Sécurité des formats OpenDocument et Open XML (OpenOffice et MS Office 2007)

Philippe Lagadec

NATO/NC3A
philippe.lagadec@laposte.net

Résumé OpenDocument et Open XML sont deux nouveaux formats de fichiers pour les documents bureautiques. OpenDocument est le format normalisé par l'ISO en mai 2006, promu par OpenOffice.org et Sun StarOffice, alors qu'Open XML est le nouveau format pour les documents de la suite Microsoft Office 2007, accepté comme standard par l'ECMA fin 2006. Ils s'appuient tous deux sur des principes de base similaires : des fichiers XML dans une archive ZIP, avec un schéma ouvert, ce qui contraste avec les anciens formats bureautiques propriétaires (MS Word, Excel, Powerpoint, ...). Cependant ces formats peuvent poser de nombreux problèmes de sécurité, proches de ceux déjà connus pour les documents MS Office actuels : il est possible d'y camoufler des contenus malveillants (chevaux de Troie, virus,...) grâce à certaines fonctionnalités « actives » comme les macros, les scripts ou les objets OLE. A cela s'ajoutent les possibilités de camouflage supplémentaires dues à XML et ZIP. Cet article montre ces problèmes de sécurité avec des détails techniques, et décrit comment concevoir un filtre pour assainir les documents grâce à leur format ouvert.

1 Introduction

Les formats de fichiers bureautiques ont toujours posé des problèmes de sécurité, en grande partie à cause de la richesse de leurs fonctionnalités « actives » comme les macros ou l'inclusion d'objets OLE [5], ou bien la fuite d'informations cachées dans les documents [1]. Plus récemment, les chercheurs en sécurité se sont penchés plus précisément sur ces formats et les applications correspondantes, et de nombreuses vulnérabilités y ont été découvertes [7,14].

Le scénario le plus classique est un utilisateur recevant un document bureautique malicieux en pièce jointe d'un courriel ou par un autre moyen. L'ouverture de ce fichier déclenche alors du code malveillant dans une macro, un objet ou un shellcode, avec ou sans action de l'utilisateur suivant les cas, et le résultat peut aller jusqu'à la compromission de la machine.

Aujourd'hui pour diverses raisons beaucoup d'utilisateurs se sentent plus en sécurité avec OpenOffice ou MS Office 2007 qu'avec les autres suites bureautiques habituelles, mais cela pourrait se révéler être une fausse impression, comme l'annonçait Eric Filiol lors de SSTIC06 [2,4,3].

Par exemple en juin 2006 un virus « preuve de concept » nommé « Stardust » a montré que la menace était réelle, et 3 vulnérabilités sérieuses ont été découvertes dans OpenOffice 2.0.2, permettant à du code malveillant de contourner la sécurité par défaut (contournement de la fenêtre d'avertissement pour les macros et bac à sable Java).

De son côté Open XML est un nouveau format basé sur des standards ouverts, les spécifications viennent tout juste d'être publiées par l'ECMA et Microsoft, et aucune analyse de sécurité complète n'a été publiée pour l'instant.

Cet article examine certains problèmes de sécurité posés par ces nouveaux formats OpenDocument et Open XML ainsi que les suites bureautiques associées, puis apporte quelques pistes pour se

protéger. Il montre également que l'utilisation de ZIP et XML apporte des possibilités de camouflage supplémentaires pour les codes malveillants, et indique les pièges à éviter pour un antivirus ou un filtre de fichiers.

