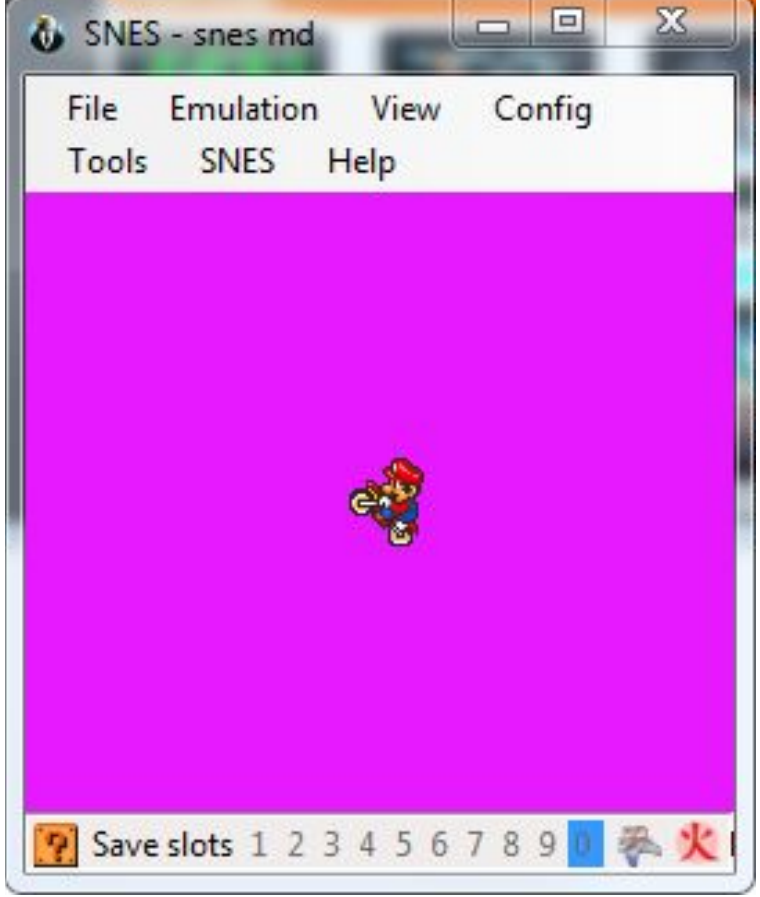
@angealbertini

ange@corkami.com

http://www.corkami.com

Welcome to my talk!

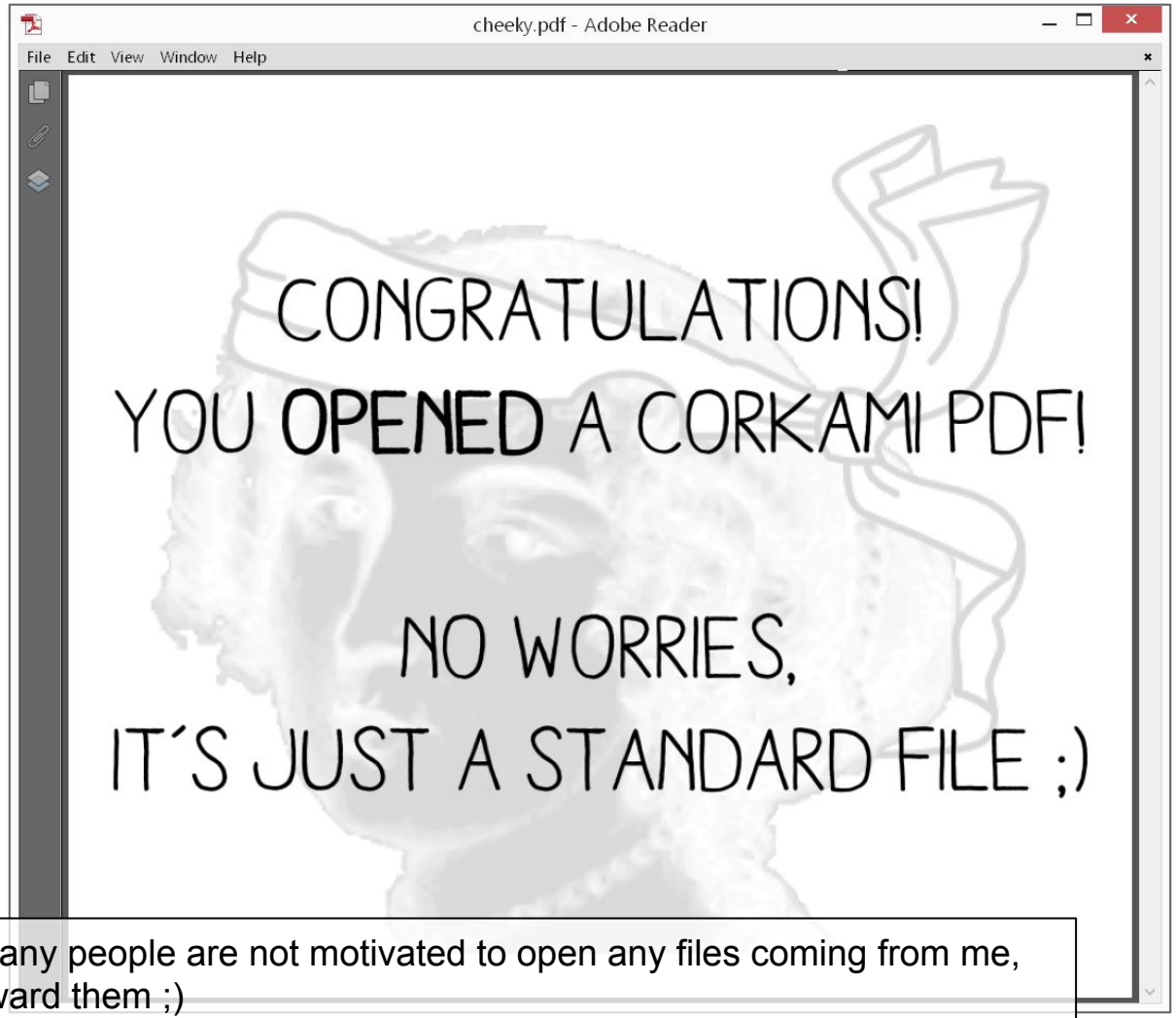
My resume is a PDF. What could go wrong ? ;)



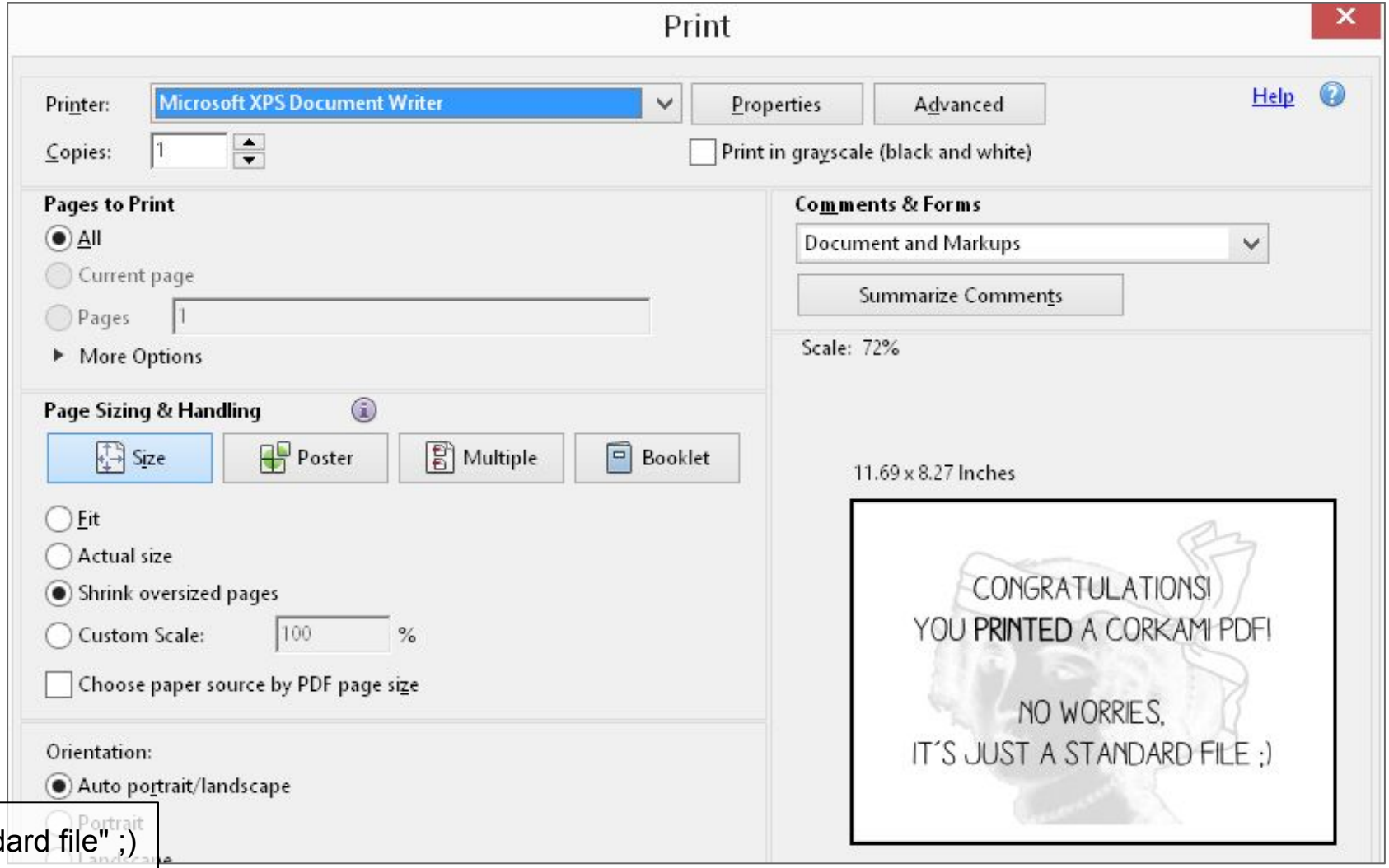
NINTENDO

Sega®

;)



For some reason, many people are not motivated to open any files coming from me, so I made this to reward them ;)



"standard file" ;)

Yes, I write files by hand...

[...and I open them in hex editors]

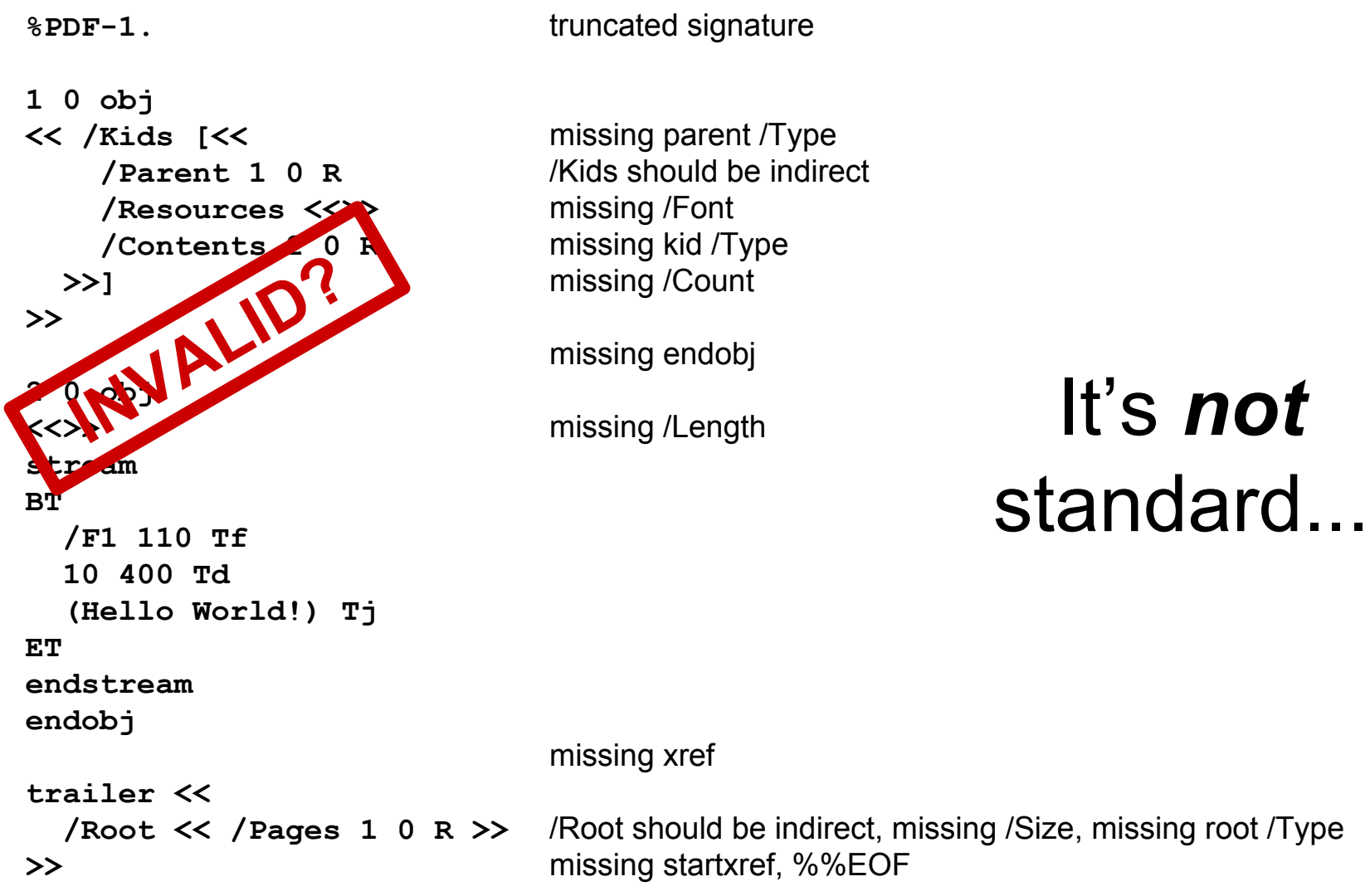
```
%PDF-1.

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

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

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

...like this one


%PDF-1.
truncated signature
1 0 obj
<< /Kids [<<
/Parent 1 0 R
/Resources <<>>
/Contents 2 0 R
>>]
>>
missing parent /Type
/Kids should be indirect
missing /Font
missing kid /Type
missing /Count
missing endobj
2 0 obj
<<>>
stream
BT
/F1 110 Tf
10 400 Td
(Hello World!) Tj
ET
endstream
endobj
missing /Length
trailer <<
/Root << /Pages 1 0 R >>
>>
missing xref
/Root should be indirect, missing /Size, missing root /Type
missing startxref, %%EOF
INVALID?
It's *not* standard...

...but it works

exactly as planned!

(without any reported error)

File formats are my playground

(and I'm beyond recovery already)

Files or file formats?

It's not a real question.

Fichier sans format n'est que ruine de l'âme ;)

LE GOUVERNEMENT
DU GRAND-DUCHÉ DE LUXEMBOURG
Administration des contributions directes

modèle 100 F

www.impotsdirects.public.lu

Bureau d'imposition:

Réinitialiser

Déclaration pour l'impôt sur le revenu de l'année 2014

Ce formulaire est destiné aux personnes physiques résidentes et non résidentes. La déclaration est à remettre remplie et signée pour le 31 mars 2015 au bureau d'imposition compétent sous peine d'un supplément d'impôt pour dépôt tardif ou non-dépôt. Les personnes physiques qui n'ont pas leur domicile fiscal ou leur séjour habituel au Luxembourg doivent remplir la rubrique "non-résidents" à la page 3.

signalétique

	contribuable	contribuable conjoint/partenaire
nom	101	102
prénom	103	104
date de naissance /	105	106

To share information, you need to use a common standard.

Forms & file formats

moving to a different country == making a PDF/SNES polyglot

same problems: all similar, but all different

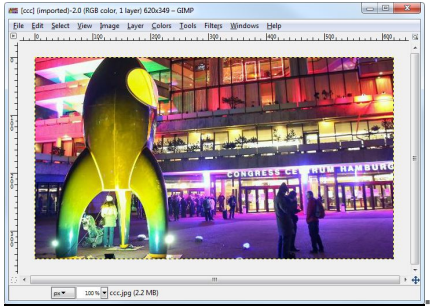
only difference: forms are rarely required to evolve

Trusting files comes with trusting their format.

Knowing that the specs will be useful and reliable.

Retrospective

JPG

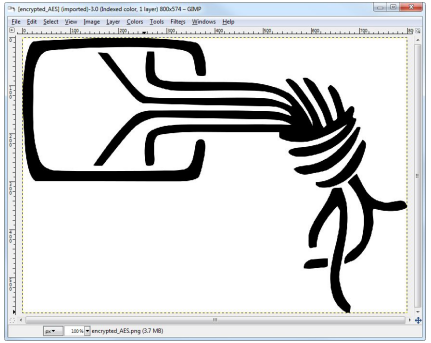


JAR
(ZIP + CLASS)

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

>
```

