OpenOffice v3.x Security Design Weaknesses

Eric Filiol and Jean-Paul Fizaine
ESIEA
Laboratoire de virologie et de cryptologie opérationnelles
38 rue des Dr Calmette et Guérin, 53 000 Laval, France
filiol@esiea.fr,fizaine@esiea-ouest.fr

March 16, 2009

Abstract

For years, document malware exist quite exclusively for Microsoft Office: the sadly known macro-viruses which still represent a numeric nuisance nowadays. However, the recent evolution of office suite towards free software – providing a high compatibility with existing office software – makes it very necessary to determine and evaluate the exact level of risk of the OpenOffice suite with respect to document malware, This paper presents an up to date in-depth evaluation of its security (release 3.0.x) based on the results established since 2006 and 2007. All those results as well as the different sources codes of our attacks have been communicated to the OpenOffice developers group in order to help them to correct the identified security weaknesses and thus enhance the overall security of the OpenOffice suite around the concept of Trusted OpenOffice suite.

While this suite has been developed towards more and more easy-to-useness, the overall security has not been modified at all since. Worrying security weaknesses that have been identified since can still be exploited. They still may be used by malware to spread through innocuous-looking documents by exploiting the feeling of trust based on encryption and digital signature. At the present time, it seems far easier to develop sophisticated document malware for OpenOffice than for Microsoft Office. It is worth mentionning that the attacks we present are NOT based on software (implementation) flaws but on conceptual weaknesses that urge a redesign of the whole software concept.

Finally this paper will discuss the pros and cons of both open and proprietary solutions, on a purely technical basis, as far as security is concerned. There is no such thing as a perfect solution. Therein lies all the complexity of doing computer security.

1 Introduction

The evolution of viral techniques and malware attacks shows that they will occur more and more through office documents. Document malware [8] are formidable attack vectors: users are not suspicious of the documents themselves and a low dose of social engineering is enough to make any user open any document. On the other side, antivirus software are still unable to effectively manage document malware as soon as they are a little bit sophisticated. The analysis of recent cases of espionage or attacks unfortunately demonstrates it dramatically.

In this context, the emergence of new office suites like OpenOffice is a potential boon for the attacker, if security is not there. The case of a macro worm like BadBunny [7] illustrated it particularly. The portability of that office suite on Windows, Unix and Mac environments increases the risk by making it multiplatform. Yet this code was far from exploiting all the weaknesses of this suite.

These conceptual weaknesses have been identified in a number of studies [2, 5, 6] conducted in 2006 and 2007. It was logical at the time of the release of version 3.0 of the suite in October 2008, announced as a major evolution of the suite, to check how these weaknesses had been managed and whether the previously identified attacks were still possible.

For our study, we have considered, without loss of generality, a multiplatform virus written in Python [4]. The goal is not to test the viral algorithmics related to OpenOffice, since it is not conceptually different from that used in Microsoft Office malware – it is just much richer because of the many native programming languages contained in OpenOffice. It is simply related to the general viral algorithmic with respect to document malware [8]. The aim is to test whether the conceptual weaknesses previously identified, mainly related to the management of encryption and digital signature [6] do not still facilitate the attacks and the implementation of those malware when considering the OpenOffice version 3.

Let us briefly recall the attacks and weaknesses that have been identified in 2006 and 2007 [2, 5, 6] for versions 2.x, which were linked to the Open Document Format 1.1.

- Changing the macros management: essentially malware can change and pervert the level of macro security [2, 5].

- Add macros in user space: it was possible to add macros in the user library in the suite. This allowed to embed a resident viral code in

the application. The latter infect any document at the opening for example.

- Modification and manipulation of the menus in the application (undermining the integrity of the application), this weakness is more subtle because it can mislead the user to click too quickly in the same place. In the end, it forces their choice without being realized by the user.

- Integrity, signature and encryption: 2.x versions (ODF 1.0 format) [6] can handle and manipulate the cryptographic functionalities in several ways:

  - Unencrypted (plain) document: the lack of integrity control of the file structure makes possible to include any type of information. Files can be added and inserted without having to declare them in the file structure. Accordingly, it is possible for example to introduce executable code (macro virus).
  - Encrypted document: the most interesting case relates to macros. Considering an encrypted document with macros, you can either simply remove the macro or replace it with (viral) executable code (viral) in plaintext, at the cost of losing the original code (unless performing an actual malware infection like prepreding or appending viral code to the existing macro code), thus without causing any alert at the recipient. It is therefore possible to introduce active code even if the document is encrypted. The attacker uses and pervert the confidence provided by encryption, while increasing the scope of a successful malware attack.
  - Digitally signed document: it was possible to remove the signature on both encrypted and unencrypted documents and thus come back to the previous cases.

In this paper, we will look back at those attacks and identify those who are still valid with the OpenOffice version 3. We shall first briefly recall the structure of the ODF format, particularly with respect to encryption and digital signature features, as well as tools used to analyze and manipulate this format. For a more detailed description, the reader will refer to previous publications [2, 5, 6]. Most of the technical information presented here relate to the Apple Mac version. However, what is presented also fully holds under Unices and Windows. Moreover, without loss of generality we will consider the case of `odt` documents (produced by the *OpenOffice*

Writer) only. All the results presented in this paper generalizes to any other OpenOffice application.

2 ODF Format and Security Features

Let us note first that according to the official website announcements on OpenOffice [10], the OpenOffice suite already supports version 1.2 of the ODF format. But at the time of writing this article, no official document has yet been published on the oasis website [1]. The site only contains old versions 1.0 and 1.1 of the format. Thus we can expect to find exactly the same weaknesses already identified [2, 5, 6].

2.1 A Formal Approach of OpenOffice Cryptographic Features Use

OpenOffice.org' s security is based on two essential mechanisms: password-based encryption and digital signature. Both supposedly aim at preventing an illegitimate use or manipulation of a document, in the particular context of document malware, any weakness with respect to any of these mechanisms could be exploited in a dramatically powerful way to fool the user's trust in both cryptographic protections.

Since there exists a lot of way of using encryption and signature to protect an OpenOffice document, we are going to use a formal graph-based approach to describe them all. Every node in our graph describes a user's action. A given path in our graph will just describe a sequence of such actions to encrypt and/or signed a document. This approach is very powerful at detecting security flaws, in other words the cases where the mechanisms are supposed to have been successfully applied while in reality they are not (the document is not encrypted or not signed contrary to the user's intent).

Our graph-based formalization aims at identifying weaknesses in the signature process, in particular with respect to the macros. To summarize, we are going to show that signature of document visible content and signature of the macros are mutually exclusive: we cannot signed them at the same time. Using technical examples, we will prove in the subsequent sections that it constitutes a serious design flaw that can be efficiently exploited by malware.

Every node describes a possible status for the document:

- **D**: modified document,
- **MD**: modified document with macro,

- **ED**: saved document,
- **EMD**: saved document with macro,
- **SED**: document is signed and saved,
- **MSD**: document with a macro added AFTER the document has been signed,
- **EMSD**: document with a macro added AFTER the document has been signed but BEFORE the document is saved,
- **SEMD**: signed and saved document with a macro.

Nodes are connected by possibly labelled directed arc. The label describes a user's command/action applied to the document:

- `add M`: add a macro,
- `save`: save document,
- `sig 1`: sign through the *File* → *Digital Signatures...* menu,
- `sig 2`: sign through *Tools* → *Macros* → *Digital Signature...* menu,
- `sig 3`: sign through the second bottom right box in the OpenOffice GUI.

The corresponding graph is depicted in Figure 1.

In Figure 1, we notice that the graph is divided into two connected components. A first sub-graph is made up of nodes {**MD, EMD, SEMD**} while the second one contains nodes {**ED, SED, MSD, EMSD**}. This supposes two different possible uses of digital signature that can be applied at any time in the life of the document. However, our experiments have proved that it may be quite different indeed. Let us explain why.

When considering signature AND encryption at the same time, our approach remains essentially the same. The set of nodes is generalized as follows:

- **D**: modified document,
- **MD**: modified document with macro,
- **(SE)MD**: encrypted and saved document with macro,

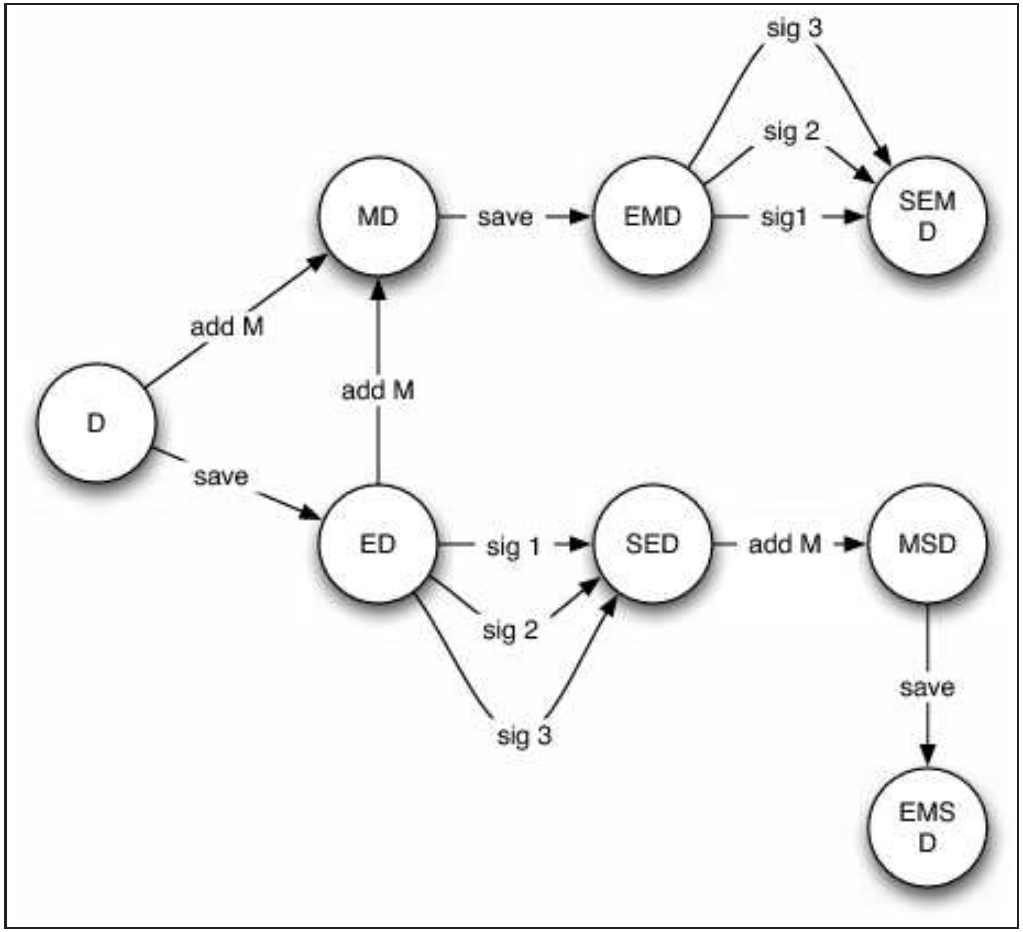


Figure 1: Formal description of digital signature use under OpenOffice

- **S(SE)MD**: signed, encrypted and saved document with a macro,
- **(SE)D**: encrypted and signed document,
- **M(E)D**: a macro is added to an encrypted and saved document,
- **SM(E)D**: a macro is added to an encrypted, saved and finally signed document,
- **S(SE)D**: modified document which has been saved, encrypted and signed,
- **MS(E)D**: a macro is added to an encrypted and signed document,
- **EMS(E)D**: signed then encrypted document with macro.

We thus obtain the graph depicted in Figure 2. It is very powerful at identifying potential misuses of digital signature in OpenOffice. Such misuses could be potentially by malware as we will show it in the next sections. Two classes of design flaws have been identified. The first class deals with the lack of OpenOffice efficient built-in document integrity management. The second class refers to problems that may occur when macros and/or document are signed. We will not discuss the critical of trust macros. They are presented in [2].

2.2 ZIP and ODF Format and Manipulation Tools

Without loss of generlity, we will consider command-line tools for Unix-like environments. Of course, there exist equivalent tools under Windows. However, Unix-like environments are more friendly to use and provide a better access for the reader who intends to reproduce the different examples presented hereafter.

A first command is the `file` command which enables to verify that an OpenOffice document has the OpenDocument format type.

File command output

```
new-host-2:Format lrv\$ file document1.odt
document1.odt: OpenDocument Text
```

Let us recall that in previous OpenOffice version, the file format was the ZIP format. But if we analyze the first few bytes of the file described in the listing given just hereafter, we can however confirm that it is indeed still a ZIP format.

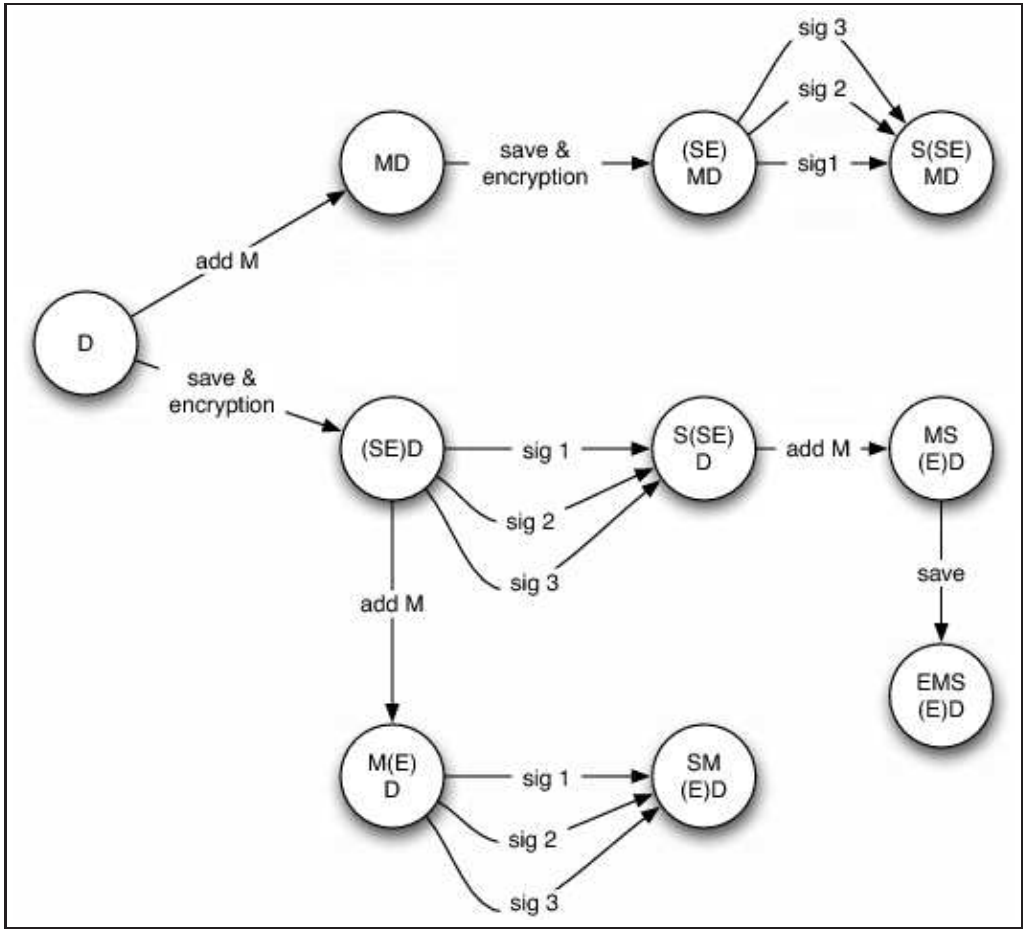


Figure 2: Formal description of encryption AND digital signature use under OpenOffice

First few bytes of OpenDocument files

```
new-host-2:Format lrv$ od -c -N 10 document1.odt
0000000    P   K 003 004 024     0     0     0     0     0
0000012
```

The character string *PK 003 004* indeed betrays the ZIP format (refer to the file `/usr/share/file/magic.mime`, line 303). This information is confirmed in the */usr/share/file/magic* file: any OpenOffice.org, KOffice and StarOffice file are simple ZIP files. The following extract of this file provide additional informations:

magic file content (extract)

```
# opendocument formats (for OpenOffice 2.x / StarOffice >= 8)
# http://lists.oasis-open.org/archives/office/200505/msg00006.html
>>>50   string  vnd.oasis.opendocument. OpenDocument
>>>>73  string  text
>>>>>77 byte    !0x2d                   Text
>>>>>77 string  -template               Text Template
>>>>>77 string  -web                    HTML Document Template
>>>>>77 string  -master                 Master Document
>>>>73  string  graphics                 Drawing
>>>>>81 string  -template               Template
>>>>73  string  presentation             Presentation
>>>>>85 string  -template               Template
>>>>73  string  spreadsheet              Spreadsheet
>>>>>84 string  -template               Template
>>>>73  string  chart                    Chart
>>>>>78 string  -template               Template
>>>>73  string  formula                  Formula
>>>>>80 string  -template               Template
>>>>73  string  database                 Database
>>>>73  string  image                    Image
```

The first column represents the offset (in bytes) from the beginning of the file. The second column defines the data type. The two last columns are of higher interest to us since they contain the document type. At the time of writing, these data hold for the OpenOffice 3.x release only.

The next command to consider enables to definitively confirm the format and document types.

File nature and type

```
new-host-2:Format lrv$ od -c -j 50 -N 35 document1.odt
0000062    v   n   d   .   o   a   s   i   s   .   o   p   e   n   d   o
0000102    c   u   m   e   n   t   .   t   e   x   t   P   K 003 004 024
0000122              0               0               0
0000125
```

It is worth mentioning that there exist a project whose goal is to provide a set of tools to directly and transparently handle and manipulate the *OpenDocument* format. The very final aim is to make the ZIP format as transparent as possible for the developper. This project is denoted **odf toolkit** [9].

2.3 OpenDocument Format Structure

As a first experiment, let us consider a sample document which contains the text *Test file...* only. We will encrypt it, then we will digitally sign it. For comparison purposes, we will then revert the order of those cryptographic functionnalities.

To first decompress an *Opendocument* format document, we simply use the *unzip* command as follows:

```
unzip -d rep file.odt
```

The `-d` option just enables to decompress the file in a specified directory.

Decompression of an OpenDocument file

```
new-host-2:Format lrv$ unzip -d Document1 document1.odt
Archive:  document1.odt
 extracting: Document1/mimetype
   creating: Document1/Configurations2/statusbar/
  inflating: Document1/Configurations2/accelerator/current.xml
   creating: Document1/Configurations2/floater/
   creating: Document1/Configurations2/popupmenu/
   creating: Document1/Configurations2/progressbar/
   creating: Document1/Configurations2/menubar/
   creating: Document1/Configurations2/toolbar/
   creating: Document1/Configurations2/images/Bitmaps/
  inflating: Document1/content.xml
  inflating: Document1/styles.xml
 extracting: Document1/meta.xml
  inflating: Document1/Thumbnails/thumbnail.png
  inflating: Document1/settings.xml
  inflating: Document1/META-INF/manifest.xml
```

The *ls-lR* command provides a detailed view of the content.