1.1 Notes

- Cette analyse de sécurité est basée sur les versions suivantes :
  - spécifications OpenDocument v1.0 [11]. La nouvelle version 1.1 [12] n'a pas encore été analysée.
  - OpenOffice v2.1
  - spécifications Open XML ECMA-376, version finale publiée en décembre 2006 : [13].
  - Microsoft Office 2007 version 12.0.4518.1014.
  - Tous les tests ont été menés sur Windows XP SP2.
- Ceci n'est pas une analyse de sécurité complète, notamment certains aspects importants comme le chiffrement et la signature de documents n'ont pas été étudiés.
- Cette analyse s'est principalement penchée sur les problèmes de sécurité liés aux fonctionnalités natives des formats et des applications. Il ne s'agissait pas de rechercher des vulnérabilités de type débordement de tampon ou d'entier.
- Les travaux de recherche ayant permis la rédaction de cet article ont été financés successivement par DGA/CELAR et par le programme de travail de recherche et développement de NATO/ACT.
- Une première version de ces travaux a été publiée en anglais pour la conférence PacSec fin novembre 2006 (cf. [6]).
- Certains points de cet article sont détaillés plus concrètement dans la présentation associée, disponible après le symposium SSTIC à l'adresse http ://actes.sstic.org.
- Dans ce document, la suite OpenOffice.org est désignée par les abréviations courantes « OpenOffice » ou « OOo », et la suite Microsoft Office par « MS Office » ou « Office ».

2 Deux nouveaux formats « ouverts »

OpenDocument version 1 est le format de fichiers employé pour tous les documents de la suite OpenOffice version 2. Ce format est également employé par d'autres applications bureautiques comme Sun StarOffice, Koffice, Abiword, ... Il est devenu standard ISO depuis mai 2006.

Open XML est le nouveau format par défaut pour la plupart des applications de la suite Microsoft Office 2007 : Word, Excel, Powerpoint. C'est un standard ECMA depuis fin 2006.

OpenDocument et Open XML ont des caractéristiques similaires :

- Ils sont tous deux basés sur des technologies ouvertes et très couramment répandues : chaque document est principalement constitué de fichiers XML compressés dans une archive ZIP.
- Leurs spécifications sont ouvertes et publiées sur Internet : [11,12,13].
- Ils ont tous deux été acceptés comme standards par des instances internationales : ISO pour OpenDocument, ECMA pour Open XML.
- Ils prennent tous deux en charge les formats classiques de documents bureautiques : document texte, feuille de calcul, présentation, dessin vectoriel.

En étudiant de plus près les spécifications publiées, on peut cependant trouver quelques différences :

- La structure d'Open XML est plus complexe et plus riche que celle d'OpenDocument. Microsoft s'est a priori attaché à reprendre intégralement toutes les fonctionnalités existantes de sa suite Office, alors qu'OpenDocument apporte un modèle de document plus simple et direct.
- D'ailleurs les spécifications d'OpenDocument ne font « que » 700 pages, alors que celles d'Open XML dépassent les 6000 pages !
- ...et pour autant les fonctionnalités qui nous préoccupent particulièrement en matière de sécurité comme les macros VBA ou les objets OLE ne sont pas encore abordées dans les spécifications ouvertes d'Open XML. Les spécifications d'OpenDocument ne sont pas exhaustives non plus.

Par rapport aux formats bureautiques propriétaires, l'analyse de la sécurité de ces formats est donc grandement facilitée par leur caractère ouvert. Cependant la complexité de leurs spécifications et l'absence de documentation de certaines fonctionnalités ne rendent pas la chose si aisée.

3 OpenDocument et OpenOffice.org

3.1 Le format OpenDocument

Les spécifications OpenDocument v1.0 ne couvrent aujourd'hui qu'une partie des formats de documents pris en charge par la suite OpenOffice v2 : Seuls les documents texte, feuilles de calcul, présentations et dessins vectoriels sont décrits. Les autres types de documents comme les bases de données, modèles HTML et formules mathématiques possèdent cependant une structure très similaire.

Le tableau suivant rappelle les différentes extensions associées aux principaux formats natifs d'OpenOffice v1 ou v2 :

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	

3.2 Structure interne

Ce format est principalement basé sur des fichiers XML dans une archive ZIP, situés à la racine ou dans des sous-répertoires.

