OpenDocument and Open XML security (OpenOffice.org and MS Office 2007)

Philippe Lagadec - philippe.lagadec(a)nc3a.nato.int

NATO/NC3A

Abstract. OpenDocument and Open XML are both new open file formats for office documents. OpenDocument is an ISO standard, promoted by OpenOffice.org and Sun StarOffice. Open XML is the new format for Microsoft Office 2007 documents, an ECMA standard. These two formats share the same basic principles: XML files within a ZIP archive, with an open schema, in contrast to good-old proprietary formats (MS Word, Excel, PowerPoint, ...). However, both of them suffer from many security issues, similar to previous Office formats: malicious people can still embed and hide malware (Trojan horses and viruses) thanks to macros, scripts, OLE objects and similar features. This paper shows the security issues with technical details, including XML and ZIP obfuscation techniques that may be used to bypass antiviruses, and describes how to design a filter to get rid of unwanted parts in a safe way.

1 Introduction

There has always been security issues with office file formats mostly because of their rich "active" features such as macros or OLE objects inclusion [SSTIC03], or because of hidden information leak from documents [OSSIR03]. More recently security researchers have increased their focus on these formats and associated applications to discover many implementation vulnerabilities [CVEMSO,CVEOOo].

The usual scenario is the following: a user receives a malicious document as attachment of an e-mail or through another means. Opening this file triggers some malicious code in a macro, an object or an exploit, with or without user interaction, and the result may end up to the machine compromise.

Currently for various reasons many users feel more secure with OpenOffice or MS Office 2007 than with former office suites. However, this might be a false feeling as Eric Filiol announced during SSTIC06 [ESAT1,ESAT2,ESAT3].

For example, in June 2006 a first "proof-of-concept" virus named Stardust showed that the threat was real against OpenOffice. As well, three vulnerabilities have been found in OpenOffice 2.0.2, allowing malicious code to bypass default security policy (bypass warning popup for macros and Java sandbox).

Open XML is a new format based on open standards, its specifications have just been published by ECMA and Microsoft in 2006, and no real complete security analysis has been published yet.

This article focuses on some security issues brought by these new formats and their associated office suites, and then proposes a few solutions to protect our systems against threats. It also shows that ZIP and XML bring additional obfuscation possibilities for malicious code, and highlights caveats to avoid for antiviruses and content filters.

The original publication is available at www.springerlink.com: http://dx.doi.org/10.1007/s11416-007-0060-2

1.1 Notes

These are the products and versions used for this analysis:

- OpenDocument specifications v1.0 [ODSPEC10]. The new version 1.1 [ODSPEC11] has not been analyzed yet.

- OpenOffice v2.2.0
- Open XML ECMA-376 specifications, final version published in December 2006: [OXSPEC].
- Microsoft Office 2007 version 12.0.4518.1014.
- All tests were performed on Windows XP SP2.

This is not a complete security analysis, some features such as encryption and digital signature have not been thoroughly tested.

This analysis is focused on security issues linked to native features of the file formats and applications. The goal was not to find vulnerabilities such as buffer overflows.

The research work that lead to this article was started at DGA/CELAR in France.

Previous versions of this work were published for the PacSec security conference in November 2006 [PACSEC06], then in French for the SSTIC security conference in June 2007 [SSTIC07].

In this document, OpenOffice.org may be called “OpenOffice” or “OOo”, and Microsoft Office “MS Office” or simply “Office”.

2 Two new “open” formats

OpenDocument v1 is the open format used by OpenOffice v2. It is also used by other office applications, like Sun StarOffice, Koffice or Abiword. Since May 2006 OpenDocument has become an ISO standard.

Office 2007 is the new version of Microsoft Office first released in December 2006. There are many changes in the product such as the user interface and the file formats. Open XML is the new default format for MS Office main applications: Word, Excel and PowerPoint. It is an ECMA standard since the end of 2006, and there is an ongoing process to make it an ISO standard.

OpenDocument and Open XML share similar characteristics:

- They are both based on open and widely used technologies: a document is mainly made of XML files in a compressed ZIP archive.
- Their specifications are open and freely available on the Internet: [ODSPEC10,ODSPEC11,OXSPEC].
- They have both been accepted as standards by international bodies: ISO for OpenDocument, ECMA for Open XML.
- They both handle common office document formats: text, spreadsheet, presentation, vector drawing.

By studying published specifications more closely, it is possible to highlight some differences:

- Open XML structure is more complex and feature-rich than OpenDocument. Microsoft’s goal was to handle all of the existing features of its Office suite; whereas, OpenDocument is more of a document model.
- OpenDocument specifications are “only” 700 pages long; whereas Open XML reach 6045 pages !
- Many of the most interesting features for security concerns, such as VBA macros, OLE objects or encryption, are not covered in the Open XML specifications. According to [ECMA] page 15, Microsoft and ECMA consider these features as proprietary technologies that are outside the scope of the Open XML standard. As a result, the Open XML file format produced by MS Office is not fully open, and the standard should only be considered as a subset of the file format. In practice, antiviruses and content analysis filters have to handle the whole format including these unspecified proprietary features.
- OpenDocument specifications are not yet complete, because this is still a work in progress rather than to protect technologies. More information can be found about security features in other documents, and the whole source code of OpenOffice.org is available.

Compared to proprietary file formats, security analysis of OpenDocument and Open XML is much easier thanks to their open status. However the complexity of their specifications and the lack of some details do not make it so straightforward.

3 A few words about exploits

Nowadays the most common threat concerning office documents is not linked to macros or OLE objects, but rather to exploits based on implementation vulnerabilities in office suites [CVEMSO,CVEOOo]. Since OpenDocument and Open XML are made of structured and well-defined XML instead of binary data, it can be assumed that implementation vulnerabilities linked to format decoding are much less likely to be found. Furthermore, if the office applications use strict XML schemas to validate data, malformed documents should not be able to trigger potential vulnerabilities.

However, OpenDocument and Open XML do not contain only XML: binary parts such as bitmap pictures and OLE objects are still present. Office suites also rely on external libraries to handle some types of data, and these libraries may be vulnerable.

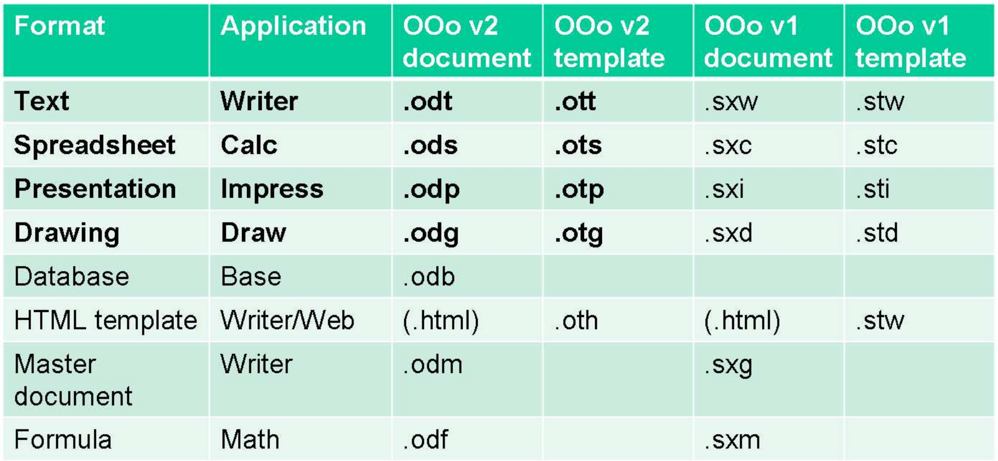
Another issue is that many third-party applications will be created to handle these formats thanks to their open specifications. As the formats are not so simple, it can be guessed that vulnerabilities will also be found in these applications.

