# **Files and Filetypes**

**REVERSE ENGINEERING** 

departamento de eletrónica, telecomunicações e informática

João Paulo Barraca

#### **Files**

- Files are containers that are parsed according to a schema
  - Parsing implies knowing the file content

- How to select the adequate parser?
  - Using the file extension
  - Using magic headers
  - Using rules provided by configuration
  - Previous knowledge
- What if the parser is wrong?

#### **File extensions**

• File extensions are words appended to the filename, after a dot

#### lecture.pptx

- File extensions are a basic mechanism to know how to handle a file
  - Operating systems uses extensions to select the correct process
  - Applications use it to filter which files are adequate (.e.g images). Mostly an usability aspect
  - Humans use extensions to differentiate files
- Popular file extensions:
  - zip, rar, bz2, gz, 7z: compressed files
  - exe, dll, so, com: executable files
  - jpg, tiff, bmp, fits, png: images

#### **File extensions**

- Knowing the file extension is important to apply the correct analysis process
  - Analyzing a JPG is different from analyzing an EXE, or even a PNG



#### **File extensions**

#### **Extensions are misleading!**

- Windows hides extension of known file types
  - Sample.pptx becomes only Sample
- Executable files may have an embedded icon
  - Freely defined by the developer
  - Explorer will show that icon
- A file named Sample.pptx.exe will be shown as Sample.pptx
  - Users recognize the extension and may think the file is safe
- In a RE task, consider that a file may have bogus extensions



#### Also known as Magic Bytes/Header

- Most files can also be recognized by a magic value in the file start/end
  - Manipulating headers can lead to incorrect detection and maybe processing
  - Some OS use the magic headers instead of the file extension
  - Also known as File Signatures
- Some magic values:
  - Office Documents: D0 CF 11 E0
  - ELF: 7F E L F
  - JPG: FF D8
  - PNG: 89 P N G 0D 0A 1A 0A
  - Java class: CA FE BA BE

Sometimes, magic headers are reused

• PK.. (50 4B 03 04) is the magic for ZIP files

#### —\$ file 8\ -\ Obfuscation.pptx ; - Obfuscation.pptx: Microsoft PowerPoint 2007+

#### └─\$ file sample.zip sample.zip: Zip arcl

sample.zip: Zip archive data, at least v2.0 to extract

PKMPXcl*a.txtUTe
eux[.r.Hr}^.EH.H.~
%((.G=
.3B.QR=.J.E.&d.U}<^~.
\kti45}>rD.n.(.p
F!i}.>4~QI.:P9in/
8J\$*h9'tmzw{{?OT\$.Oz*%
D.h.&AK.yj. .'D.oiY%\$M
OQZ0.}A~i,N.+bV.)+{O <qv< td=""></qv<>
*qRD1}.I.q2Hk.s
< WvtN.x)"tDK/)M
.X z.[no".rQ. %SM
x#x^hz.t)rHX.R.
%.w@^.]%\$1.IIiZyqwE?.dH .V
~.{ S8@*.+co^P.>a#ZD&e
.kh>~.e&.{fiQ.Y@.,M=W
7.`WVZ2 <q}:}9]j\$1z< td=""></q}:}9]j\$1z<>
.o]bR]e?NEZ.\j7y

Sometimes, magic headers are reused

• Actually, pptx are zip files

\$ unzip -	-1 8\ -\ Obfu	uscation	.pptx
Archive:	8 - Obfuscat	tion.ppt>	K
Length	Date	Time	Name
5179	1980-01-01	00:00	<pre>ppt/presentation.xml</pre>
12041	1980-01-01	00:00	customXml/item1.xml
1203	1980-01-01	00:00	customXml/itemProps1.xml
219	1980-01-01	00:00	customXml/item2.xml
335	1980-01-01	00:00	customXml/itemProps2.xml
394	1980-01-01	00:00	customXml/item3.xml
606	1980-01-01	00:00	customXml/itemProps3.xml
33895	1980-01-01	00:00	<pre>ppt/slideMasters/slideMaster1.xml</pre>
2477	1980-01-01	00:00	<pre>ppt/slides/slide1.xml</pre>
4665	1980-01-01	00:00	<pre>ppt/slides/slide2.xml</pre>
4384	1980-01-01	00:00	<pre>ppt/slides/slide3.xml</pre>
4003	1980-01-01	00:00	<pre>ppt/slides/slide4.xml</pre>
4719	1980-01-01	00:00	<pre>ppt/slides/slide5.xml</pre>