AES_{K_1}



PNG

AES_{K_2}

FLV

3DES

PDF

I gave an entertaining presentation with many funky binary creations.
Check it if you want more binary magic tricks ;)

pocorgtfo07.pdf - Adobe Reader

File Edit View Window Help

Open 18 / 64 139% Tools Fill & Sign Comment

6 Abusing file formats; or, Corkami, the Novella

by Ange Albertini

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.

Like beef cuts, which vary with the country's standards by which the animal is cut, a file is subject to interpretations of the standard. The beauty of standards is that there are so many interpretations to choose from!

Because these standards are sometimes unclear, incomplete, or difficult to understand, a variety of abuses are possible, even if the files are considered valid by individual parsers.

A *Polyglot* is a file that has different types simultaneously, which may bypass filters and avoid security counter-measures. A *Schizophrenic* file is one that is interpreted differently depending on the parser. These files may look innocent (or corrupted) to one interpreter, malicious to another. A *Chimera* is a polyglot where the same data is interpreted as different types, which is a more advanced kind of filter bypass.

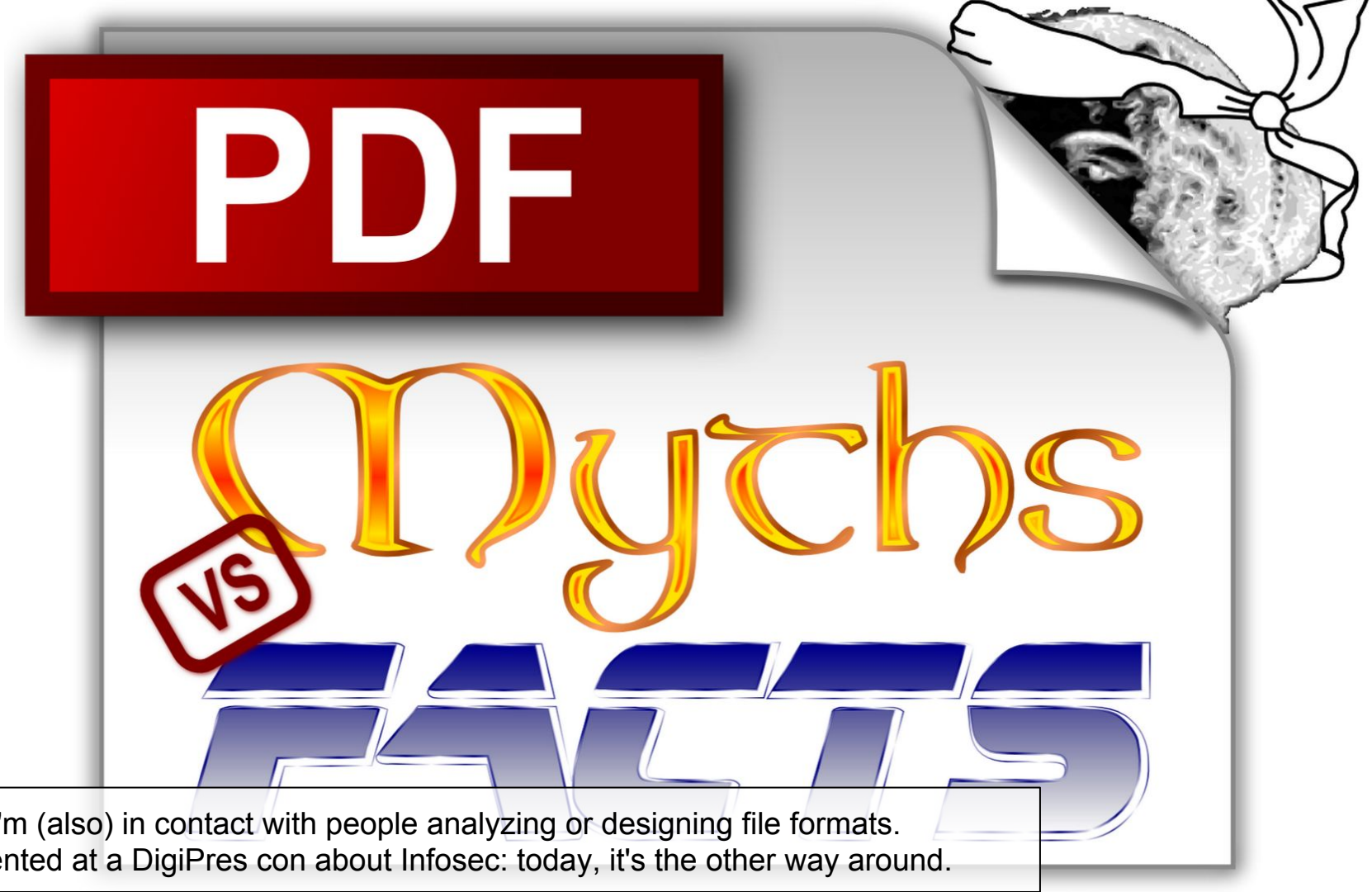
This paper is a classification of various file techniques, many of which have already been mentioned in [illegible] overview and comparison of them, not to necessarily [illegible]

I wrote a technical paper classifying all my file format abuses:
So far, I was (only) playing with file formats.

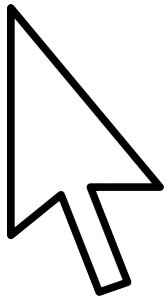
Why?

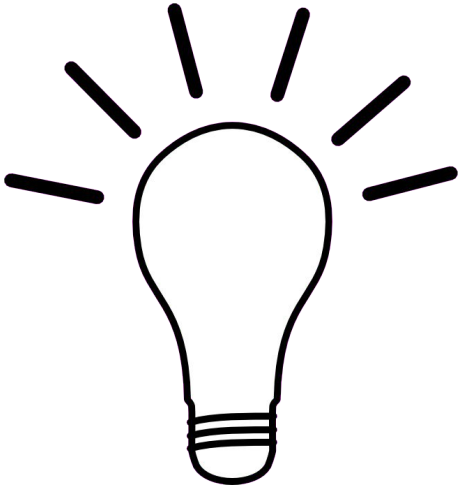
Why are all these abuses possible?
What could we (try to) do about it?

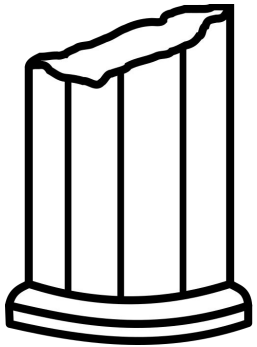
The big question...



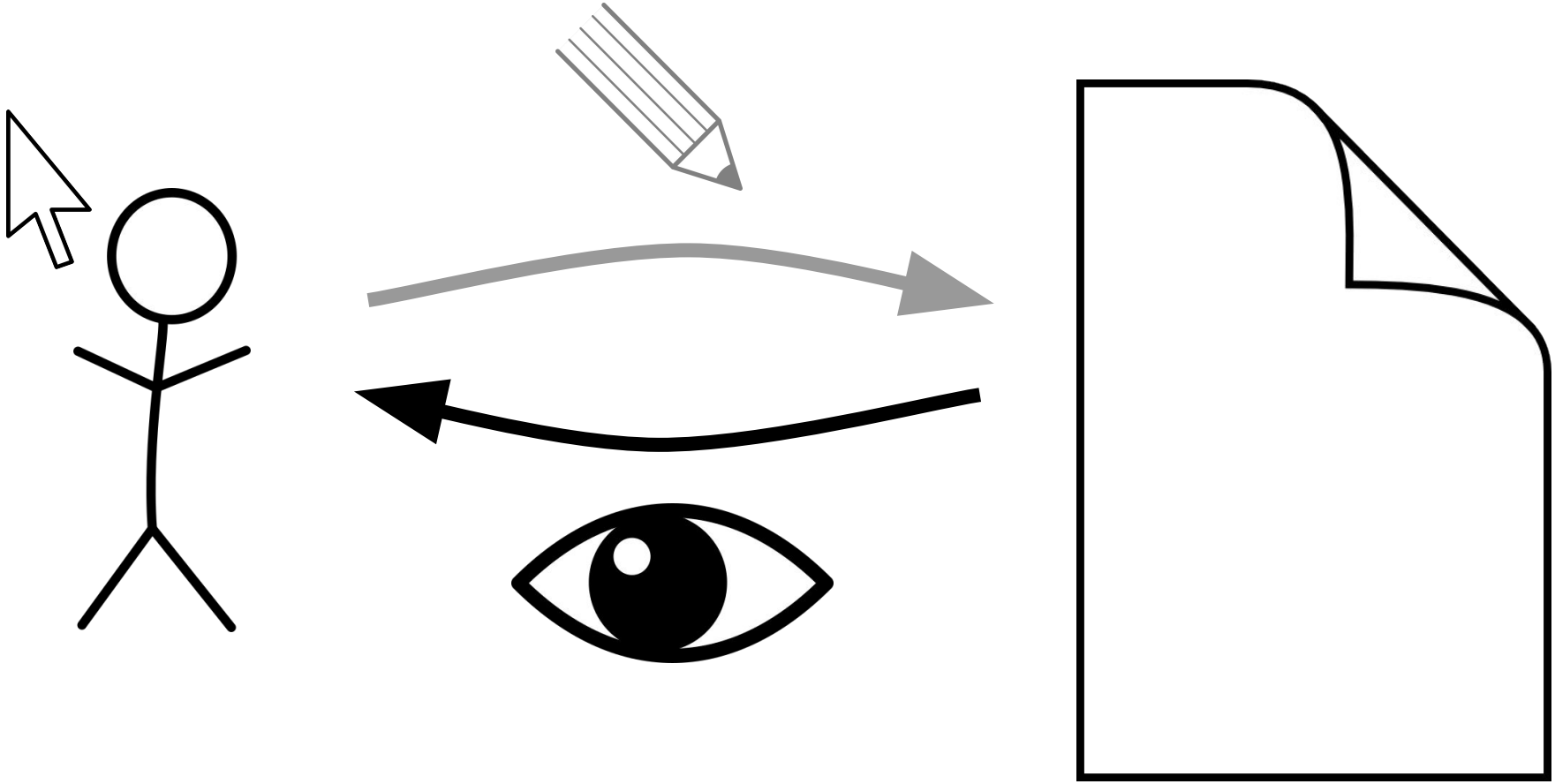
Now, I'm (also) in contact with people analyzing or designing file formats.
I presented at a DigiPres con about Infosec: today, it's the other way around.