Voici les principaux fichiers XML :

- **content.xml** : corps du document
- **styles.xml** : styles
- **meta.xml** : méta-données (titre, auteur, ...)
- **settings.xml** : paramètres globaux pour le document
- **META-INF/manifest.xml** : description de chaque fichier de l'archive

L'archive peut également contenir d'autres types de fichiers :
- **images et miniatures** : JPEG, PNG, SVG, ... (répertoires Pictures et Thumbnails)
- **objets inclus** : dessins vectoriels, diagrammes, graphiques, objets OLE, ...

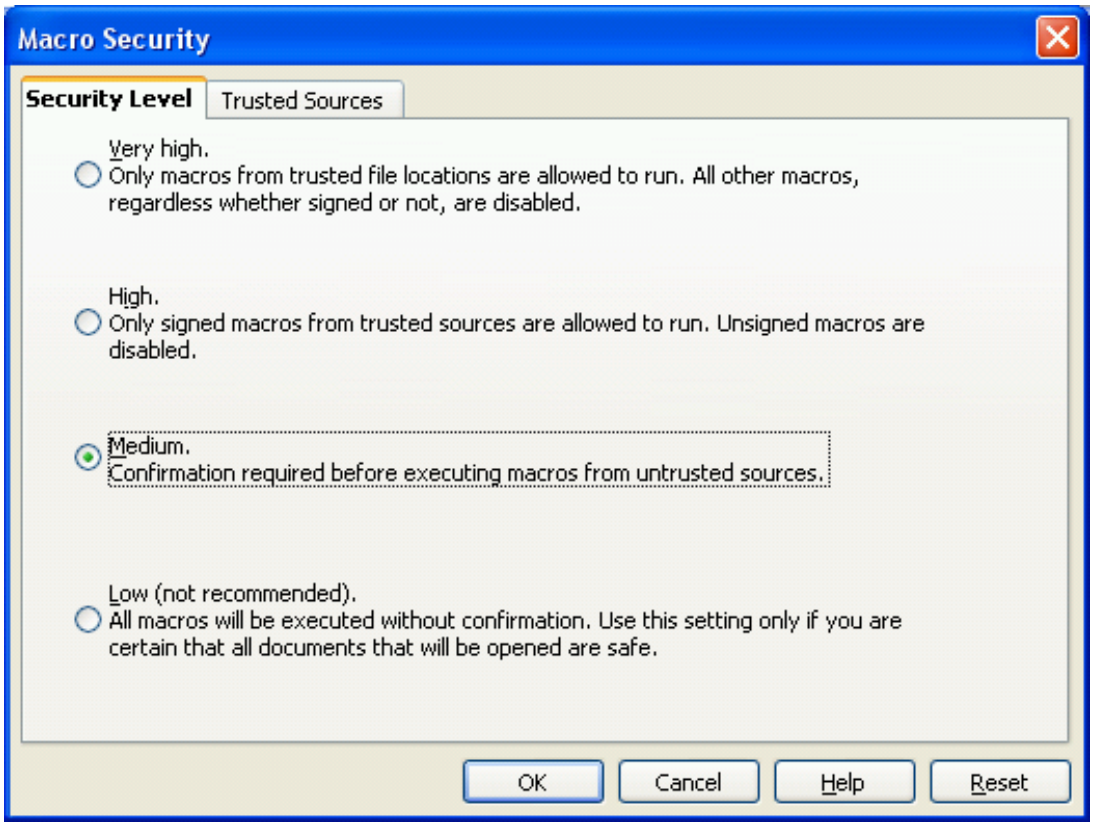
3.3 Macros

Le problème de sécurité principal provient des macros. OpenOffice v2 permet d'écrire des macros dans **4 langages différents : Basic, Javascript, Java (Beanshell) et Python**. D'autres langages pourront être ajoutés à l'avenir. Une installation standard d'OpenOffice contient tous les interpréteurs nécessaires ou s'appuie sur des moteurs externes comme la JVM.