PKppt/presentati
on.xmln.8;@PJt"T'.
t.t\$CS(z.wxw>k.> E
WV2%/.w0~I5.SaZ.`K
.E.~x7].[aRr`?Tq.%a3
wQ6>.K)Z.;.`%.^MpZ)
u.7B^r.cDS*v.B.Qm87\$z
w1[.kr4?0)l\!
#'pv).Qrpu=nC{uR.u
N0.]z>k~x]~i.uaa
G\~~d5.fKf
.Y/Rr/?r.8u?.A
.G"k}AV ~G":.Hu:~\$
+^.:obRK?L.1.MiM#
JO.J.g.Z>.752q<.^.tCj

#### Magic Headers can be manipulated if the content is known

- PyInstaller allows conversion of a Python application to an executable application
  - It packs the pyc files into a container. Container is extracted on runtime and compiled python code is executed
  - Headers are omitted from pyc files. If header is added, extracted file executes as a standard pyc file

		0	-	4	5		0		0	) A	вс	0120400	10 JABC			U 1	4 .		5 0	/ 0	, ,	AD	0	0123430708ABC		
	00000000	E 3	0.0	00 0	0 0	0 0	0 0 0	00	00 0	00 00	00 00	<u>.</u>		^	00000000	<u>55</u> 01	) OD 0	<b>A</b> 01	00 00	00 0	0 CD	D2 B9	9A	<u>u</u>	^	
	0000000D	0.0	0.0	00 0	0 00	6 0	0 0 0	00	40 0	00 00	00 73		.@s		000000D	DD <b>7</b> 3	FC E	<b>3</b> 00	00 00	00 0	0 0 0	00 00	0.0	. s		
	0000001A	02	01	00 0	00 6	4 0	0 64	01	6C (	00 5A	00 64	d.d.	.l.Z.d		0000001A	00 00	00 0	0 0 0	00 00	06 0	0 0 0	00 40	0.0	· · · · · · · · · · · · · · · · · @ · ·		
	00000027	0.0	64	02 🤅	6C 0	1 6	<b>D</b> 02	5A	02 0	01 00	64 00	.d.l.m.2	Zd.		00000027	00 00	73 0	2 01	00 00	64 0	0 64	01 6C	0.0	sd.d.l.		
	00000034	64	03	6C (	03 6	D 0	<b>4</b> 5A	04	01 0	0 65	00 A0	d.l.m.Z.	e		00000034	5A 00	64 0	0 64	02 6C	01 6	D 02	5A 02	01	Z.d.d.l.m.Z		
	00000041	0.0	A1	00 5	5 <b>a</b> 0	5 6	5 0.5	5 A0	06 6	55 00	6A 07	Z.e	e.j.		00000041	00 64	00 6	4 03	6C 03	6D 0	4 5A	04 01	0.0	.d.d.l.m.Z		
	0000004E	65	00	6A (	08 6	4 0	<b>4</b> A1	03	01 0	0 65	05 A0	e.j.d	e		0000004E	65 00	AO 0	0 A1	00 5A	05 6	5 05	A0 06	65	eZ.ee		
	0000005B	09	64	05 Z	A1 0	1 0	1 00	65	05 <b>Z</b>	<b>10</b> 0A	64 06	.de	əd.		0000005B	00 <b>6</b> 7	07 6	5 00	6A 08	64 0	4 A1	03 01	0.0	.j.e.j.d		
	00000068	A1	01	01 0	00 6	5 0	<b>5</b> A0	) 0B	A1 (	00 5C	02 5A	e	\.z		00000068	65 05	AO 0	9 64	05 A1	01 0	1 00	65 05	A0	ede		
	00000075	00	5A	OD 🤅	65 0	CA	<b>0</b> 0 E	64	07 Z	<b>A1</b> 01	5A 0F	.Z.ed	dZ.		00000075	0A 64	06 A	<b>1</b> 01	01 00	65 0	5 A0	0B A1	0.0	.de		
	00000082	65	10	65 0	)F 8	3 0	1 64	07	6B (	03 72	<b>7A</b> 65	e.ed.	.k.rze		00000082	5C 02	5A 0	C 5A	OD 65	0C A	0 0E	64 07	A1	\.Z.Z.ed		
	0000008F	00	A0	11 Z	A1 0	0 0	1 00	) 71	4E 6	55 OF	A0 12		qNe		0000008F	01 5 <b>A</b>	0F 6	5 10	65 OF	83 0	1 64	07 6B	03	.Z.e.ed.k.		
	0000009C	A1	00	65 0	02 6	4 0	8 83	01	A0 1	L3 A1	00 6B	e.d	k		0000009C	72 <b>7</b> 2	<u>65</u> 0	C AO	<b>11</b> A1	00 0	1 00	71 <b>4</b> E	65	rzeqNe		
	000000A9	03	72	A2 🤅	65 0	CA	.0 14	64	09 Z	<b>\1</b> 01	01 00	.r.ed	d		000000A9	0F A0	) 12 A	1 00	65 02	64 0	8 83	01 A0	13	e.d		
	000000B6	65	0C	A0 1	11 A	.1 0	0 01	00	71 4	<b>E</b> 65	0C A0	e	.qNe		000000B6	A1 00	6B 0	3 72	<b>A2</b> 65	0C A	0 14	64 09	A1	k.r.ed		