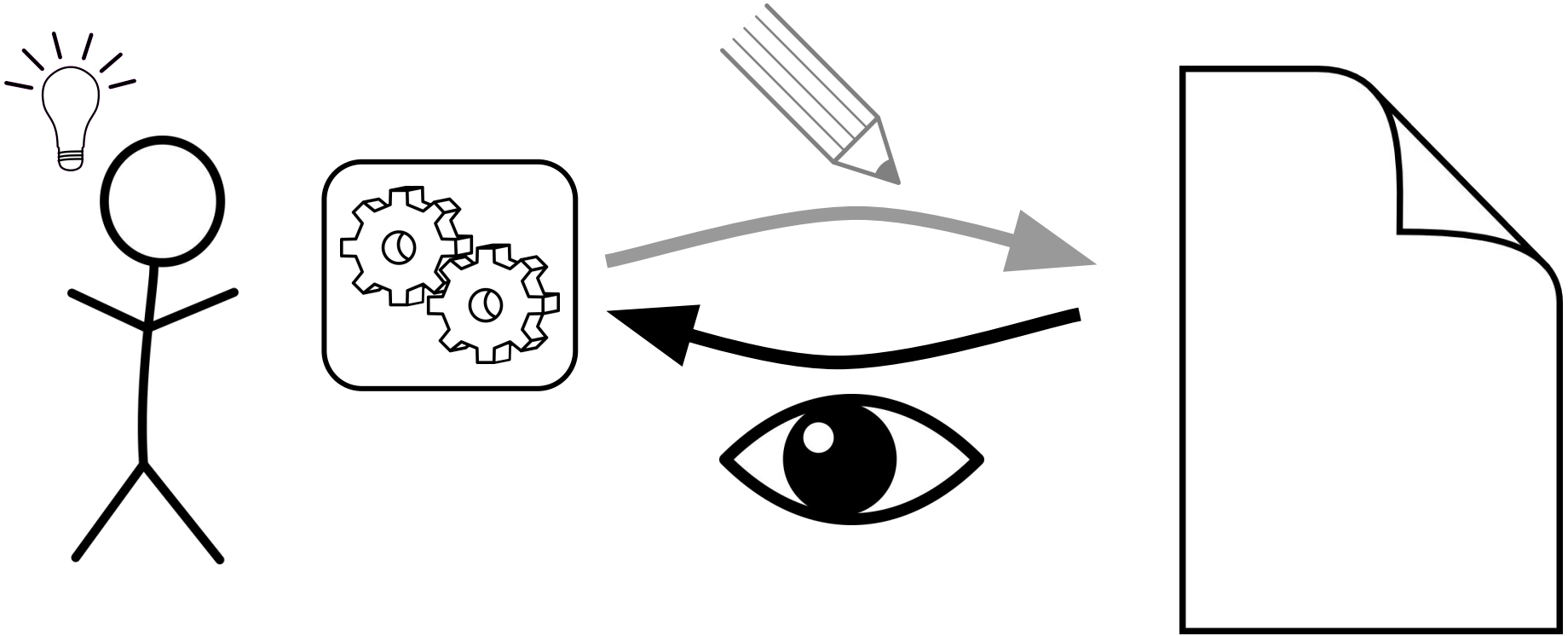




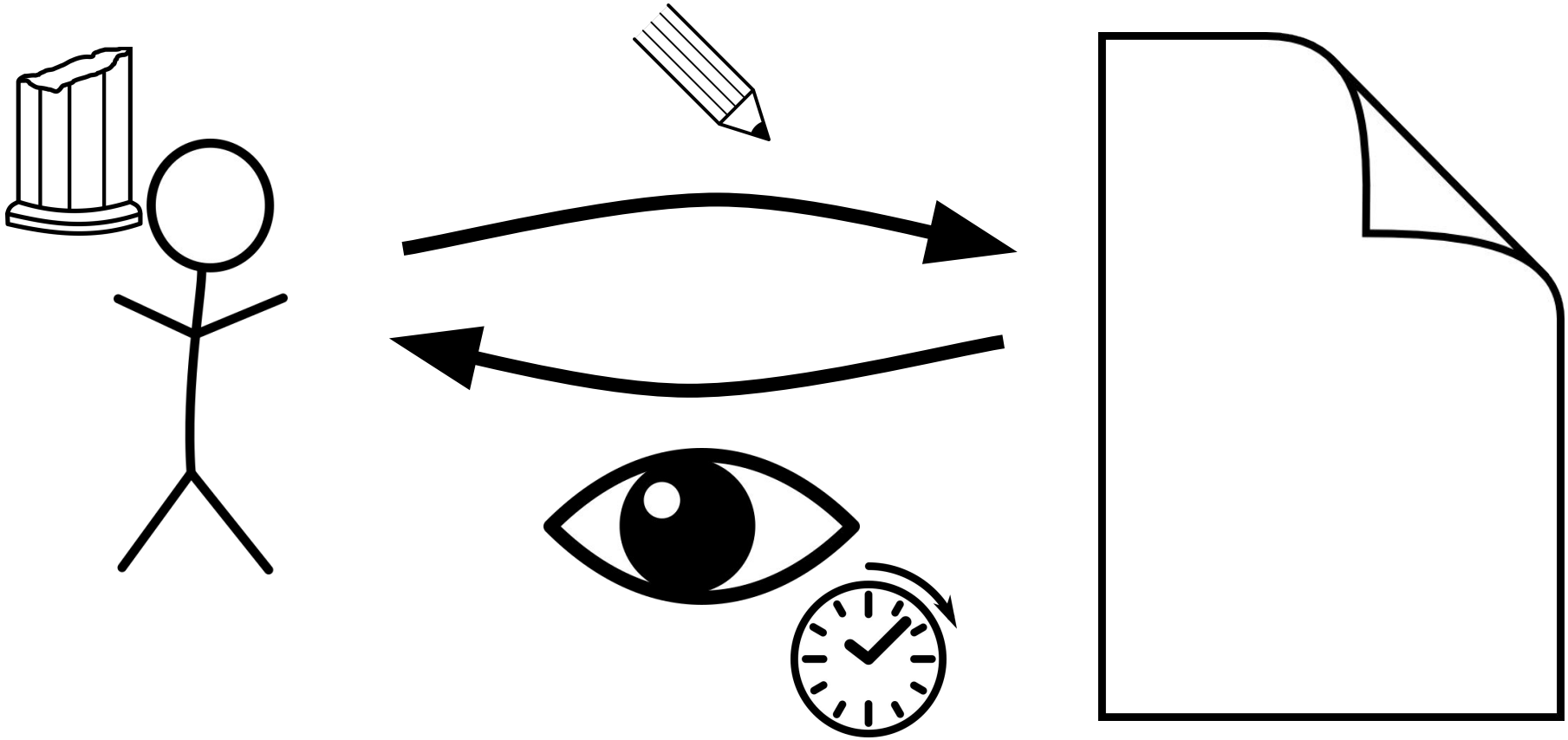
There are different kinds of expectations for files.



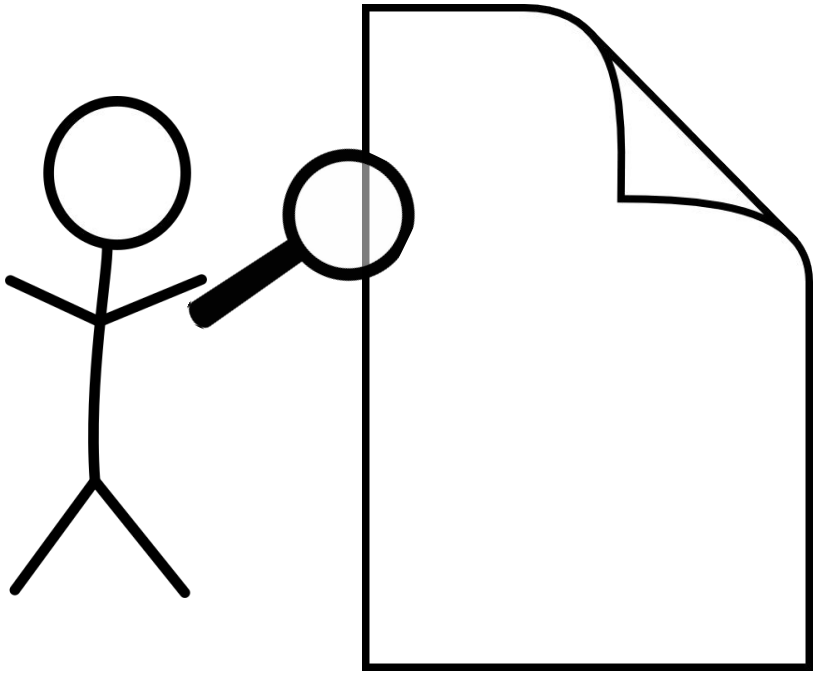
The end-user just wants to view external files, store his own information and re-use it.



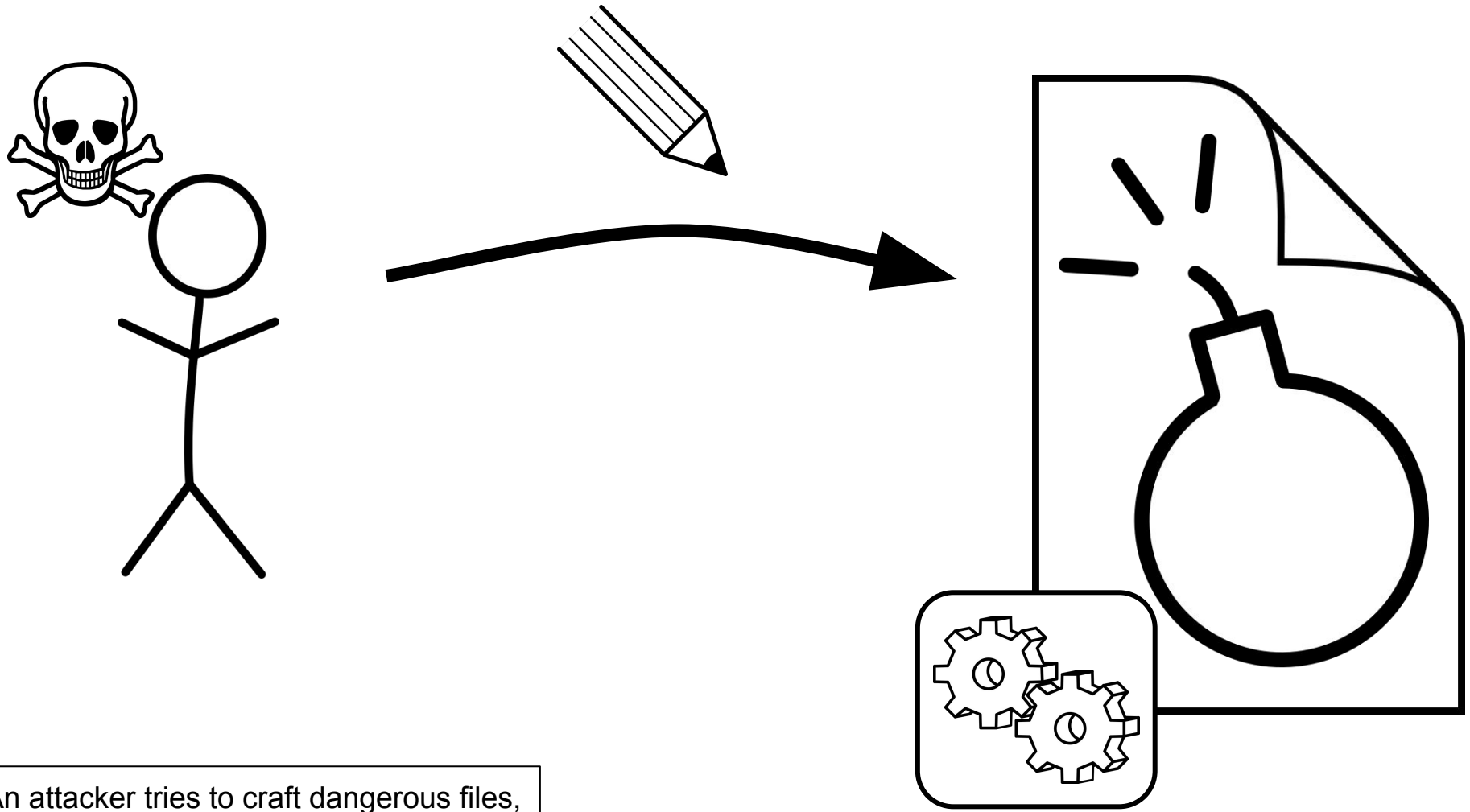
The developer relies on the specifications to add support in his library.



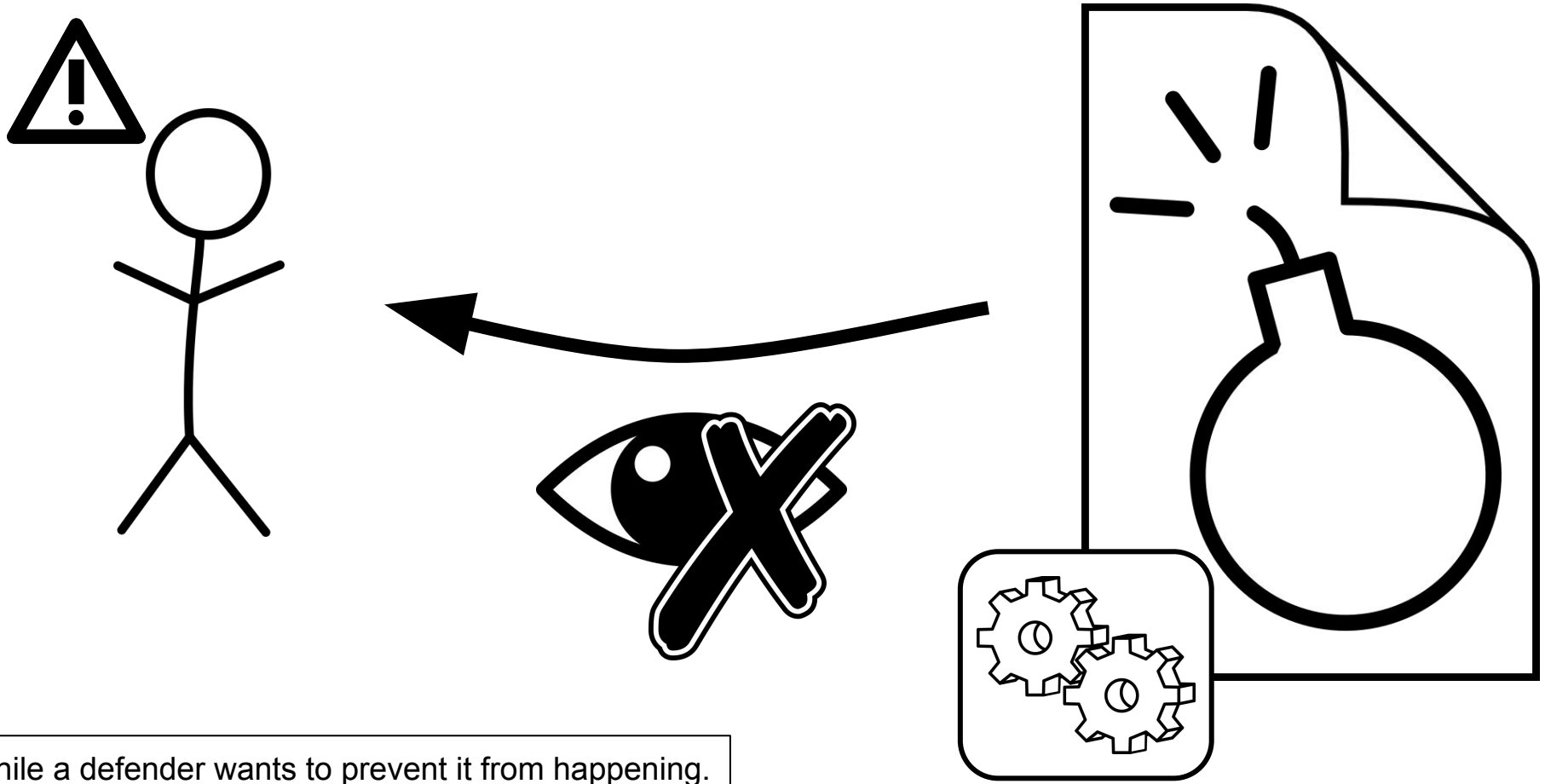
The archivist wants to make sure that his data will be re-usable *much* later.