Opendocument archive content

```
new-host-2:Format lrv$ ls -lR Document1
total 64
drwxr-xr-x  10 lrv  lrv    340 30 nov 22:40 Configurations2
drwxr-xr-x   3 lrv  lrv    102 30 nov 22:40 META-INF
drwxr-xr-x   3 lrv  lrv    102 30 nov 22:40 Thumbnails
-rw-r--r--   1 lrv  lrv   2726 30 nov 18:06 content.xml
-rw-r--r--   1 lrv  lrv    866 30 nov 18:06 meta.xml
-rw-r--r--   1 lrv  lrv     39 30 nov 18:06 mimetype
-rw-r--r--   1 lrv  lrv   7992 30 nov 18:06 settings.xml
-rw-r--r--   1 lrv  lrv  10615 30 nov 18:06 styles.xml
Document1/Configurations2:
total 0
drwxr-xr-x  3 lrv  lrv  102 30 nov 22:40 accelerator
drwxr-xr-x  2 lrv  lrv   68 30 nov 18:06 floater
drwxr-xr-x  3 lrv  lrv  102 30 nov 22:40 images
drwxr-xr-x  2 lrv  lrv   68 30 nov 18:06 menubar
drwxr-xr-x  2 lrv  lrv   68 30 nov 18:06 popupmenu
drwxr-xr-x  2 lrv  lrv   68 30 nov 18:06 progressbar
drwxr-xr-x  2 lrv  lrv   68 30 nov 18:06 statusbar
drwxr-xr-x  2 lrv  lrv   68 30 nov 18:06 toolbar

Document1/Configurations2/accelerator:
total 0
-rw-r--r--  1 lrv  lrv  0 30 nov 18:06 current.xml

Document1/Configurations2/floater:

Document1/Configurations2/images:
total 0
drwxr-xr-x  2 lrv  lrv  68 30 nov 18:06 Bitmaps

Document1/Configurations2/images/Bitmaps:

Document1/Configurations2/menubar:

Document1/Configurations2/popupmenu:

Document1/Configurations2/progressbar:

Document1/Configurations2/statusbar:
```

```
Document1/Configurations2/toolbar:

Document1/META-INF:
total 8
-rw-r--r--  1 lrv  lrv  1889 30 nov 18:06 manifest.xml

Document1/Thumbnails:
total 8
-rw-r--r--  1 lrv  lrv  807 30 nov 18:06 thumbnail.png
new-host-2:Format lrv$
```

At the very first look, we notice that the structure is quite complex and that there is a number of directories. Official documentation explains that the `META-INF/manifest.xml` file describes the whole file structure while the document apparent content (payload) is located in the `content.xml` file (see listing on the next page).

content.xml

```
new-host-2:Document1 lrv$ more content.xml
<?xml version="1.0" encoding="UTF-8"?>
<office:document-content xmlns:office="urn:oasis:names:tc:opendocument:
xmlns:office:1.0" xmlns:style="urn:oasis:names:tc:opendocument:xmlns:
style:1.0" xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
xmlns:table="urn:oasis:names:tc:opendocument:xmlns:table:1.0" xmlns:
draw="urn:oasis:names:tc:opendocument:xmlns:drawing:1.0" xmlns:fo="urn:
oasis:names:tc:opendocument:xmlns:xsl-fo-compatible:1.0" xmlns:xlink=
"http://www.w3.org/1999/xlink" xmlns:dc="http://purl.org/dc/elements/
1.1/" xmlns:meta="urn:oasis:names:tc:opendocument:xmlns:meta:1.0" xmlns:
number="urn:oasis:names:tc:opendocument:xmlns:datastyle:1.0" xmlns:svg=
"urn:oasis:names:tc:opendocument:xmlns:svg-compatible:1.0" xmlns:chart=
"urn:oasis:names:tc:opendocument:xmlns:chart:1.0" xmlns:dr3d="urn:oasis:
names:tc:opendocument:xmlns:dr3d:1.0" xmlns:math="http://www.w3.org/1998
/Math/MathML" xmlns:form="urn:oasis:names:tc:opendocument:xmlns:form:1.0"
xmlns:script="urn:oasis:names:tc:opendocument:xmlns:script:1.0" xmlns:ooo
="http://openoffice.org/2004/office" xmlns:ooow="http://openoffice.org/
2004/writer" xmlns:oooc="http://openoffice. org/2004/calc" xmlns:dom=
"http://www.w3.org/2001/xml-events" xmlns:xforms="http://www.w3.org/2002/
xforms" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance" xmlns:rpt="http://openoffice.
org/2005/report" xmlns:of="urn:oasis:names:tc:opendocument:xmlns:of:1.2"
xmlns:rdfa="http://docs.oasis-open.org/opendocument/meta/rdfa# " office:
version="1.2"><office:scripts/><office:font-face-decls><style:font-face
style:name="Tahoma1" svg:font-family="Tahoma"/><style:font-face style:
name="Times New Roman" svg:font-family="&apos;Times New Roman&apos;"
style:font-family-generic="roman" style:font-pitch="variable"/><style:
font-face style:name="Arial" svg:font-family="Arial" style:font-family-
generic="swiss" style:font-pitch="variable"/><style:font-face style:name=
"Arial Unicode MS" svg:font-family="&apos;Arial Unicode MS&apos;" style:
font-family-generic="system" style:font-pitch="variable"/><style:font-face
style:name="Tahoma" svg:font-family="Tahoma" style:font-family-generic=
"system" style:font-pitch="variable"/></office:font-face-decls><office:
automatic-styles/><office:body><office:text><text:sequence-decls><text:
sequence-decl text:display-outline-level="0" text:name="Illustration"/>
<text:sequence-decl text:display-outline-level="0" text:name="Table"/>
<text:sequence-decl text:display-outline-level="0" text:name="Text"/>
<text:sequence-decl text:display-outline-level="0" text:name="Drawing"/>
</text:sequence-decls><text:p text:style-name="Standard">
Test file...</text:p></office:text></office:body></office:document-content>
```

The only significant difference with the former version of the format relates to the use of a different file extension. Every file in the archive have the `xml` extension. From the `document1.odt` structure and the OpenDocument documentation, we have built the file organization tree given in Figure 3.

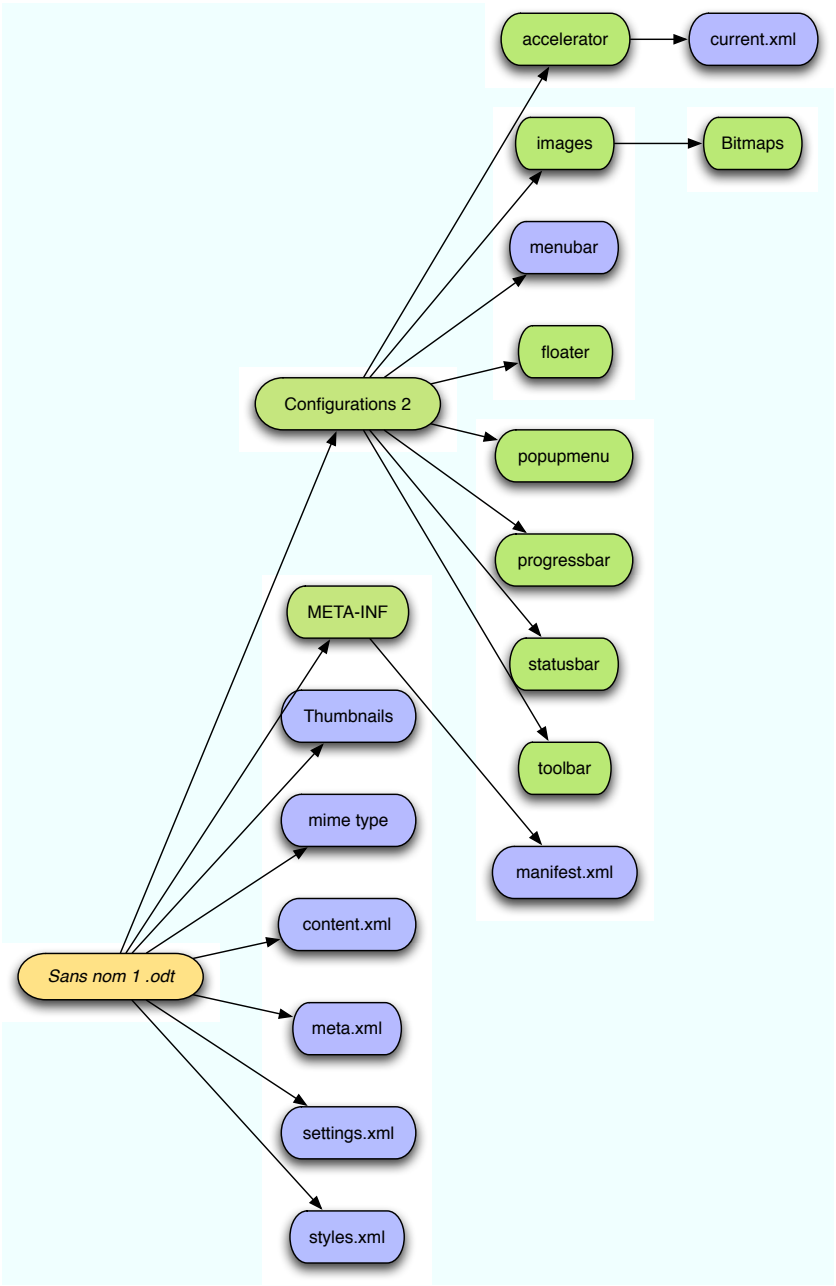


Figure 3: OpenDocument File Internal Structure Tree (here for the `Sans nom 1.odt` document

2.3.1 Role of the `manifest.xml` file

This file somehow plays the role of spinal column for the OpenDocument format. It completely describes files and directories within the ZIP file. To illustrate this, let us consider the following listing with respect to our sample file:

manifest.xml file

```
new-host-2:Sans nom 1 lrv$ more META-INF/manifest.xml
<?xml version="1.0" encoding="UTF-8"?>
<manifest:manifest xmlns:manifest="urn:oasis:names:tc:opendocument:
   xmlns:manifest:1.0"><manifest:file-entry manifest:media-type=
   "application/vnd.oasis.opendocument.text" manifest:version="1.2"
   manifest:full-path="/"/>
 <manifest:file-entry manifest:media-type="" manifest:
      full-path="Configurations2/statusbar/"/>
 <manifest:file-entry manifest:media-type="" manifest:
      full-path="Configurations2/accelerator/current.xml"/>
 <manifest:file-entry manifest:media-type="" manifest:
      full-path="Configurations2/accelerator/"/>
 <manifest:file-entry manifest:media-type="" manifest:
      full-path="Configurations2/floater/"/>
 <manifest:file-entry manifest:media-type="" manifest:
      full-path="Configurations2/popupmenu/"/>
 <manifest:file-entry manifest:media-type="" manifest:
      full-path="Configurations2/progressbar/"/>
 <manifest:file-entry manifest:media-type="" manifest:
      full-path="Configurations2/menubar/"/>
 <manifest:file-entry manifest:media-type="" manifest:
      full-path="Configurations2/menubar/"/>
 <manifest:file-entry manifest:media-type="" manifest:
      full-path="Configurations2/toolbar/"/>
 <manifest:file-entry manifest:media-type="" manifest:
      full-path="Configurations2/images/Bitmaps/"/>
 <manifest:file-entry manifest:media-type="" manifest:
      full-path="Configurations2/images/"/>
 <manifest:file-entry manifest:media-type="application/
   vnd.sun.xml.ui.configuration" manifest:
   full-path="Configurations2/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:
   full-path="content.xml"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:
   full-path="styles.xml"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:
   full-path="meta.xml"/>
```

```
 <manifest:file-entry manifest:media-type="" manifest:
   full-path="Thumbnails/thumbnail.png"/>
 <manifest:file-entry manifest:media-type="" manifest:
   full-path="Thumbnails/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:
   full-path="settings.xml"/>
</manifest:manifest>
```

It is worth mentioning that all attacks identified with respect to the OpenOffice version 2.x [2, 6, 5], have been realized by precisely modifying/manipulating the *manifest.xml*

2.3.2 Macro location

Let us now consider how the OpenDocument format manages the most critical part in terms of security: the (in)famous macros. We have added a macro to the previous sample document, whose task, without loss of generality, consists just in displaying the classical message "*Hello Word*" in a a message box. For clarity sake's and once again without loss of generalty, this macro is not attached to any event and its security level has not been modified. For more details about those aspects the reader will refer to [2].

We obtain the following listing in which the location of macros is described (use the command `unzip -d doc_macro_dir doc_macro.odt`.

Content of a document with macro

```
New-host-2:Format lrv$ unzip -d doc_macro_dir doc_macro.odt
Archive:  doc_macro.odt
 extracting: doc_macro_dir/mimetype
   creating: doc_macro_dir/Configurations2/statusbar/
  inflating: doc_macro_dir/Configurations2/accelerator/current.xml
   creating: doc_macro_dir/Configurations2/floater/
   creating: doc_macro_dir/Configurations2/popupmenu/
   creating: doc_macro_dir/Configurations2/progressbar/
   creating: doc_macro_dir/Configurations2/menubar/
   creating: doc_macro_dir/Configurations2/toolbar/
   creating: doc_macro_dir/Configurations2/images/Bitmaps/
  inflating: doc_macro_dir/content.xml
  inflating: doc_macro_dir/Basic/Standard/Hello.xml
  inflating: doc_macro_dir/Basic/Standard/script-lb.xml
  inflating: doc_macro_dir/Basic/script-lc.xml
  inflating: doc_macro_dir/styles.xml
 extracting: doc_macro_dir/meta.xml
  inflating: doc_macro_dir/Thumbnails/thumbnail.png
```

```
  inflating: doc_macro_dir/settings.xml
  inflating: doc_macro_dir/META-INF/manifest.xml
new-host-2:Format lrv$
```

Compared to the same document without macro, we have now three new files (marked in red color in the previous listing):

- `hello.xml`,
- `script-lb.xml`,
- `script-lc.xml`.

We notice that a new directory, denoted *Basic*, has been created as well. It contains the *script-lc.xml* file whose content is given hereafter:

Content of the file *script-lc.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE library:libraries PUBLIC "-//OpenOffice.org//DTD Office
Document 1.0//EN" "libraries.dtd">
<library:libraries xmlns:library="http://openoffice.org/2000/library"
 xmlns:xlink="http://www.w3.org/1999/xlink">
 <library:library library:name="Standard" library
:link="false"/>
</library:libraries>
```

This file gives a few information about the macro, its relevant library, its directory location within the archive... The `Basic/Standard/script-lb.xml` file (see content hereafter) gives additional information about the macro, including very interesting data related to the document security whose obvious nature does not require further comments:

- library:readonly=”false”
- library:passwordprotected=”false”

This just implies that simple text manipulations of this file enables to dramatically change the security status of the document. This is of critical interest for malware actions.

script-lb.xml file content

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE library:library PUBLIC "-//OpenOffice.org//DTD OfficeDocu
ment 1.0//EN" "library.dtd">
```

```
<library:library xmlns:library="http://openoffice.org/2000/library"
library:name="Standard" library:readonly="false"
library:passwordprotected="false">
 <library:element library:name="Hello"/>
</library:library>
```

Finally, the macro content itself (the code) is contained in the last file `Hello.xml`.

Hello.xml macro content

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE script:module PUBLIC "-//OpenOffice.org//DTD OfficeDocu
ment 1.0//EN" "module.dtd">
<script:module xmlns:script="http://openoffice.org/2000/script" s
cript:name="Hello" script:language="StarBasic">REM  *****  BASIC
*****

Sub Main        msgbox(&quot;Hello Word&quot;)End Sub
</script:module>
```

2.3.3 Document encryption

The official documentation for the OpenDocument format 1.1 mentions that the encryption algorithm used in the suite is the *Blowfish* algorithm in CFB mode (*Cipher Feedback Mode*[1]). As for the integrity management, the SHA-1 primitive has been considered along a 64-bit encoding.

Let us consider a first OpenOffice encrypted document, without any macro. It is very surprising to notice that the `manifest.xml` file is itself not encrypted as shown in the listing hereafter. Indeed this file contains the whole archive structure description, which is a critical information for any malware action.

Content of the *manifest.xml* file of an encrypted document

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest:manifest xmlns:manifest="urn:oasis:names:tc:opendocument:
 xmlns:manifest:1.0">
 <manifest:file-entry manifest:media-type="application/vnd.oasis.
  opendocument.text" manifest:version="1.2" manifest:full-path="/"/>
```

[1] Interestingly, it is not the AES which has been chosen!

```
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/statusbar/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/accelerator/current.xml" manifest:size="0">
<manifest:encryption-data manifest:checksum-type=
"SHA1/1K" manifest:checksum="aIk0hF8iBJyxRmiDLvoz1FATtrk=">
<manifest:algorithm manifest:algorithm-name="Blowfish CFB"
manifest:initialisation-vector="QFdB1tmVK0Q="/>
<manifest:key-derivation manifest:key-derivation-name=
"PBKDF2" manifest:iteration-count="1024" manifest:salt=
"t53RvVUPZHU8FTOZvhUgoQ=="/></manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/accelerator/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/floater/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/popupmenu/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/progressbar/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/menubar/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/toolbar/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/images/Bitmaps/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/images/"/>
 <manifest:file-entry manifest:media-type="application/vnd.sun.
  xml.ui.configuration" manifest:full-path="Configurations2/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:
  full-path="content.xml" manifest:size="2758">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K"
  manifest:checksum="2ovqs54pAuEsJqNREheELGWH+mc=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
 manifest:initialisation-vector="s8J+ff5cKuk="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
     manifest:iteration-count="1024" manifest:salt=
 "pme8h2TQh4PQCzKdfWGWHw==/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="text/xml" manifest:
   full-path="styles.xml" manifest:size="10615">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K"
    manifest:checksum="o/VqxYDQKkIkdZrjmNGefhKth3g=">
```

```
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
   manifest:initialisation-vector="xqNDc9J+npg="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
    manifest:iteration-count="1024" manifest:salt=
     "/117PUAWtAOfr/SfcHv9aw=="/>
 </manifest:encryption-data>
</manifest:file-entry>
<manifest:file-entry manifest:media-type="text/xml" manifest:
  full-path="meta.xml" manifest:size="866">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
  manifest:checksum="FOG/1gCsxEJzIGfuuXRzzsKJvVA=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
   manifest:initialisation-vector="F4bH2haWXOk="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
   manifest:iteration-count="1024" manifest:salt=
   "u0/A4LVJq3zmvl+XGNqlVQ=="/>
 </manifest:encryption-data>
</manifest:file-entry>
<manifest:file-entry manifest:media-type="" manifest:full-path="
 Thumbnails/thumbnail.png" manifest:size="4252">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K" manifest:
  checksum="oJf7JAjmPn/7q76QPXSxjNdN8RM=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
   manifest:initialisation-vector="WFmPHr6IAbU="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
   manifest:iteration-count="1024" manifest:salt=
   "QoAPYhYoJZKy2N5o4YIGAg=="/>
 </manifest:encryption-data>
</manifest:file-entry>
<manifest:file-entry manifest:media-type="" manifest:full-path=
  "Thumbnails/"/><manifest:file-entry manifest:media-type="text/xml"
  manifest:full-path="settings.xml" manifest:size="7993">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
  manifest:checksum="QFtS4asMnG37DnXmeGGlLNz7KXc=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
   manifest:initialisation-vector="JhidkVmh1fI="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
   manifest:iteration-count="1024" manifest:salt="
   b5RCsexsBGkH1673w6aORw=="/>
 </manifest:encryption-data>
</manifest:file-entry>
</manifest:manifest>
```

As this listing clearly shows, a significant number of interesting informa-

tion are then directly available:

- the encryption mechanism itself and its operating way with respect to the integrity management,
- data with respect to the encryption initialization vector (IV) used to differentiate the base key,
- the key derivation algorithm (PBKDF2) and iteration number (1,024), the seed (random value generated for every new encryption)...