	000000C3	0 E	64	04 <b>Z</b>	A1 0	1 5	A 15	65	OC Z	<b>10</b> 0E	65 16	.dZ.e	ee.		000000C3	01 01	. 00 <b>6</b>	5 0C.	A0 11	A1 0	0 01	00 71	4 E	$\dots e \dots q N$		
Extracted	000000000	A0	17	65 1	15 6	4 0;	A Al	02	A1 (	<b>)1</b> 5A	18 65	e.d	Z.e		000000000	65 <b>0</b> 0	: AO O	<b>E</b> 64	04 A1	01 5	A 15	65 OC	A0	edZ.e		Reconstructed
LALIALLEU	000000DD	18	A0	19 6	64 0	BA	<b>1</b> 01	. 73	D2 6	55 OC	A0 11	ds	s.e		000000DD	0E 65	5 16 A	0 17	65 15	64 0	A Al	02 A1	01	.ee.d		
	000000EA	A1	00	01 0	00 7	1 47	E 65	5 18	A0 1	LA 64	<b>0B</b> 64	qNe.	d.d		000000EA	5A 18	65 1	8 A0	19 64	0B A	1 01	73 D2	65	Z.eds.e		
	000000F7	00	A1	02 5	5A 1	.8 6	<b>5</b> 04	65	18 6	5 <b>4</b> OD	64 OE	Z.e.e	e.d.d.		000000F7	OC AC	) 11 A	1 00	01 00	71 4	E 65	18 AO	1A	qNe		
	00000104	8 D	02	5A 1	LB 6	5 0	C AO	) 14	65 1	LB Al	01 01	z.e	.e		00000104	64 OE	64 0	C Al	02 5A	18 6	5 04	65 18	64	d.dZ.e.e.d		
	00000111	0.0	65	OC Z	AO 1	.1 A	1 00	01	00 7	71 4E	64 01	.e	qNd.		00000111	0D 64	0E 8	D 02	5A 1B	65 0	C AO	14 65	1B	.dZ.ee.		
	0000011E	53	00	29 0	DF E	.9 0	0 00	00	00 4	<b>1</b> E 29	01 DA	s.)	N)		0000011E	A1 01	. 01 <b>0</b>	0 65	0C A0	11 A	1 00	01 00	71	q		
	0000012B	03	6D	64 3	35 2	.9 0	1 DA	7 0C	63 6	58 65	63 6B	.md5)	.check		0000012B	4E 64	01 5	3 00	29 OF	E9 0	0 0 0	00 00	4 E	Nd.S.)N		
	00000138	5 F	6F	75 7	747	0 7	5 74	E 9	01 0	00 00	00 29	_output.	)		00000138	29 01	. DA 0	<b>3</b> 6D	64 35	29 0	1 DA	0C 63	68	)md5)ch		
	00000145	02	7A	07 3	30 2	.E 3	0 2E	: 30	2E 3	<b>30</b> 69	51 11	.z.0.0.0	0.0iQ.		00000145	65 63	6B 5	F 6F	75 74	70 7	574	E9 01	0.0	eck_output		
	00000152	0.0	00	E9 (	05 0	0 0	0 00	) E9	20 0	00 00	00 73		s		00000152	00 00	29 0	2 7A	07 30	2E 3	0 2E	30 2E	30	).z.0.0.0.0		
	0000015F	0 E	00	00 0	00 7	3 3	4 76	5 33	5F 7	74 68	33 5F	s4v3	3_th3_		0000015F	69 51	. 11 0	0 00	E9 05	00 0	0 0 0	E9 20	0.0	iQ		
	0000016C	77	30	72 e	6C 6	,4 7	3 07	00	00 0	00 49	6E 76	w0rlds	Inv		0000016C	00 00	73 0	<b>E</b> 00	00 00	73 3	4 76	33 <b>5</b> F	74	ss4v3_t		
	00000179	61	6C	69 <b>6</b>	64 D	)A 0	6 6C	69	74 7	7 <b>4</b> 60	65 73	alidli	ittles		00000179	68 33	5F 7	<b>7</b> 30	72 6C	64 7	3 07	00 00	00	h3_w0rlds		
	00000186	0.8	00	00 0	00 6	3 6	F 6D	) 6D	61 6	5E 64	<b>3</b> A F3	comm	mand:.		00000186	49 <b>6</b> E	76 6	1 6C	69 64	DA 0	6 6C	69 74	74	Invalidlitt		
	00000193	0.0	00	00 0	00 5	4 2	9 01	DA	05	73 68	<b>65</b> 60	T)	shel		00000193	6C 65	73 0	8 00	00 00	63 6	F 6D	6D 61	6E	lescomman		
João Paulo	000001A0	6C	29	1C I	DA O	6 7	3 6F	63	6B <b>6</b>	55 74	DA 07	1)soc	cket		000001A0	64 <b>3</b> A	F3 0	0 00	00 00	54 2	9 01	DA 05	73	d:T)s		NGINEERING 9
	000001AD	68	61	73 6	58 6	C 6	9 62	2 72	02 (	00 00	00 DA	hashlibi	r	*	000001AD	68 65	60 6	C 29	1C DA	06 7	3 6F	63 6B	65	hell)socke	×	
			/																							