The digital investigator looks for clues in a suspect's system.



An attacker tries to craft dangerous files,



while a defender wants to prevent it from happening.

Common points

We're blind believers:

- believing that we'll be able to reuse our information
- believing that in any case, we can just rely on the specs to help us, like a religious book.

"the cult of the (useless) specs" ;)

It's not just an Infosec problem.

Bad specs make it harder for devs, DFIR, digipres, defenders...

Theory: check official specs

Reality:
check unofficial specs & blog posts
analyse/reverse libraries
gather ITW (clean & malware) samples

Does it ring a bell ?

Bad specs are why attackers and DFIR devs can make so much money ;)

It's not specs reading anymore, it's reversing.

Not all abuse of file formats turn into exploits.

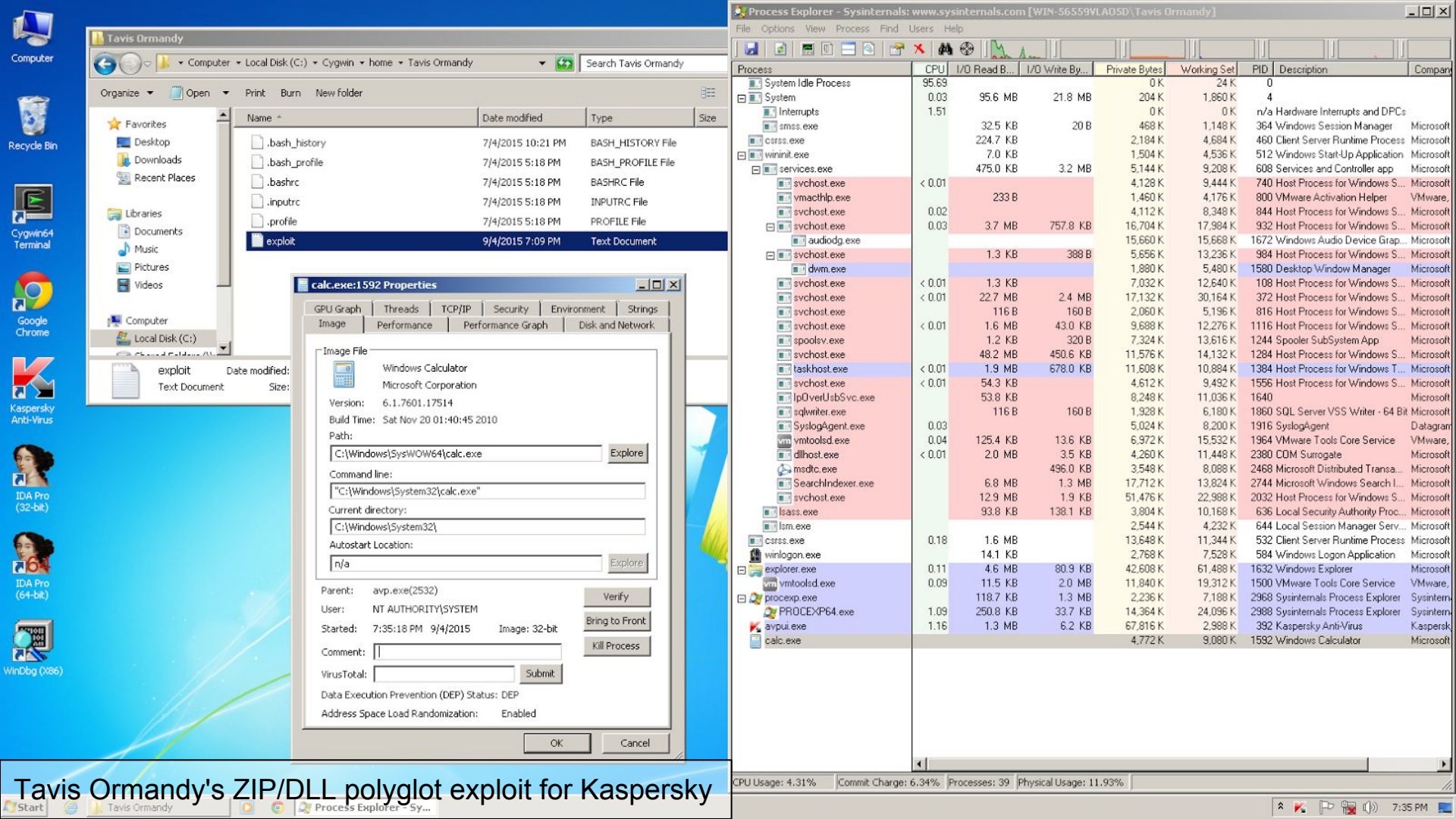
But why should we only fix what's pwning you?

"Short term fix" anyone?

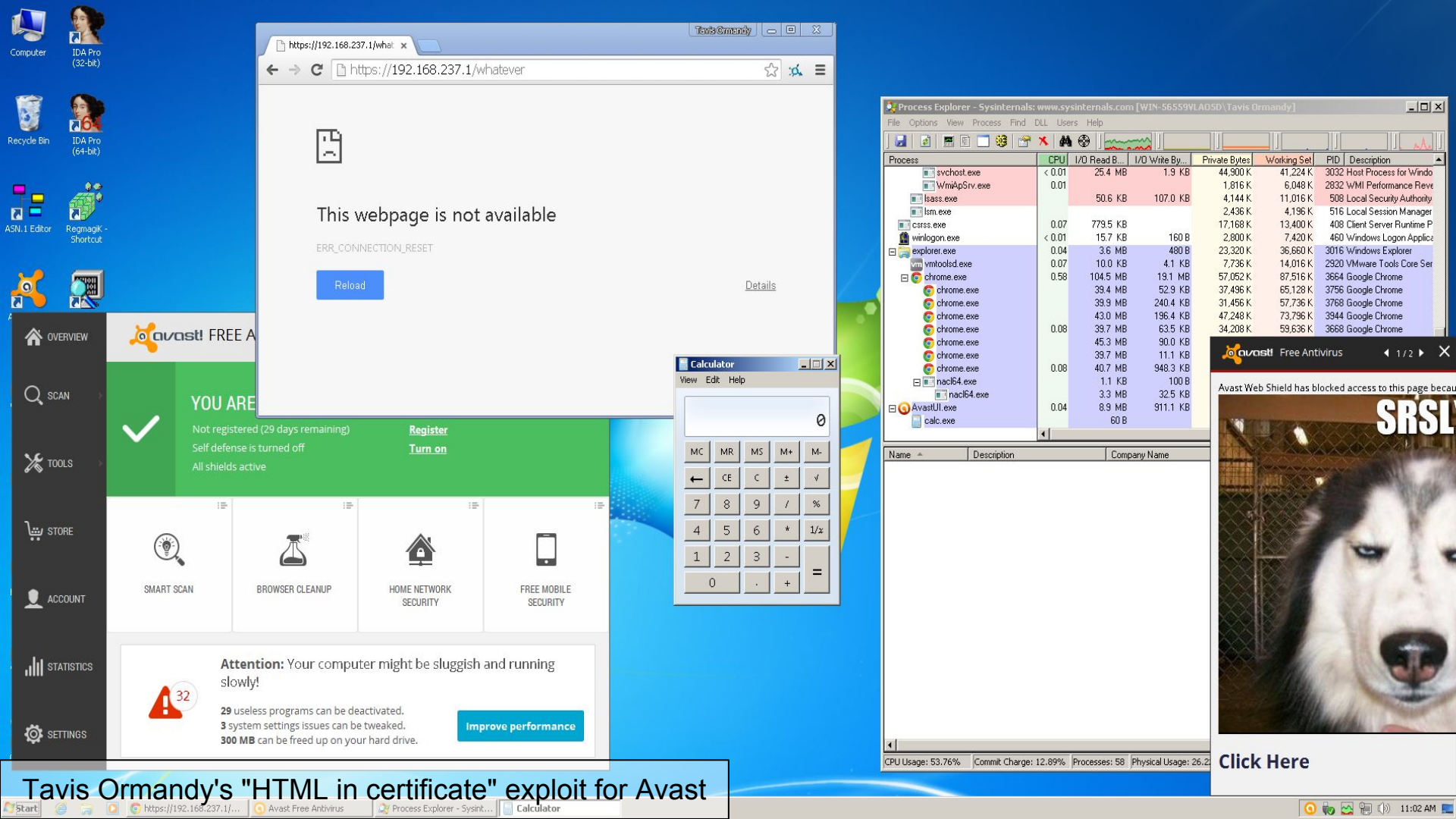
We just care about code, and "cyber attacks".

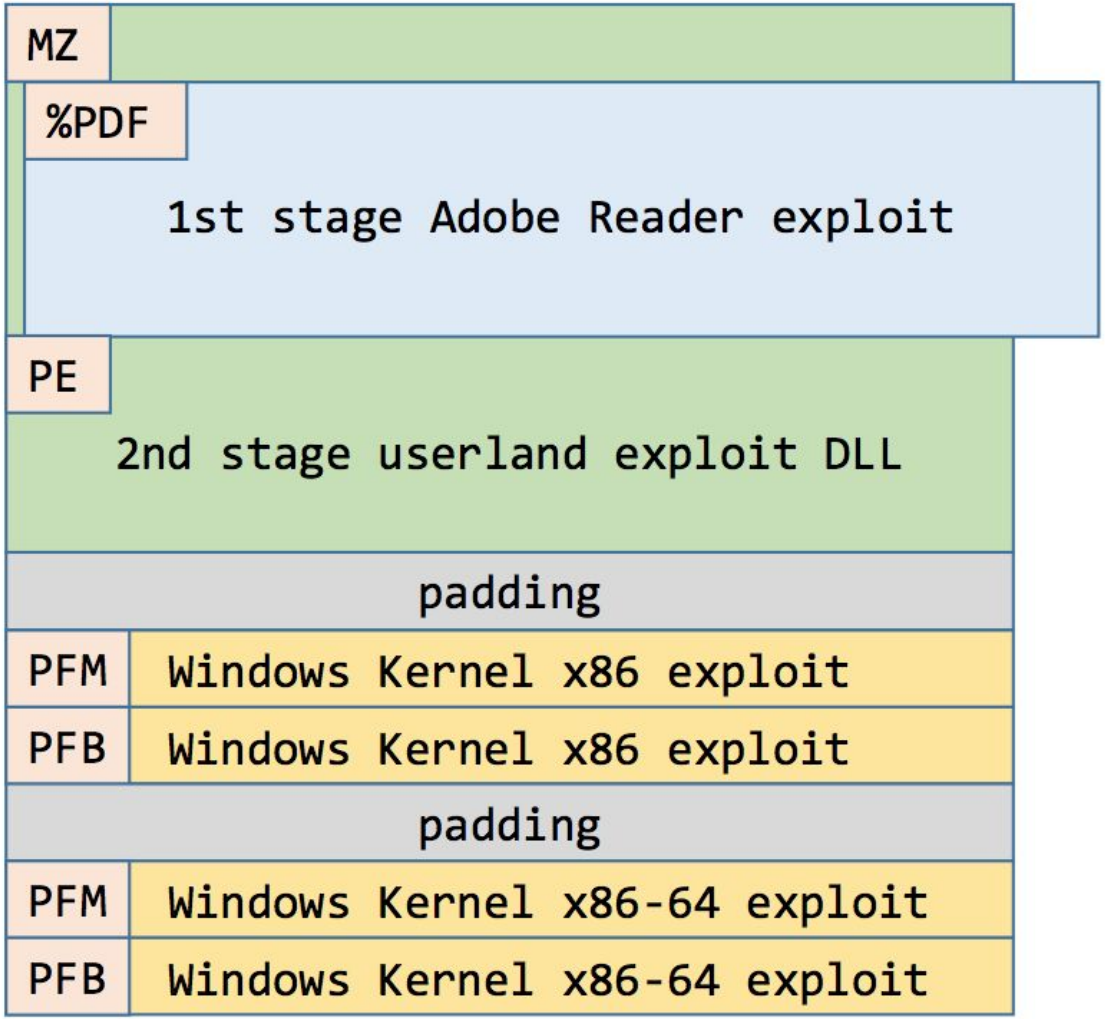
Files tricks go under the radar.

Usually… a few exceptions...



