

When your slides read themselves a binary inception

Ange Albertini, September 2013
(written in April 2014)



disclaimer

this technique was already used in my [presentation](#) at 44con, however I didn't give the full details at the time.

And since Adobe blacklists PDF starting with PEs signature, it can't be published in PoC||GTFO.

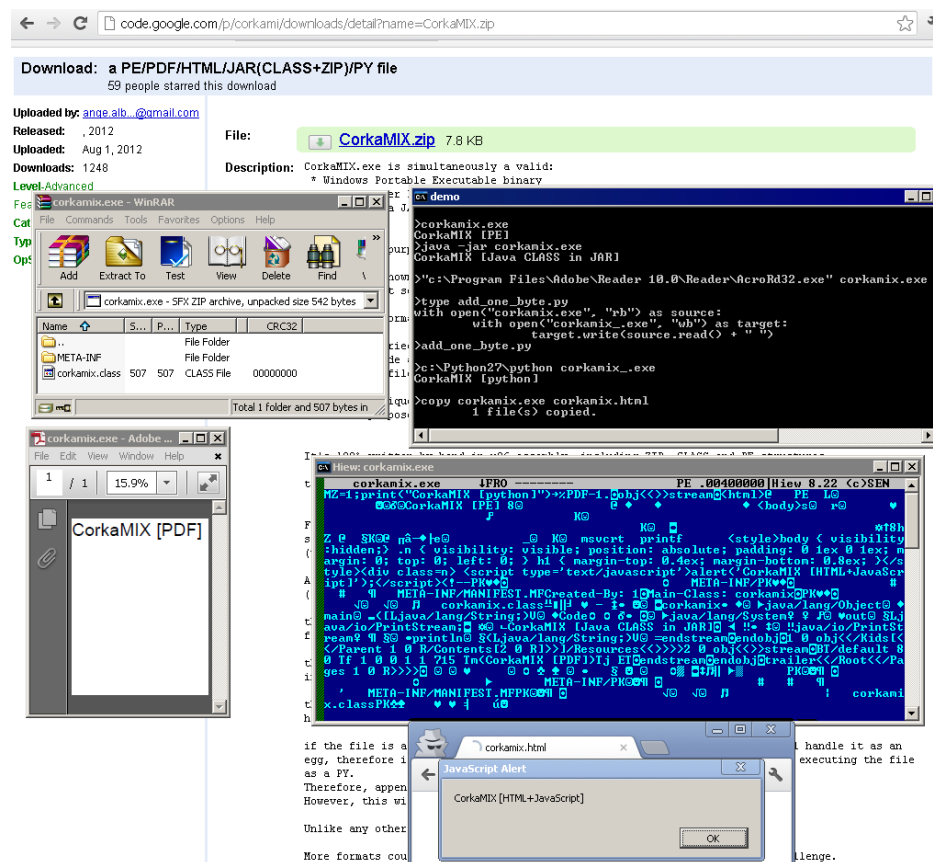
so here are the details, before I forget...

**Messing with
binary formats**



Before

I crafted several funky PoCs in the past, such as a PDF + PE + ... (in ASM)



The image shows a sequence of screenshots demonstrating the execution of a PoC (Proof of Concept) file named CorkaMIX.zip. The browser window displays the download page with the file name "CorkaMIX.zip" and a size of 7.8 KB. The description states: "CorkaMIX.exe is simultaneously a valid: * Windows Portable Executable binary".

The terminal window shows the following commands and output:

```
demo
>corkamix.exe
CorkaMIX [PE]
>java -jar corkamix.exe
CorkaMIX [Java CLASS in JAR]
>"c:\Program Files\Adobe\Reader 10.0\Reader\AcroRd32.exe" corkamix.exe
>type add_one_byte.py
with open("corkamix.exe", "rb") as source:
    with open("corkamix_-.exe", "wb") as target:
        target.write(source.read())
>add_one_byte.py
>"c:\Python27\python corkamix_-.exe
CorkaMIX [python]
>copy corkamix.exe corkamix.html
1 file(s) copied.
```

The terminal window also shows the execution of the file, which results in a JavaScript alert dialog box with the message "CorkaMIX [HTML+JavaScript]".

The PDF viewer window shows the document "CorkaMIX [PDF]".