Whenever we apply encryption to an OpenOffice document with macros (see listing given hereafter), every file implied in the macro management are themselves encrypted.

manifest.xml file of an encrypted document with macros

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest:manifest xmlns:manifest="urn:oasis:names:tc:opendocument:
  xmlns:manifest:1.0">
 <manifest:file-entry manifest:media-type="application/vnd.oasis.
  opendocument.text" manifest:version="1.2" manifest:full-path="/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/statusbar/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/accelerator/current.xml" manifest:size="0">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K"
  manifest:checksum="aIk0hF8iBJyxRmiDLvoz1FATtrk=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
  manifest:initialisation-vector="voDWTz2nmCI="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
  manifest:iteration-count="1024" manifest:salt=
   "tGw7GCUGjqGEcR19OxJs5A=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="" manifest:full-path="
  Configurations2/accelerator/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="
  Configurations2/floater/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="
  Configurations2/popupmenu/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="
  Configurations2/progressbar/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="
  Configurations2/menubar/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path="
```

```
 Configurations2/toolbar/"/>
<manifest:file-entry manifest:media-type="" manifest:full-path="
 Configurations2/images/Bitmaps/"/>
<manifest:file-entry manifest:media-type="" manifest:full-path="
 Configurations2/images/"/>
<manifest:file-entry manifest:media-type="application/vnd.sun.
 xml.ui.configuration" manifest:full-path="Configurations2/"/>
<manifest:file-entry manifest:media-type="text/xml" manifest:
 full-path="content.xml" manifest:size="2760">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
   manifest:checksum="K+aKAbMvuagJXpbCmP9nVAljr4c=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
   manifest:initialisation-vector="L6IXvbylJNY="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
   manifest:iteration-count="1024" manifest:salt=
  "xi4t6AaaghaTtjL8YeGc6Q=="/>
 </manifest:encryption-data>
</manifest:file-entry>
<manifest:file-entry manifest:media-type="text/xml"
manifest:full-path="Basic/Standard/Hello.xml" manifest:size="340">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
 manifest:checksum="SqB+WsDl2im8HwWMgR2dyTXSjeg=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
  manifest:initialisation-vector="2kLgOfQG4O4="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
  manifest:iteration-count="1024" manifest:salt=
    "n0yXDaA1N8OFpG4lP5rhXg=="/></manifest:encryption-data>
</manifest:file-entry>
<manifest:file-entry manifest:media-type="text/xml"
manifest:full-path="Basic/Standard/script-lb.xml"
manifest:size="346">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
manifest:checksum="AVJqugo0F2xvU9KaiKcanc17mgE=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
 manifest:initialisation-vector="oeIXds7tJyI="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
 manifest:iteration-count="1024" manifest:salt=
 "zAeOU1AAUZHCNULNSznFOQ=="/></manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
"Basic/Standard/"/>
 <manifest:file-entry manifest:media-type="text/xml" manifest:
full-path="Basic/script-lc.xml" manifest:size="338">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
 manifest:checksum="EClic6byHiSVEsuYf5VZ85y2C5A=">
```

```
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
manifest:initialisation-vector="T7AJHQ2d4cQ="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
manifest:iteration-count="1024" manifest:salt=
"gNsBJSVPGniC9vvdTooWSg=="/> </manifest:encryption-data>
</manifest:file-entry> <manifest:file-entry
manifest:media-type="" manifest:full-path="Basic/"/>
<manifest:file-entry manifest:media-type="text/xml"
 manifest:full-path="styles.xml" manifest:size="10615">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
  manifest:checksum="o/VqxYDQKkIkdZrjmNGefhKth3g=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
   manifest:initialisation-vector="M6sfPMj8ay4="/>
  <manifest:key-derivation manifest:key-derivation-name=
   "PBKDF2" manifest:iteration-count="1024" manifest:salt=
   "8FR2WjHPfiUcFprbh56kSw=="/> </manifest:encryption-data>
</manifest:file-entry>
<manifest:file-entry manifest:media-type="text/xml" manifest:
 full-path="meta.xml" manifest:size="866">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
  manifest:checksum="lhixNNmsues8WUN9e7g115jEtR0=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
   manifest:initialisation-vector="EbyAuuIUoH8="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
  manifest:iteration-count="1024" manifest:salt=
  "BJtN3SyHUOM90g0gK1x2/w=="/></manifest:encryption-data>
</manifest:file-entry>
<manifest:file-entry manifest:media-type="" manifest:full-path=
  "Thumbnails/thumbnail.png" manifest:size="4252">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
  manifest:checksum="oJf7JAjmPn/7q76QPXSxjNdN8RM=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
  manifest:initialisation-vector="6Xm8dSyC0JA="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
  manifest:iteration-count="1024" manifest:salt=
   "xnEx1ZWULMHWWf2M5rPpZw=="/></manifest:encryption-data>
</manifest:file-entry>
<manifest:file-entry manifest:media-type="" manifest:full-path=
 "Thumbnails/"/><manifest:file-entry manifest:media-type="text/xml"
 manifest:full-path="settings.xml" manifest:size="7993">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
  manifest:checksum="9WBT7d86SlNWWj+z7X5I+hjxO1w=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
  manifest:initialisation-vector="CQsxPNmEuTo="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
```

```
   manifest:iteration-count="1024"
   manifest:salt="IuR5SWSnLdH5x8sNf7w4Wg==">
  </manifest:encryption-data>
 </manifest:file-entry>
</manifest:manifest>
```

The macro file itself is encrypted:

Contenu de la macro *Hello.xml*

```
^L%^Z ‘\23401f^D\234~
r4K%^\F^X\223Ci1]^m^F.^@‘=Y^Ej\227#‘\217B^@-\207
^Vg#pRd\232,+W\203G\213%)j\230P#0I4^Mge^O^Cr^GgGNs|(y\205^R/h-
\204Gh&Qv2\2148\223s\231^F\207G+{"j|‘8li\216\211oA^OR13k0>^@
\206bb&^]*^RS/^?V^Uil^OIj\216^A\206/Z
```

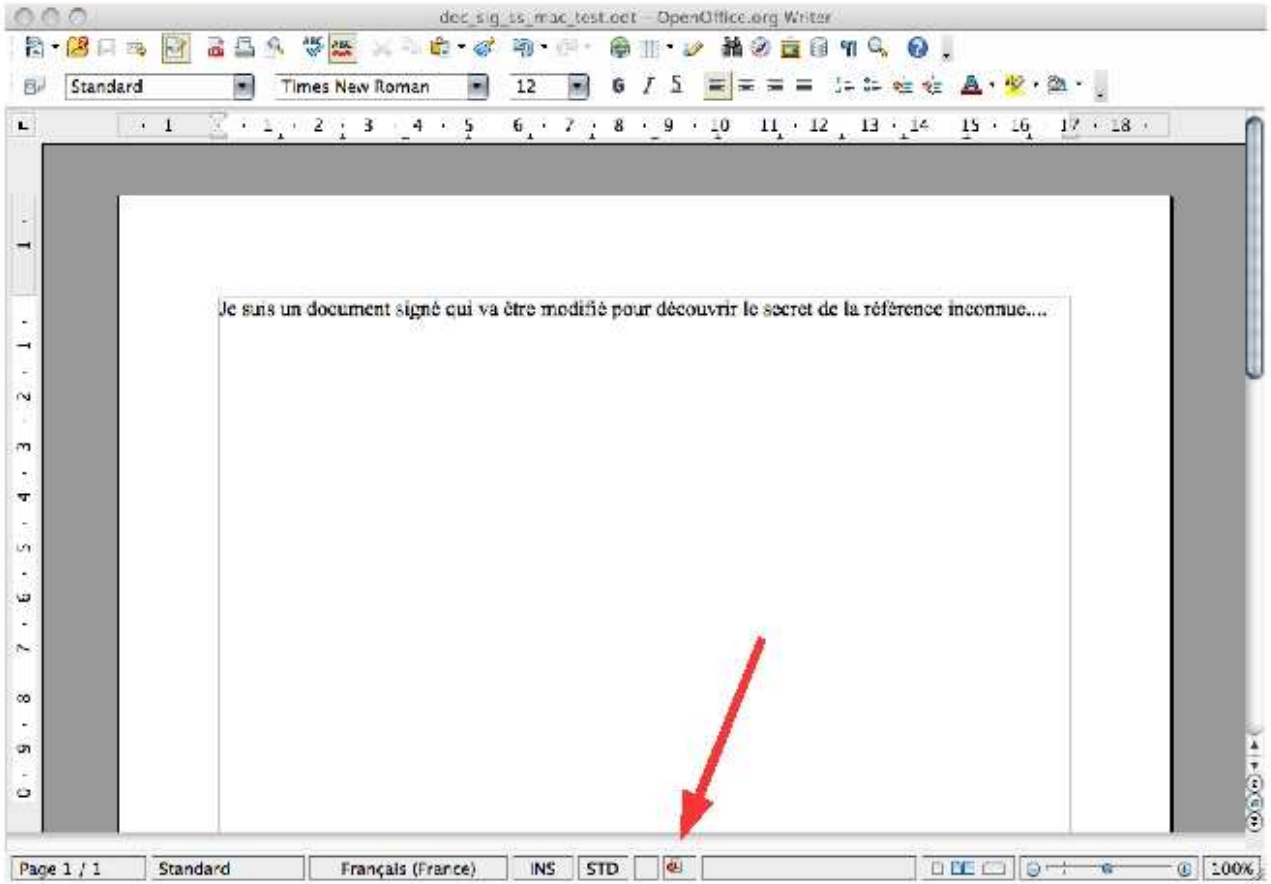
As a first conclusion, we see that the protection of the `manifest.xml` file is not sufficient. Despite the fact that the document seems to use strong digital signature, partly for strong integrity purposes, we will show in Section 5 that this protection can be easily bypassed.

2.3.4 Digital signature of documents

In order to identify and locate the different components involved in the digital signature, we will proceed in the same way as for encryption (refer to the previous section). Let us first consider a document without and then with macro. In OpenOffice 2.x versions, the digital signature was managed at the *Firefox* browser level. For that purpose, it has required to add an environment variable to OpenOffice.org in order to set up the certificate management service (site). It was necessary to import a valid PKCS12 certificate into the browser. Finally, it was possible to digitally sign a document before saving the document.

In the newly released version (OpenOffice.org v3.x), quite almost nothing has changed from a functional point of view. The signature maganement tools are still located either in the *Tools* → *Macros* → *Digital Signature...* menu or in the `File` → `Digital Signature...` menu (see Section 2.3.6). Once a document has been signed, a dedicated symbol appears in the bottom middle of the application (see Figure 4.

The analysis of the archive files shows (see listing hereafter) that the information with respect to the signature are located in an additional file which is not the `manifest.xml` file.


doc_sig_ts_mac_test.odt – OpenOffice.org Writer
Standard
Times New Roman
12
Je suis un document signé qui va être modifié pour découvrir le secret de la référence inconnue....
Page 1 / 1
Standard
Français (France)
INS
STD
100%

Digitally signed document without

```
ZZR:sig_ss_macro lrv$ unzip -d doc_ss_macro_nchiff_dir
         doc_ss_macro_nchiff.odt
Archive:  doc_ss_macro_nchiff.odt
extracting: doc_ss_macro_nchiff_dir/mimetype
  creating: doc_ss_macro_nchiff_dir/Configurations2/statusbar/
 inflating: doc_ss_macro_nchiff_dir/Configurations2/accelerator/
            current.xml
  creating: doc_ss_macro_nchiff_dir/Configurations2/floater/
  creating: doc_ss_macro_nchiff_dir/Configurations2/popupmenu/
  creating: doc_ss_macro_nchiff_dir/Configurations2/progressbar/
  creating: doc_ss_macro_nchiff_dir/Configurations2/menubar/
  creating: doc_ss_macro_nchiff_dir/Configurations2/toolbar/
  creating: doc_ss_macro_nchiff_dir/Configurations2/images/Bitmaps/
inflating: doc_ss_macro_nchiff_dir/META-INF/
   documentsignatures.xml
 inflating: doc_ss_macro_nchiff_dir/content.xml
 inflating: doc_ss_macro_nchiff_dir/styles.xml
 extracting: doc_ss_macro_nchiff_dir/meta.xml
 inflating: doc_ss_macro_nchiff_dir/Thumbnails/thumbnail.png
 inflating: doc_ss_macro_nchiff_dir/settings.xml
 inflating: doc_ss_macro_nchiff_dir/META-INF/manifest.xml
ZZR:sig_ss_macro lrv$
```

All the information are located in the `META-INF/documentsignatures.xml` file. At the beginning of this file the *SignatureMethod* tag indicates which signature algorithm is used: *rsa-sha1* algorithm. For every file described by the *Reference* tag, there are two additional tags:

1. the *DigestMethod* tag for the integrity algorithm,
2. the *DigestValue* which contains the hashh value itself.

Then, we find the certificate details with respect to the document signer (within the *KeyInfo* tags): name, town, email address of the certificate owner. Finally the X509 certificate itself follows in the *X509Certificate*. The certificate is reproduced three times. Here follows the corresponding listing for illustration purposes:

documentsignature.xml file content

```
<?xml version="1.0" encoding="UTF-8"?>
<document-signatures xmlns="urn:oasis:names:tc:opendocument:
  xmlns:digitalsignature:1.0">
```

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#"
  Id="ID_00af00330079002b00f7008f00480072008000910003f00e300f7003200
      1400ac">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/
     REC-xml-c14n-20010315"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#
     rsa-sha1"/>
    <Reference URI="Configurations2/accelerator/current.xml">
      <DigestMethod Algorithm="http://www.w3.org/
     2000/09/xmldsig#sha1"/>
      <DigestValue>2jmj7l5rSw0yVb/vlWAYkK/YBwk=</DigestValue>
      </Reference>
    <Reference URI="content.xml">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/TR/2001/
          REC-xml-c14n-20010315"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#
       sha1"/>
      <DigestValue>33FY1rcgFPlB7lQo2XhvSRu25Lg=</DigestValue>
    </Reference>
    <Reference URI="styles.xml">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/TR/2001/
         REC-xml-c14n-20010315"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#
         sha1"/>
      <DigestValue>6j3xV9OoFUv9fjP6SttOiKO+vG0=</DigestValue>
    </Reference>
    <Reference URI="meta.xml">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/TR/2001/
         REC-xml-c14n-20010315"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#
         sha1"/>
      <DigestValue>OOqkeMjV5Y889YBxcUlE+77nVlk=</DigestValue>
    </Reference>
    <Reference URI="Thumbnails/thumbnail.png">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#
       sha1"/>
      <DigestValue>wLZnbiBkTUid9POT8M3UPIOxLN8=</DigestValue>
    </Reference>
```

```
  <Reference URI="settings.xml">
    <Transforms>
      <Transform Algorithm="http://www.w3.org/TR/2001/
       REC-xml-c14n-20010315"/>
    </Transforms>
    <DigestMethod Algorithm="http://www.w3.org/2000/09/
       xmldsig#sha1"/>
    <DigestValue>b3EdF+Nb7saYiveC17OvydU0Ss4=</DigestValue>
  </Reference>
  <Reference URI="#ID_0039006700b700a9006d008
       8004600e9009f00d9005f00f400f500c4000b0032">
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#
       sha1"/>
    <DigestValue>C3ymiFAn3V+n7AkEGLvGddVI2Q8=</DigestValue>
  </Reference>
</SignedInfo>
<SignatureValue>rPcs/4mmY5Cm0jY40IXEPEuxPVd43TjbjYuaghe3hscXAVYI
 FErIyzziIV0Dv/ERwNcs5ulwnb0VpQeoLMU1z5KNQ2xu7BZqWXYUk9xo0UTa13i
 rHMbAPG0qDMbd4ntLHGnUUGS14SbChgtwben5zO2ZR/LxwYo3CMXXl2Rqj3A=
 </SignatureValue>
<KeyInfo>
  <X509Data>
    <X509IssuerSerial>
      <X509IssuerName>E=jpfizaine@gmail.com,CN=Fizaine,
       OU=Recherche,O=Lrv,L=Poitiers,ST=Vienne,C=FR
      </X509IssuerName>
      <X509SerialNumber>16553245699451631157</X509SerialNumber>
    </X509IssuerSerial>
    <X509Certificate>MIIDhDCCAu2gAwIBAgIJAOW48ULDk7Y1MA0GCSq
    GSIb3DQEBBAUAMIGJMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGVmllbm5
    lMREwDwYDVQQHEwhQb2l0aWVyczEMMAoGA1UEChMDTHJ2MRIwEAYDVQQ
    LEwlSZWNoZXJjaGUxEDAOBgNVBAMTB0ZpemFpbmUxIjAgBgkqhkiG9w0
    BCQEWE2pwZml6YWluZUBnbWFpbC5jb20wHhcNMDgxMjE5MTAxNzA3Whc
    NMDkwMTE4MTAxNzA4WjCBiTELMAkGA1UEBhMCRlIxDzANBgNVBAgTBlZ
    pZW5uZTERMA8GA1UEBxMIUG9pdGllcnMxDDAKBgNVBAoTA0xydjESMBA
    GA1UECxMJUmVjaGVyY2hlMRAwDgYDVQQDEwdGaXphaW5lMSIwIAYJKoZ
    IhvcNAQkBFhNqcGZpemFpbmVAZ21haWwuY29tMIGfMA0GCSqGSIb3DQE
    BAQUAA4GNADCBiQKBgQDiYnM6+ILIDtySkMWNkK082gk3MzlC8witnym
    GWGpkJDa5TIiFHke1w8PyDSYJ+X9cIhjXlc+FdDcZnHk0LxiKu4JbaEZ
    4f/tFdAHNkHRimLWa8fvwVziftO/1+vRBTvmnVhwrTBtDe3JVkWMJDT9
    YCOjr2ey2I/TjQVAYbiE8cwIDAQABo4HxMIHuMB0GA1UdDgQWBBQYm3x
    6c8waECap4izFiwfmSKeCnjCBvgYDVR0jBIG2MIGzgBQYm3x6c8waECa
    p4izFiwfmSKeCnqGBj6SBjDCBiTELMAkGA1UEBhMCRlIxDzANBgNVBAg
    TBlZpZW5uZTERMA8GA1UEBxMIUG9pdGllcnMxDDAKBgNVBAoTA0xydjE
    SMBAGA1UECxMJUmVjaGVyY2hlMRAwDgYDVQQDEwdGaXphaW5lMSIwIAY
```

```
        JKoZIhvcNAQkBFhNqcGZpemFpbmVAZ21haWwuY29tggkA5bjxQsOTtjU
        wDAYDVR0TBAUwAwEB/zANBgkqhkiG9w0BAQQFAAOBgQCVAFl6e56hJaO
        QxjMzMfD8AVxwCkjZ6X6FCAEgPjMaNHObjYnMrXsXfhFZmWKfGzGM9YA
        B6emv+/Jpk1u1wLSoteidEYA2+tflBfK7aNmicIUWmFzZxZ41GKWMlov
        WrrWYyjcXbx9Gwng6ggmxu9PtssFzOBZkQ77spwOpdvh5SA=
        =</X509Certificate>
        <x509certificate>MIIDhDCCAu2gAwIBAgIJAOW48ULDk7Y1MA0GCSq
        GSIb3DQEBBAUAMIGJMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGVmllbm5
        lMRE  [...]
        e56hJaOQxjMzMfD8AVxwCkjZ6X6FCAEgPjMaNHObjYnMrXsXfhFZmWKf
        GzGM9YAB6emv+/Jpk1u1wLSoteidEYA2+tflBfK7aNmicIUWmFzZxZ41
        GKWMlovWrrWYyjcXbx9Gwng6ggmxu9PtssFzOBZkQ77spwOpdvh5SA==
        </X509Certificate>
        <x509certificate>MIIDhDCCAu2gAwIBAgIJAOW48ULDk7Y1MA0GCSq
        GSIb3DQEBBAUAMIGJMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGVmllbm5
        lMRE  [...]
        e56hJaOQxjMzMfD8AVxwCkjZ6X6FCAEgPjMaNHObjYnMrXsXfhFZmWKf
        GzGM9YAB6emv+/Jpk1u1wLSoteidEYA2+tflBfK7aNmicIUWmFzZxZ41
        GKWMlovWrrWYyjcXbx9Gwng6ggmxu9PtssFzOBZkQ77spwOpdvh5SA==
        </X509Certificate>
      </X509Data>
    </KeyInfo>
    <Object>
      <SignatureProperties>
        <SignatureProperty Id="ID_0039006700b700a9006d0088004600
          e9009f00d9005f00f400f500c4000b0032" Target="#ID_00af00
          330079002b00f7008f00480072008000910003f00e300f700320014
          00ac">
          <dc:date xmlns:dc="http://purl.org/dc/elements/1.1/">
           2008-12-19T14:56:19</dc:date>
        </SignatureProperty>
      </SignatureProperties>
    </Object>
</Signature>
</document-signatures>
```

The following files only of the archive are digitally signed:

- `Configurations2/accelerator/current.xml`,
- `content.xml`,
- `style.xml`,
- `meta.xml`,

- `Thumbnails/thumbnail.png`,
- `setting.xml`,

It is worth mentioning that the *META-INF/manifest.xml* and *META-INF/-documentsignatures.xml* themselves are not signed. This constitutes a dramatic weakness as we will see later. However, there is another strange reference (enlighted in orange in the previous listing) which seems to relate in a way or another to the `META-INF/manifest.xml` file. A detailed analysis in fact shows that it is actually a self-reference of the file to itself, in order to recursively managed its own integrity. But this does not constitutes a limitation when perverting the signature.