Tavis Ormandy's ZIP/DLL polyglot exploit for Kaspersky


Computer
IDA Pro (32-bit)
Recycle Bin
IDA Pro (64-bit)
ASN.1 Editor
RegmagiK - Shortcut
https://192.168.237.1/whatever
This webpage is not available
ERR_CONNECTION_RESET
Reload
Details
Tavis Ormandy
avast! FREE A
OVERVIEW
SCAN
TOOLS
STORE
ACCOUNT
STATISTICS
SETTINGS
YOU ARE
Not registered (29 days remaining)
Register
Self defense is turned off
Turn on
All shields active
SMART SCAN
BROWSER CLEANUP
HOME NETWORK SECURITY
FREE MOBILE SECURITY
Attention: Your computer might be sluggish and running slowly!
29 useless programs can be deactivated.
3 system settings issues can be tweaked.
300 MB can be freed up on your hard drive.
Improve performance
Calculator
View Edit Help
Process Explorer - Sysinternals: www.sysinternals.com [WIN-56559VLAO5D\Tavis Ormandy]
avast! Free Antivirus
Avast Web Shield has blocked access to this page becau
SRSL
Click Here
Tavis Ormandy's "HTML in certificate" exploit for Avast



J00ru's font vulnerability (Recon 2015)

If we don't understand how it really works, we can't: parse it, preserve it, tell if corrupted or malicious.

Crafting a file format

File format is not just "data structure"

Protobuf / XML doesn't solve everything.
They're just the high-level layer.

Data structure need to be logical and make sense from a dev perspective.
So at least, use a magic number/signature,
and enforce version numbers, sizes... ;)

Failure is still possible

Office file format is a … filesystem!

You can defragment it!

And it has different kinds of FAT ;)

We're a bunch of developers from IBM, ask us anything!

TECHNOLOGY submitted 1 month ago * by CrazyAboutCode

Hey Reddit! We're a bunch of developers who like to talk to people. So stereotypes be damned. We work at IBM and like to talk about app infrastructure, app delivery and app tool projects (some of our favorite projects:

https://www.reddit.com/r/IAmA/comments/3ilzey/were_a_bunch_of_developers_from_ibm_ask_us/?sort=top

A file format is not just an "algorithm"

Your algorithm is great, but the file format will be the interface between your algorithm and all its users and other applications.

finish your specs! double-check them!

provide test cases!

A file format is a map

Every street should follow the same rules,

Otherwise you must expect many violations.

Wherever there is a 'surprise', bad things happen.

Consistency ^ (Compatibility || Schizophrenia)

"PSD makes inconsistency an art form"

XeePhotoshopLoade ×

https://code.google.com/p/xee/source/browse/XeePhotoshopLoader.m?r=f16763d221dfca6253983824b470adf553a19e06#108

```
107                 off_t nextchunk=[fh offsetInFile]+((chunklen+3)&~3);
108                 // At this point, I'd like to take a moment to speak to you about the Adobe PSD format.
109                 // PSD is not a good format. PSD is not even a bad format. Calling it such would be an
110                 // insult to other bad formats, such as PCX or JPEG. No, PSD is an abysmal format. Having
111                 // worked on this code for several weeks now, my hate for PSD has grown to a raging fire
112                 // that burns with the fierce passion of a million suns.
113                 // If there are two different ways of doing something, PSD will do both, in different
114                 // places. It will then make up three more ways no sane human would think of, and do those
115                 // too. PSD makes inconsistency an art form. Why, for instance, did it suddenly decide
116                 // that *these* particular chunks should be aligned to four bytes, and that this alignement
117                 // should *not* be included in the size? Other chunks in other places are either unaligned,
118                 // or aligned with the alignment included in the size. Here, though, it is not included.
119                 // Either one of these three behaviours would be fine. A sane format would pick one. PSD,
120                 // of course, uses all three, and more.
121                 // Trying to get data out of a PSD file is like trying to find something in the attic of
122                 // your eccentric old uncle who died in a freak freshwater shark attack on his 58th
123                 // birthday. That last detail may not be important for the purposes of the simile, but
124                 // at this point I am spending a lot of time imagining amusing fates for the people
125                 // responsible for this Rube Goldberg of a file format.
126                 // Earlier, I tried to get a hold of the latest specs for the PSD file format. To do this,
127                 // I had to apply to them for permission to apply to them to have them consider sending
128                 // me this sacred tome. This would have involved faxing them a copy of some document or
129                 // other, probably signed in blood. I can only imagine that they make this process so
130                 // difficult because they are intensely ashamed of having created this abomination. I
131                 // was naturally not gullible enough to go through with this procedure, but if I had done
132                 // so, I would have printed out every single page of the spec, and set them all on fire.
133                 // Were it within my power, I would gather every single copy of those specs, and launch
134                 // them on a spaceship directly into the sun.
135                 //
136                 // PSD is not my favourite file format.
137
138                 if(sign!='8BIM') break; // sanity check
```

https://code.google.com/p/xee/source/browse/XeePhotoshopLoader.m?r=f16763d221dfca6253983824b470adf553a19e06#108

Not just specs

A default open implementation?

with test cases for the code, and free-licenced examples cases provided.

Too many 'features from the specs' are never seen in the wild.

Life of a file format

1. define a format (if possible)
2. implement it in your software
3. end :(

if you're lucky:

your software becomes standard

along with its file format. That's all.

Becoming a de-facto standard doesn't require anything: it's your niche market. No official requirements. Just business directions. no "long term plan"

You end up with a standard that was never properly designed or documented in the first place.

Have fun preserving it or making it secure!

I wrote a simple "Hello World" PDF,
that works on every reader.
Yet, it's not 100% standard (only 99%)
That's a bad start :(

Thinking about bundling?

Hint: don't.

int bundle(trust){return trust--;}

Evolution of a format

(divergence)

Evolution

1. Tool X creates bogus file
2. StandardTool adapts silently to support them
3. Now StandardTool goes beyond the specs

Specs are now even more useless.

Ex: ColorTrac scanners, PDF readers

Implementations slowly diverge from the specs

⇒ the specs become theoretical and useless in the wild.

Yet nothing exists to replace them.

Once it's a standard, it's too late to fix it.

Before it's a standard, no one really cares.

And too few people care anyway ;)

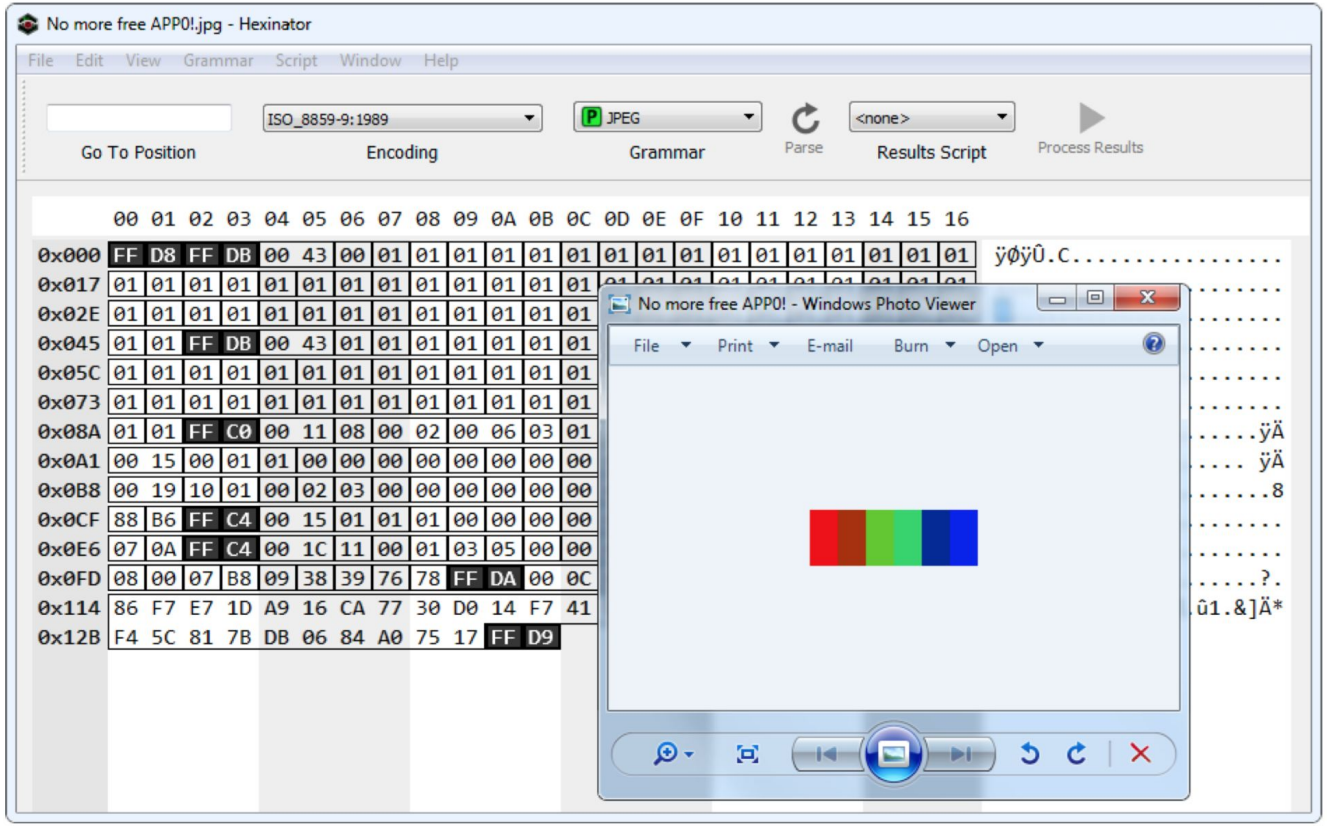
JPEG 1/2

JPEG (1992) is not a file format!

Open source library: LibJPEG → that's great!

LibJPEG goes beyond the specs:

- recovers standard types of App0 chunks
  - including the one specific to Adobe
- unnecessary functions (headless JPEG (!))
  - "let's add this in case" ⇔ design by committee ?



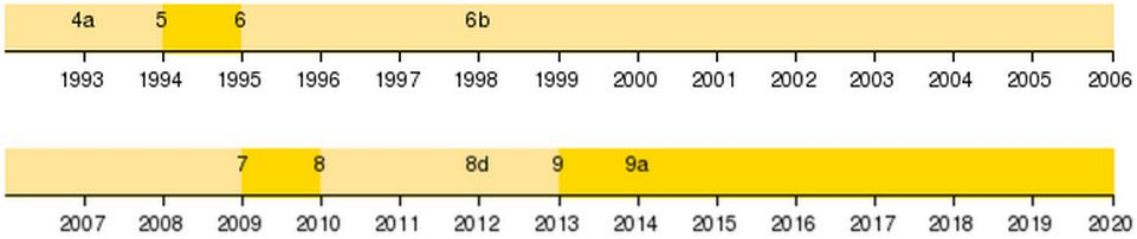
A JPG without a 'required' APPx segment

JPEG 2/2

JPEG is ‘de facto’ libJPEG-turbo v6b.

Explore corner-cases, and then you fail Adobe or Safari:

⇒ their test cases are not big enough



Major problems (so many!)