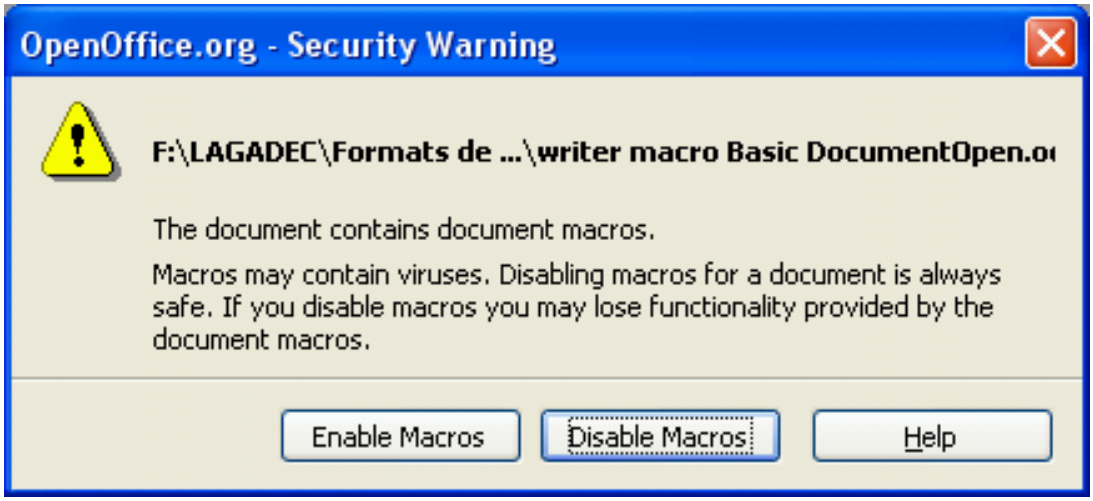
Chaque langage de macro accède à une même API appelée UNO (Universal Network Objects), qui permet d'agir sur les documents, OpenOffice ou le système d'exploitation. La richesse de cette API permet de concevoir un code malveillant tout à fait efficace.

De plus les macros peuvent être associées à des événements, il est donc **possible de les déclencher à l'ouverture ou à la fermeture d'un document**.

Pour protéger l'utilisateur contre des macros malicieuses, OpenOffice propose **4 niveaux de sécurité**, semblables à ceux de MS Office 2000/XP/2003.



Le niveau par défaut est « moyen » : si une macro quelconque est présente dans le document, une fenêtre de confirmation demande à l'utilisateur s'il souhaite activer ou non les macros, avant qu'il puisse voir le contenu du document.



On peut noter qu'il s'agit du même niveau par défaut que MS Office 97. Depuis la version 2000, MS Office est passé à un niveau « haut » par défaut, qui n'autorise que les macros signées ou provenant d'un emplacement de confiance.

Mi-2006, une vulnérabilité a été découverte dans OpenOffice 2.0.2, qui permettait de contourner cette sécurité.

Stockage des macros : Les macros sont stockées dans 2 sous-répertoires de l'archive suivant leur langage :

- Dans le répertoire « Basic » sous forme de fichiers XML pour les macros OOo Basic.
- Dans le répertoire « Scripts » sous forme de scripts pour les macros en Java/Beanshell, Javascript ou Python.

Voici quelques exemples :

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

Avec OpenOffice v2 il est possible de créer et modifier les macros depuis l'application pour les langages Basic, Java/Beanshell et Javascript. Pour l'instant l'inclusion de macros Python demande par contre des opérations manuelles (cf. [9]) : ajout de fichiers dans l'archive, et modification du fichier manifest.xml.

Signature des macros : Un des principaux problèmes découverts par les chercheurs de l'ESAT est que les macros ne sont pas signées avec le reste du document. Il est donc possible de modifier les macros d'un document signé et de leurrer l'utilisateur qui pense ouvrir un document authentique [2,4,3].