4 OpenDocument and OpenOffice.org

4.1 The OpenDocument format

Current OpenDocument specifications [ODSPEC11] cover only a subset of the various file formats handled by the OpenOffice v2 suite. Only text documents, spreadsheets, presentations and vector drawings are described. Though other document types such as databases, HTML templates and mathematical formulas share a very similar structure.

The following table shows the extensions associated to most of the native formats used by OpenOffice v1 and v2. Formats described in the OpenDocument specifications are highlighted in bold.



Format	Application	OOo v2 document	OOo v2 template	OOo v1 document	OOo v1 template
Text	**Writer**	**.odt**	**.ott**	.sxw	.stw
Spreadsheet	**Calc**	**.ods**	**.ots**	.sxc	.stc
Presentation	**Impress**	**.odp**	**.otp**	.sxi	.sti
Drawing	**Draw**	**.odg**	**.otg**	.sxd	.std
Database	Base	.odb			
HTML template	Writer/Web	(.html)	.oth	(.html)	.stw
Master document	Writer	.odm		.sxg	
Formula	Math	.odf		.sxm	

4.2 Internal structure

Each document is stored in a ZIP compressed archive.

It is basically made of several XML files, which are found at the root of the archive or in subdirectories.

Here are the main XML files inside a common document:

- **content.xml**: document body

- **styles.xml**: style data
- **meta.xml**: metadata (author, title, ...)
- **settings.xml**: OOo settings for the document
- META-INF/**manifest.xml**: files description

Note that other *non-XML* files may also be stored in the archive:

- Pictures and thumbnails: JPEG, PNG, SVG, etc.
- Embedded charts/drawings/documents, OLE objects.

4.3 Macros

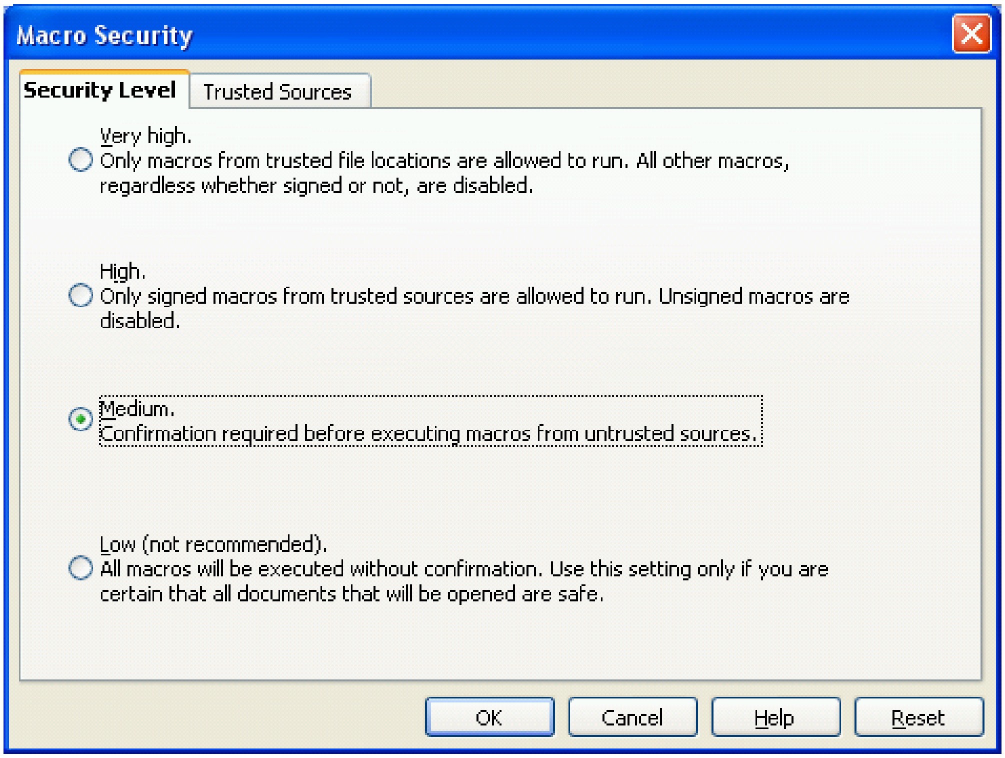
The most important security issue involves macros. OpenOffice v2 provides 4 different languages to write macros: Basic, JavaScript, Java (Beanshell), and Python. More languages may be added in the future. A regular OpenOffice installation contains these interpreters or relies on external ones like the JVM (Java Virtual Machine).

Each macro language gives access to a very powerful API, named UNO (Universal Network Objects), which is able to act on the operating system. Therefore it is possible to write effective malware. Furthermore, macros can be assigned to events and can be launched automatically while opening or reading a document.

At least 3 known attempts to write viruses using OpenDocument macros have been reported by antivirus vendors since 2006: Stardust, Starbugs and BadBunny. However, none of them had the ability to run without user confirmation.

To protect the user against malicious macros, OpenOffice provides **4 security levels**, quite similar to those of MS Office 2000/XP/2003:

- **Low** (to be avoided): no protection at all.
- **Medium** (default): macros can be enabled by the user before any access to the document (Simple popup warning).
- **High**: only signed macros or trusted directories are allowed. No warning if signature authority was already accepted or from a trusted location.
- **Very high**: only trusted locations, no signature, no warnings.



Macro Security

Security Level | Trusted Sources

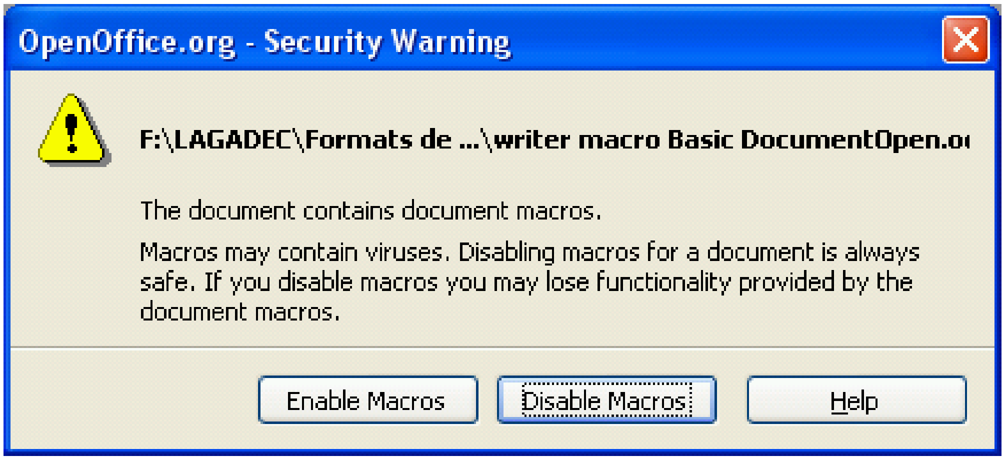
○ Very high.
Only macros from trusted file locations are allowed to run. All other macros, regardless whether signed or not, are disabled.

○ High.
Only signed macros from trusted sources are allowed to run. Unsigned macros are disabled.

● Medium.
Confirmation required before executing macros from untrusted sources.

○ Low (not recommended).
All macros will be executed without confirmation. Use this setting only if you are certain that all documents that will be opened are safe.

The default level is medium, so if there is a macro in a document a popup window asks the user whether he wants to enable or disable macros before he is allowed to view the file contents.



OpenOffice.org - Security Warning

F:\LAGADEC\Formats de ...\writer macro Basic DocumentOpen.o

The document contains document macros.

Macros may contain viruses. Disabling macros for a document is always safe. If you disable macros you may lose functionality provided by the document macros.

Notice that this is the same default level as in MS Office 97. Since version 2000, MS Office default level has been changed to high, and only signed macros are authorized to run.

However, the default security level should be changed to “High” in the next release OpenOffice v2.3 scheduled in September 2007.

In mid-2006, a weakness was found in OpenOffice 2.0.2 (and fixed since then), which could allow an attacker to bypass the macro security warning.