specs really come last: absent, or TBD

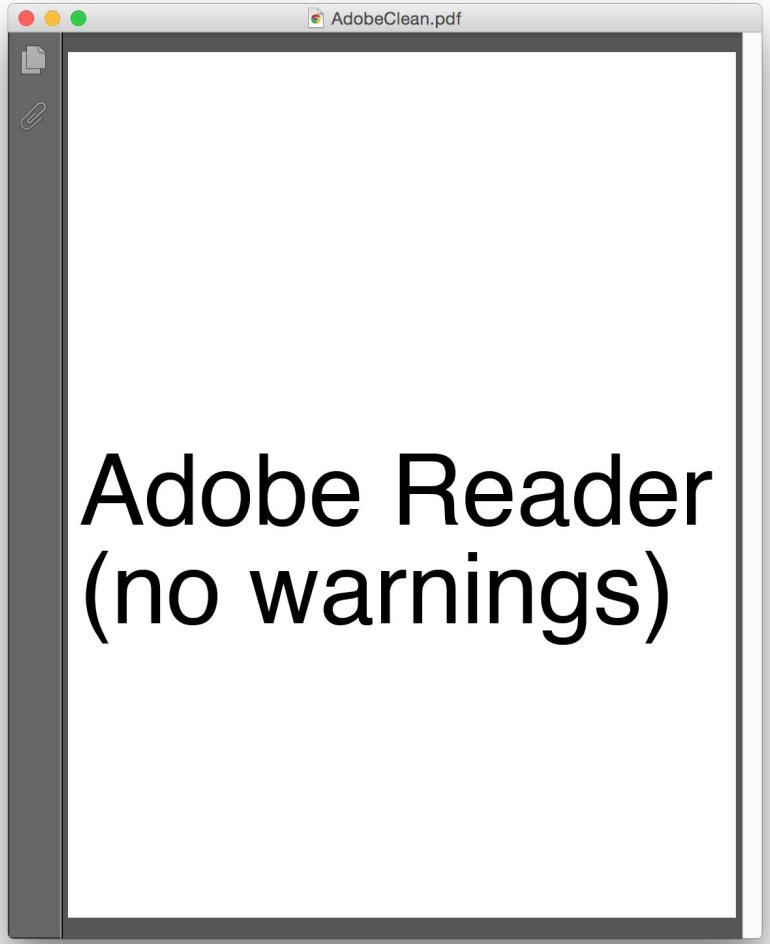
incomplete specs: BPG, ZIP, PDF

incoherent specs: PDF

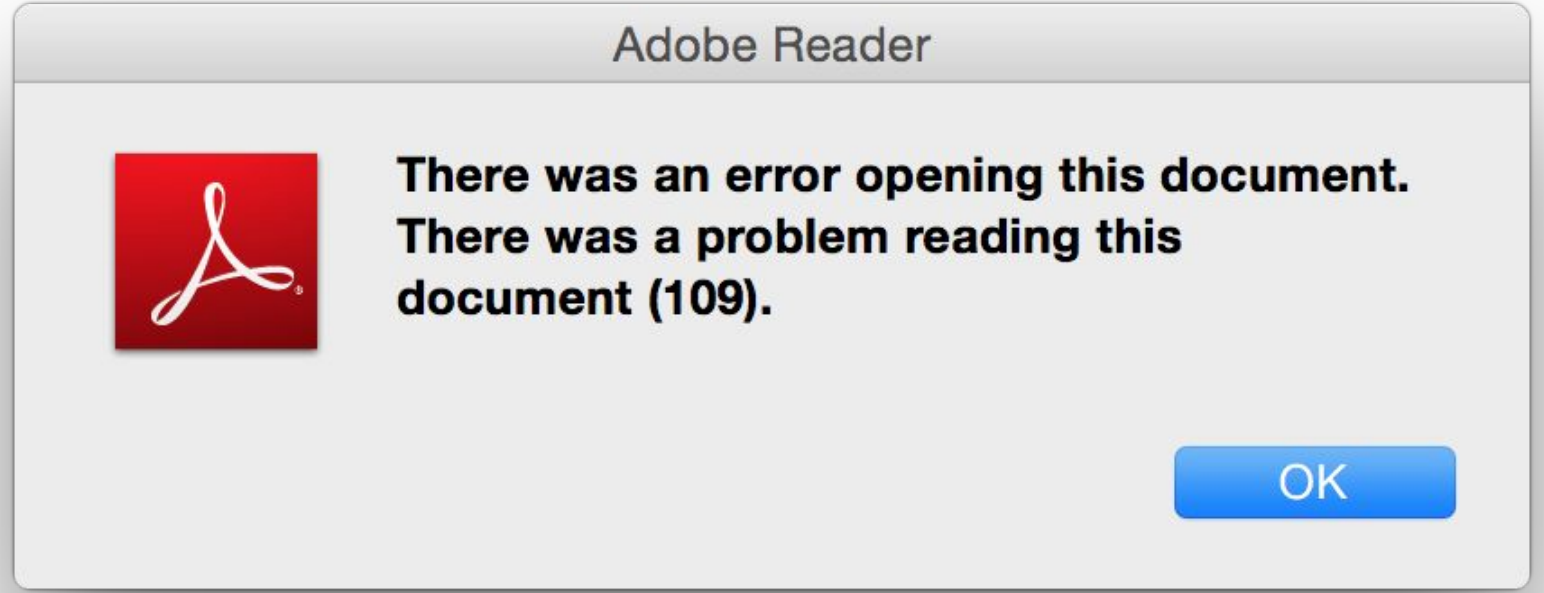
non-free specs.

Recovering broken files

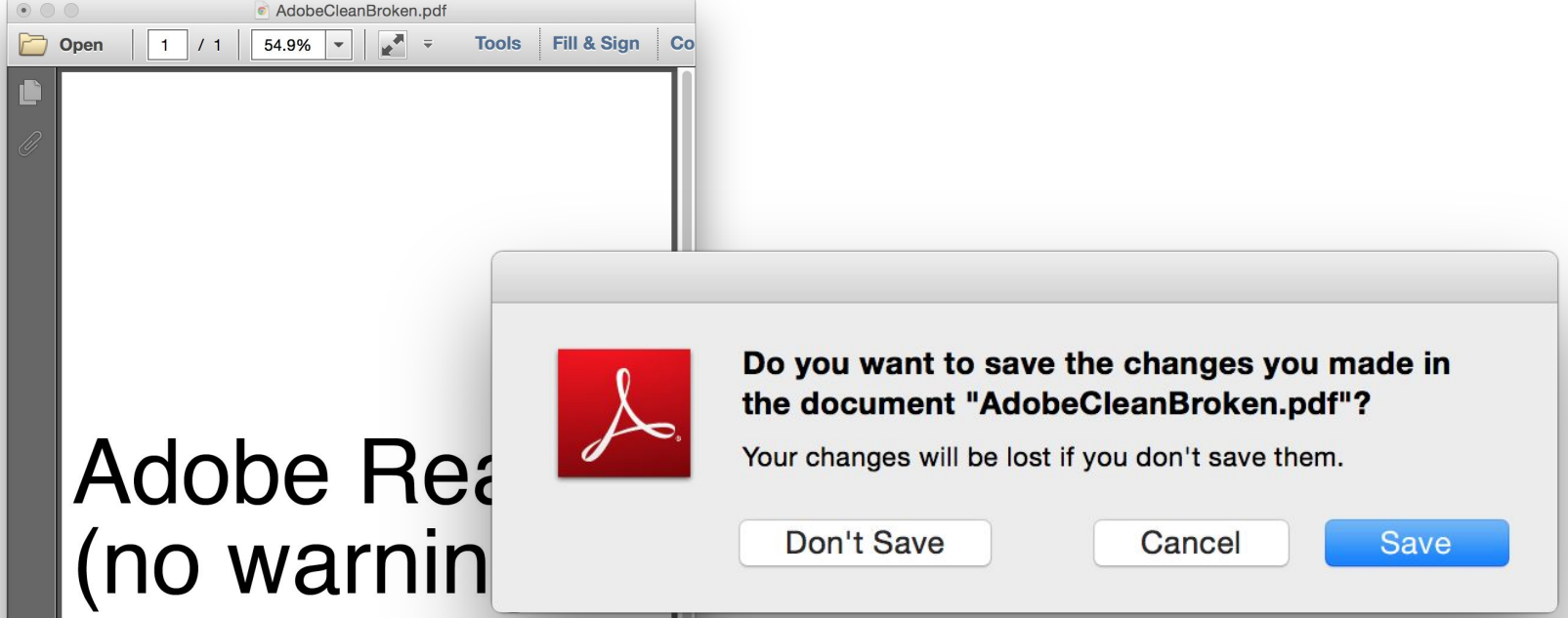
AKA "hidden mode"



Take a fully working PDF.



Change one byte at the wrong place (in the XREF) ⇒ OMG it's corrupted!



But if you remove its XREF entirely, it now miraculously works,
with just a (misleading) dialog on closing, that actually means:
"we found some bugs, do you want to save as a valid but bloated file?"

Standard programs typically embed a (silent) recovery mode.

Nightmare for devs/defenders

These modes try their best to recover "broken" files. Far beyond the specs.

To improve security
and format reliability:
turn auto-recovery into
dialog box warnings?
or reject these files and log the error?
That would make vendors act.
"This file is not correct,
please contact your vendor"...

"helping" the end-user by triggering no warning? (even temporarily) OK

What about identifying bad practices to make them stop eventually?

Forcibly deprecate?

Like crypto? Sounds good, but...

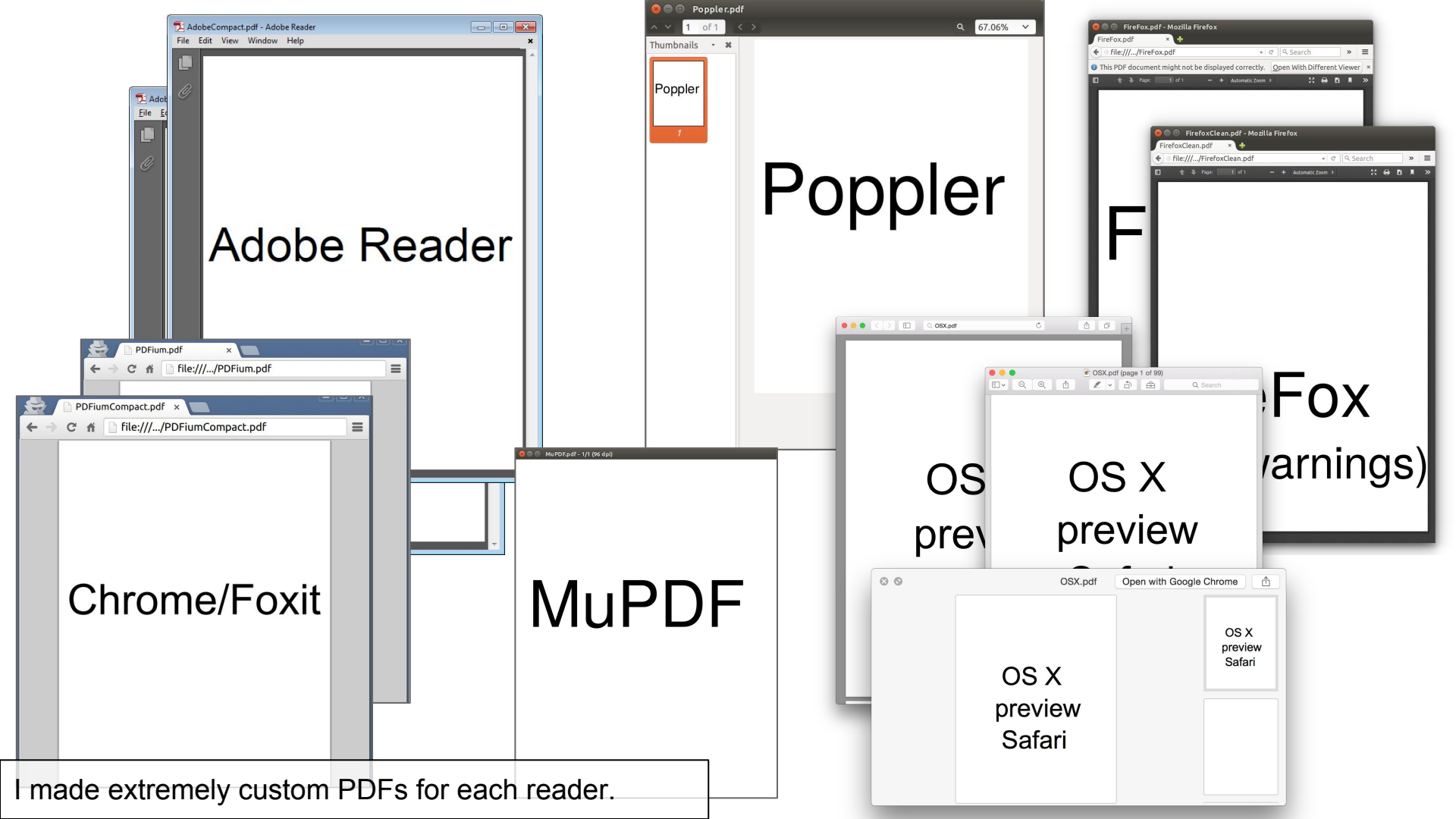
Not going to happen:

Broken crypto leads to fast and mass pwnage.

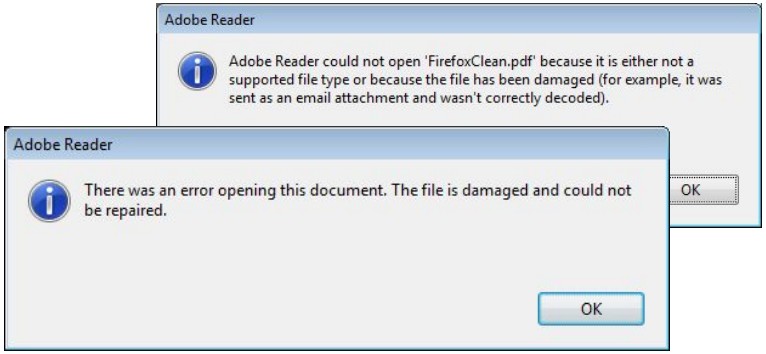
Broken file formats mostly just lead to headache - no incentive to avoid that.

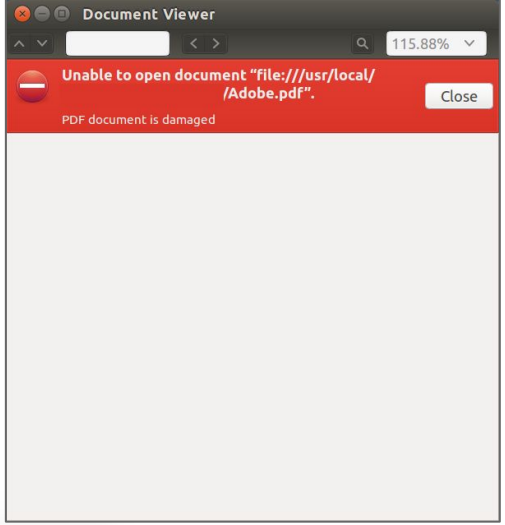
Not enough "Android master key" bugs yet.

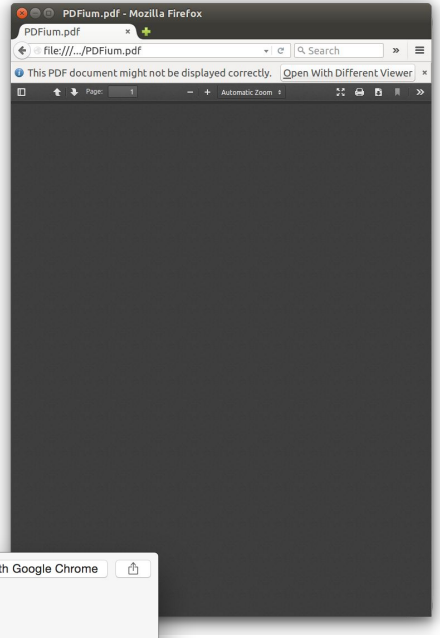
"one" standard ?

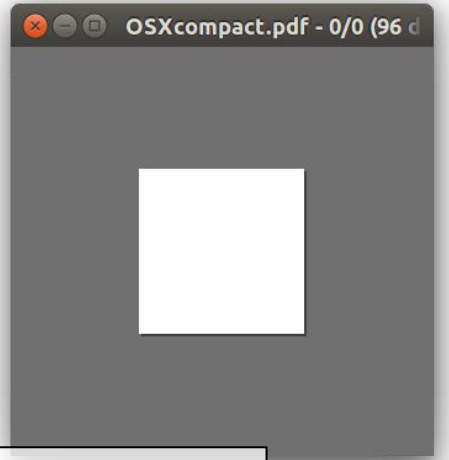


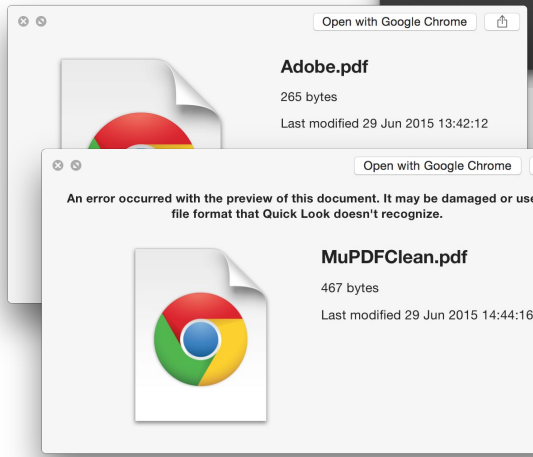
I made extremely custom PDFs for each reader.