Added header

#### Magic Headers can be manipulated if the content is known

- Direct Visualization may help
  - Direct byte visualization, Mapping to an image, Entropy Analysis, Tuples

shell32.dll	Network traffic	Compressed data

Network traffic

#### Compressed data

Greg Conti, Sergei Bratus, "Voyage of the Reverser A Visual Study of Binary Species"

#### **Content Type Obfuscation**

Polyglots

A file that has different types simultaneously, which may bypass filters and avoid security counter-measures.

pocorgtfo19.pdf (alchemistowl.org)

Technical Note: This file, pocorgtfo19.pdf, **is valid as a PDF** document, **a ZIP archive**, and **a HTML page**. It is also available as a **Windows PE executable**, a **PNG image** and an **MP4 video**, all of which have the same MD5 as this PDF



Types

- Simple Polyglot file: file <u>has</u> different types, accessed depending on how it is handled
- Ambiguous file: is one that <u>is interpreted</u> differently depending on the parser. One parser may crash or fail to process it, while other may return a valid file.
- Chimera file: file has some data that is interpreted as different types

**Use in Malware** 

https://nvd.nist.gov/vuln/detail/CVE-2009-1862

...allows remote attackers to **execute arbitrary code** or cause a **denial of service** (memory corruption) via (1) a **crafted Flash application in a .pdf file** or (2) **a crafted .swf file**, related to authplay.dll, as exploited in the wild in July 2009.

#### **Strategies**

- Stacks: Data is appended to the file
- Cavities: Uses blank (non used space) in the file
- Parasites: Uses comments or metadata fields that allow content to be written
- Zippers: mutual comments

#### **Empty Space**

- Files sometimes allow empty or unused space
  - Before, in the middle or after actual content (appended)
  - Most common in Block formats (ISO and ROM dumps, TAR archives)
    - NAND dumps, ROM dumps, ISOs are directly mapped to sectors
  - Some formats allow arbitrary bytes before file start (e.g. PDF)
    - PDFs are processed from the end
- "Empty space" can be abused to inject crafted content

bash-pdf.pdf

\$ file bash-pdf.pdf
bash-pdf.pdf: POSIX shell script executable (binary data)
\$ ./bash-pdf.pdf
Hello World