Macros Storage: Macros files are located in 2 different subdirectories inside an OpenDocument file:

- Basic macros are stored in XML files in the "Basic" directory of the archive.
- Java (Beanshell), JavaScript and Python macros are stored in script files, in the "Scripts" directory.

Examples:

- **Basic**/Standard/Module1**.xml**
- **Scripts**/beanshell/Library1/MyMacro**.bsh**
- **Scripts**/javascript/Library1/MyMacro**.js**
- **Scripts**/python/MyMacro**.py**

With OpenOffice v2 it is possible to create and edit macros from the application for Basic, Java and Javascript. Currently Python macros still require manual operations (see [OSFP]) : to include files in the archive, and edit manifest.xml.

Macros signature: One of the main issues found by the ESAT researchers is that macros are not signed with the rest of the document. Then it is possible to modify macros in a signed document and fool the user who thinks he is opening a genuine document [ESAT1,ESAT2,ESAT3].

VBA macros : Since version 2, OpenOffice has been able to read VBA macros source code in MS Office documents. Currently it is not possible to actually run VBA macros from OpenOffice.

When a MS Office document is converted to an OpenDocument format, VBA macros are stored in cleartext in a comment of an OOo Basic macro. Consequently, the document triggers the same warning as normal macros even if there is no active code. However, if the document is converted back to MS Office format, VBA macros are reactivated.

There is work in progress to provide native VBA macros support from OpenOffice with a UNO wrapper. In 2007 some alternative versions already provide limited support for Excel macros.

4.4 OLE objects

After macros, the second major security issue concerns embedded OLE objects. It is possible to store many types of OLE objects in OpenDocument files and to open them at least on Windows.

OLE objects are usually stored in binary files named "Object xxx" at the root of the archive, in Microsoft OLE2 format (also called structured storage). So even if OpenDocument is an open format, it can hold closed-format parts.

The most dangerous OLE object type is OLE Package: it may contain any file, including executables, or any command line. If the user double-clicks on the object, the file or the command is launched by the system, after a popup warning.

The warning before opening OLE Package objects only comes from Windows (packager.exe), not from OpenOffice. Therefore this can be a big security issue on old Windows systems. For example on Windows 2000 SP4, there is no warning at all.

In addition, a vulnerability was discovered in 2006 in Windows XP and 2003 [MS06-065], that makes it possible to trick the user into launching a command line. Just add a slash and a filename at the end of the command line, and the OLE Package object appears as if it was a harmless text file. Here is an example:

```
cmd.exe /c [...malicious commands...] /joke.txt
```

4.5 Scripts

It is possible to embed HTML scripts inside documents, in Javascript or VBscript language. These are not directly run by OpenOffice. However, if the document is saved as HTML, they may be launched by the browser, and they can trigger exploits.

Scripts are stored in content.xml, in a <text:script> tag, for example:

```
<text:script script:language=”JavaScript”>
alert(&quot;test script&quot;);
</text:script>
```

Source code of the script can be either directly included in the tag, or in an external script file, referenced by a “xlink:href” attribute.

4.6 Java applets

It is also possible to store Java applets in documents. This is not the same as macros written in Java language. These compiled applets run in a Java sandbox, from the inside of OpenOffice. Thanks to the sandbox, a malicious applet cannot do harmful things. However a sandbox can sometimes be evaded if there are vulnerabilities, as it was the case in OpenOffice 2.0.2.

4.7 URL links

A document can contain URL links. When the user clicks on them, OpenOffice opens the browser with the provided address if it is an external URL. Hopefully it seems not possible to launch javascript or vbscript URLs this way, because URLs starting with “javascript:...” and “vbscript:...” are filtered.

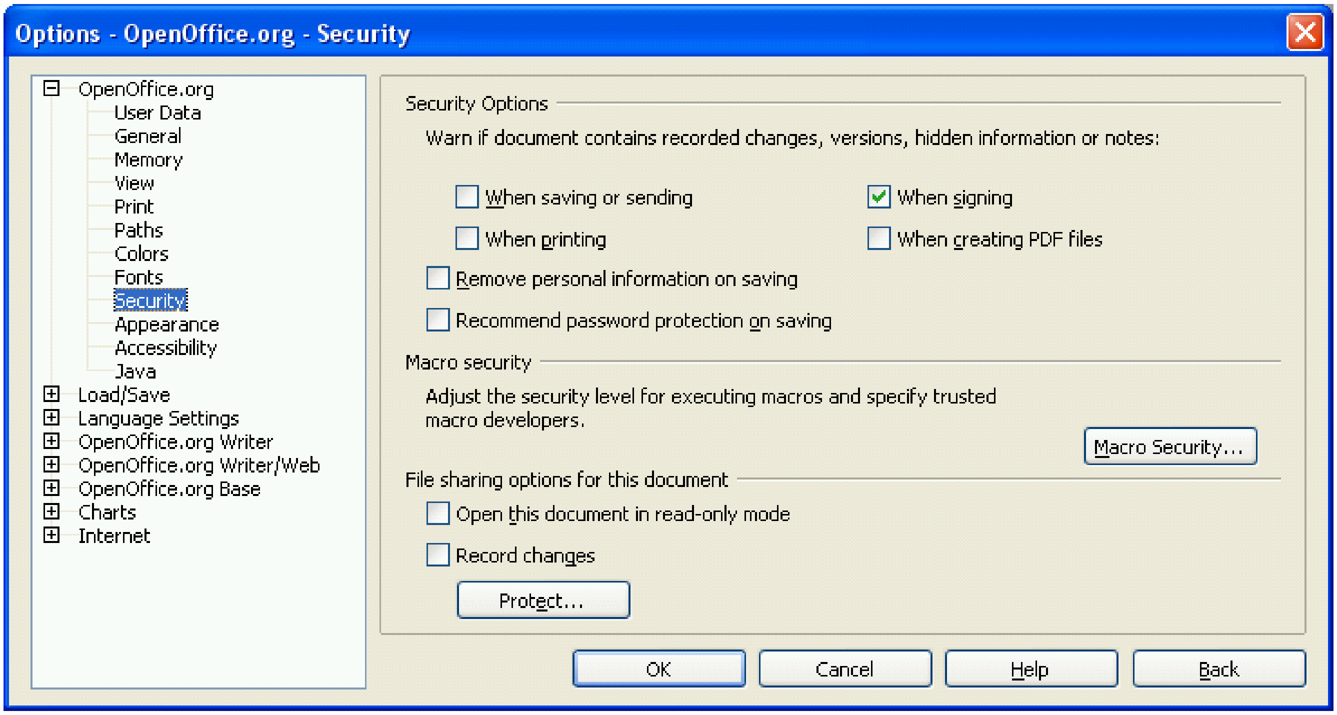
However it is still possible to direct the user to a malicious website, or to launch XSRF attacks (cross-site request forgery, [XSRF]) if the browser is already authenticated in a sensitive web application.

In addition, a vulnerability recently discovered in OpenOffice v2.1 made it possible to execute local commands on Linux and Solaris, by inserting escape characters for the shell in the URL [OOoURL].

4.8 Hidden data - Information leak

Just like MS Office, OpenDocument files may hide sensitive data to the user: metadata, hidden text, comments, revision marks, etc... That may become an problem when the document is published on the Internet, or sent outside the corporate network.

To address this issue, OpenOffice provides interesting features to warn about hidden information when signing, exporting as PDF or saving. However that is not sufficient in all cases, because this does not include data that may be hidden in OLE objects.



Security Options

Warn if document contains recorded changes, versions, hidden information or notes:

□ When saving or sending ☑ When signing

□ When printing □ When creating PDF files

□ Remove personal information on saving

□ Recommend password protection on saving

Macro security

Adjust the security level for executing macros and specify trusted macro developers.