These "extreme" PDFs fail on any other reader.

Consequence

We have 6 PDF reader 'standards' in practice:
these may be extreme examples, but
OTOH "Hello World" is not so complex

"Nothing to fix"
"Specs are subject to interpretation"

PDF Schizophrenia?

- Sumatra / Chrome-1 / Others
- Chrome-2 / Others
- Safari / Others
- Poppler / Others

It's not even funny anymore…

⇒ any unclear area may lead to schizophrenia

PDF = portable?

Most readers are okay to read 'standard' docs.
any advanced functions? Adobe Reader
(printing, forms, JavaScript, 3D).
Also, no more Linux version.

PDF, a clean standard?

Non-free specs.

Only the "standard" 1.7 doc is free.

No free examples.

Incomplete + missing specs

no shareable samples

Non-free specs?
No free sample-set?

And you wish to stay a "standard" in 2015?

PDF for archiving?

PDF/A already has 8 sub-standards
Adobe Preflight is not very updated

⇒ Preservation is **not** a business model,
nor a legal requirement of any kind.

How long before "support is discontinued"?

PDF 2.0

No new security stuff, specs are now 170 CHF.

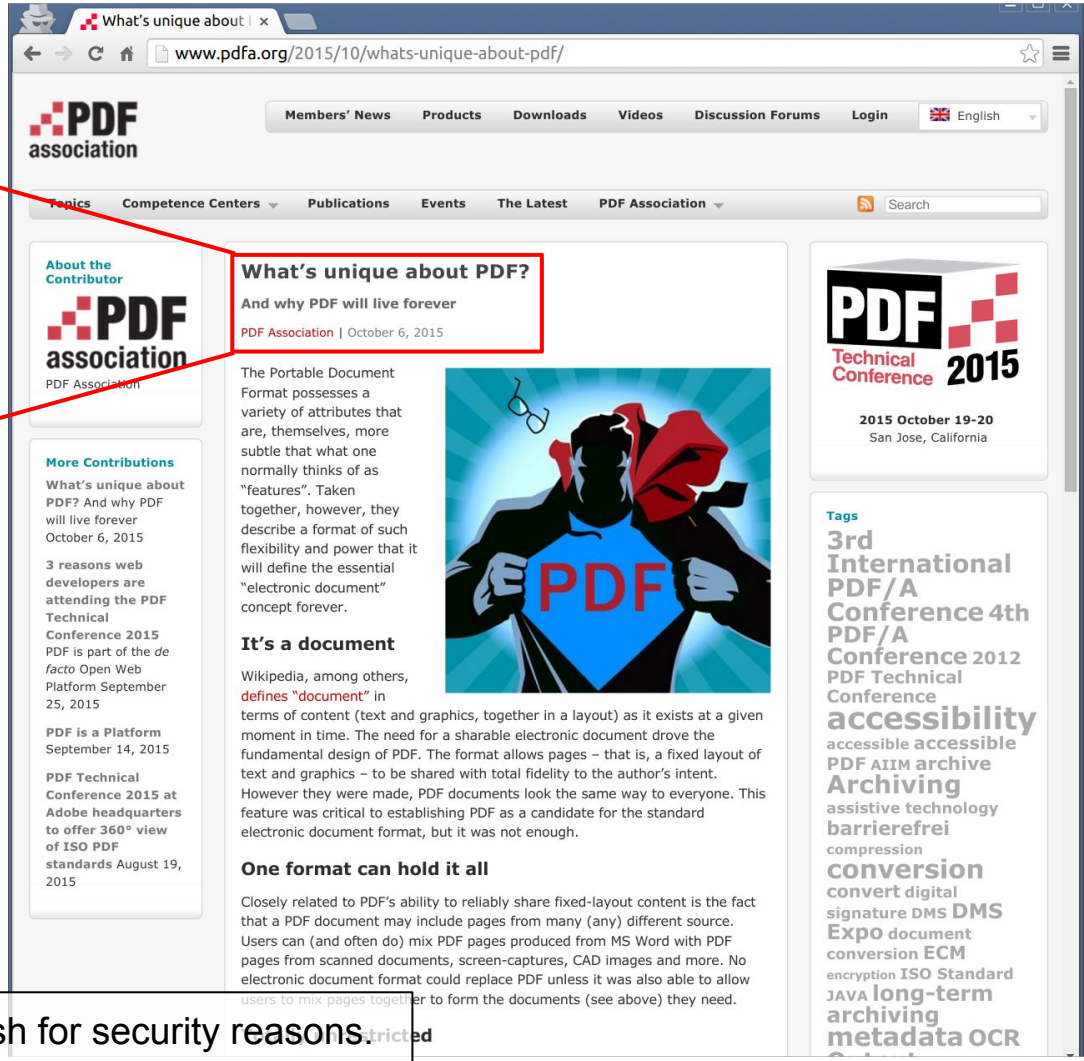
New printing features, new insecure features: embedding files anyone?

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/

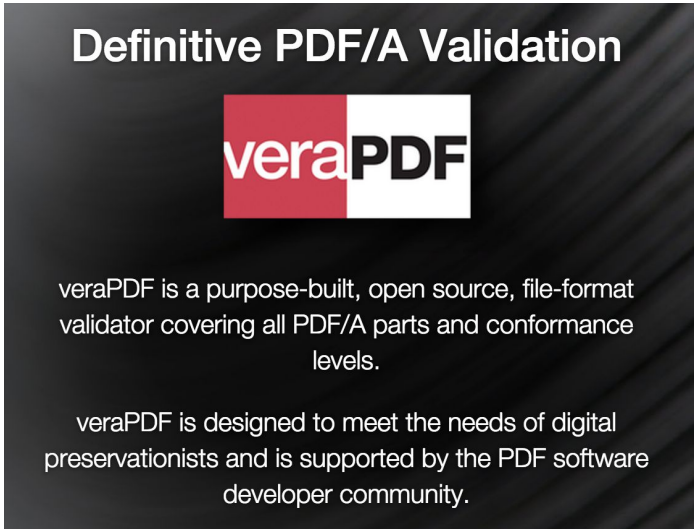


I'm not so sure about it - after all, we're killing Flash for security reasons.

A (tiny) ray of hope

VeraPDF.org:

open source PDF/A validator.



Preservation

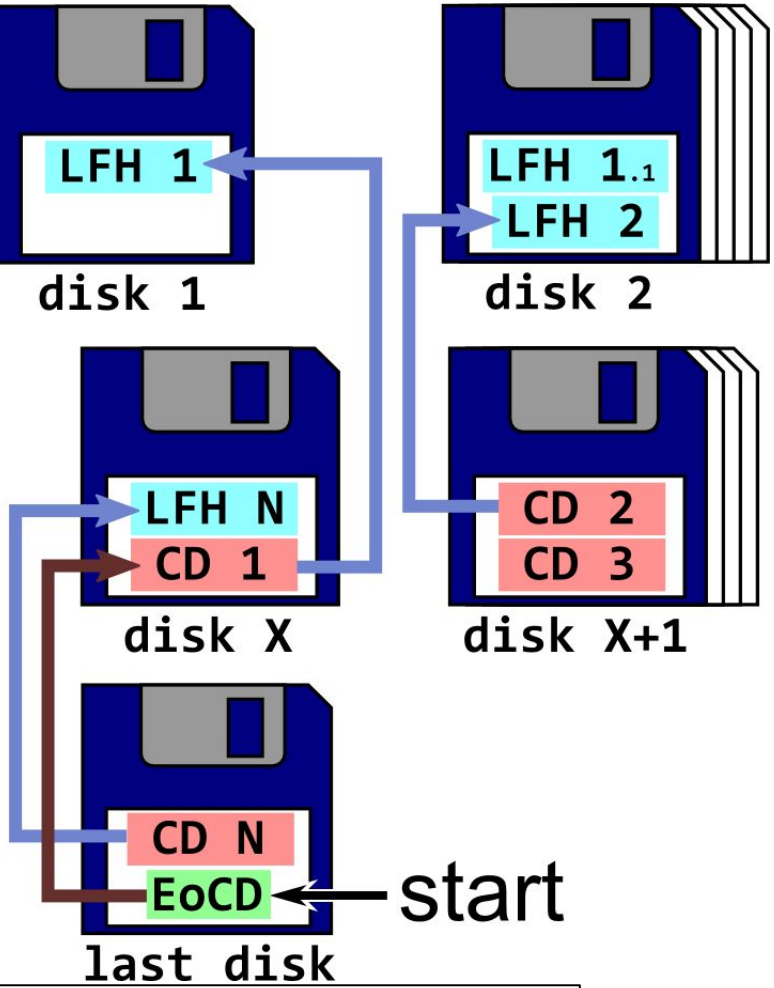
portable compiler + toolchain

portable source

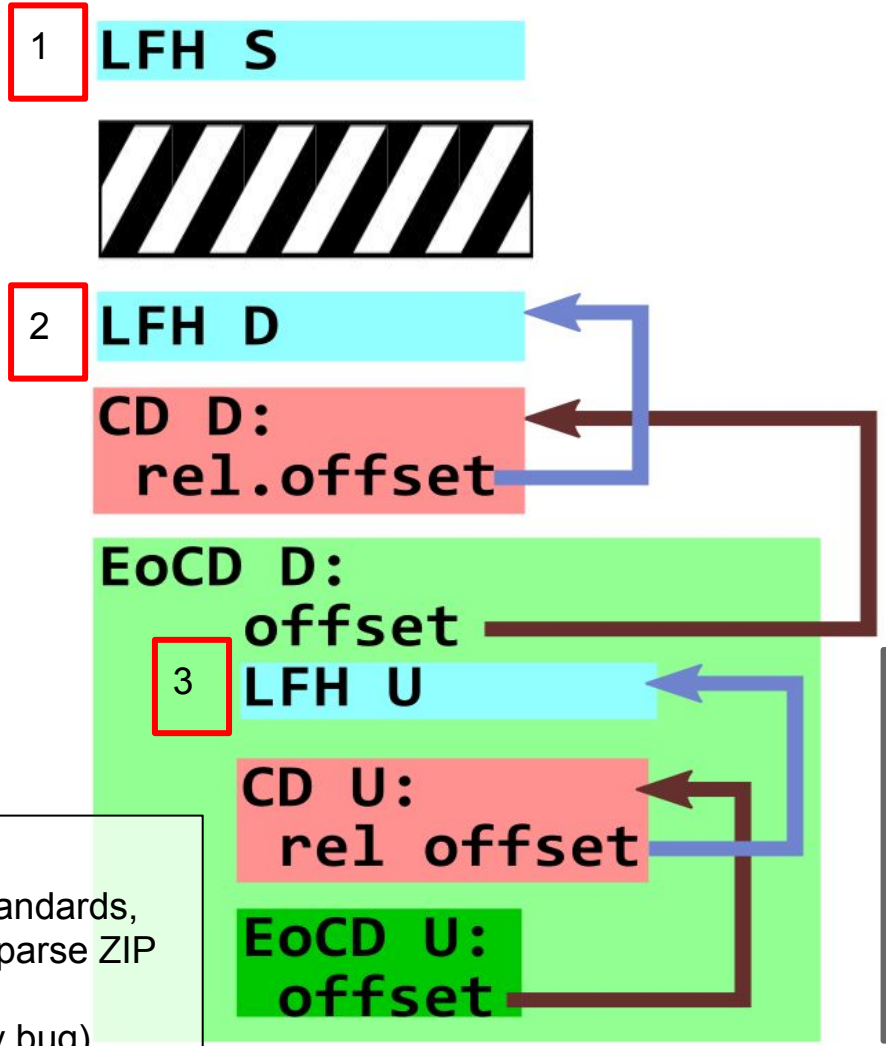
no OS dependency at all ?

preservation via closed-source software?

⇒ "emulation as a service" has a great future :(



ZIP archives already made for multiple floppy support.



Because it's awkward
and suboptimal for modern standards,
there are now 3 ways ITW to parse ZIP
(can be abused
like in the Android Master Key bug)

Do we still need floppy support?

ZIP (1989) is still updated.

ZIP added AES, LZMA, 64 bits, Unicode.

But still this awkward obsolete structure?

Why not just reorder structures, enforcing values, and slowly preventing abuses ?

Not re-inventing the format, just forking it.

Seriously

Do we still really need Tape Archives?

Floppy-oriented, backward-parsed ZIP?

Any generated PDF that doesn't have its magic at offset 0?

FTR:

OpenSSL still supports WinCE, BeOS….

Windows bitmap fonts are stored as 16 bits NE executables (copyright 1989).