- 1 **#!/bin/bash**
- 2 echo "Hello World"; exit
- 3 %PDF-1.7
- 4 **%µµµµ**
- 5 **10 obj**
- 5 <</Type/Catalog/Pages 2 0 R/Lang(en-US) /StructTreeRoot 11 0 R/MarkInfo<</Marked true>>/Metadata 23 0 R/ViewerPreferences 24 0 R>>
- 7 endobj
- 3 20 obj
- 9 <</Type/Pages/Count 1/Kids[ 3 0 R] >>
- 10 endobj
- 11 30 obj
- 12 <</Type/Page/Parent 2 0 R/Resources<</Font<</F1 5 0 R>>/ExtGState<</GS7 7 0 R/GS8 8 0 R>>/XObject<</Image9 9 0 R>>/ProcSet[/PDF/Text/ImageB/ImageC/Image1]
  >>/MediaBox[ 0 0 612 792] /Contents 4 0 R/Group<</Type/Group/S/Transparency/CS/DeviceRGB>>/Tabs/S/StructParents 0>>
- 13 endobj
- 14 40 obj
- 15 <</Filter/FlateDecode/Length 245>>
- 16 stream

#### Why?

- PDF is a collection of objects
  - Objects are dictionaries of properties with a named type
  - Called "CosObjects" or Carousel Object System
  - Simply added to file. New revisions will create new objects that are appended
  - A PDF can have unused object
  - Objects can contain executable code (the code is not executed by the pdf reader!)
    - Objects can contain anything!
    - Well.... There is the LAUNCH action, and Javascript is a valid object type...

A simple object

1 0 obj <</length 100>> stream ...100 bytes.. endstream endobj

**Two objects** 

1 0 obj <</length 100>> stream ...100 bytes.. endstream Endobj 2 0 obj <</length 100>> stream ...100 bytes.. endstream endobj

João Paulo Barrac

#### Two objects and something else that is not parsed

```
1 0 obj
<</length 100>>
stream
...100 bytes..
endstream
Endobj
I should not be here, but who cares. And I could be anywhere
2 0 obj
<</length 100>>
stream
...100 bytes..
endstream
endobj
```

SE ENGINEERING

#### The XREF Table

- PDF have a table with the offset of every object
  - In the end!
  - Reader skips to the end of the file, reads the table and parses the objects
    - That's one reason why it ignores garbage between objects

- XREF table also defines where the file magic (%PDF-1.5 $n\n$ ) is
  - There may be some bytes before the magic
  - Actually, 1024 random bytes are allowed

1 xref 0 26 2 0000000011\_65535 f 0000000017 00000 n 0000000166 00000 n 0000000222 00000 ... 0000000511 00000 n 000000830 00000 n 000000998 00000 n 0000001237 00000 n 10 0000001290 00000 n 11 0000001343 00000 n 12 0000055720 00000 n 13 0000000012 65535 f 14 15 0000000013 65535 f 0000000014 65535 f 16 0000000015 65535 f 17 0000000016 65535 f 18 0000000017 65535 f 19 20 0000000018 65535 f 0000000019 65535 f 21 0000000020 65535 f 22 0000000000 65535 f 23 0000056466 00000 n 24 25 0000056683 00000 n 0000083140 00000 n 26 27 0000086318 00000 n 0000086363 00000 n trailer 29 <</size 26/Root 1 0 R/Info 10 0 R/ID[<85F88F67066D2E4AAB78E636585E887B><85F88F67066D2E4AAB78E636585E887B>] >> 30 startxref 32 86664 %%EOF 34 xref 00 trailer

## **Offsets of object locations**

<</size 26/Root 1 0 R/Info 10 0 R/ID[<85F88F67066D2E4AAB78E636585E887B><85F88F67066D2E4AAB78E636585E887B>] /Prev 86664/XRefStm 86363>>

- startxref
- 87341
- %%EOF

#### ZIP





João Paulo Barraca

Large Format Matrix at https://github.com/Polydet/polyglot-database

#### **Practical application**

- Malware makes use of polyglots are means to circumvent filters
  - A Packet/Email/Web application firewall will block executables, but will it block JPGs?
    - If it does, can it be done with a low rate of false positives?

- General process involves download a polyglot and a decoder
  - Polyglot contains malicious code
  - Decode is implemented in a less suspicious manner (e.g., Javascript)
- From a Reversing Perspective: how much effort will we spend analyzing a JPG?
  - Automated tools such as binwalk, TrId and file can help (but are limited)