File sharing options for this document

□ Open this document in read-only mode

□ Record changes

4.9 Conclusion about OpenDocument / OpenOffice security

There are many ways to include active content into OpenDocument files, and potentially to launch malicious code. Even if there are some protections to avoid this, none of them is absolutely safe. Furthermore, vulnerabilities are now regularly found in OpenOffice [CVEOOo], which requires to stay vigilant and often update the software.

Even if OpenDocument is an open format, files may sometimes hold closed-format parts, such as Microsoft OLE objects.

In conclusion, it can be said that OpenOffice is not absolutely more or less secure than Microsoft Office. We have seen that we can find similar security issues in both of them concerning malware or hidden data. There are of course some differences and some features are more secure in one product than the other.

However, it is really much simpler to analyze and filter active content or hidden data in OpenDocument format than in usual proprietary office formats.

5 Open XML and Microsoft Office 2007

Open XML is the new default file format for the main MS Office 2007 applications: Word, Excel and PowerPoint. Open XML files have new extensions:

- **Word**: .doc**x**, .doc**m**, .dot**x**, .dot**m**
- **Excel**: .xls**x**, .xls**m**, .xlt**x**, .xlt**m**, .xls**b**, .xla**m**
- **PowerPoint**: .ppt**x**, .ppt**m**, .pps**x**, .pps**m**
- Access: .accdb (new binary format, not Open XML)

Old binary formats from previous Office versions (OLE2) can still be read and written thanks to a “compatibility mode”.

A converter pack is also freely available to read and write Open XML documents from MS Office 2000, XP and 2003. Most of the issues described below concern these versions if the converter pack is installed.

5.1 Open XML structure

Basically an Open XML document is a ZIP compressed archive with XML files just like OpenDocument. However, the structure is a little more complex. It follows the new Microsoft Open Packaging Conventions, or OPC (see [OXSPEC] part 2), shared by other formats like XPS [XPS] (XPS is a new Microsoft format similar to PDF).

In an OPC archive, each data file is called a **part**. The type of each part is described in a file “**[Content_Types].xml**” at the root of the archive. There are also “**.rels**” files that store indirect relationships between parts. There are no direct relationships unlike in OpenDocument.

In Open XML documents, data is stored in several XML files, which are different for each application. Here is an example for a Word document:

- word/**document.xml**: document body
- word/**styles.xml**: style data
- word/**settings.xml**: settings for the document
- docProps/**app.xml** and **core.xml**: metadata (author, title, ...)

There may also be optional binary files:

- Pictures and other media: JPEG, PNG, GIF, TIFF, WMF, ...
- OLE objects, VBA macros, printer settings, ...

5.2 VBA macros

As for OpenDocument, the main security issue is that Open XML documents may contain macros, and these macros provide enough features to write effective malicious code. These are VBA macros just like previous MS Office formats.

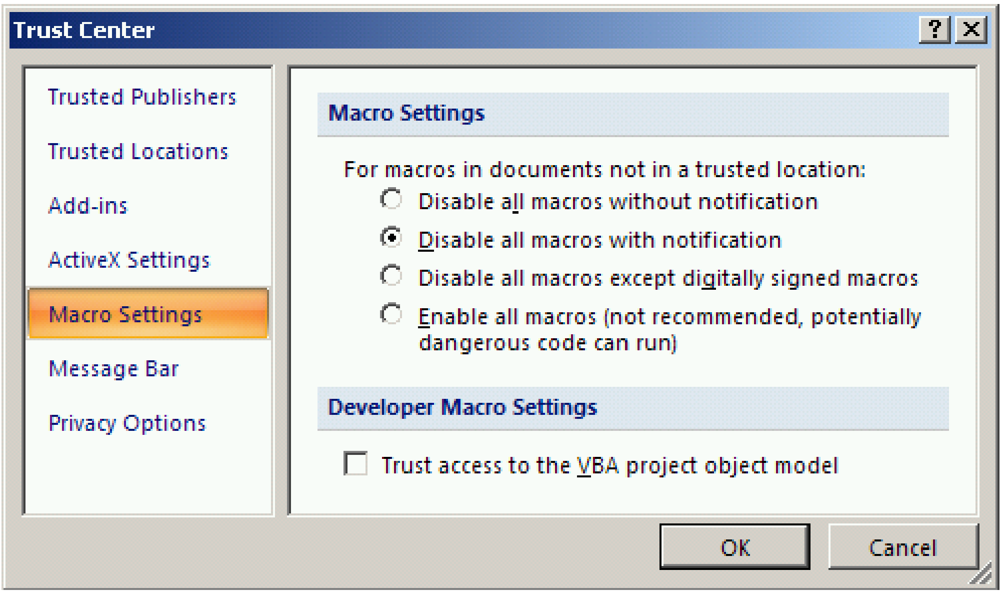
A big change is that MS Office 2007 distinguishes “normal” from “macro-enabled” documents. Normal documents end with an “X” (such as DOCX, XLSX, PPTX) and cannot contain macros. To store a macro in a document, it has to be saved in a “macro-enabled” format, which ends with an “M” (such as DOCM, XLSM, PPTM).

If a DOCM file is renamed to DOCX, it will be rejected by Office 2007 as “corrupted”.

Security levels : There is also a big change in Office 2007 macros security.

In previous Office versions, by default only signed or trusted macros can be launched by the user, because “high security” mode is on by default.

In Office 2007, there are no more medium or high security levels. The new default level is called “Disable all macros with notification”. And there are also new other levels, available in the new Trust Center which is a central place to set all security parameters.



Macro Settings

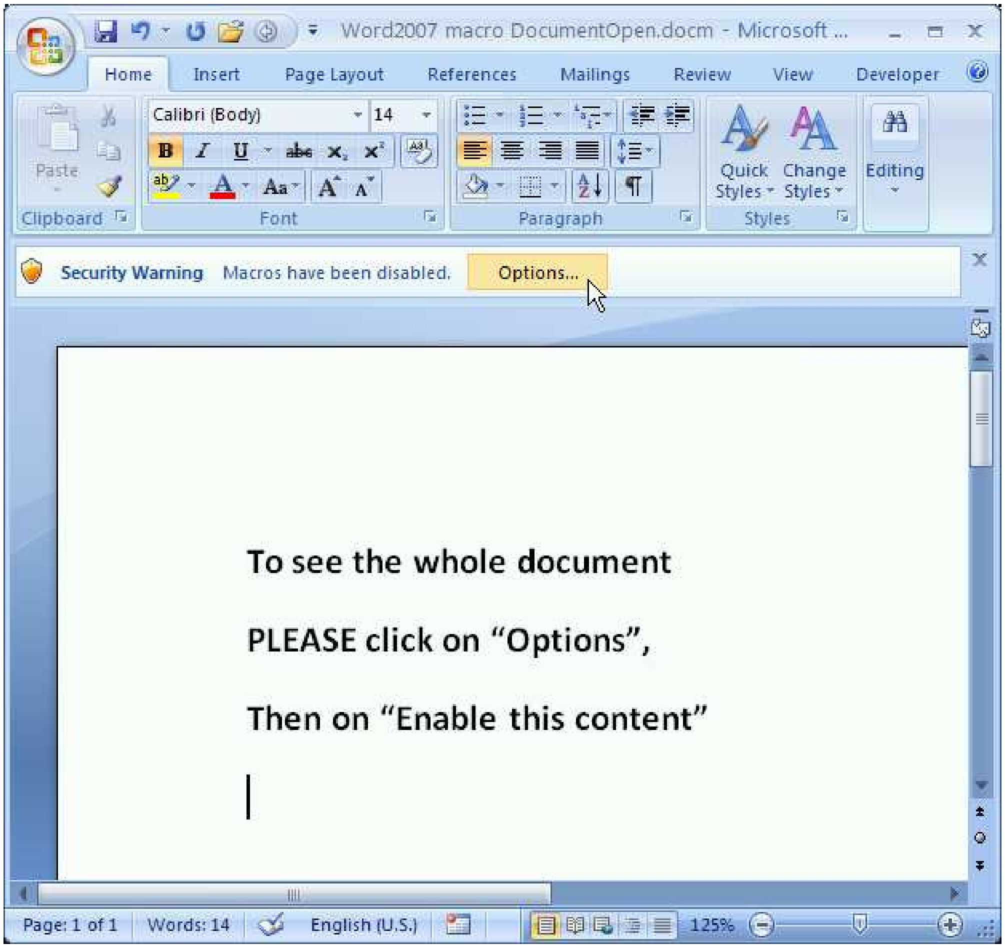
For macros in documents not in a trusted location:

- ○ Disable all macros without notification
- ● Disable all macros with notification
- ○ Disable all macros except digitally signed macros
- ○ Enable all macros (not recommended, potentially dangerous code can run)

Developer Macro Settings

□ Trust access to the VBA project object model

Whenever a user opens a document with macros, the content is displayed, macros are disabled, and a warning message appears under the “Ribbon”.