The JavaScript alert dialog box is titled "JavaScript Alert" and contains the message "CorkaMIX [HTML+JavaScript]".

the idea

As soon as it's hand-made, one may not expect it could work in any case.

What would be a perfect demo?

What would convince the most the audience that it really works?

What has the audience seen until the moment of your demo?

just yourself and your slides

slides == demo

so let's make the demo with the slides themselves.

So, by the time you're announcing the demo, you can say:

actually, we've been in the demo all along.

the slides are the demo

⇒ inception :)

the slides are the demo ?

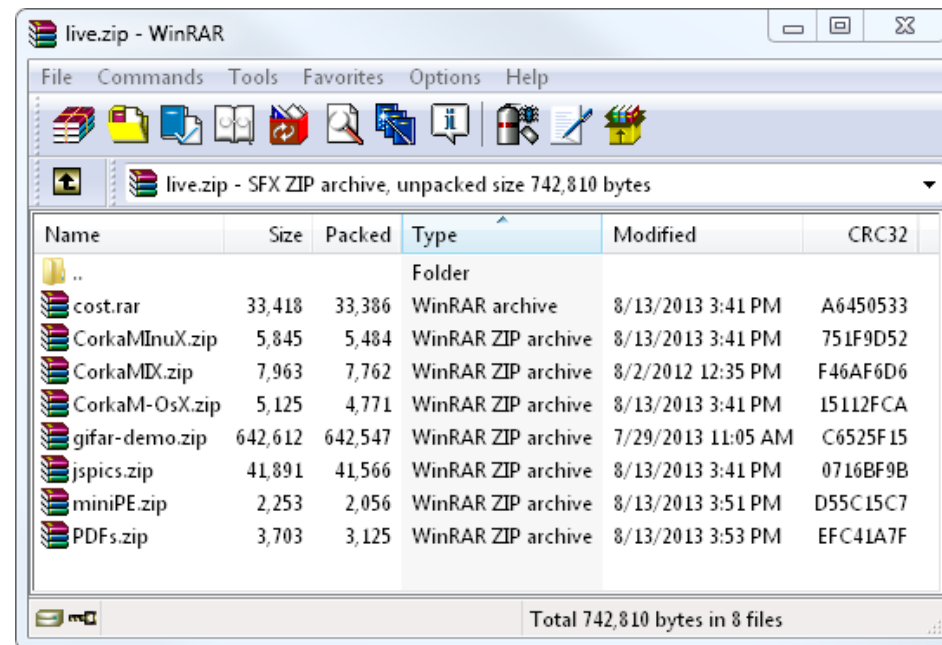
the slides are typically in PDF (YMMV)

So let's merge a genuine PDF slide deck and a genuine PDF viewer

is that all?

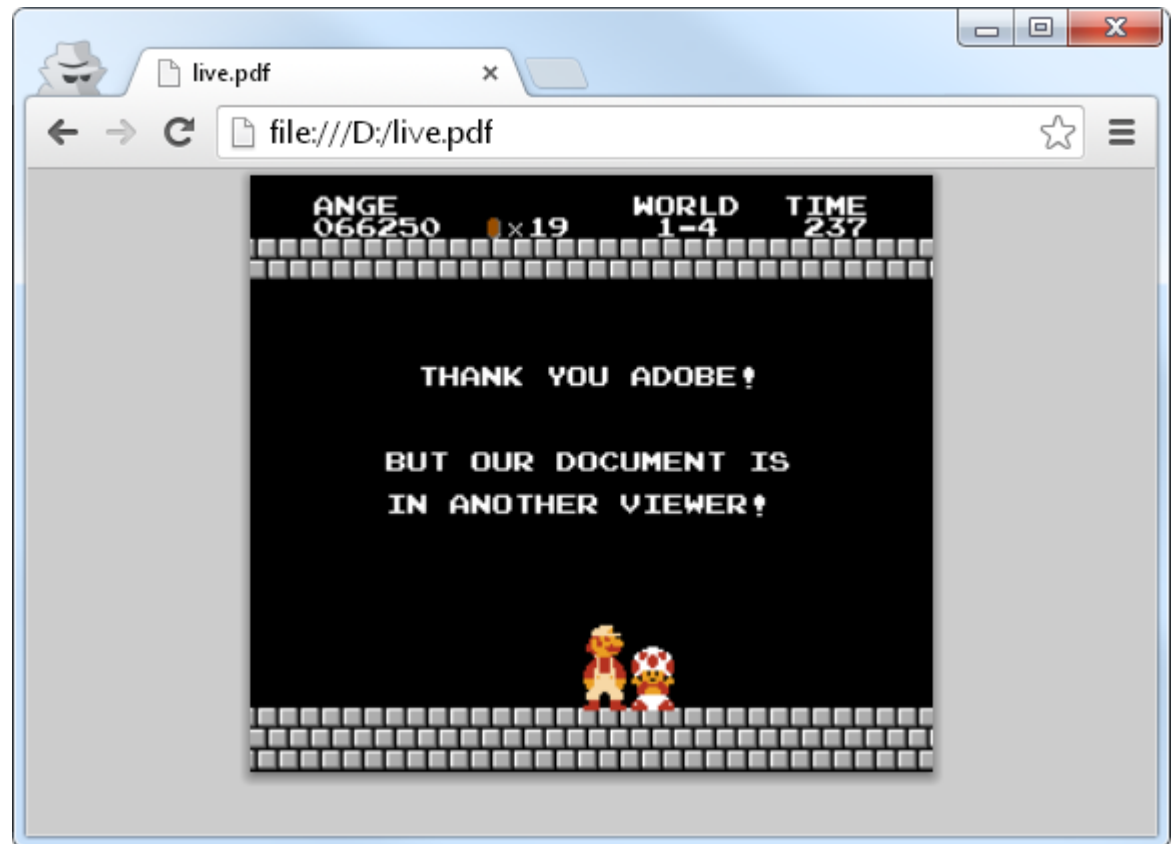
let's also make it:

- a ZIP
 - to bundle the PoCs
 - not detailed here - see my [44con slides](#) for details

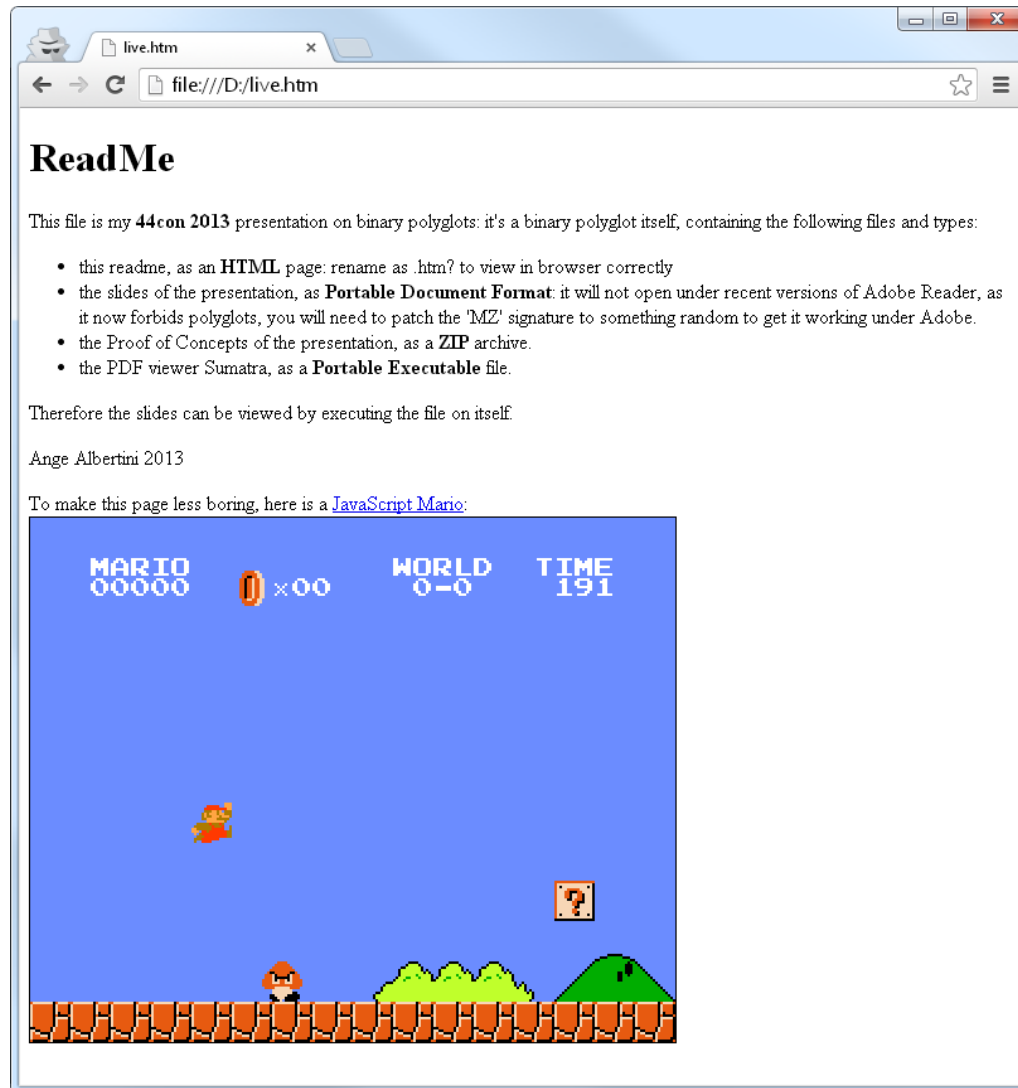


...and an alternate PDF... (when opened with Chrome)

(not detailed here)



...and also an HTML webpage



a PDF viewer in single PE?

Sumatra

- single executable
- no installation required
- lightweight

Perfect!

Merging PE & PDF

- not covered here, see slides for the general case
- however a couple of extra problems had to be solved

Problem 1

- PE/PDF payload before & after the HTML page

Solution:

- surround with comments to hide most stuff

```
<!-- garbage -->
```

```
  <html>
```

```
  ...
```

```
  </html>
```

```
<!-- garbage
```

Problem 1.5

- file has to start with MZ
 - can't be hidden via comment

Solution:

- CSS to the rescue:
 - define body hidden by default
 - trick from [lcamtuf](#)

```
MZ<!--  
...  
--><html>  
<body onload="Mario(true, 2);">  
<style>  
body { visibility: hidden; }  
.n {  
    visibility: visible;  
...  
}</style>  
<div class='n'>  
<h1>ReadMe</h1>  
...
```

Problem 2

- a compressed PDF data might accidentally contains “-->”
 - which would kill your HTML part

Solution:

- apply an ASCIIHexDecode filter on each binary stream of your PDF
 - Guillaume Delugré's Origami will handle that magically for us

Problem 2.5

Warning:

- forcing any filter blindly will break JPEG images: they require DCTDecode filter

Origami script

```
begin
  require 'origami'
rescue LoadError
  ORIGAMIDIR = "#{File.dirname(__FILE__)}/../../lib"