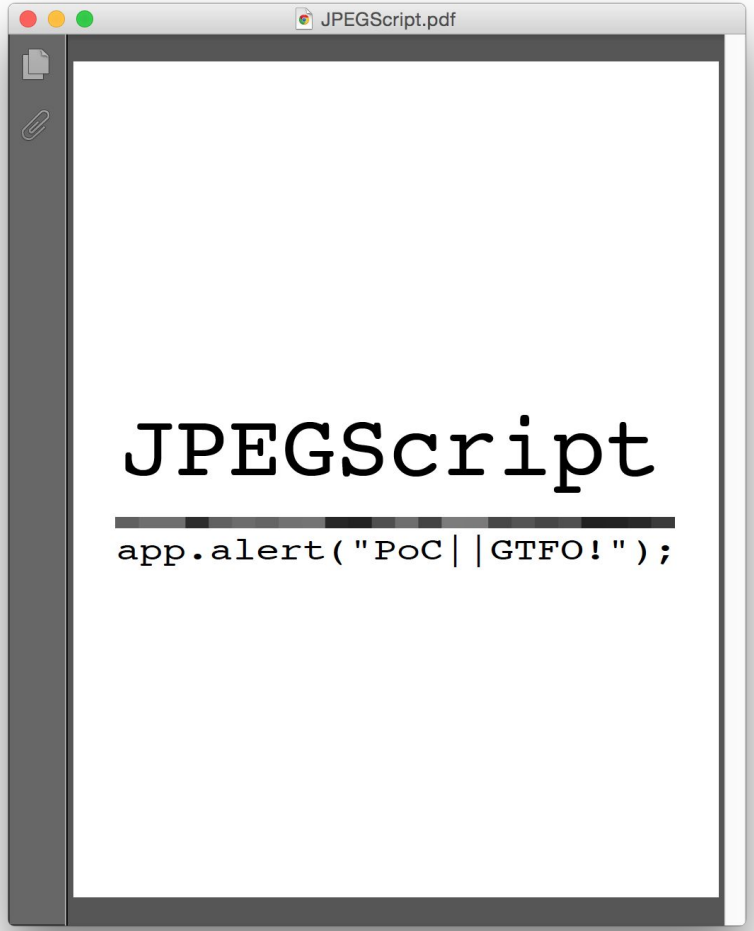
Pure digital preservation

New documents are born digital:

the problem is shifted:

the 'master' copy already depends on:

source+compiler+toolchain+(OS+CPU).



A PDF with a JPEG-compressed script

JPEG, but not an image?

It's not against the specs,

but it was removed without any warning nor tracking.

⇒ breaks backward compatibility

If your document was using it, now it's broken.

If this document is born digital, you lost your source document.

Backward compatibility

Everywhere.

In case, you never know.

The customer is always right.

Perhaps except for security things ;)

Our kids will probably ask us one day
why we kept all these things for so long...

Windows compatibility

Windows is becoming progressively (but silently) more strict for the PE format, slowly killing several packers.

Have you heard anyone complaining?

(the official PE doc still totally sucks though)

breaking backward compatibility

It's ok if it's for valid reasons,

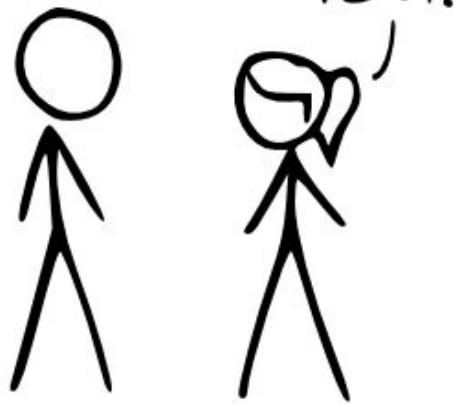
but keep track of changes, enforce version numbers, and update the specs accordingly at the same time!

Nowadays,
a file format is an evolving entity for security reasons,
not something sacred written in stone

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.



SOON:

SITUATION:
THERE ARE
15 COMPETING
STANDARDS

Multiple formats is not the problem: we have different needs.

But documentation never reflect reality in any case.

There are many benefits to know definitively what a valid file is or isn't.

Cleaning up

Terse Executable is a cleaned-up version of the Portable Executable
(but for UEFI, not to replace it).

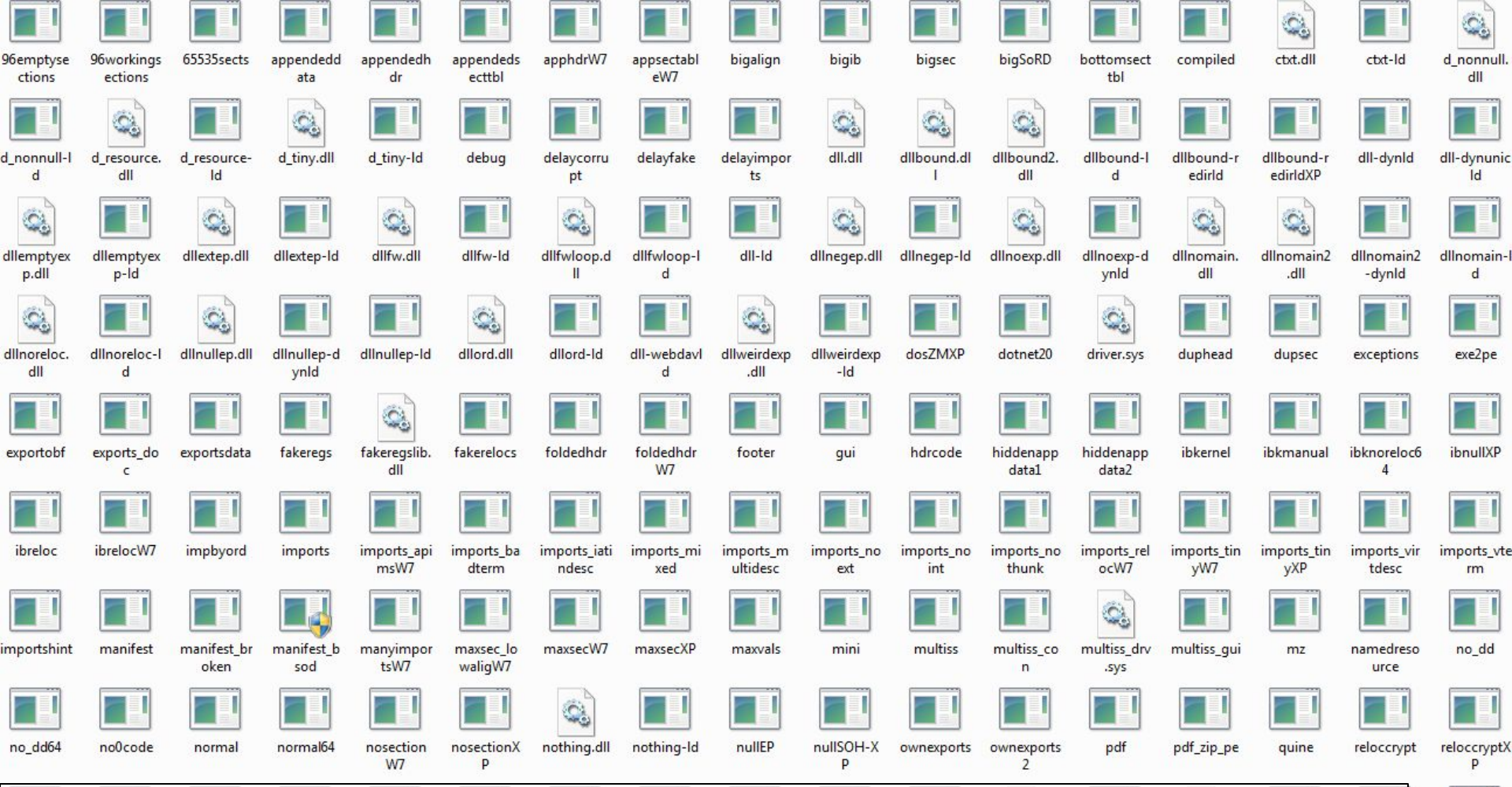
Only example of forking that makes sense?

We just stack features...

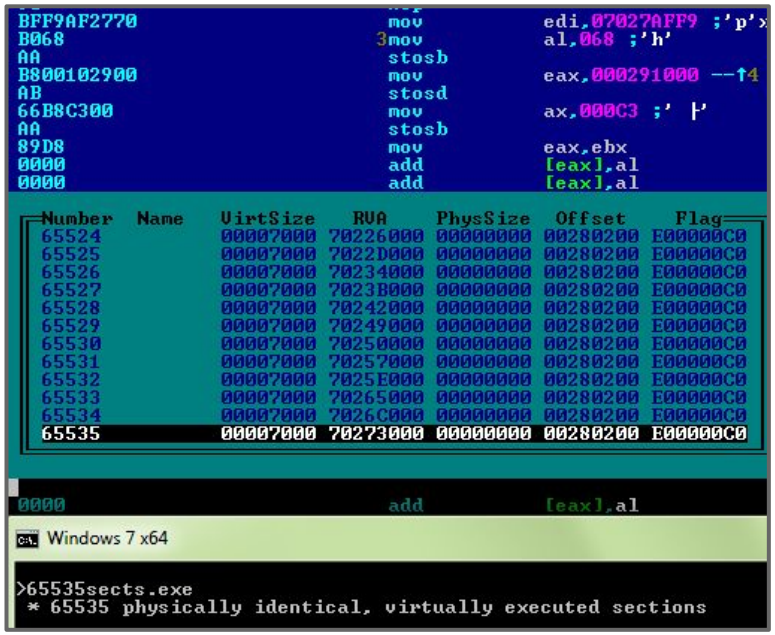
There's no standard for file format specifications

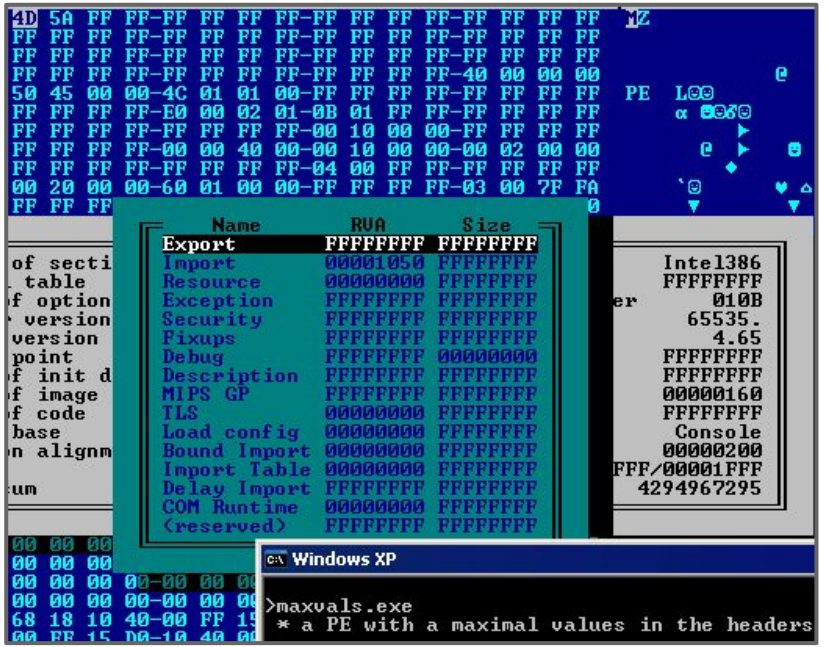
different style of writing, may be incomplete unclear, non free...

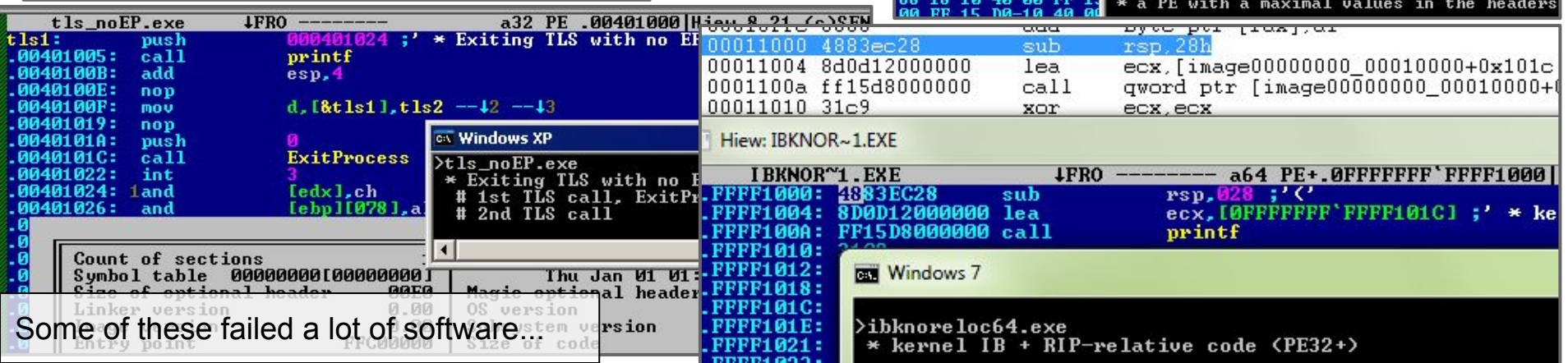
Something I tried



PE.corkami.com: my own collection of hand-made executables and "documentation" (completely free).







Some of these failed a lot of software...

Consequence?

- 'corkami-proof' software
- raises the bar for everyone
- become a hub of knowledge
  - "I can't share the sample", but from the knowledge, my own file will be shared
    ⇒ even useful for the original contact

Conclusion

We're 

-ed

We probably have to witness the burning of a *digital* "Library of Alexandria" before we change anything.

(because money)

No matter the kind of format, we can't trust files: "specifications" ?

more like *gentle introductions*!

Or maybe something like
religious texts (with philosophical suggestions)
not accurate descriptions of reality.

Many more file abuses will come!
It doesn't get you any bug bounty,
but plenty of new classes
of abuse to discover:
compression, network, cryptography,
file systems...

Rules of thumb

- abuse your own format
  - double-check your specs -- with a twisted mind!
- open-source, unit-tested library
- consistency, technical common sense
- stop stacking features!

How you can help?

test-case binaries

- share your testing suite
- fuzzing results (seen from code coverage)

⇒ raises the bar for all industries

A format evolves

- deprecate!
- enforce version numbers
- make it public

we can set open ultimatum for crypto,
we should do the same for bad files.

Ack

Phil Paul Arindam Jacob Alex Christophe Travis Tavis Sergey Kurt Gabor Miki Gyn Mat Bart Max

…

Thank you!

Corkami: 10 years!
time to evolve !

More PoCs, posters, book(s)...
+ some side projects

⇒ no more [personal] presentations for now

FAQ: "do you have any recommended PDF reader"

Only Adobe Reader handles complex documents and functionalities.

Other are more or less equivalent.

Not a very satisfying answer, I know ;)

@angealbertini

corkami.com