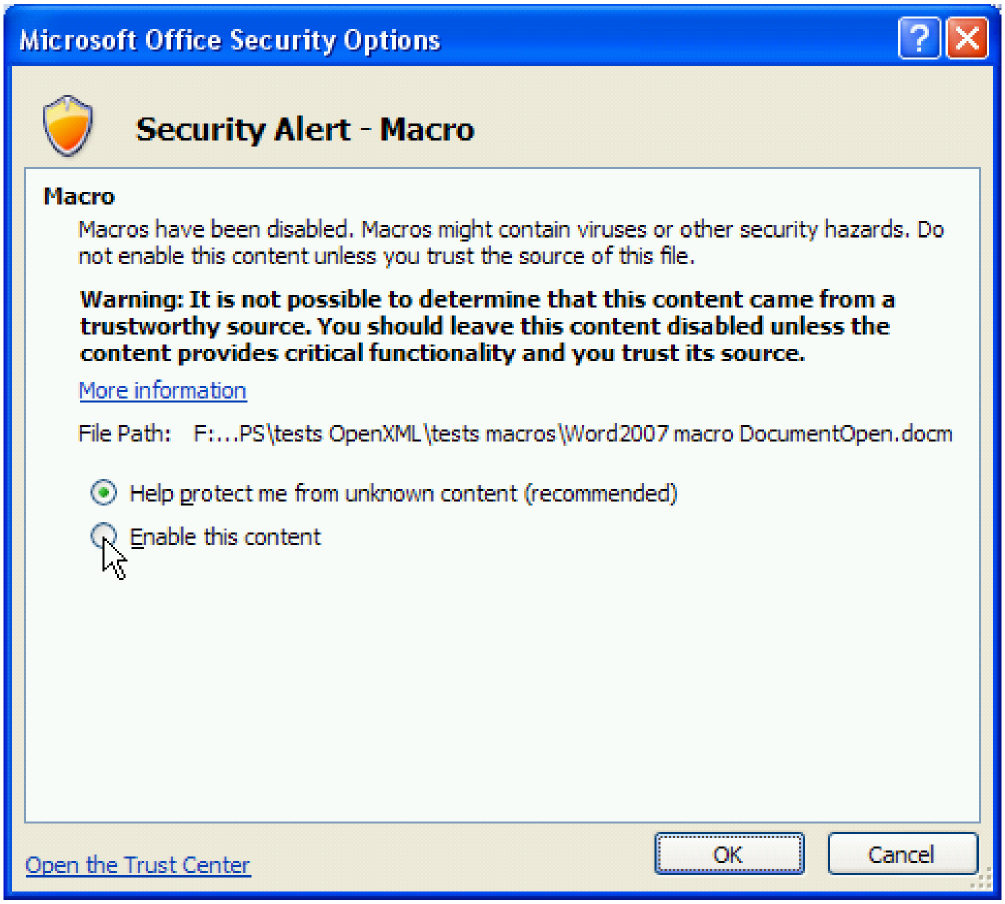
Security Warning Macros have been disabled. Options...

To see the whole document

PLEASE click on "Options",

Then on "Enable this content"

By clicking on this message the user obtains a window that allows him to enable macros be they signed or not.



Macro

Macros have been disabled. Macros might contain viruses or other security hazards. Do not enable this content unless you trust the source of this file.

Warning: It is not possible to determine that this content came from a trustworthy source. You should leave this content disabled unless the content provides critical functionality and you trust its source.

File Path: F:...PS\tests OpenXML\tests macros\Word2007 macro DocumentOpen.docm

● Help protect me from unknown content (recommended)

○ Enable this content

The new default security levels then allows a user to launch an unsigned macro with 3 clicks, if he doesn't read carefully all the warning messages.

Moreover, the document can be read before deciding to enable macros, which offers some social engineering possibilities.

Macros storage : VBA macros are stored in a file named vbaProject.bin, which path in the archive depends on the application:

- **Word**: word/vbaProject.bin
- **Excel**: xl/vbaProject.bin
- **PowerPoint**: ppt/vbaProject.bin

This is a binary file using Microsoft OLE2 format (structured storage), and this is not described in the current Open XML specifications [OXSPEC].

If a macro has a specific name, for example "Document_Open" for Word, it may be triggered automatically when the document is opened.

5.3 OLE objects

Like OpenDocument and previous Office versions, it is possible to store OLE objects in Open XML documents with the same security issues.

These objects are usually stored in their original format, in various places in the archive according to the application: for example “word/embeddings” for Word. An OLE Package object is stored using the binary OLE2 format.

Like macros, OLE objects storage is not described in the current Open XML specifications.

Some situations can lead to strange results. For instance, it is possible to store an Excel workbook with macros (.xlsm) as an object inside a normal Word document without macros (.docx). There is no warning when the Word document is opened. However, there is a popup window whenever the Excel object is activated by the user, which asks whether macros should be enabled or not. This is always the case even if the security level is “Disable all macros without notifications”...

5.4 Excel 2007 binary workbooks

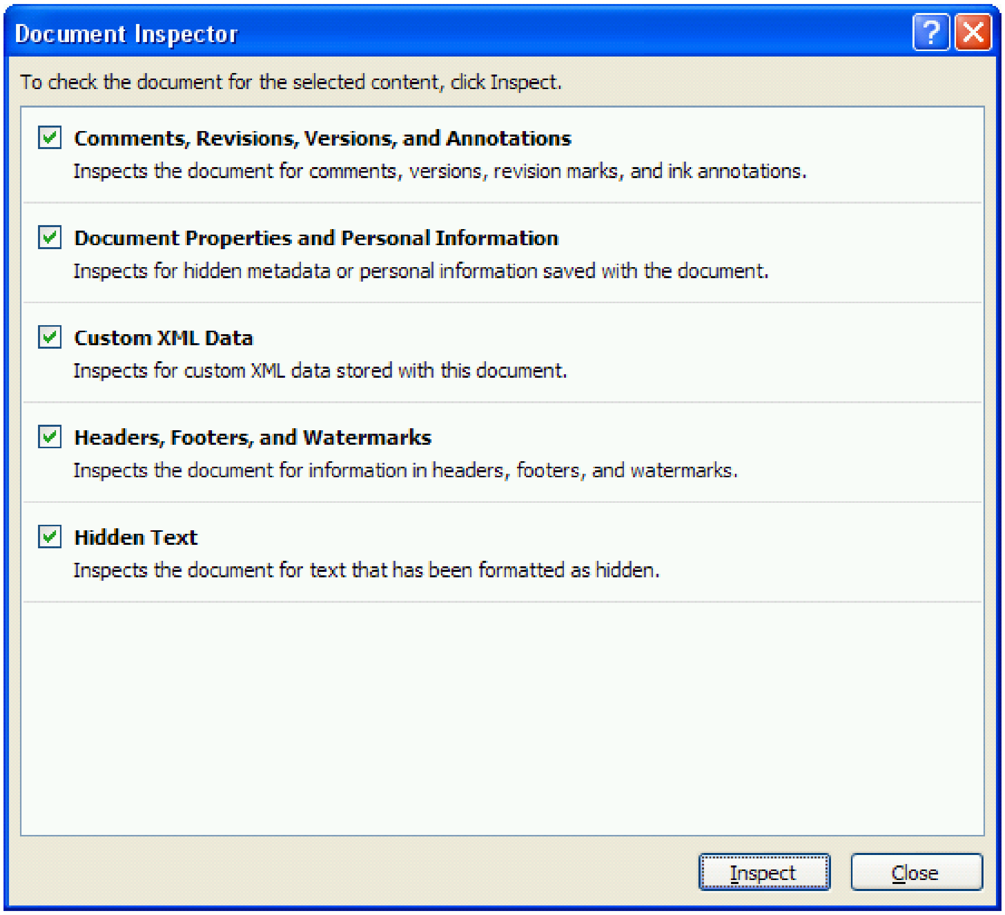
Excel 2007 can save a workbook in a hybrid Open XML format, called “binary workbook”, with extension “.xlsb”. This format is very close to Open XML, except that part of the data is stored in binary files instead of XML. This undocumented format looks like BIFF8 used by previous Excel versions.