Macros VBA : Pour l'instant OpenOffice n'est pas capable d'interpréter les macros en langage VBA. Cependant lorsqu'un document MS Office est converti au format OpenDocument, le code

source des macros VBA est extrait « en clair » et stocké dans les commentaires d'une macro OOo Basic vide. Ceci active donc l'avertissement de sécurité normal des macros.

Si ce document est ensuite exporté de retour au format MS Office, les macros VBA sont réactivées sous leur forme d'origine.

Des travaux sont en cours pour permettre l'exécution native de macros VBA dans une future version d'OpenOffice, et certaines versions alternatives commencent déjà à proposer un support limité des macros Excel.

3.4 Objets OLE

Un document peut inclure des objets provenant d'autres modules de la suite OpenOffice, par exemple un graphique dans un document texte. Il peut également inclure des objets provenant d'autres applications grâce à la technologie OLE sous Windows. Les objets OLE les plus problématiques sont les objets Package, qui peuvent contenir n'importe quel fichier, y compris un exécutable binaire, ou bien une simple ligne de commande (cf. [5] pour des exemples concrets).

Les objets OLE sont généralement stockés dans des fichiers « Object xxx » à la racine de l'archive. Ce sont des fichiers au format binaire propriétaire OLE2 (encore appelé Structured Storage) de Microsoft. Dans ce cas le fichier OpenDocument n'a plus un format 100% ouvert.

Les objets OLE peuvent être ouverts par un double-clic de l'utilisateur, directement sans confirmation et sans avertissement de la part d'OpenOffice. Sur un système Windows récent, c'est le gestionnaire de liaisons de Windows (packager.exe) qui demande confirmation à l'ouverture d'un objet OLE Package. Sur un système plus ancien (par exemple Windows 2000 SP4 non mis à jour), l'ouverture est directe sans confirmation (comportement similaire à WordPad).

OpenOffice est du coup tributaire du système d'exploitation : par exemple en 2006, une vulnérabilité de Windows XP dans packager.exe permettait de camoufler la ligne de commande d'un objet OLE en faisant passer celui-ci pour un fichier inoffensif [17].

3.5 Scripts

OOo permet l'inclusion de code Javascript ou VBscript dans les documents. Cependant ces scripts ne sont pas exécutés dans l'application. Ils sont prévus pour être inclus dans le code HTML dans le cas d'un export ou d'un import. Ces scripts peuvent présenter une menace, car il est possible d'inciter l'utilisateur à exporter un document en HTML pour activer un script malveillant dans son navigateur.

Les scripts sont stockés dans content.xml, dans une balise `<text:script>`, par exemple :

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

Le code source du script peut être soit inclus directement dans la balise (comme ci-dessus), soit dans un fichier script à part, référencé par un attribut « `xlink:href` ».

3.6 Applets Java

Un document peut contenir des applets, qui sont exécutées à son ouverture dans OOo. Ces applets sont exécutées dans une sandbox, similaire à celle d'un navigateur (pas d'accès aux fichiers locaux). Cela devrait donc être sûr, mais par exemple OpenOffice 2.0.2 souffrait d'une vulnérabilité qui permettait à des applets malicieuses de contourner les protections de la sandbox.

3.7 Liens URLs

Un document peut contenir des liens de type URL. Lorsqu'il s'agit d'une URL externe, son activation provoque l'ouverture du navigateur par défaut. Une URL locale (pointant vers un fichier d'un disque local) provoque l'ouverture du gestionnaire de fichiers, après confirmation. Les URLs contenant des scripts, commençant par « javascript :... » ou « vbscript :... » par exemple, sont filtrées et interdites par OpenOffice.

Une vulnérabilité récemment découverte dans OpenOffice v2.1 permettait l'exécution de commandes locales sous Linux ou Solaris en insérant des caractères d'échappement du shell dans une URL [15].