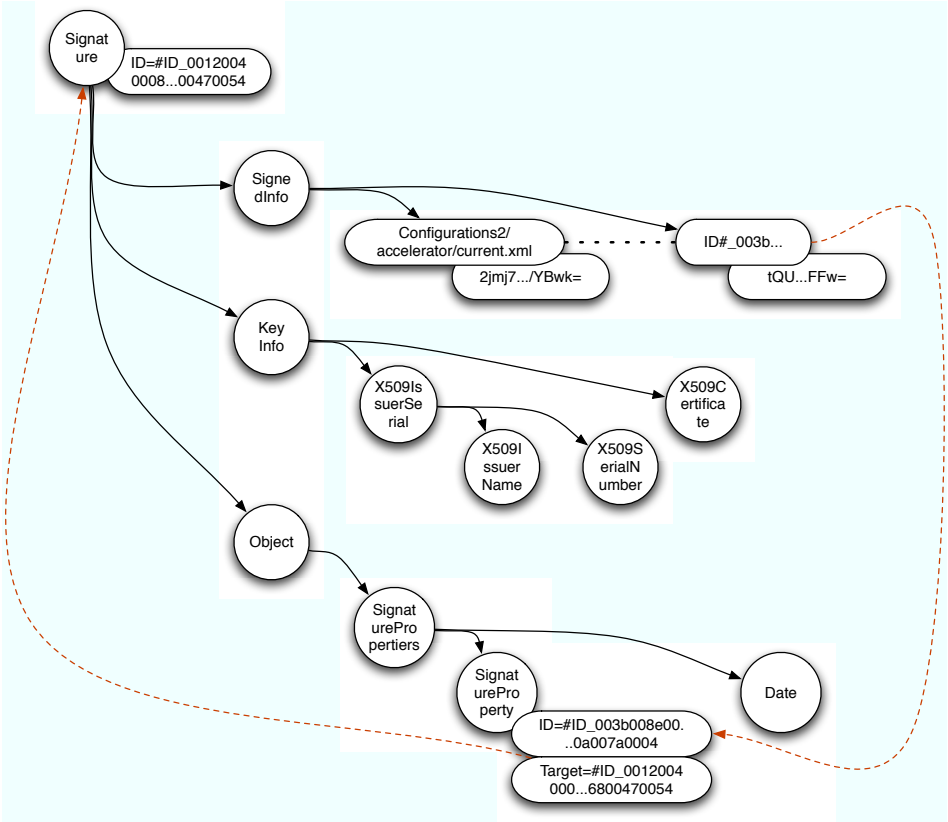


Figure 5: Structural graph of the *documentsignatures.xml* file structure

The digital signature relies on the *XML-DSIG* norm [3]. However, it is itself not standardized in the OpenDocument format release 1.2 (at the

present time no more information is available and the only reference is the version 1.1 [1]) yet. To summarize the complex organization and structure within the *documentsignature.xml*, we have established its structural graph given in Figure 5. It models both the information hierarchy and the relationships existing between a file and its integrity value (arcs in red color).

From the analysis of the OpenOffice source code related to the digital signature management, we can summarize the main steps operated whenever a document is signed:

1. The signer certificate which has been selected is checked and loaded.
2. The `aSignatureHelper.AddForSigning` primitive is called from `Current/xmlsecurity/source/helper/xmlsignaturehelper.cxx`.
3. The `mpXSecController->signAStream` primitive is then called from `Current/xmlsecurity/source/helper/xsecsign.cxx`.
4. The `m_vInternalSignatureInformations[index].addReference` primitive is called from `Current/xmlsecurity/source/helper/xsecctl.hxx`.
5. Call of the `aSignatureHelper.CreateAndWriteSignature` primitive from `Current/xmlsecurity/source/helper/xmlsignaturehelper.cxx`.
6. Call of the `WriteSignature` primitive from `Current/xmlsecurity/source/helper/xsecsign.cxx` in order to create an instance of the document by means of `CreateDocumentHandlerWithHeader` in `Current/xmlsecurity/source/helper/xmlsignaturehelper.cxx`
7. Insertion of signature `WriteSignature` from `Current/xmlsecurity/source/helper/xsecsign.cxx` in the previous document instance,
8. Formatting step by calling the `prepareSignatureToWrite` primitive, then call of the `exportSignature` primitive from `Current/xmlsecurity/source/helper/xsecctl.cxx`.

The different structures used to store all the information related to the signature process are located in `/Current/xmlsecurity/inc/xmlsecurity/sigstruct.hxx`. The `SignatureReferenceInformation` structure stores the integrity value for a given file. The `struct SignatureInformation` structure stores all those generated during the signature, in particular `SignatureReferenceInformations vSignatureReferenceInfors` which is itself a list

of `SignatureReferenceInformation` structures. The `rtl::OUString ouSignatureValue` variable contains the signature value itself. All the aspects related to signature could be interesting for any malware which would operate directly in memory and could thus manipulate the signature during its production.

As a conclusion, the digital signature in OpenOffice 3.x is only partially secure: while it complies with signature standards, it is far from being correctly and securely implemented: the critical files are not encrypted and critical data can be accessed very easily.

2.3.5 Digital signature combined with encryption

Whenever a document is both signed and encrypted, the overall structure remains the same. The slight difference lies in the fact that the integrity value is computed from the encrypted version of the files. However, we have noticed that the `META-INF/documentsignatures.xml` file is itself not encrypted.

manifest.xml file content of a signed, encrypted document without macro

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest:manifest xmlns:manifest="urn:oasis:names:tc:opendocument:
  xmlns:manifest:1.0">
 <manifest:file-entry manifest:media-type="application/vnd.oasis.
  opendocument.text" manifest:version="1.2" manifest:full-path="/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/statusbar/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/accelerator/current.xml" manifest:size="0">
  <manifest:encryption-data manifest:checksum-type="SHA1/1K"
  manifest:checksum="aIk0hF8iBJyxRmiDLvoz1FATtrk=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
  manifest:initialisation-vector="OgPf8Zzjzvw="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
  manifest:iteration-count="1024" manifest:salt=
  "YQBpOWEEtDWvtzA1y50vsw=="/>
  </manifest:encryption-data>
 </manifest:file-entry>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/accelerator/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/floater/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
  "Configurations2/popupmenu/"/>
 <manifest:file-entry manifest:media-type="" manifest:full-path=
```

```
 "Configurations2/progressbar/"/>
<manifest:file-entry manifest:media-type="" manifest:full-path=
 "Configurations2/menubar/"/>
<manifest:file-entry manifest:media-type="" manifest:full-path=
 "Configurations2/toolbar/"/>
<manifest:file-entry manifest:media-type="" manifest:full-path=
 "Configurations2/images/Bitmaps/"/>
<manifest:file-entry manifest:media-type="" manifest:full-path=
 "Configurations2/images/"/>
<manifest:file-entry manifest:media-type="application/vnd.sun.
 xml.ui.configuration" manifest:full-path="Configurations2/"/>
<manifest:file-entry manifest:media-type="" manifest:full-path=
 "META-INF/documentsignatures.xml"/>
<manifest:file-entry manifest:media-type="" manifest:full-path=
 "META-INF/"/>
<manifest:file-entry manifest:media-type="text/xml" manifest:
 full-path="content.xml" manifest:size="2761">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
 manifest:checksum="bC4i5ZGRXrj75cKbDJab1J2hsUY=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
 manifest:initialisation-vector="SOV70HUn4Oo="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
 manifest:iteration-count="1024" manifest:salt=
 "2o0OXq0jrJ/7TCpmaBudig=="/>
 </manifest:encryption-data>
</manifest:file-entry>
<manifest:file-entry manifest:media-type="text/xml" manifest:
 full-path="styles.xml" manifest:size="10615">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
 manifest:checksum="o/VqxYDQKkIkdZrjmNGefhKth3g=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
 manifest:initialisation-vector="u97RW7vYTUE="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
 manifest:iteration-count="1024" manifest:salt=
 "Z6MYPg4I+bldkLE0C1vP8A=="/>
 </manifest:encryption-data>
</manifest:file-entry>
<manifest:file-entry manifest:media-type="text/xml"
 manifest:full-path="meta.xml" manifest:size="866">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
 manifest:checksum="KTBc6tsuUW6ipNwxrGNb2ENgbcY=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
 manifest:initialisation-vector="9WK0PpHB8Ps="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
 manifest:iteration-count="1024" manifest:salt=
```

```
 "VjqjV6h4yeSH8PsOgfpJFg==" />
 </manifest:encryption-data>
</manifest:file-entry>
<manifest:file-entry manifest:media-type="" manifest:full-path=
 "Thumbnails/thumbnail.png" manifest:size="4252">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
 manifest:checksum="oJf7JAjmPn/7q76QPXSxjNdN8RM=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
 manifest:initialisation-vector="om5AUfkgRUk="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
 manifest:iteration-count="1024" manifest:salt=
 "kJ6sRqSoGV4CXrri81PPaA==" />
 </manifest:encryption-data>
</manifest:file-entry>
<manifest:file-entry manifest:media-type="" manifest:
 full-path="Thumbnails/"/>
<manifest:file-entry manifest:media-type="text/xml" manifest:
 full-path="settings.xml" manifest:size="7993">
 <manifest:encryption-data manifest:checksum-type="SHA1/1K"
 manifest:checksum="QAu2847Mq+yA/p/QtO3SpZBimBc=">
  <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
 manifest:initialisation-vector="MAN/+6/q3tw="/>
  <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
 manifest:iteration-count="1024" manifest:salt=
 "3evR4PQWQajIOSPxtTkxvA==" />
 </manifest:encryption-data>
</manifest:file-entry>
</manifest:manifest>
```

Additionally, the `documentsignatures.xml` is still not encrypted. This is worth noticing since it is a major weakness which will enable us to build powerful attacks (see further).

2.3.6 Signature and macros

Let us now consider the case of documents with macros. The following listing clearly shows that the file relating to macros are signed, that is to say:

- `Basic/Standard/Helloword.xml`,
- `Basic/Standard/script-lb.xml`,
- `Basic/script-lc.xml`.

documentsignatures.xml file content for an encrypted, signed document

<?xml version="1.0" encoding="UTF-8"?>
<document-signatures xmlns="urn:oasis:names:tc:opendocument:xmlns:
digitalsignature:1.0"><Signature xmlns="http://www.w3.org/2000/09/
xmldsig#" Id="ID_004500bb0048003900cb00df004f004000bc002900ce00c6
00ad008d00900045"><SignedInfo><CanonicalizationMethod Algorithm=
"http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/><SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/><Reference
URI="Configurations2/accelerator/current.xml"><DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/><DigestValue>
XyZEi1Pop5CELspLAY3Cu0yEN2k=</DigestValue></Reference><Reference
URI="content.xml"><DigestMethod Algorithm="http://www.w3.org/2000/
09/xmldsig#sha1"/><DigestValue>V3pF751rQvy/ZRl0UmYbn9f+Mns=
</DigestValue></Reference><Reference URI="Basic/Standard/
script-lb.xml"><DigestMethod Algorithm="http://www.w3.org/2000/09/
xmldsig#sha1"/><DigestValue>j7ZLBv2s983y4d45JUSsITyV4cQ=
</DigestValue></Reference><Reference URI="Basic/Standard/
helloword.xml"><DigestMethod Algorithm="http://www.w3.org/2000/09/
xmldsig#sha1"/><DigestValue>KI0dwenjcLYG2E0FTw8ev1hMgL4=</DigestValue>
</Reference><Reference URI="Basic/script-lc.xml"><DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/><DigestValue>
OyFNXrX3ZH1IoiS1MXR+D/6SoG4=</DigestValue></Reference><Reference
URI="styles.xml"><DigestMethod Algorithm="http://www.w3.org/2000/
09/xmldsig#sha1"/><DigestValue>6D58vNujzFAkVmx1G+IIVVb3Yx8=
</DigestValue></Reference><Reference URI="meta.xml"><DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/><DigestValue>
abJwazDRPrAyAx2FPMPSmDtTeS0=</DigestValue></Reference><Reference
URI="Thumbnails/thumbnail.png"><DigestMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/><DigestValue>73/8vrqsq1
yG+CGuCnUMHJGF0yU=</DigestValue></Reference><Reference
URI="settings.xml"><DigestMethod Algorithm="http://www.w3.org/2000/
09/xmldsig#sha1"/><DigestValue>Ulw3BbqePP7wvJHAscb00mbcBuo=
</DigestValue></Reference><Reference URI="#ID_002a006c00b2001900c80
0a7004c00ec0096006100aa0031008d00cd006a008b"><DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/><DigestValue>
R6jxmsjbHN3MJArSzcvXBALTfEk=</DigestValue></Reference></SignedInfo>
<SignatureValue>W0J4/7TVpQ+AL91SWWboWhe3laYuYnTaXfl1TVVKu57Xt/9+0mP
+84VBqu4BAM9tWviM/07yzuS8fg7+24I5Xj5k2ZdJeH42JTzNfLO5oTITIyuE/gFdNr
Zs5XE4onagN7pQZJzN7pEAoMZ0qZnDSdHBiEb7m+gK0jeasuWmnjk=</Signature
Value><KeyInfo><X509Data><X509IssuerSerial><X509IssuerName>E=jpfiza
ine@gmail.com,CN=Fizaine,OU=Recherche,O=Lrv,L=Poitiers,ST=Vienne,
C=FR</X509IssuerName><X509SerialNumber>16553245699451631157
</X509SerialNumber></X509IssuerSerial><X509Certificate>MIIDhDCCAu2g
AwIBAgIJAOW48ULDk7Y1MA0GCSqGSIb3DQEBBAUAMIGJMQswCQYDVQQGEwJGUjEPMA0

```
GA1UECBMGVmllbm5lMREwDwYDVQQHEwhQb2l0aWVyczEMMAoGA1UEChMDTHJ2MRIwEA
YDVQQLEwlSZWNoZXJjaGUxEDAOBgNVBAMTB0ZpemFpbmUxIjAgBgkqhkiG9w0BCQEWE
2pwZml6YWluZUBnbWFpbC5jb20wHhcNMDgxMjE5MTAxNzA3WhcNMDkwMTE4MTAxNzA4
WjCBiTELMAkGA1UEBhMCRlIxDzANBgNVBAgTBlZpZW5uZTERMA8GA1UEBxMIUG9pdGl
lcnMxDDAKBgNVBAoTA0xydjESMBAGA1UECxMJUmVjaGVyY2hlMRAwDgYDVQQDEwdGaX
phaW5lMSIwIAYJKoZIhvcNAQkBFhNqcGZpemFpbmVAZ21haWwuY29tMIGfMA0GCSqGS
Ib3DQEBAQUAA4GNADCBiQKBgQDiYnM6+ILIDtySkMWNkK082gk3MzlC8witnymGWGpk
JDa5TIiFHke1w8PyDSYJ+X9cIhjXlc+FdDcZnHk0LxiKu4JbaEZ4f/tFdAHNkHRimLW
a8fvwVziftO/1+vRBTvmnVhwrTBtDe3JVkWMJDT9YCOjr2ey2I/TjQVAYbiE8cwIDAQ
ABo4HxMIHuMB0GA1UdDgQWBBQYm3x6c8waECap4izFiwfmSKeCnjCBvgYDVR0jBIG2M
IGzgBQYm3x6c8waECap4izFiwfmSKeCnqGBj6SBjDCBiTELMAkGA1UEBhMCRlIxDzAN
BgNVBAgTBlZpZW5uZTERMA8GA1UEBxMIUG9pdGllcnMxDDAKBgNVBAoTA0xydjESMBA
GA1UECxMJUmVjaGVyY2hlMRAwDgYDVQQDEwdGaXphaW5lMSIwIAYJKoZIhvcNAQkBFh
NqcGZpemFpbmVAZ21haWwuY29tggkA5bjxQsOTtjUwDAYDVR0TBAUwAwEB/zANBgkqh
kiG9w0BAQQFAAOBgQCVAFl6e56hJaOQxjMzMfD8AVxwCkjZ6X6FCAEgPjMaNHObjYnM
rXsXfhFZmWKfGzGM9YAB6emv+/Jpk1u1wLSoteidEYA2+tflBfK7aNmicIUWmFzZxZ4
1GKWMlovWrrWYyjcXbx9Gwng6ggmxu9PtssFzOBZkQ77spwOpdvh5SA==
</X509Certificate><X509Certificate>MIIDhDCCAu2gAwIBAgIJAOW48ULDk7Y1
[.......]
XsXfhFZmWKfGzGM9YAB6emv+/Jpk1u1wLSoteidEYA2+tflBfK7aNmicIUWmFzZxZ41
GKWMlovWrrWYyjcXbx9Gwng6ggmxu9PtssFzOBZkQ77spwOpdvh5SA==
</X509Certificate>
<X509Certificate>MIIDhDCCAu2gAwIBAgIJAOW48ULDk7Y1MA0GCSqGSIb3DQEBBA
[.........]
Z41GKWMlovWrrWYyjcXbx9Gwng6ggmxu9PtssFzOBZkQ77spwOpdvh5SA==
</X509Certificate>
</X509Data></KeyInfo><Object><SignatureProperties><SignatureProperty
Id="ID_002a006c00b2001900c800a7004c00ec0096006100aa0031008d00cd006a
008b" Target="#ID_004500bb0048003900cb00df004f004000bc002900ce00c600
ad008d00900045"><dc:date xmlns:dc="http://purl.org/dc/elements/1.1/">
2008-12-20T22:07:58</dc:date></SignatureProperty></Signature
Properties></Object></Signature></document-signatures>
```

Macros are signed BUT the `documentsignatures.xml` is itself neither encrypted nor signed.

Let us go deeper inside the signature management of document with macros. We are going to check whether there is a significant evolution from OpenOffice 2.x to OpenOffice 3.x. In [6], serious weaknesses have been identified, Indeed, digital signature was only applying to the document content only. Macros themselves were not signed. As a consequence, it was possible to replace a macro with another (possibly malicious) macro, without triggering any integrity violation alert.

In OpenOffice 3.x, there are still two ways to digitally sign a document. The first one is to use the `File` → `Digital Signature...` menu while the second one considers the `Tools` → `Macro` → `Digital Signature...` menu. According to the method used, there are significant differences inside the corresponding archive. In the first case, the information with respect to signature are located in the `META-INF/documentsignature.xml` while in the second case, the signature process stores them into the `META-INF/macrosignature.xml`

`Comparaison between the signature methods (documents without macros)`