Binary workbooks may contain macros.

5.5 Hidden data - Information leak

Office 2007 provides a new tool called “Document Inspector” to detect and remove several types of hidden data in a document. It is an improved version of the RHDTool that can be installed with Office 2003.

This is of course a very interesting feature. Just like OpenOffice, OLE objects are not detected as potential hidden data, so the results are not always fully accurate.



Document Inspector

To check the document for the selected content, click Inspect.

☑ **Comments, Revisions, Versions, and Annotations**
Inspects the document for comments, versions, revision marks, and ink annotations.

☑ **Document Properties and Personal Information**
Inspects for hidden metadata or personal information saved with the document.

☑ **Custom XML Data**
Inspects for custom XML data stored with this document.

☑ **Headers, Footers, and Watermarks**
Inspects the document for information in headers, footers, and watermarks.

☑ **Hidden Text**
Inspects the document for text that has been formatted as hidden.

5.6 Conclusion about Open XML and MS Office 2007 security

According to this analysis, the new Open XML format has the same malicious code and hidden data security issues as previous Office binary formats. And for some aspects such as macros, default security settings may even be more relaxed than before.

Even if Open XML is based on open specifications, documents produced by Microsoft Office 2007 can contain some parts in proprietary undocumented formats (MS OLE2 or BIFF for example), and these parts are often important for security matters.

Moreover, some major features such as macros are not part of the Open XML specifications, so the actual format cannot be considered as 100% open from a security point of view.

Like OpenDocument, Open XML is still much easier to analyze and filter than closed formats; however, its internal structure is more complex and requires more sophisticated processing.

6 How to protect against these security issues

There are two main complementary technical solutions to protect systems against malicious content and information leak due to office documents :

- To harden security settings of the office suites
- To filter documents on gateways or on removable devices

We do not mention organizational solutions which could consist for example in better informing users about malicious contents and information leak through documents.

6.1 Secure settings for OpenOffice and MS Office 2007

Here are a few generic principles to improve security settings of both office suites according to needs:

- Of course, start by applying security updates, if possible automatically.
- Harden the macros and ActiveX security levels at the highest position, according to the users *real* needs.
- If macros are necessary, use digital signature with a corporate PKI. If not possible, use trusted directories with a careful setup. (Avoid self-signed certificates, except for personal use)
- Disable all trusted directories if they are not used. At least disable those for which users have write access.
- Protect against OLE Package objects, which are rarely used:
  - forbid execution of “C:\Windows\System32\Packager.exe”, using file permissions.

Specific settings for MS Office 2007 :

- Disable notification messages, to avoid unsigned macros.
- Alternatively, it is even possible to disable the VBA engine, with some loss of functionality, by setting this registry key:
  - HKLM\SOFTWARE\Microsoft\Office\12.0\Common\VBAOff = 1
- Deploy the security settings on the network to improve administration tasks, for example using GPO. Look for “2007 Office System Administrative Templates” on Microsoft website to obtain the templates.
- If this is not possible, for example on a standalone workstation, protect security settings so that users cannot modify them, using HKLM registry keys or file/registry permissions. HKLM and HKCU registry keys for Office 2007 can be found in the templates mentioned above. Unlike former Office versions it seems that not all security settings can be protected using HKLM keys. For example the macros security level can always be modified by the user in HKCU, unless registry permissions are set to avoid it.
- For more details, Microsoft provides a document with some of these recommendations: http://go.microsoft.com/fwlink/?LinkID=85671

Specific settings for OpenOffice : For OpenOffice, security settings such as macros security level are located in XML files (.xcu) among other settings. At first global settings are read from XCU files in the application directory (Program Files), and then user-specific settings are read from the user profile (Documents and Settings). It is also possible to store some settings in a LDAP directory, to easily deploy them on a network.

By default, a user can override any setting in his profile. It is however possible to protect selected settings from users mistakes by adding “finalized” and “mandatory” attributes to XML tags in XCU files.

For example, it is possible to disable macros support for all users by adding the following lines to the file Common.xcu (in Program Files\OpenOffice.org 2.2\share\registry\data\org\openoffice\Office\). This will also hide macros security settings in the application.

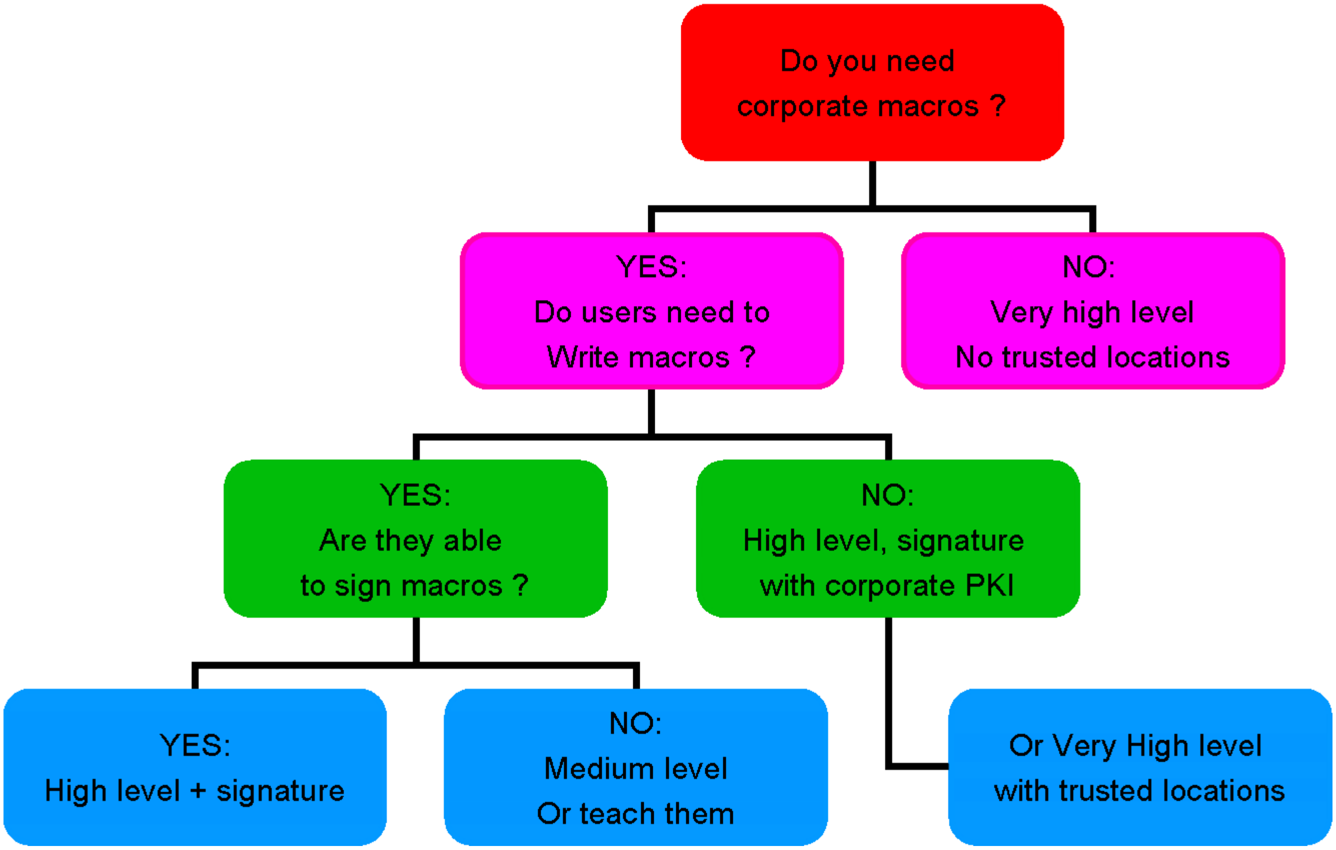
```
<node oor:name="Security">
<node oor:name="Scripting">
```

```
<prop oor:name="MacroSecurityLevel" oor:type="xs:int" oor:finalized="true" oor:mandatory="tru
<value>3</value>
</prop>
<prop oor:name="DisableMacrosExecution" oor:type="xs:boolean" oor:finalized="true" oor:mandato
<value>true</value>
</prop>
</node>
</node>
```