  $: << ORIGAMIDIR
  require 'origami'
end
include Origami

pdf = PDF.read "doc.pdf"

pdf.root_objects.find_all{|o| o.is_a? Stream}.each {|s|

  # decode stream
  decoded = s.data

  # add a final filter
  s.Filter = [s.Filter || []].flatten.unshift(:ASCIIHexDecode) # or ASCII85Decode

  # force the stream to be re-encoded
  s.data = decoded
}

pdf.save "docASCII.pdf"
```

Problem 3

- Sumatra is a PDF viewer
 - it contains PDF keywords
 - which interferes with PDF parsing ;)

⇒ use a packer

but compressed data might contain "-->"

⇒ same problem again!

⇒ keep trying various packers/algo until it doesn't :D

UPX with LZMA eventually worked

Problem 4

- Sumatra has a Manifest
 - it's XML and still present once packed
 - contains a --> comment
 - removing the Manifest entirely doesn't work well :)

just remove (only) the comment in the Manifest

Messing with binary formats




Ange Albertini
2013/09/13

London, England



live.exe:5124 Properties

TCP/IP	Security	Environment
Image	Performance	Performance Gi
Image File		
	SumatraPDF (Unable to verify) Krzysztof Kowalczyk	
Version:	2.2.1.0	
Time:	8/26/2013 5:12 PM	
Path (Image is probably packed):		
d:\live.exe		
Command line:		
d:\live.exe d:\live.exe		
Current directory:		
d:\		

Victoly !

Conclusion

- completely useless? :D
- really works on any PDF

the most important:

a convinced audience

ACK

Guillaume Delugré, Icamtuf, Jonas Magazinius...

Questions/suggestions?

@angealbertini

Want more?

read PoC||GTFO !