3.8 Fuite d'information

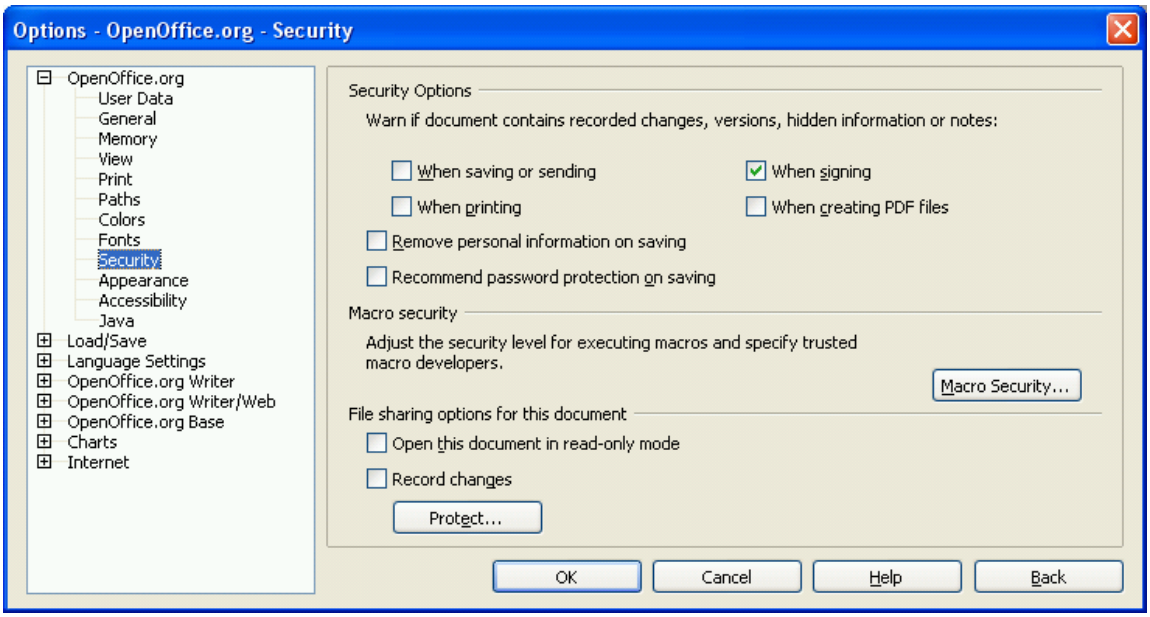
Le format OpenDocument peut contenir diverses informations cachées, qui créent un risque de fuite d'informations sensibles, comme cela existe dans les formats bureautiques propriétaires [1] :

- méta-données : titre, auteur, ...
- marques de révision, suivi des modifications
- commentaires
- fichiers inclus (objets OLE ou OpenDocument)
- paragraphes et champs cachés
- formules

Ces informations cachées sont problématiques lorsque des documents sont publiés vers l'extérieur, ou bien dans le cadre de la signature électronique.

OpenOffice apporte cependant des fonctionnalités intéressantes, car il est possible d'être mis en garde si un document contient des données cachées avant de le signer, de l'exporter en PDF, ou de le sauver.

On peut toutefois noter que cette protection ne prend pas en compte la présence d'objets OLE.



3.9 Conclusion sur la sécurité du format OpenDocument et d'OpenOffice

Il existe donc de multiples fonctionnalités permettant d'inclure un contenu actif potentiellement malveillant dans un fichier OpenDocument. Même s'il existe aujourd'hui des garde-fous pour éviter qu'un utilisateur puisse déclencher directement un code malveillant, les protections existantes ne sont pas absolues. De plus, des vulnérabilités sont régulièrement découvertes dans OpenOffice, ce qui impose de rester vigilant et d'appliquer toutes les mises à jour.

On peut également constater que même si le format OpenDocument est ouvert, il peut dans certains cas contenir des parties en format propriétaire (MS OLE2 par exemple).

Au final, OpenOffice n'est pas fondamentalement plus sûr que Microsoft Office, en ce qui concerne le code malveillant ou la fuite d'information. Cependant le format ouvert OpenDocument est bien plus facile à analyser et à filtrer qu'un format propriétaire fermé.