Please note this is not meant to obtain a fully trusted OpenOffice configuration on highly sensitive systems, as advanced users are usually able to run the application from alternate directories or override settings by editing some files. However this protection is always better than default settings.

Currently, there is no known tool and no comprehensive documentation to help administrators set security settings. The best way to discover available settings is to look at XCS files located in Program Files\OpenOffice.org 2.2\share\registry\schema\org\openoffice\, especially Common.xcs. The tag <group oor:name="Security"> holds most security settings with comments about their effect.

Here is an example of a decision tree to choose security settings for OpenOffice macros:



Do you need corporate macros ?
YES: Do users need to Write macros ?
NO: Very high level No trusted locations
YES: Are they able to sign macros ?
NO: High level, signature with corporate PKI
YES: High level + signature
NO: Medium level Or teach them
Or Very High level with trusted locations

6.2 Filtering documents - Content analysis and Antiviruses

Documents can be filtered on a gateway (for e-mail, web, file transfers, ...) or on removable devices. It can consist in a usual antivirus analysis or more sophisticated filtering process. An example of such a process would be to remove all active content from documents (macros, scripts, OLE objects, ...) or hidden data.

As both OpenDocument and Open XML use standard technologies such as ZIP and XML, it can be assumed that it is easy to analyze and filter these formats using common tools and libraries. In effect all active elements such as macros and objects can be easily found.

Sample filter for OpenDocument: To remove all active content:

- Macros: remove any file in Basic and Scripts directories.
- OLE objects : remove any file whose name starts with “Object”.
- In content.xml:
  - remove OLE objects: <draw:object-ole>
  - remove scripts : <text:script>
  - remove applets : <draw:applet>
  - update any tag linked to macros, for example: <office:event-listeners>

Sample filter for Open XML: To remove all active content:

- Macros: remove any file “vbaProject.bin” and “vbaData.xml”
- OLE objects: remove any file “*.bin”

Very simple filter for OpenDocument and Open XML in Python: Here is a very simple filter in Python, which only removes potentially active files in a document.

```python
import zipfile, sys
try:
        infile = zipfile.ZipFile(sys.argv[1], "r")
        outfile = zipfile.ZipFile(sys.argv[2], "w")
except:
        sys.exit("usage: %s infile outfile" % __file__)
for f in infile.infolist():
        fname = f.filename.lower()
        if  not fname.startswith("scripts") \
        and not fname.startswith("basic") \
        and not fname.startswith("object") \
        and not fname.endswith(".bin") :
                data = infile.read(f.filename)
                outfile.writestr(f, data)
```

6.3 Bypassing antiviruses and filters

As we have seen, it is very tempting to analyze or filter these open formats with simple techniques using common ZIP tools and libraries, or text-based search in XML files.

However, an attacker may obfuscate malicious content in several ways to bypass a filtering gateway or an antivirus. For this it is only necessary to use native features from office suites, XML and ZIP formats. Here are some potential obfuscation tricks:

Rename Open XML documents with macros : At first, file extensions alone are not reliable to filter out Open XML documents with macros. Even if a ".docx" file can never contain macros, it is always possible to rename a ".docm" document to ".doc" if ".docm" is blocked on a gateway.

Rename OpenDocument macros : It is possible to rename files containing macros inside an OpenDocument archive to replace .xml, .bsh, .js or .py with any other extension. For this it is only necessary to edit manifest.xml and content.xml to change the links to the macro file. In conclusion, file extension is not a good criteria to detect macros inside OpenDocument.

Hopefully, directories "Basic" and "Scripts" have fixed names in the current implementation and they only contain files related to macros, so this is a safe way to detect or remove macros.

Rename VBA macros in Open XML: Because of the modular structure of Open XML (OPC), it is possible to rename the file "vbaProject.bin" which contain macros with any name. For example in a Word document:

- rename "vbaProject.bin" to "no_macros_here.txt"
- update relationships in "word/_rels/document.xml.rels"
- in "[Content_Types].xml", replace "bin" by "txt"

This simple manipulation allows to bypass the Python filter shown above, keeping macros active. Therefore it is not possible to rely on filenames to detect macros in Open XML. A safer solution is to use a real XML parser to detect "vbaProject" and "vbaData" parts in [Content_Types].xml (or "oleObject" for OLE objects). Another solution would be to analyze each file content looking for binary OLE2 header, with potential false-positives.

Open XML - US-ASCII encoding and "obfuscation bit": Like Internet Explorer (cf. [IEASCII]), Office 2007 handles "US-ASCII" encoding rather strangely: All characters which ASCII code is above 127 have their 8th bit simply removed before XML content is parsed. Therefore this behaviour allows a very simple obfuscation against filters/antiviruses which wouldn't check it. Here is an example where <HIDDENTAG> tags are obfuscated:

```
<?xml version="1.0" encoding="us-ascii" standalone="yes"?>
1/4HIDDENTAG3/4 malware[...] 1/4/HIDDENTAG3/4
```

Note: in this example "1/4" and "3/4" respectively represents characters with ASCII codes 188 and 190.

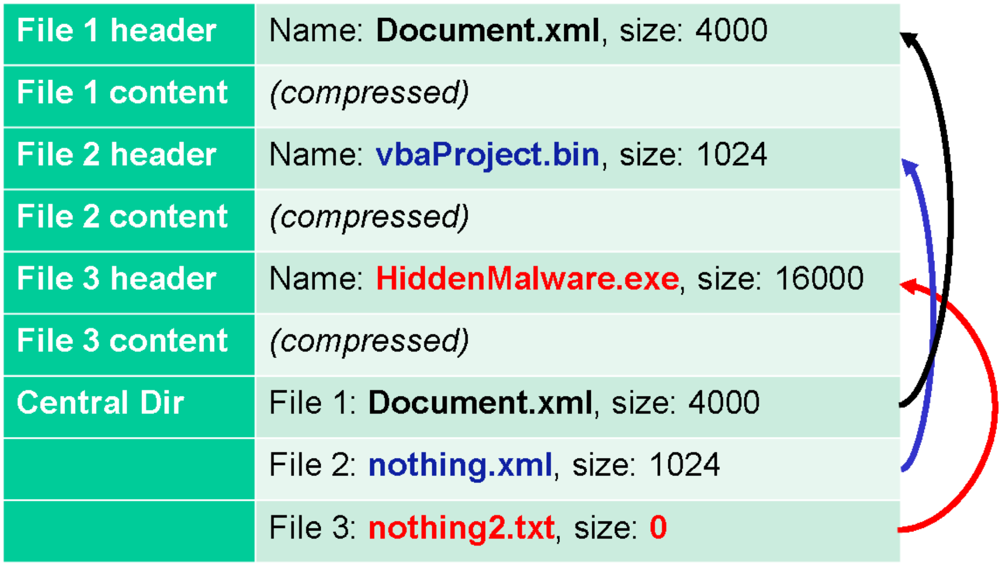
Open XML - UTF-7 encoding: Following the same principle, it is possible to hide XML tags using UTF-7 encoding, and alternative character representations. This is now forbidden for UTF encodings, however Office 2007 allows it (just like Internet Explorer). Here is an example:

```
<?xml version="1.0" encoding="UTF-7" standalone="yes"?>
+ADw-HIDDENTAG+AD4- malware[...] +ADw-/HIDDENTAG+AD4-
```

However, Open XML specifications [OXSPEC] clearly state that only UTF-8 and UTF-16 should be allowed in XML files.