```
  ZZR:CreationSig lrv$ unzip -d doc_sig2_ss_mac_dit
         doc_sig2_ss_mac.odt
Archive:  doc_sig2_ss_mac.odt
 extracting: doc_sig2_ss_mac_dit/mimetype
   creating: doc_sig2_ss_mac_dit/Configurations2/statusbar/
  inflating: doc_sig2_ss_mac_dit/Configurations2/accelerator/
              current.xml
   creating: doc_sig2_ss_mac_dit/Configurations2/floater/
   creating: doc_sig2_ss_mac_dit/Configurations2/popupmenu/
   creating: doc_sig2_ss_mac_dit/Configurations2/progressbar/
   creating: doc_sig2_ss_mac_dit/Configurations2/menubar/
   creating: doc_sig2_ss_mac_dit/Configurations2/toolbar/
   creating: doc_sig2_ss_mac_dit/Configurations2/images/Bitmaps/
  inflating: doc_sig2_ss_mac_dit/META-INF/macrosignatures.xml
  inflating: doc_sig2_ss_mac_dit/content.xml
  inflating: doc_sig2_ss_mac_dit/styles.xml
 extracting: doc_sig2_ss_mac_dit/meta.xml
  inflating: doc_sig2_ss_mac_dit/Thumbnails/thumbnail.png
  inflating: doc_sig2_ss_mac_dit/settings.xml
  inflating: doc_sig2_ss_mac_dit/META-INF/manifest.xml

ZZR:CreationSig lrv$ unzip -d  doc_sig1_ss_mac_dir
                        doc_sig1_ss_mac.odt
Archive:  doc_sig1_ss_mac.odt
 extracting: doc_sig1_ss_mac_dir/mimetype
   creating: doc_sig1_ss_mac_dir/Configurations2/statusbar/
  inflating: doc_sig1_ss_mac_dir/Configurations2/accelerator/
              current.xml
   creating: doc_sig1_ss_mac_dir/Configurations2/floater/
   creating: doc_sig1_ss_mac_dir/Configurations2/popupmenu/
   creating: doc_sig1_ss_mac_dir/Configurations2/progressbar/
   creating: doc_sig1_ss_mac_dir/Configurations2/menubar/
   creating: doc_sig1_ss_mac_dir/Configurations2/toolbar/
   creating: doc_sig1_ss_mac_dir/Configurations2/images/Bitmaps/
```

```
  inflating: doc_sig1_ss_mac_dir/META-INF/documentsignatures.xml
  inflating: doc_sig1_ss_mac_dir/content.xml
  inflating: doc_sig1_ss_mac_dir/styles.xml
 extracting: doc_sig1_ss_mac_dir/meta.xml
  inflating: doc_sig1_ss_mac_dir/Thumbnails/thumbnail.png
  inflating: doc_sig1_ss_mac_dir/settings.xml
  inflating: doc_sig1_ss_mac_dir/META-INF/manifest.xml
```

When exploring deeper into those two files, differences are noticeably more important. Whenever the signature is applied through the `Tools` → `Macros ...` menu, there is no signature component or information with respect to the files of the archive. This means that if the user sign wrongly signs the document when no macro is present (just mistaken the two methods), the document is in fact no protected. This is a major design weakness.

META-INF/macrosignatures.xml file content

```
<?xml version="1.0" encoding="UTF-8"?>
<document-signatures xmlns="urn:oasis:names:tc:opendocument:
xmlns:digitalsignature:1.0"><Signature xmlns="http://www.w3.
org/2000/09/xmldsig#" Id="ID_00ef00da00e5006600ac00c4004b00
6a00a4005a00a4008f008100950042005f"><SignedInfo>
<CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001
/REC-xml-c14n-20010315"/><SignatureMethod Algorithm="http://
www.w3.org/2000/09/xmldsig#rsa-sha1"/><Reference URI="#ID_00e
7006700d4008f00d7006c004a005a00a00032001b006500fd00c70084009c
"><DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig
#sha1"/><DigestValue>XRYZWS27mS5/uHZ7QzHVwHYBnIQ=</DigestValue>
</Reference></SignedInfo>
<SignatureValue>qV8syoItE7yeuKsHpd9IvkFqQVy2we8MMolmPTSZ/1oc1
yLWOxaHKgafRlRBgdG7kz30HIlM+6yUEPaTml1/XgDHXjf1e3VTDntsaxCc4W
+bsYcIicEUOf5fd7gkSAHMnsmN3wVORVmYnTBIDn7AaBLBVtSgrMVd1DDveUO
WmsE=</SignatureValue><KeyInfo><X509Data><X509IssuerSerial>
<X509IssuerName>E=jpfizaine@gmail.com,CN=Fizaine,OU=virology
research,O=Lrv,L=Poitiers,ST=France,C=FR</X509IssuerName>
<X509SerialNumber>14423087784341907841</X509SerialNumber>
</X509IssuerSerial>
<X509Certificate>MIIDnDCCAwWgAwIBAgIJAMgpGexN+amBMA0GCSqGSIb3DQE
BBAUAMIGRMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGRnJhbmNlMREwDwYDVQQHEwh
[.....................................................]
8T7sr7erG+EKwksN8Iyo9XHaI6B8K9nOX06BfW2KkYSx8WXf1++C3dWct01pFg==
</X509Certificate>
<X509Certificate>MIIDnDCCAwWgAwIBAgIJAMgpGexN+amBMA0GCSqGSIb3DQE
BBAUAMIGRMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGRnJhbmNlMREwDwYDVQQHEwh
[.....................................................]
```

```
8T7sr7erG+EKwksN8Iyo9XHaI6B8K9nOX06BfW2KkYSx8WXf1++C3dWct01pFg==
</X509Certificate>
<X509Certificate>MIIDnDCCAwWgAwIBAgIJAMgpGexN+amBMA0GCSqGSIb3DQE
BBAUAMIGRMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGRnJhbmNlMREwDwYDVQQHEwh
[.....................................................]
8T7sr7erG+EKwksN8Iyo9XHaI6B8K9nOX06BfW2KkYSx8WXf1++C3dWct01pFg==
</X509Certificate>
</X509Data></KeyInfo><Object><SignatureProperties><SignatureProperty
Id="ID_00e7006700d4008f00d7006c004a005a00a00032001b006500fd00c700
84009c" Target="#ID_00ef00da00e5006600ac00c4004b006a00a4005a00a40
08f008100950042005f">
<dc:date xmlns:dc="http://purl.org/dc/elements/1.1/">2009-01-31
T22:20:38</dc:date></SignatureProperty></SignatureProperties>
</Object></Signature></document-signatures>
```

This is to be compared to the first method (`File` → `Digital Signature...`):

META-INF/documentsignatures.xml file content

```
<?xml version="1.0" encoding="UTF-8"?>
<document-signatures xmlns="urn:oasis:names:tc:opendocument:xmlns:
digitalsignature:1.0"><Signature xmlns="http://www.w3.org/2000/09/
xmldsig#" Id="ID_0071008a00a00022000c00960045009a00a400930031005c
000500cf00900071"><SignedInfo><CanonicalizationMethod Algorithm=
"http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/><SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/><Reference
URI="Configurations2/accelerator/current.xml"><DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/><DigestValue>
2jmj7l5rSw0yVb/vlWAYkK/YBwk=</DigestValue></Reference><Reference
URI="content.xml"><Transforms><Transform Algorithm="http://www.w3.
org/TR/2001/REC-xml-c14n-20010315"/></Transforms><DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/><DigestValue>
9PZW/DAW2OdGeCZar71HwqLkkVo=</DigestValue></Reference><Reference
URI="styles.xml"><Transforms><Transform Algorithm="http://www.w3.
org/TR/2001/REC-xml-c14n-20010315"/></Transforms><DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/><DigestValue>
6j3xV9OoFUv9fjP6SttOiKO+vG0=</DigestValue></Reference><Reference
URI="meta.xml"><Transforms><Transform Algorithm="http://www.w3.
org/TR/2001/REC-xml-c14n-20010315"/></Transforms><DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/><DigestValue>
OT5hMqRDznKqynCkuw21X2TNfSE=</DigestValue></Reference><Reference
URI="Thumbnails/thumbnail.png"><DigestMethod Algorithm="http://
www.w3.org/2000/09/xmldsig#sha1"/><DigestValue>/PXPspxYoQzuGmzE
7D8u8jM2V/c=</DigestValue></Reference><Reference URI="settings.
xml"><Transforms><Transform Algorithm="http://www.w3.org/TR/2001
```

```
/REC-xml-c14n-20010315"/></Transforms><DigestMethod Algorithm="
http://www.w3.org/2000/09/xmldsig#sha1"/><DigestValue>O/jKf8oxm
Q1YrjoCxjxJWWPDYdU=</DigestValue></Reference><Reference URI="
#ID_00050069008400ee008f002b0044000a00bc00a900bc00dd00ee008600e
90032"><DigestMethod Algorithm="http://www.w3.org/2000/09/xml
dsig#sha1"/><DigestValue>EjfF4iS0dS0oPSVJImPd/iA8zhs=
</DigestValue></Reference></SignedInfo><SignatureValue>rdyhtM8fn
rZoK7EI00Vhayk47gceFDHcNSP3XSX9+FPHX7VUzWRws4zckWXzijvHMaHvwgsfo
EJdF3duvhJjh/e+Vdle0Fh/gXv8sHUxTm7f/mYO2N/yfqDN4d1gBQ62koSwgZDah
8KlpDMGuvUg9SYghyHyMcqLcjYqJXCQyik=</SignatureValue><KeyInfo>
<X509Data><X509IssuerSerial><X509IssuerName>E=jpfizaine@gmail.com,
CN=Fizaine,OU=virology research,O=Lrv,L=Poitiers,ST=France,C=FR
</X509IssuerName><X509SerialNumber>14423087784341907841
</X509SerialNumber></X509IssuerSerial>
<X509Certificate>MIIDnDCCAwWgAwIBAgIJAMgpGexN+amBMA0GCSqGSIb3DQE
BBAUAMIGRMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGRnJhbmNlMREwDwYDVQQHEwh
[..........................................................]
8T7sr7erG+EKwksN8Iyo9XHaI6B8K9nOX06BfW2KkYSx8WXf1++C3dWct01pFg==
</X509Certificate>
<X509Certificate>MIIDnDCCAwWgAwIBAgIJAMgpGexN+amBMA0GCSqGSIb3DQE
BBAUAMIGRMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGRnJhbmNlMREwDwYDVQQHEwh
[..........................................................]
8T7sr7erG+EKwksN8Iyo9XHaI6B8K9nOX06BfW2KkYSx8WXf1++C3dWct01pFg==
</X509Certificate>
<X509Certificate>MIIDnDCCAwWgAwIBAgIJAMgpGexN+amBMA0GCSqGSIb3DQE
BBAUAMIGRMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGRnJhbmNlMREwDwYDVQQHEwh
[..........................................................]
8T7sr7erG+EKwksN8Iyo9XHaI6B8K9nOX06BfW2KkYSx8WXf1++C3dWct01pFg==
</X509Certificate>
</X509Data></KeyInfo><Object><SignatureProperties><SignatureProperty
Id="ID_00050069008400ee008f002b0044000a00bc00a900bc00dd00ee00860
0e90032" Target="#ID_0071008a00a00022000c00960045009a00a40093003
1005c000500cf00900071"><dc:date xmlns:dc="http://purl.org/dc/
elements/1.1/">2009-01-31T22:23:45</dc:date></SignatureProperty>
</SignatureProperties></Object></Signature></document-signatures>
```


It is obvious that signature is actually protecting the document in the case of the `File` → `Digital Signature...` method only. But it is worth mentioning that it is an apparent protection as we will show it in Section 5.

Now what about document containing macros? With the first method (`File` → `Digital Signature...`), the whole document content is signed, including existing macros. This is a significant evolution since in OpenOffice 2.x, macros themselves were not signed. As a consequence, attacks identified

in [6] are no longer working, at least directly. But as we will see in Section 5, it is still possible to bypass the signature use by *man-in-the-middle* attacks.

META-INF/documentsignatures.xml file content

```
<?xml version="1.0" encoding="UTF-8"?>
<document-signatures xmlns="urn:oasis:names:tc:
  opendocument:xmlns:digitalsignature:1.0">
  <Signature xmlns="http://www.w3.org/2000/09/
  xmldsig#" Id="ID_009400270095003e006c007d004
  d00e7009000e000e800de00d400f300a00015">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http:
     //www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/
      2000/09/xmldsig#rsa-sha1"/>
      <Reference URI="Configurations2/accelerator/
      current.xml">
        <DigestMethod Algorithm="http://www.w3.org/
      2000/09/xmldsig#sha1"/>
        <DigestValue>2jmj7l5rSw0yVb/vlWAYkK/YBwk=</DigestValue>
      </Reference>
      <Reference URI="content.xml">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/
      TR/2001/REC-xml-c14n-20010315"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/
      2000/09/xmldsig#sha1"/>
        <DigestValue>YKbZUeCcs0fJHmvT75X3PQurqwo=</DigestValue>
      </Reference>
        <Reference URI="Basic/Standard/script-lb.xml">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/TR/2001/
          REC-xml-c14n-20010315"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/
        09/xmldsig#sha1"/>
        <DigestValue>QT+KDg92khkio8BypiQmQZhKVQQ=</DigestValue>
      </Reference>
        <Reference URI="Basic/Standard/Mess.xml">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/TR/2001/
          REC-xml-c14n-20010315"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/
```

```
   09/xmldsig#sha1"/>
  <DigestValue>3IaXMPEv9oz1ClZRnwH2Z7KihmU=</DigestValue>
</Reference>
  <Reference URI="Basic/script-lc.xml">
  <Transforms>
    <Transform Algorithm="http://www.w3.org/TR/2001/
   REC-xml-c14n-20010315"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/2000/09/
   xmldsig#sha1"/>
  <DigestValue>xebEZyd1OwOyLL6Wp9QELPl+UEY=</DigestValue>
</Reference>
<Reference URI="styles.xml">
  <Transforms>
    <Transform Algorithm="http://www.w3.org/TR/2001/
   REC-xml-c14n-20010315"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/2000/09/
   xmldsig#sha1"/>
  <DigestValue>6j3xV9OoFUv9fjP6SttOiKO+vG0=</DigestValue>
</Reference>
<Reference URI="meta.xml">
  <Transforms>
    <Transform Algorithm="http://www.w3.org/TR/2001/
   REC-xml-c14n-20010315"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/2000/09/
   xmldsig#sha1"/>
  <DigestValue>1fAh4stpvpSZv+Vwsj3tjCjDCAk=</DigestValue>
</Reference>
<Reference URI="Thumbnails/thumbnail.png">
  <DigestMethod Algorithm="http://www.w3.org/2000/09/
   xmldsig#sha1"/>
  <DigestValue>fhbAJvPqzcJVM0ge79JhlWJX49k=</DigestValue>
</Reference>
<Reference URI="settings.xml">
  <Transforms>
    <Transform Algorithm="http://www.w3.org/TR/2001/
   REC-xml-c14n-20010315"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/2000/09/
   xmldsig#sha1"/>
  <DigestValue>LozqLt1yQvg6a2UVdza4IUC1p5Y=</DigestValue>
</Reference>
<Reference URI="#ID_00f500ab00ef00d800eb00ec0040003400
```

```
          be00d30052007a00dd00aa003100bc">
         <DigestMethod Algorithm="http://www.w3.org/2000/09/
          xmldsig#sha1"/>
         <DigestValue>jizkaarcRaqipxZFt+yZbtCkN9A=</DigestValue>
    </Reference>
    </SignedInfo>
    <SignatureValue>MQ+Zd5SaVTcahI0OojbcH/8BHCwDkjAodXhOF3s3g
     KriQjS+hdN9fbt/kTrjORfHI2cOuq7hS9edqWBb7dTkTQrj0n5mr8C4w
     rbEa6OQ8276GV89oi2O90e3ozfrBDK9OknG8h8WRRzagAqVruY+YUKg7
     tVHH2dlfZbXycAtnZg=</SignatureValue><KeyInfo><X509Data>
<X509IssuerSerial><X509IssuerName>E=jpfizaine@gmail.com,
CN=Fizaine,OU=virology research,O=Lrv,L=Poitiers,ST=France,
C=FR</X509IssuerName><X509SerialNumber>14423087784341907841
</X509SerialNumber></X509IssuerSerial>
<X509Certificate>MIIDnDCCAwWgAwIBAgIJAMgpGexN+amBMA0GCSqGSIb3DQ
EBBAUAMIGRMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGRnJhbmNlMREwDwYDVQQHE
whQb2l0aWVyczEMMAoGA1UEChMDTHJ2MRowGAYDVQQLExF2aXJvbG9neSByZXNl
[.....................................................]
T7sr7erG+EKwksN8Iyo9XHaI6B8K9nOX06BfW2KkYSx8WXf1++C3dWct01pFg==
</X509Certificate>
<X509Certificate>MIIDnDCCAwWgAwIBAgIJAMgpGexN+amBMA0GCSqGSIb3DQ
EBBAUAMIGRMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGRnJhbmNlMREwDwYDVQQHE
[.....................................................]
T7sr7erG+EKwksN8Iyo9XHaI6B8K9nOX06BfW2KkYSx8WXf1++C3dWct01pFg==
</X509Certificate>
<X509Certificate>MIIDnDCCAwWgAwIBAgIJAMgpGexN+amBMA0GCSqGSIb3DQ
EBBAUAMIGRMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGRnJhbmNlMREwDwYDVQQHE
[.....................................................]
T7sr7erG+EKwksN8Iyo9XHaI6B8K9nOX06BfW2KkYSx8WXf1++C3dWct01pFg==
</X509Certificate>
</X509Data></KeyInfo><Object><SignatureProperties>
<SignatureProperty Id="ID_00f500ab00ef00d800eb00ec0040003400be0
0d30052007a00dd00aa003100bc" Target="#ID_009400270095003e006c00
7d004d00e7009000e000e800de00d400f300a00015"><dc:date xmlns:dc=
"http://purl.org/dc/elements/1.1/">2009-01-31T22:33:43</dc:date>
</SignatureProperty></SignatureProperties></Object></Signature>
</document-signatures>
```

In the second method, the macro tree only is signed (including the macros). This is anothor major design weakness since documents signed in this way can be partially modified while the user is not aware of the fact that only the part relevant to macros are actually signed.

META-INF/macrosignatures.xml file content

```
<?xml version="1.0" encoding="UTF-8"?>
<document-signatures xmlns="urn:oasis:names:tc:
 opendocument:xmlns:digitalsignature:1.0">
  <Signature xmlns="http://www.w3.org/2000/09/
 xmldsig#" Id="ID_008800c70019008600b00073004300
 c300a6007d00c00069001300950059002d">
    <SignedInfo>
      <CanonicalizationMethod Algorithm=
 "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/
       2000/09/xmldsig#rsa-sha1"/>
      <Reference URI="Basic/Standard/script-lb.xml">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/TR/
          2001/REC-xml-c14n-20010315"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/
          2000/09/xmldsig#sha1"/>
        <DigestValue>QT+KDg92khkio8BypiQmQZhKVQQ=</DigestValue>
      </Reference>
      <Reference URI="Basic/Standard/Mess.xml">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/TR/
          2001/REC-xml-c14n-20010315"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/
          2000/09/xmldsig#sha1"/>
        <DigestValue>3IaXMPEv9oz1ClZRnwH2Z7KihmU=</DigestValue>
      </Reference>
      <Reference URI="Basic/script-lc.xml">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/TR/
          2001/REC-xml-c14n-20010315"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/
          2000/09/xmldsig#sha1"/>
        <DigestValue>xebEZyd1OwOyLL6Wp9QELPl+UEY=</DigestValue>
      </Reference>
      <Reference URI="#ID_00d4001900e4002a0032008f004b0023009d00
          c3003a002f00c000df002800ab"><DigestMethod Algorithm=
          "http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>tGsZaB7NQd3J2WUo0OfP9kUwUsc=</DigestValue></Reference>
</SignedInfo><SignatureValue>ivC3n5rd2Tac1mq3tp50cuj4higiEnArGpTEYs
wE6wjMsCCq81TcsO7xgAMbeRtqwYSV7FEXMvPuYH61ZJrIAjgfJjTGaXbePzqBawH+di
WZEHCUyvkjTfgdG0tsC5txPcvWNGk+c6dHO16pjJ8BX1jN6GrPWyqyyPn9Qvc8RXA=
```

```
</SignatureValue><KeyInfo><X509Data><X509IssuerSerial>
<X509IssuerName>E=jpfizaine@gmail.com,CN=Fizaine,OU=virology research,
O=Lrv,L=Poitiers,ST=France,C=FR</X509IssuerName>
<X509SerialNumber>14423087784341907841</X509SerialNumber></X509IssuerSerial>
<X509Certificate>MIIDnDCCAwWgAwIBAgIJAMgpGexN+amBMA0GCSqGSIb3DQ
EBBAUAMIGRMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGRnJhbmNlMREwDwYDVQQHE
whQb2l0aWVyczEMMAoGA1UEChMDTHJ2MRowGAYDVQQLExF2aXJvbG9neSByZXNl
[.....................................................]
T7sr7erG+EKwksN8Iyo9XHaI6B8K9nOX06BfW2KkYSx8WXf1++C3dWct01pFg==
</X509Certificate>
<X509Certificate>MIIDnDCCAwWgAwIBAgIJAMgpGexN+amBMA0GCSqGSIb3DQ
EBBAUAMIGRMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGRnJhbmNlMREwDwYDVQQHE
[.....................................................]
T7sr7erG+EKwksN8Iyo9XHaI6B8K9nOX06BfW2KkYSx8WXf1++C3dWct01pFg==
</X509Certificate>
<X509Certificate>MIIDnDCCAwWgAwIBAgIJAMgpGexN+amBMA0GCSqGSIb3DQ
EBBAUAMIGRMQswCQYDVQQGEwJGUjEPMA0GA1UECBMGRnJhbmNlMREwDwYDVQQHE
[.....................................................]
T7sr7erG+EKwksN8Iyo9XHaI6B8K9nOX06BfW2KkYSx8WXf1++C3dWct01pFg==
</X509Certificate>
</X509Data></KeyInfo><Object><SignatureProperties>
<SignatureProperty Id="ID_00d4001900e4002a0032008f004b0023009d0
0c3003a002f00c000df002800ab" Target="#ID_008800c70019008600b000
73004300c300a6007d00c0006900130095005900 2d"><dc:date xmlns:dc=
"http://purl.org/dc/elements/1.1/">2009-01-31T22:35:16</dc:date>
</SignatureProperty></SignatureProperties></Object></Signature>
</document-signatures>
```

Finally, let us stress on the fact that the `documentsignatures.xml` is itself not signed.

3 Viral Attacks Through Plain OpenOffice Documents

There is absolutely no significant changes compared to the previous version [2]. Simple archive manipulations (using zip/unzip utility and a simple text editor) enable to perform a lot of attacks.

If we intend to modify the document payload itself (the document visible text), the principle consists in modifying the information contained within the suitable tags. As an example, you can modify the content within the `<text>` tags in the `META-INF/manifest.xml` file. Since no integrity check is

performed, the modification remains unnoticed by the user.

To illustrate this, we modify the following original file at the `content.xml` file level

Orginal *content.xml* file (extract)

```
<?xml version="1.0" encoding="UTF-8"?>
<office:document-content xmlns:office="urn:oasis:names:tc:
opendocument:xmlns:office:1.0" xmlns:style="urn:oasis:
names:tc:opendocument:xmlns:style:1.0" xmlns:text="urn:
oasis:names:tc:opendocument:xmlns:text:1.0" xmlns:table=
[.......]
<text:p text:style-name="Standard">I am a simple
document.</text:p></office:text></office:body>
</office:document-content>
```

to produce the following (slightly) modified file

Modified *content.xml* file

```
<?xml version="1.0" encoding="UTF-8"?>
<office:document-content xmlns:office="urn:oasis:names:tc:
opendocument:xmlns:office:1.0" xmlns:style="urn:oasis:
names:tc:opendocument:xmlns:style:1.0" xmlns:text="urn:
oasis:names:tc:opendocument:xmlns:text:1.0" xmlns:table=
[.......]
<text:p text:style-name="Standard">I am a simple simple
(modified) document.</text:p></office:text></office:body>
</office:document-content>
```

It is possible to modify any other format description data (XML file description) in the same way. We will not describe the possible attacks through unprotected (encryption and/or signed) documents. There is nothing new compared to OpenOffice.org version 2.x [2]. Among many other possibilities, the main powerful attack can then

- add non declared file (in particular one or more malicious macros). This is particularly interesting when considering data (file) theft. It is possible to insert stolen data into an OpenOffice.org file.
- macro substitution (replace macros with other malicious macros). This is the core approach for any malware attack.

From a technical point of view, the main point is to respect the XML structure and syntax in the file. Any incorrect modification will be detected during the XML parsing step. Any XML compliant modification will remain undetected.

4 Viral Attacks Through Encrypted OpenOffice Documents

In this section as well as in Section 5 we are going to explain how to exploit the users' trust in cryptographic primitives that are apparently enforced by OpenOffice.org. The core approach is to modify encrypted/signed document to fool the users in such a way that no encryption bypassing or integrity violation has apparently occurred. In the case of encrypted documents, the user must still use his password to access the document while it has been infected during the transmission.

4.1 Critical Aspect of the `Content.xml` File

We have performed a lot of experiments in order to transparently bypass the encryption. As a general conclusion, as soon as the `content.xml` file is modified – or any XML information related to it, for example in the `manifest.xml` file –, the modification/attack partly or totally fails: either the document opening crashes or no password is required. In the latter case, even if the user succeeds in opening the document, the absence of password request is generally bound to trigger an alert in the user's mind since the document is supposed to be encrypted. Aside the critical impact of the `content.xml` file, a few other manipulation of encrypted documents seemed to failed while the latter file was not concerned. As an example, we failed to simply replace the `current.xml` file only.

Consequently there are a few significant differences compared to the previously identified weaknesses for OpenOffice.org version 2.x [6]. But those differences have a limited impact or no impact at all for all the other attacks.

However when trying to understand why some of those attacks failed while they were successful for the OpenOffice version 2.x, we discovered that the related difference was rather less significant than expected. In a first time, we detected a difference between a document produced by the application and a document forged by hand (or by a malware). The most surprising point lies in the fact that both seemingly have the same content.

```
ZZR:Etude lrv$ ls -l doc_application.odt doc_manual_zip_1.odt
-rw-r--r--  1 lrv  lrv  12456 14 jan 01:09 doc_manual_zip_1.odt
-rw-r--r--  1 lrv  lrv  11334 14 jan 00:42 doc_application.odt
ZZR:Etude lrv$
```

This clearly proves that there exist some additional information within the archive that have a significant impact. Let us recall firt that encrypted files cannot be compressed further (since the entropy is quite optimally reduced).

Now every file in an encrypted archive are themselves encrypted with the noticeable exception of the `META-INF/manifest.xml` file. The comparison of both archive contents unveils the difference.

```
  ZZR:Etude lrv$ unzip -l doc_application.odt
Archive:  doc_application.odt
  Length     Date   Time    Name
 --------    ----   ----    ----
       39  01-13-09 23:42   mimetype
        0  01-13-09 23:42   Configurations2/statusbar/
        2  01-13-09 23:42   Configurations2/accelerator/current.xml
        0  01-13-09 23:42   Configurations2/floater/
        0  01-13-09 23:42   Configurations2/popupmenu/
        0  01-13-09 23:42   Configurations2/progressbar/
        0  01-13-09 23:42   Configurations2/menubar/
        0  01-13-09 23:42   Configurations2/toolbar/
        0  01-13-09 23:42   Configurations2/images/Bitmaps/
      757  01-13-09 23:42   content.xml
     1912  01-13-09 23:42   styles.xml
      393  01-13-09 23:42   meta.xml
     4193  01-13-09 23:42   Thumbnails/thumbnail.png
     1269  01-13-09 23:42   settings.xml
     4502  01-13-09 23:42   META-INF/manifest.xml
 --------                   -------
    13067                   15 files
```

```
  ZZR:Etude lrv$ unzip -l doc_manual_zip_1.odt
Archive:  doc_manual_zip_1.odt
  Length     Date   Time    Name
 --------    ----   ----    ----
        0  01-14-09 01:09   Configurations2/
        0  01-14-09 01:09   Configurations2/accelerator/
```

```
        2  01-13-09 23:42   Configurations2/accelerator/current.xml
        0  01-13-09 23:42   Configurations2/floater/
        0  01-14-09 01:09   Configurations2/images/
        0  01-13-09 23:42   Configurations2/images/Bitmaps/
        0  01-13-09 23:42   Configurations2/menubar/
        0  01-13-09 23:42   Configurations2/popupmenu/
        0  01-13-09 23:42   Configurations2/progressbar/
        0  01-13-09 23:42   Configurations2/statusbar/
        0  01-13-09 23:42   Configurations2/toolbar/
      757  01-13-09 23:42   content.xml
        0  01-14-09 01:09   META-INF/
     4502  01-13-09 23:42   META-INF/manifest.xml
      393  01-13-09 23:42   meta.xml
       39  01-13-09 23:42   mimetype
     1269  01-13-09 23:42   settings.xml
     1912  01-13-09 23:42   styles.xml
        0  01-14-09 01:09   Thumbnails/
     4193  01-13-09 23:42   Thumbnails/thumbnail.png
--------                    -------
    13067                   20 files
```

While the total size of the files in both `doc_manual_zip_1.odt` and `doc_application.odt` archives are the same, five extra files are present in the first one, which constitutes a significant difference. To successfully bypass encryption, we suppose that we have to recreate the archive by hand (or by a malware) in such a way that those five extra files are not present. Instead of using the formely efficient command (for OpenOffice.org 2.x)

```
zip -r ../doc.odt *
```

we now use the following command instead

```
zip -r -x Configuarations2/ Configurations2/accelerator/ \
 Configurations2/images/ META-INF/ Thumbnails/ /doc.odt *
```

The result is a success as long as we do not modify the `content.xml` file.

4.2 Successful Attacks Through Encrypted OpenOffice 3.x Documents

Still a large number of attacks remains possible by means of encrypted OpenOffice 3.x documents. Due to lack of space, we will not details of of them. Figure 6 summarizes all of them (green cells), along with those

which now are failing (pink cells) while they were successful for OpenOffice 2.x [6].

We are going to focus on the most interesting one with respect to malware attacks: replacing all encrypted macros with malicious (of course unencrypted) macros. Whenever the intended recipient opens the infected, encrypted document, the latter asks for a password in the same way as it would for a clean, non infected archive. Everything looks correct and the recipient's trust in encryption turns against him. We will not cover the trivial case of inserting such malicious macros into an encrypted OpenOffice 3.x document without macro. This is performed very easily by first inserting the macro themselves and creating their references into the archive (in particular in the `manifest.xml` file) without forgetting to strictly comply to the XML syntax and structure.

Let us consider then the case which consists in replacing an encrypted macro with a plaintext (malicious) macro. First the original encrypted macro is simply replaced with the code of a malicious macro we intend to use for our attack:

Malicious substituted macro

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE script:module PUBLIC "-//OpenOffice.org//DTD
  OfficeDocument 1.0//EN" "module.dtd">
<script:module xmlns:script="http://openoffice.org/2000/
  script" script:name="Mess" script:language="StarBasic">
  REM  *****  BASIC  *****

Sub Main
msgbox(&quot;Hello man, I&apos;m the Gremlins hehehehe !!!&quot;)
End Sub
</script:module>
```

In a next step, we modify the `META-INF/manifest.xml` file to remove the encryption reference with respect to the macro only. For that purpose, we remove the following lines:

```
<manifest:encryption-data manifest:checksum-type="SHA1/1K"
  manifest:checksum="emEQ5qqz7rZWFM0AHmSEo828kWs=">
   <manifest:algorithm manifest:algorithm-name="Blowfish CFB"
    manifest:initialisation-vector="tEhzORcLihY="/>
   <manifest:key-derivation manifest:key-derivation-name="PBKDF2"
    manifest:iteration-count="1024"
```

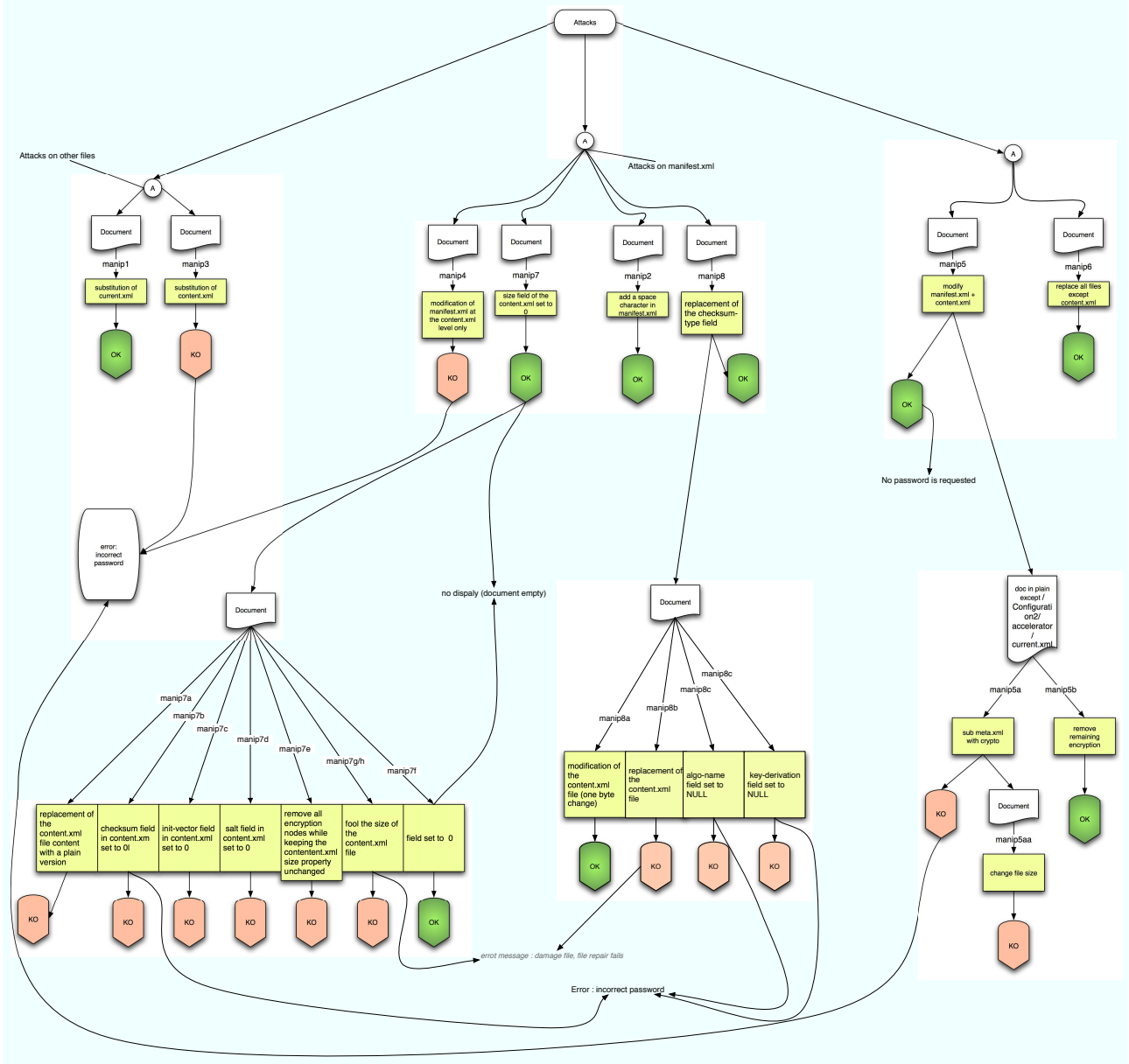


Figure 6: Summary of attacks through encrypted OpenOffice 3.x documents

```
      manifest:salt="jf7Cy6mFcQa0XH8IBVE3NQ==" />
  </manifest:encryption-data>
 </manifest:file-entry>
```

and we modify the line

```
  <manifest:file-entry manifest:media-type="text/xml"
   manifest:full-path="Basic/Standard/Mess.xml" manifest:size="368">
```

to obtain

```
<manifest:file-entry manifest:media-type="text/xml" manifest:
  full-path="Basic/Standard/Mess.xml"/>
```

Once the `META-INF/manifest.xml` has been modified, the archive is rebuilt. At the opening the password is requested and the document successfully opens while the malicious macro operates.

We can perform far more sophisticated attacks by replacing the whole tree of encrypted macros with a complete tree of malicious macros. The principle remains quite the same. We have to modify the `manifest.xml` file in such a way that we remove every reference to encryption with respect to any part of the macro tree.

5 Viral Attacks Through Digitally Signed OpenOffice Documents

From the different data presented in Section 2.3.4, it follows that most of the attacks that have been indentified in [6] are no longer valid for most of them. Let us summarize them:

- add macro to a signed document,
- replace macro with another macro.

But since critical files are not encrypted and especially there is no external secure management of digital signature keys and certificate (use of PKI), it is dramatically easy to forge fake `X509` certificate and play man-in-the-middle attacks:

1. Alice the sender signs her document and sends it to the legitimate recipient Bob.

2. Charlie the attacker intercepts the message and gets rid of the signature. Then he forges a fake Alice's certificate and public key and then impersonates Alice by signing the document as she would do. Let us note that Charlie could directly produce such a malicious document instead of intercepting an existing one. In both cases, Charlie sends the document to Bob in Alice's place (impersonation attacks).

3. Bob the recipient verifies the signature supposedly put by Alice. Everything is OK and Bob trusts in the document. Charlie's attack is successful simply by turning the signature confidence against Bob.

Let us now detail those three steps.

5.1 Step 1: Alice Signs Her Document

As in any signature scheme, Alice has first to produce her public/private keys and `X509` certificate. When checking the validity of the certificate, the OpenOffice application displays the following information to Alice (Figure 8 on left). More detailed information about it are available as well (Figure 8 on right). When checking the certificate, we obtain what Figure 7 displays.

Let us recall that Charlie can retrieve all this information by simply checking any document signed by Alice. This information will be very useful to him in order to forge a fake Alice's certificate.

5.2 Step 2: Charlie forges Alice's signature

Let us recall that Charlie can retrieve all those information by simply checking any document signed by Alice. Those information will be very useful to him in order to forge a fake Alice's certificate.

From a practical point of view, Charlie needs a lot of Alice's personal data to forge this certificate. Two different files in an OpenOffice archive are then critical. The first one is the `meta.xml` file which contains generic data about the signer. The second one is either the `documentsignatures.xml` file or the `macrosignatures.xml` depending on the method used to sign the document (see Section 2.3.6).