It can be noted that the OpenOffice XML parser does not allow these obfuscations.

Malformed ZIP archives - Duplicate filenames: In a standard ZIP archive, filenames are duplicated in 2 places, in the central directory at the end of the archive and in the header before each file content. This is also the case for the file size and other information. It is possible to create malformed ZIP archives by modifying only one of the filenames. Many applications do not check the coherence of these names and some only rely on one location or the other. Here is an example of such a malformed ZIP archive:



File 1 header	Name: **Document.xml**, size: 4000
File 1 content	*(compressed)*
File 2 header	Name: **vbaProject.bin**, size: 1024
File 2 content	*(compressed)*
File 3 header	Name: **HiddenMalware.exe**, size: 16000
File 3 content	*(compressed)*
Central Dir	File 1: **Document.xml**, size: 4000
	File 2: **nothing.xml**, size: 1024
	File 3: **nothing2.txt**, size: **0**

OpenOffice relies on the central directory only. If a filter or an antivirus only reads file headers, it is possible to fool it with this technique.

MS Office 2007 checks coherence between the 2 filenames. However, if it detects any problem, it asks the user if the file should be repaired. Strangely, the file is always fixed so that macros are kept executable whether the header or the central directory was changed ! This technique may then be used to bypass any filter/antivirus which only relies on one of the filenames.

Zip64 compression : ZIP format has been improved in the last few years in order to increase the maximal size of the archives and the compression rate. Open XML specifications explicitly allow the use of the new Zip64 format; therefore, some filters/antiviruses may be bypassed if they do not handle this format properly.

Recommendations for a robust filter or antivirus : OpenDocument and Open XML analysis must not be seen as an easy task even though these are open formats based on ZIP and XML. Here are some important facts to check for a safe analysis:

- Use a robust ZIP library, able to detect malformed archives.
- In ZIP archives, reject any incoherence between central directory and file headers.
- Use case-insensitive functions to handle filenames and paths within ZIP archives.
- Reject any ZIP archive whose format is not supported by the library, or if it is not allowed by the specifications: Zip64, new compression algorithm, encryption, ...
- Always use a full and robust XML parser. Never use a simple text search or regular expressions for XML files.

    - Open XML parsing is especially more complex because it requires a deeper analysis of XML data, following the OPC principles (Open Packaging Conventions, see [OXSPEC] part 2).
- Check XML files encoding: reject any encoding that is not allowed by specifications, and use strict decoding mode to reject any abnormal character.
- If possible take advantage of XML schemas provided by vendors with open specifications.
- Reject any incoherence between internal structure and filename of the document. (For example an Open XML document should never be named ".doc")

7 Conclusion

Both new office formats OpenDocument and Open XML are very promising, and their open specifications are very useful from a security point of view. Active content and hidden data filtering are much easier than before.

However they still suffer from the same security issues as former proprietary formats, and there is no real reason to feel more secure. Some new features of these formats or office suites may even add security issues, such as XML and ZIP obfuscation techniques.

It can be guessed that it will take some time before all antiviruses and content analysis software are able to handle OpenDocument and Open XML safely. This article brings a few ideas to improve analysis of these formats.

References

[ESAT1] *In-depth Analysis of the Viral Threats with OpenOffice.org Documents*, De Drézigué, Fizaine, Hansma (ESAT), Journal in Computer Virology, 2006. http://www.springerlink.com/content/1772-9904/?k=openoffice

[ESAT2] *Le risque viral sous OpenOffice 2.0.x,* Filiol, Fizaine (ESAT), MISC magazine n˚27, 09/2006.

[FSECURE] *OpenOffice security*, S. Rautiainen (F-Secure), VB2003 conference, http://www.f-secure.com/weblog/archives/openoffice_security.pdf

[PACSEC06] *OpenOffice/OpenDocument and MS Open XML security,* P. Lagadec, PacSec 2006 conference, http://pacsec.jp/psj06archive.html

[SSTIC07] *Sécurité des formats OpenDocument et Open XML*, P. Lagadec, http ://actes.sstic.org/SSTIC07/Securite_OpenDocument_OpenXML/

[ECMA] Ecma International, *National Body Comments from 30-Day Review of the Fast Track Ballot for ISO/IEC DIS 29500 (ECMA-376) "Office Open XML File Formats", Ecma/TC45/2007/006,* http://www.ecma-international.org/news/TC45_current_work/Ecma%20responses.pdf

[SSTIC03] *Formats de fichiers et code malveillant,* P. Lagadec, SSTIC03, http ://actes.sstic.org/SSTIC03/Formats_de_fichiers/

[CVEMSO] *Common Vulnerabilities and Exposures, keywords "Microsoft Office",* http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=microsoft+office

[CVEOOo] *Common Vulnerabilities and Exposures, keyword "OpenOffice",* http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=openoffice

[ESAT3] *Analyse du risque viral sous OpenOffice.org 2.0.x,* E. Filiol (ESAT), rump sessions SSTIC06, http ://actes.sstic.org/SSTIC06/Rump_sessions/SSTIC06-rump-Filiol-Risque_viral_sous_OpenOffice.pdf

[OOoURL] *OpenOffice.org URL Handling Security Vulnerability (Linux/Solaris),* http://www.openoffice.org/security/CVE-2007-0239.html

[XSRF] *Cross-site request forgery*, Wikipedia, http://en.wikipedia.org/wiki/XSRF

[OSSIR03] *La fuite d'informations dans les documents propriétaires,* P. Chambet (EdelWeb), Eric Filiol (ESAT), E. Detoisien, OSSIR 6/10/2003, http ://www.ossir.org/windows/supports/2003/2003-10-06/OSSIR-Fuite%20infos.pdf

[ODSPEC10] *Open Document Format for Office Applications (OpenDocument) v1.0,* OASIS Standard, 1 May 2005, http://docs.oasis-open.org/office/v1.0/OpenDocument-v1.0-os.pdf

[ODSPEC11] *Open Document Format for Office Applications (OpenDocument) v1.1,* OASIS Standard, 1 Feb 2007, http://docs.oasis-open.org/office/v1.1/OpenDocument-v1.1.pdf

[OXSPEC] *Office Open XML File Formats - Standard ECMA-376*, http://www.ecma-international.org/publications/standards/Ecma-376.htm

[OSFP] *OOo scripting framework and Python*, http://udk.openoffice.org/python/scriptingframework/index.html

[OOoPy] *OOoPy*, Python module to edit OpenDocument, http://ooopy.sourceforge.net

[MS06-065] *Secunia advisory for MS06-065*, http://secunia.com/advisories/20717

[XPS] *Microsoft XML Paper Specification - XPS*, http://www.microsoft.com/whdc/xps/default.mspx

[IEASCII] http://www.securityfocus.com/archive/1/437948