5.2.1 Recovering Alice's personal data

Here are the different steps to be performed by Charlie:

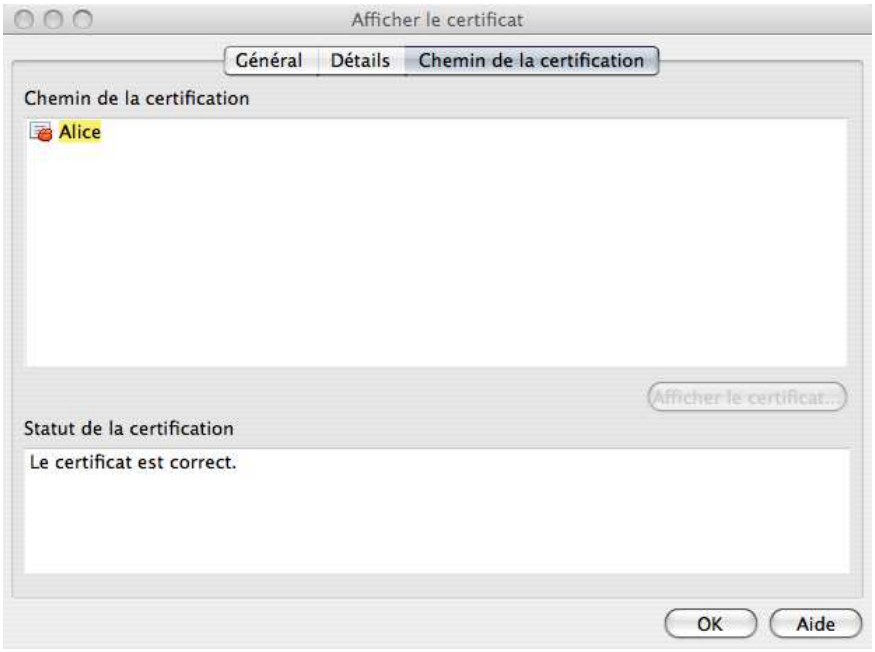


Figure 7: Alice's certificate (verification)

1. Unzip any document signed by Alice with the command `unzip -d AliceForBob_dir AliceForBob.odt`.

2. Extract generic Alice's data from the `meta.xml` file.

Alice's generic data

```
<?xml version="1.0" encoding="UTF-8"?>
<office:document-meta xmlns:office="urn:oasis:names:tc:opendocu
ment:xmlns:office:1.0" xmlns:xlink="http://www.w3.org/1999/xlin
k" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:meta="urn:
oasis:names:tc:opendocument:xmlns:meta:1.0" xmlns:ooo="http://
openoffice.org/2004/office" office:version="1.2"><office:meta>
<meta:creation-date>2009-02-28T21:56:37</meta:creation-date>
<meta:document-statistic meta:table-count="0" meta:image-count=
"0" meta:object-count="0" meta:page-count="1" meta:paragraph-
count="1" meta:word-count="15" meta:character-count="79"/><dc:
date>2009-03-01T14:23:02</dc:date><meta:editing-duration>PT16H
26M50S</meta:editing-duration><meta:editing-cycles>1</meta:edi
ting-cycles><meta:generator>OpenOffice.org/3.0$Unix OpenOffice
.org_project/300m9$Build-9358</meta:generator></office:meta>
</office:document-meta>
```

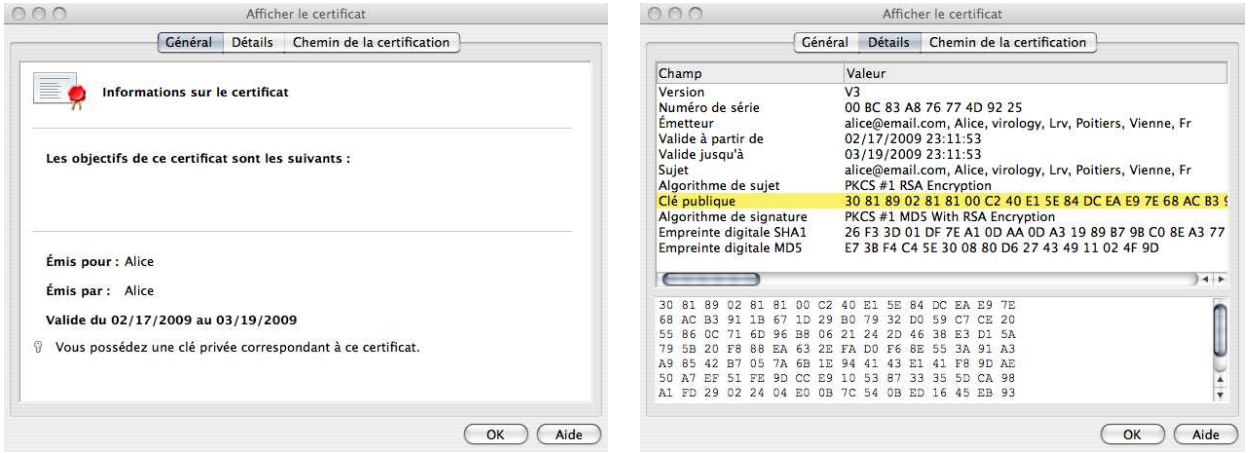


Figure 8: Alice's certificate (general) (left) - Alice's certificate (detailed view) (right)

Alice's generic data II

```
<?xml version="1.0" encoding="UTF-8"?>
<document-signatures xmlns="urn:oasis:names:tc:opendocu
ment:xmlns:digitalsignature:1.0"><Signature xmlns="http:
//www.w3.org/2000/09/xmldsig#" Id="ID_004100b600e2007300
3f003b0044001300b50089001300ab00ed001500280078"><Signed
Info><CanonicalizationMethod Algorithm="http://www.w3.
org/TR/2001/REC-xml-c14n-20010315"/><SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<Reference URI="Configurations2/accelerator/current.xml">
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmld
sig#sha1"/><DigestValue>2jmj7l5rSw0yVb/vlWAYkK/YBwk=
</DigestValue></Reference><Reference URI="content.xml">
<Transforms><Transform Algorithm="http://www.w3.org/TR/
2001/REC-xml-c14n-20010315"/></Transforms><DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>QCP8G2JsaWUm4txZoqB69HKhy4U=</DigestValue>
</Reference><Reference URI="styles.xml"><Transforms>
<Transform Algorithm="http://www.w3.org/TR/2001/REC-xml
-c14n-20010315"/></Transforms><DigestMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/><DigestValue>
6j3xV9OoFUv9fjP6SttOiKO+vG0=</DigestValue></Reference>
<Reference URI="meta.xml"><Transforms><Transform
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010
315"/></Transforms><DigestMethod Algorithm="http://www.
w3.org/2000/09/xmldsig#sha1"/><DigestValue>u7k0Zb4N3kBi
J2rXUNrfIN5Ugfs=</DigestValue></Reference><Reference
URI="Thumbnails/thumbnail.png"><DigestMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/><DigestValue>
9b0t2UV8EIOTWSXYv39IfIpl40M=</DigestValue></Reference>
<Reference URI="settings.xml"><Transforms><Transform
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010
315"/></Transforms><DigestMethod Algorithm="http://www.
w3.org/2000/09/xmldsig#sha1"/><DigestValue>atRt/IYEszjC
xuWS+yd4CKcGWIQ=</DigestValue></Reference><Reference
URI="#ID_002600ef004b00590098003a004b000700b20000000900
87005300770059 00b2"><DigestMethod Algorithm="http://www.
w3.org/2000/09/xmldsig#sha1"/><DigestValue>v1nIL2GhPwj4
rPeqJN6/ljYV/So=</DigestValue></Reference></SignedInfo>
<SignatureValue>ThKF4G4T8wyfd2JAzTCZJYFKCvwFk7fbRyw6JZw
fw1gIhdn9ACmXvkHvthn+kIKGNVd9cDNhagmGtfiFaRl+iqjXzjW6wy
kakqJCHfVr/Y2bMu9qjhuO0YHzggqrekIL/8qIcnVe/LQXzMa+5VtxC
```

```
YtwHjHkpqImkP5CN5z+gc4=</SignatureValue><KeyInfo><X509
Data><X509IssuerSerial><X509IssuerName>
E=alice@email.com,CN=Alice,OU=virology,
O=Lrv,L=Poitiers,ST=Vienne,C=Fr
</X509IssuerName><X509SerialNumber>1358
3886127840727589</X509SerialNumber>
</X509IssuerSerial>
<X509Certificate>MIIDbzCCAtigAwIBAgIJALyDqHZ3TZIlMA0GCSq
GSIb3DQEBBAUAMIGCMQswCQYDVQQGEwJGcjEPMA0GA1UECBMGVmllbm5
[.....................................................]
21zCILQyCQ==</X509Certificate>
<X509Certificate>MIIDbzCCAtigAwIBAgIJALyDqHZ3TZIlMA0GCSq
[.....................................................]
21zCILQyCQ==</X509Certificate>
<X509Certificate>MIIDbzCCAtigAwIBAgIJALyDqHZ3TZIlMA0GCSq
[.....................................................]
21zCILQyCQ==</X509Certificate>
</X509Data></KeyInfo><Object><SignatureProperties>
<SignatureProperty Id="ID_002600ef004b00590098003a004b00
0700b2000000090087005300770059 00b2" Target="#ID_004100b6
00e20073003f003b0044001300b50089001300ab00ed001500280078
"><dc:date xmlns:dc="http://purl.org/dc/elements/1.1/">
2009-03-01T14:29:42</dc:date></SignatureProperty>
</SignatureProperties></Object></Signature>
</document-signatures>
```

We then recover the following data:

the email adress in the `E=` field,

the usual surname in the `CN=` field,

the organisation or company department in the `OU=` field,

the organisation or company name in the `O=` field,

the town in the `L=` field,

the state (USA) or province (other) in the `ST=` field,

the country in the `C=` field.

5.2.2 How to forge the signature

From the data collected, Charlie is now able to forge a new certificate to impersonate Alice's identity as well as a new public/private keys pair. He first has to generate a `X509` certificate by using the command:

```
openssl req -newkey rsa:1024 -x509 -keyout CharlieKeys.pem
   -out CharlieCert.cert
```

Then Charlie export the X509 certificate to the PKCS#12 format with the command:

```
openssl pkcs12 -export -in CharlieCertX509.pem
    -inkey CharlieKeys.pem -out CharlieCertX509.p12
```

Charlie then chooses a passphrase (as Alice would do) to activate Alice's (fake) private key. Finally Charlie adds the forged certificate in the Firefox certificate path and signs the document as Alice usually do.

5.2.3 Charlie generates a malicious signed document

Charlie opens Alice's document with OpenOffice, copies its content and paste it into a new document. Then he adds a malicious macro (see Figure 9). Then he signs the new document using Alice's forged signature.

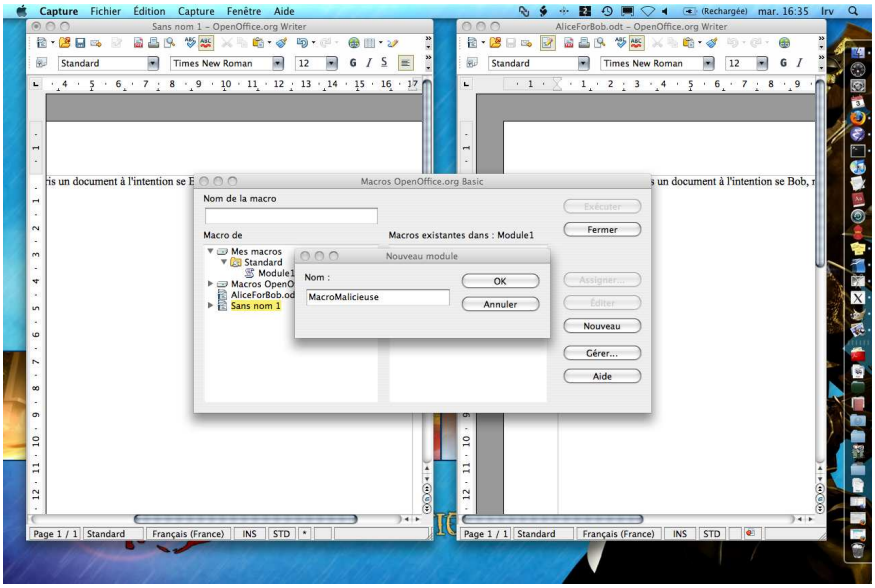


Figure 9: Charlie adds a macro

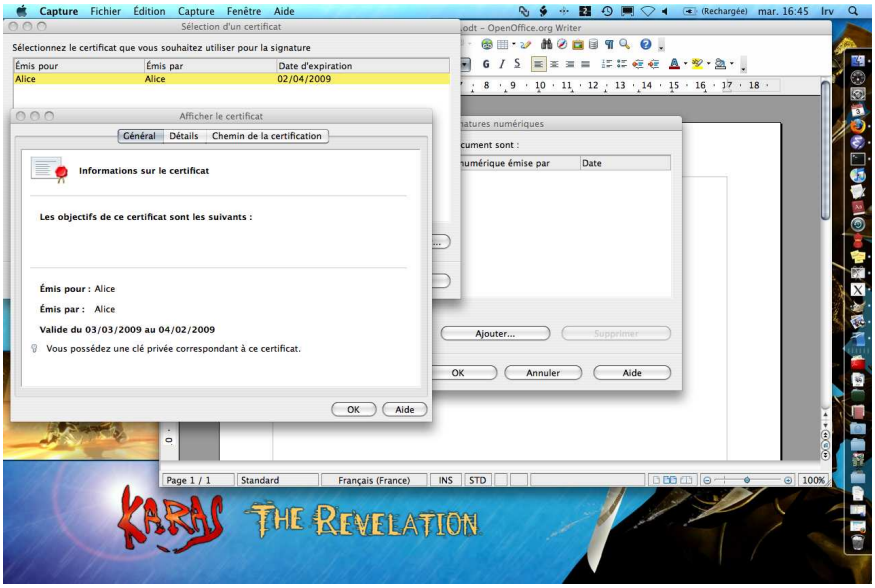


Figure 10: Charlie signs the document

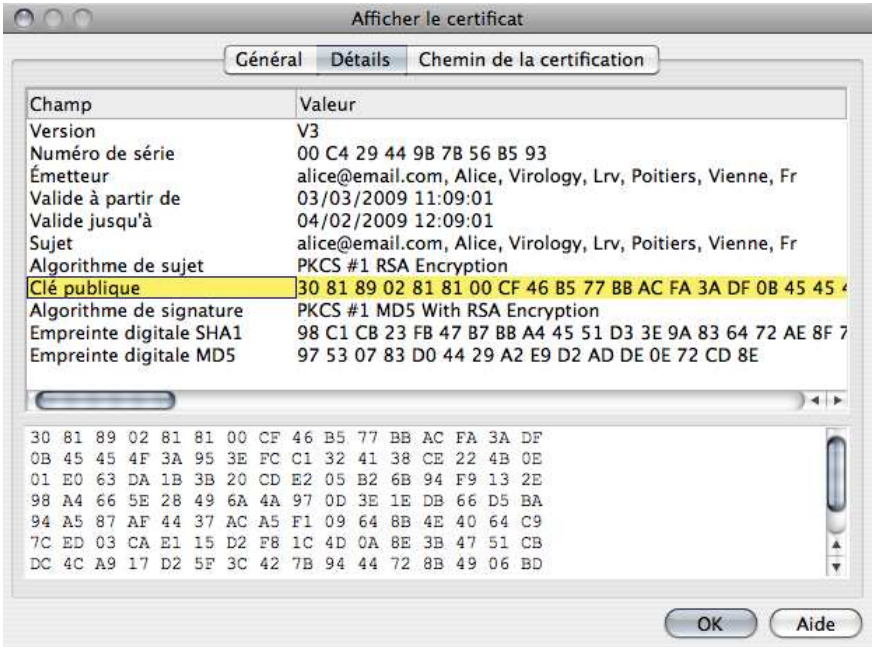


Figure 11: Charlie signs the document (II)

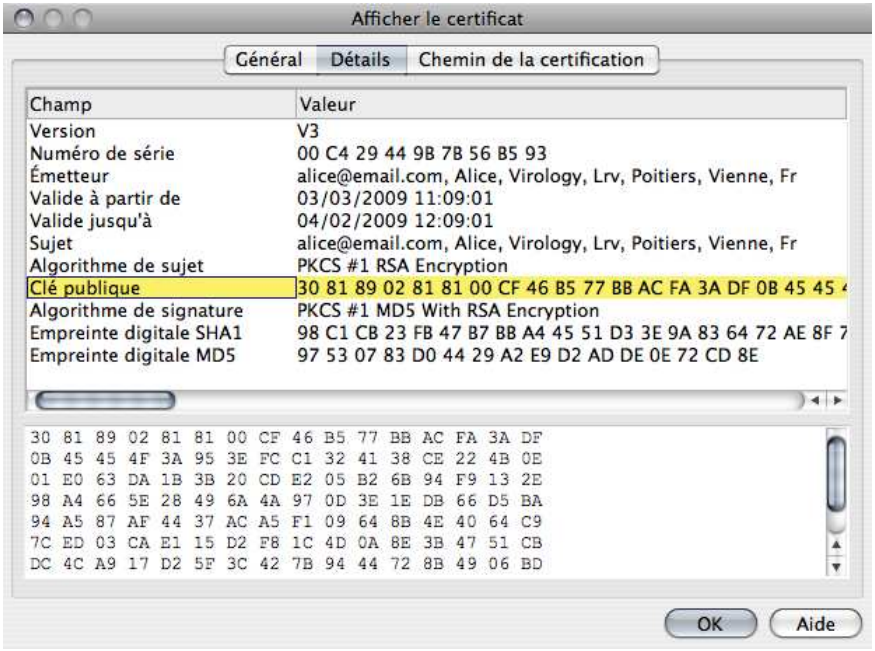


Figure 12: Detail of the forged certificate

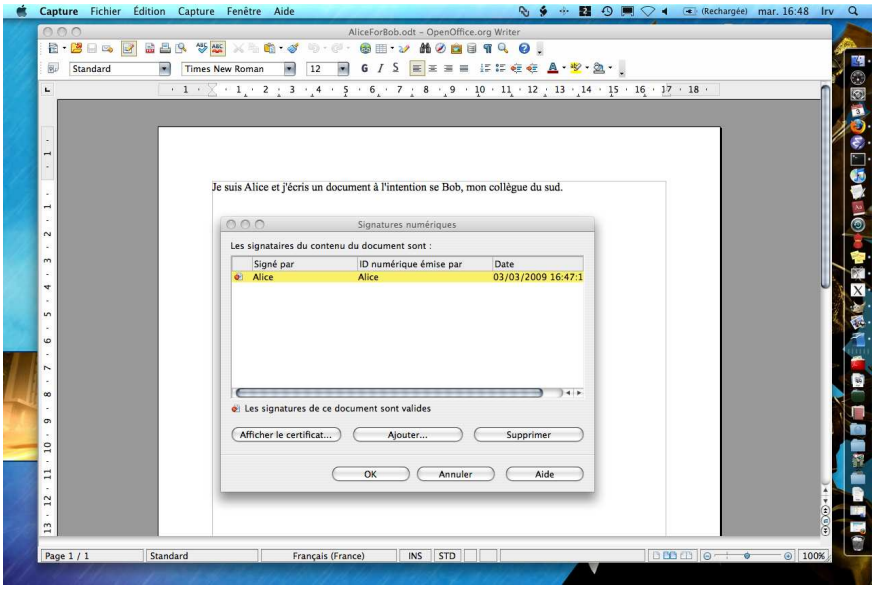


Figure 13: Charlie signs the document (III)

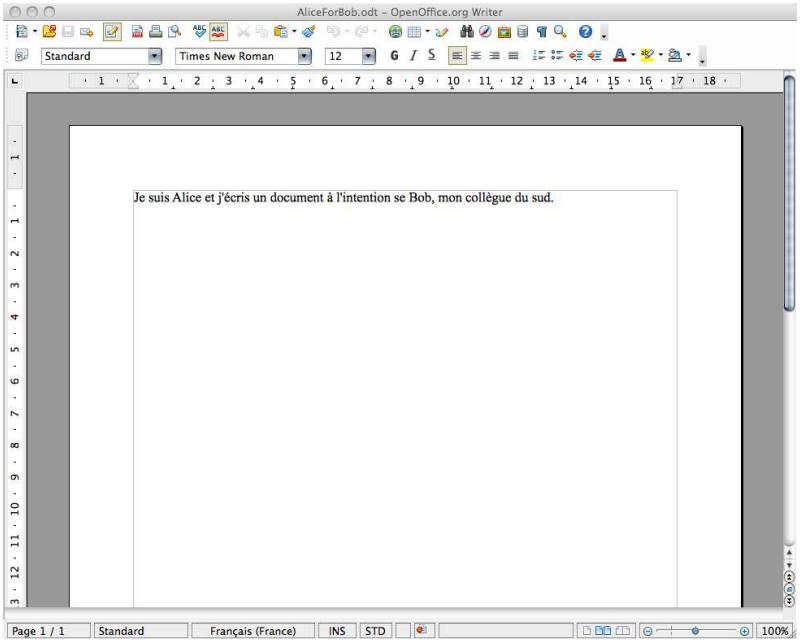


Figure 14: Charlie's final (malicious) document

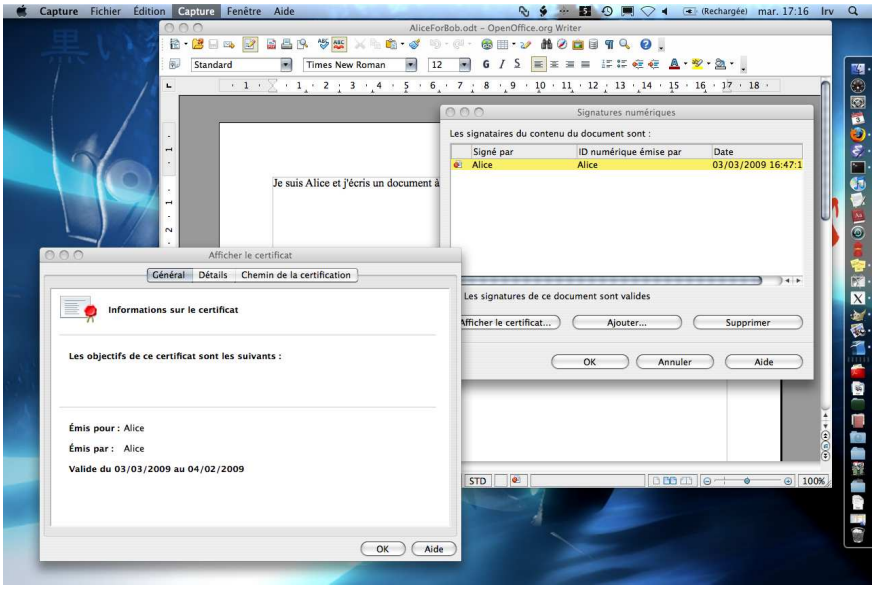


Figure 15: Certificate as seen by Bob

5.3 Step 3: Bob Checks Alice's Signature

Once Bob has received the document sent by Alice, he opens it and sees that it is apparently signed by Alice. Then he checks Alice's certificate (Figure 15).

Advanced checking confirms that the document actually is signed by Alice (Figures 16 and 17). Nothing can alert Bob that the document has

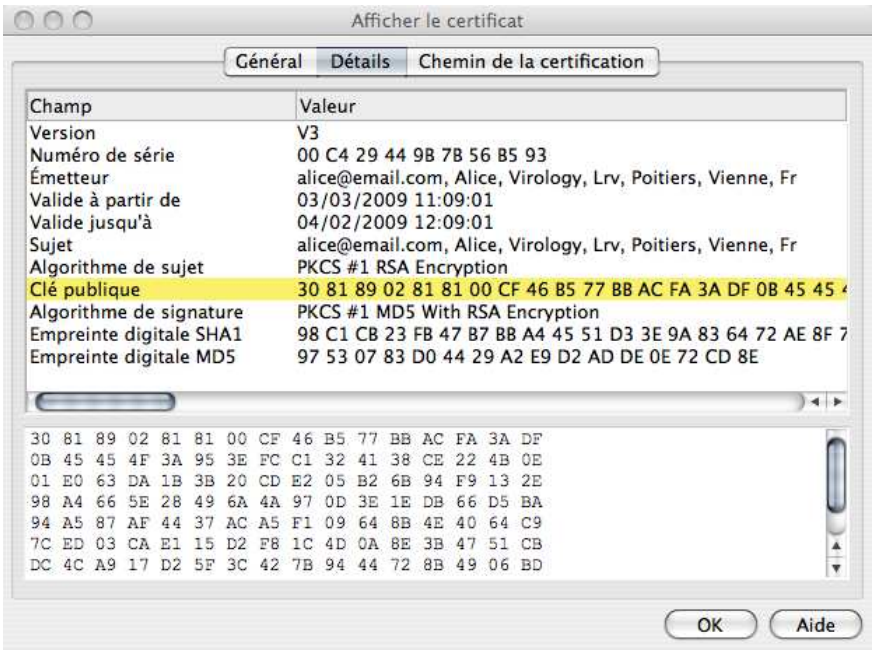


Figure 16: Detailed view of the certificate as seen by Bob (I)

actually not been signed by Alice unless he compares the true Alice's public key (that he has previously stored) with the forged key.

Figure 17: Detailed view of the certificate as seen by Bob (II)

5.4 A Close Look at Certificates

Let us now have a closer look directly at the certificates to identify the differences

Alice's certificate

```
<?xml version="1.0" encoding="UTF-8"?>
<document-signatures xmlns="urn:oasis:names:tc:opendo
 cument:xmlns:digitalsignature:1.0"><Signature xmlns=
 "http://www.w3.org/2000/09/xmldsig#" Id="ID_004100b6
 00e20073003f003b0044001300b50089001300ab00ed00150028
 0078">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.
        w3.org/TR/2001/REC-xml-c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/
        2000/09/xmldsig#rsa-sha1"/>
      <Reference URI="Configurations2/accelerator/
        current.xml">
        <DigestMethod Algorithm="http://www.w3.org/
        2000/09/xmldsig#sha1"/>
```

```
  <DigestValue>2jmj7l5rSw0yVb/vlWAYkK/YBwk=
  </DigestValue>
</Reference>
<Reference URI="content.xml">
  <Transforms>
    <Transform Algorithm="http://www.w3.org/
  TR/2001/REC-xml-c14n-20010315"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/
  2000/09/xmldsig#sha1"/>
  <DigestValue>QCP8G2JsaWUm4txZoqB69HKhy4U=
  </DigestValue>
</Reference>
<Reference URI="styles.xml">
  <Transforms>
    <Transform Algorithm="http://www.w3.org/
  TR/2001/REC-xml-c14n-20010315"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/
  2000/09/xmldsig#sha1"/>
  <DigestValue>6j3xV9OoFUv9fjP6SttOiKO+vG0=
  </DigestValue>
</Reference>
<Reference URI="meta.xml">
  <Transforms>
    <Transform Algorithm="http://www.w3.org/
  TR/2001/REC-xml-c14n-20010315"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/
  2000/09/xmldsig#sha1"/>
  <DigestValue>u7k0Zb4N3kBiJ2rXUNrfIN5Ugfs=
  </DigestValue>
</Reference>
<Reference URI="Thumbnails/thumbnail.png">
  <DigestMethod Algorithm="http://www.w3.org/
  2000/09/xmldsig#sha1"/>
  <DigestValue>9b0t2UV8EIOTWSXYv39IfIpl40M=
  </DigestValue>
</Reference>
<Reference URI="settings.xml">
  <Transforms>
    <Transform Algorithm="http://www.w3.org/
  TR/2001/REC-xml-c14n-20010315"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/
```

```
        2000/09/xmldsig#sha1"/>
        <DigestValue>atRt/IYEszjCxuWS+yd4CKcGWIQ=
        </DigestValue>
      </Reference>
      <Reference URI="#ID_002600ef004b00590098003a
        004b000700b2000000009008700530077005900b2">
        <DigestMethod Algorithm="http://www.w3.org/
        2000/09/xmldsig#sha1"/>
        <DigestValue>v1nIL2GhPwj4rPeqJN6/1jYV/So=
        </DigestValue>
    </Reference>
    </SignedInfo>
    <SignatureValue>ThKF4G4T8wyfd2JAzTCZJYFKCvwFk
    7fbRyw6JZwfw1gIhdn9ACmXvkHvthn+kIKGNVd9cDNhag
    mGtfiFaRl+iqjXzjW6wykakqJCHfVr/Y2bMu9qjhuOOYH
    zggqrekIL/8qIcnVe/LQXzMa+5VtxCYtwHjHkpqImkP5C
    N5z+gc4=</SignatureValue>
    <KeyInfo>
      <X509Data>
        <X509IssuerSerial>
          <X509IssuerName>E=alice@email.com,CN=Alice,OU=virology,
          O=Lrv,L=Poitiers,ST=Vienne,C=Fr</X509IssuerName>
<X509SerialNumber>13583886127840727589</X509SerialNumber>
        </X509IssuerSerial>
<X509Certificate>MIIDbzCCAtigAwIBAgIJALyDqHZ3TZI1MA0GCSqGSIb3DQ
[...............................................................]
VvUwaUExFCX01woZE3DKs0goSssiGOV0g21zCILQyCQ==</X509Certificate>
<X509Certificate>MIIDbzCCAtigAwIBAgIJALyDqHZ3TZI1MA0GCSqGSIb3DQ
[...............................................................]
VvUwaUExFCX01woZE3DKs0goSssiGOV0g21zCILQyCQ==</X509Certificate>
<X509Certificate>MIIDbzCCAtigAwIBAgIJALyDqHZ3TZI1MA0GCSqGSIb3DQ
[...............................................................]
VvUwaUExFCX01woZE3DKs0goSssiGOV0g21zCILQyCQ==</X509Certificate>
      </X509Data>
    </KeyInfo>
    <Object>
      <SignatureProperties>
        <SignatureProperty Id="ID_002600ef004b00590098003a004b0
        00700b2000000009008700530077005900b2" Target="#ID_004100
        b600e20073003f003b0044001300b50089001300ab00ed001500280
        078">
          <dc:date xmlns:dc="http://purl.org/dc/elements/1.1/">
        2009-03-01T14:29:42</dc:date>
        </SignatureProperty>
      </SignatureProperties>
```

```
    </Object>
  </Signature>
</document-signatures>
```

Charlie's forged certificate

```
<?xml version="1.0" encoding="UTF-8"?>
<document-signatures xmlns="urn:oasis:names:tc:opendocu
  ment:xmlns:digitalsignature:1.0">
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#"
  Id="ID_005f004b001e0070000f00eb004200720081003300ff00
     5d002b0088009e0064">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.
      org/TR/2001/REC-xml-c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/2000/
      09/xmldsig#rsa-sha1"/>
      <Reference URI="Configurations2/accelerator/
      current.xml">
        <DigestMethod Algorithm="http://www.w3.org/2000/
        09/xmldsig#sha1"/>
        <DigestValue>2jmj7l5rSw0yVb/vlWAYkK/YBwk=
        </DigestValue>
      </Reference>
      <Reference URI="content.xml">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/TR/
          2001/REC-xml-c14n-20010315"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/
          2000/09/xmldsig#sha1"/>
        <DigestValue>QCP8G2JsaWUm4txZoqB69HKhy4U=
          </DigestValue>
      </Reference>
      <Reference URI="Basic/Standard/MaliciousMacro.xml">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/TR/
          2001/REC-xml-c14n-20010315"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/
          2000/09/xmldsig#sha1"/>
        <DigestValue>9AxtmQVMkaMwIzw+pyRQmK2Pwhs=
          </DigestValue>
      </Reference>
```

```
<Reference URI="Basic/Standard/script-lb.xml">
  <Transforms>
    <Transform Algorithm="http://www.w3.org/TR/
    2001/REC-xml-c14n-20010315"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/
    2000/09/xmldsig#sha1"/>
  <DigestValue>Oj1iqlqXmednVoN53MCVkYHOAAE=
    </DigestValue>
</Reference>
<Reference URI="Basic/script-lc.xml">
  <Transforms>
    <Transform Algorithm="http://www.w3.org/
    TR/2001/REC-xml-c14n-20010315"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/
    2000/09/xmldsig#sha1"/>
  <DigestValue>xebEZyd1OwOyLL6Wp9QELPl+UEY=
    </DigestValue>
</Reference>
<Reference URI="styles.xml">
  <Transforms>
    <Transform Algorithm="http://www.w3.org/
    TR/2001/REC-xml-c14n-20010315"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/
    2000/09/xmldsig#sha1"/>
  <DigestValue>6j3xV9OoFUv9fjP6SttOiKO+vGO=
    </DigestValue>
</Reference>
<Reference URI="meta.xml">
  <Transforms>
    <Transform Algorithm="http://www.w3.org/
    TR/2001/REC-xml-c14n-20010315"/>
  </Transforms>
  <DigestMethod Algorithm="http://www.w3.org/
    2000/09/xmldsig#sha1"/>
  <DigestValue>jj3b9+WEJzgDs+JM/rlYzvYHROQ=
    </DigestValue>
</Reference>
<Reference URI="Thumbnails/thumbnail.png">
  <DigestMethod Algorithm="http://www.w3.org/
    2000/09/xmldsig#sha1"/>
  <DigestValue>9b0t2UV8EIOTWSXYv39IfIpl4OM=
    </DigestValue>
```

```
      </Reference>
      <Reference URI="settings.xml">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/
          TR/2001/REC-xml-c14n-20010315"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/
          2000/09/xmldsig#sha1"/>
        <DigestValue>1pWReD5PDj2CVDiOX92bxAdnbGo=
          </DigestValue>
      </Reference>
      <Reference URI="#ID_006e001b004e0072006700d
      8004c0080008d0048002d003c008b0088009000be">
        <DigestMethod Algorithm="http://www.w3.org/
        2000/09/xmldsig#sha1"/>
        <DigestValue>05Z0CUbnAf/FmEbEdXMNvoHUIyg=
        </DigestValue>
    </Reference>
    </SignedInfo>
<SignatureValue>T2Kr6bJsoUx0UdhGmVLZTbiFgHyxmzY
u9pTpDR+NtEY2+nhvJF4T892TQrkCDfGwVUn1OPtbbSbyhw
cd7G4h/Z/mtVkw5SZ6162XSL6FnINd5IMawhKrwtrW/tpFb
Moi1SbhWnLjEyU0rNf6jgl5xsnA3wzu9mnjGKyEm6taWHg=
</SignatureValue>
<KeyInfo>
  <X509Data>
    <X509IssuerSerial>
<X509IssuerName>E=alice@email.com,CN=Alice,OU=Virology,
O=Lrv,L=Poitiers,ST=Vienne,C=Fr</X509IssuerName>
<X509SerialNumber>14134904340058912147</X509SerialNumber>
    </X509IssuerSerial>
<X509Certificate>MIIDbzCCAtigAwIBAgIJAMQpRJt7VrWTMA0GCSqG
SIb3DQEBBAUAMIGCMQswCQYDVQQGEwJGcjEPMA0GA1UECBMGVmllbm51M
[......................................................]
3eGDMiHvnJxvZ7XYhZa5Wx7PgtC4NM3BP9GKg==</X509Certificate>
<X509Certificate>MIIDbzCCAtigAwIBAgIJAMQpRJt7VrWTMA0GCSqG
SIb3DQEBBAUAMIGCMQswCQYDVQQGEwJGcjEPMA0GA1UECBMGVmllbm51M
[......................................................]
3eGDMiHvnJxvZ7XYhZa5Wx7PgtC4NM3BP9GKg==</X509Certificate>
<X509Certificate>MIIDbzCCAtigAwIBAgIJAMQpRJt7VrWTMA0GCSqG
SIb3DQEBBAUAMIGCMQswCQYDVQQGEwJGcjEPMA0GA1UECBMGVmllbm51M
[......................................................]
3eGDMiHvnJxvZ7XYhZa5Wx7PgtC4NM3BP9GKg==</X509Certificate>
  </X509Data>
</KeyInfo>
```

```
<Object>
  <SignatureProperties>
    <SignatureProperty Id="ID_006e001b004e0072006700d8004
    c0080008d0048002d003c008b0088009000be" Target="#ID_00
    5f004b001e0070000f00eb00420072008100330 0ff005d002b008
    8009e0064">
      <dc:date xmlns:dc="http://purl.org/dc/elements/1.1/
    ">2009-03-03T16:47:15</dc:date>
    </SignatureProperty>
  </SignatureProperties>
</Object>
  </Signature>
</document-signatures>
```

5.5 Conclusion

The attack is not only very efficient but also completely transparent for Bob. Charlie has forged a fake cerificate with the help of Alice's collected data. He has also generated a pair of public/private keys and has just to impersonate Alice's identity. He will additionally modify a few data in the `meta.xml` file to play the attack more efficiently but it is important to mention that those data are not taken into account whenever the signature is verified by Bob. Those data modifications just intend to manage more suspicious recipents. Finally, any operation performed by Charlie can be automated by a malware.

6 Conclusion: Enhancing OpenOffice Security

From all the study presented in this paper, it is clear that at the present time it is still to bypass cryptographic protections in OpenOffice 3.x and more worringly to turn them against the users to mount powerful malware attacks using the the confidence put in cryptography. Moreover, the "natural" confidence in free software as well as their attractivity both for technical and ideological reasons, could result in a dramatical underestimation of the risk attached to malware attacks.

At the present time, the use of OpenOffice documents for official use or critical use should be postponed until the suite manages security in a more acceptable way. But it is likely that targeted attacks will rise using the design weaknesses of the suite.

Is it possible to enhance OpenOffice security. The question is definitively positive provided that there is a will to do it. The best solution should be to consider the suggestion we made in [6, 5] to produce a trusted OpenOffice suite in which any site administrator could configure/disable/enable some features with respect to the security policy in place. A simple administrator password would be necessary. This would enable to disable macros as an exemple or to add proprietary plug-in for a better integrity management.

But it is also possible to change some aspects which would contribute to enhance the security. The list given hereafter is probably not exhaustive but it relates to some very important points:

- whenever cryptographic protections are used, the `manisfest.xml` and `meta.xml` files, as well as any other critical files, should be better protected (encrypted). This would prevent the attacker to access the internal structure of the archive and to manipulate it;
- in a more formal way, there is no semantic verification of the archive. The application is just performing a syntactic verification of the XML specification. Such a semantic verification should be considered and implemented.

As far as signature is concerned, there is no efficient protection possible without a public key infrastructure or something equivalent. This means that unless the OpenOffice project proposes an offer equivalent to what Microsoft is itself doing, no true efficient security will be possible. Archive manipulation and man-in-the-middle attacks will be otherwise always possible. The other solution is probably to close some parts of the application but this is clearly against the free software philosophy. As soon as any system is totally open to any user, it is also open to any attacker. There is no other solution to break that duality than at least partially closing what relates to security.

References

[1] Brauer, M., Weir, R. and McRae, M.(2008), OASIS Open Document Format for Office Applications, `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office`,

[2] De Drézigué, D., Fizaine, J.-P. and Hansma, N. (2006), In-depth Analysis of the Viral Threats with OpenOffice.org Documents, *Journal in Computer Virology*, (2)-3, 2006.

[3] Bartel, M. Boyer, J., Fox, B., LaMacchia, B. and Simon, E. (2008), XML Signature Syntax and Processing, http://www.w3.org/Signature/, Http://www.w3.org/TR/xmldsig-core/

[4] Filiol, E. and Fizaine, J.-P. (2007), Les virus applicatifs multiplate-formes, *MISC - Le journal de la Sécurité Informatique*, vol. 34, pp. 64–72.

[5] Filiol, E and Fizaine, J.-P. (2006), Le risque viral sous OpenOffice, *MISC - Le journal de la Sécurité Informatique*, vol. 27.

[6] Filiol, E and Fizaine, J.-P. (2007), OpenOffice security and viral risk, *Virus Bulletin*, part one september 2007, pp. 11–17, part two october 2007, pp. 8–12, `www.virusbtn.com` journal = VirusBulletin

[7] Filiol, E. (2007), Analyse du macro-ver OpenOffice/BadBunny, *MISC - Le journal de la Sécurité Informatique*, vol. 34, pp. 18–20.

[8] Filiol, E (2008), *Computer Viruses: from Theory to Applications*, IRIS International Series, 2nd edition, Springer Verlag France.

[9] Odf Toolkit Website, `http://odftoolkit.org/`

[10] OpenOffice website, `http://www.openoffice.org/dev_docs/features/3.0/#ODF_1.2_support`,