4 Open XML et Microsoft Office 2007

Office 2007 est la nouvelle version de la suite bureautique Microsoft Office, dont les premières versions ont été diffusées en décembre 2006. Open XML est le nouveau format par défaut pour la plupart des documents de cette suite.

4.1 Le format Open XML

Les documents Open XML d'Office 2007 emploient de nouvelles extensions :
- Word : .docx, .docm, .dotx, .dotm
- Excel : .xlsx, .xlsm, .xltx, .xltm, .xlsb, .xlam
- Powerpoint : .pptx, .pptm, .ppsx, .ppsm
- Access : .accdb (nouveau format binaire, différent d'Open XML)

Office 2007 est toujours capable de lire et écrire les documents MS Office des versions précédentes (format binaire OLE2) grâce à un « mode de compatibilité ». Il existe également un « Converter pack » pour lire et écrire des documents Open XML depuis les versions précédentes d'Office (2000, XP, 2003).

4.2 Structure interne

Comme OpenDocument, le format Open XML est principalement basé sur une archive ZIP contenant des fichiers XML. Cependant la structure interne est un peu plus complexe. Open XML s'appuie en fait sur un autre nouveau format générique de conteneur proposé par Microsoft : OPC, pour « Open Packaging Conventions ». OPC sert de base à d'autres formats comme XPS [8], similaire à PDF.

Voici les principaux fichiers XML à l'intérieur de l'archive ZIP, par exemple pour un document Word 2007 :
- [Content_Types].xml : décrit chaque fichier dans l'archive
- fichiers .rels : précisent les relations entre les objets des différents fichiers XML
- word/document.xml : corps du document
- word/styles.xml : styles
- word/settings.xml : paramètres pour le document
- docProps/app.xml and core.xml : méta-données (titre, auteur, ...)

On peut également y trouver d'autres types de fichiers :
- images : JPEG, PNG, GIF, TIFF, WMF, ...
- fichiers binaires OLE2 : macros VBA, objets OLE, paramètres d'impression, ...

Contrairement à OpenDocument, il n'y a pas de relations directes entre les différents fichiers XML. Les fichiers « .rels » établissent des relations indirectes entre les fichiers XML, ce qui rend l'analyse du document plus complexe.

4.3 Macros VBA

Il est possible d'inclure des macros VBA dans les documents XML. Ces macros ont les mêmes fonctionnalités que pour les versions précédentes d'Office, et posent donc les mêmes problèmes de sécurité.

Une grande nouveauté est que les documents Open XML avec macros sont clairement distingués des documents sans macros : Leur extension est différente, par exemple un document Word sans macro se terminera par « .docx », et « .docm » s'il contient des macros. Si on renomme un document avec macros en « .docx », Office refusera de l'ouvrir.

Niveaux de sécurité : Les niveaux de sécurité concernant les macros ont été également remaniés. Dans les versions précédentes d'Office, le niveau par défaut était « haut », ce qui n'autorisait que l'exécution des macros signées ou provenant d'un emplacement de confiance. L'utilisateur devait modifier la configuration pour pouvoir exécuter des macros non signées.

Office 2007 fournit une nouvelle interface graphique (le « ruban »), et le nouveau niveau de sécurité par défaut est « Désactiver toutes les macros avec notification ». Lorsqu'un utilisateur ouvre un document avec une macro, un message s'affiche sous le ruban. En cliquant sur ce message, il accède à une fenêtre qui lui permet d'autoriser l'exécution des macros, qu'elles soient signées ou non. Le nouveau mode par défaut permet donc à un utilisateur d'exécuter une macro non signée en 3 clics de souris, s'il ne lit pas consciencieusement tous les messages d'avertissement.

De plus, l'utilisateur peut lire le document avant de décider d'activer les macros, ce qui offre quelques possibilités de « social engineering ».

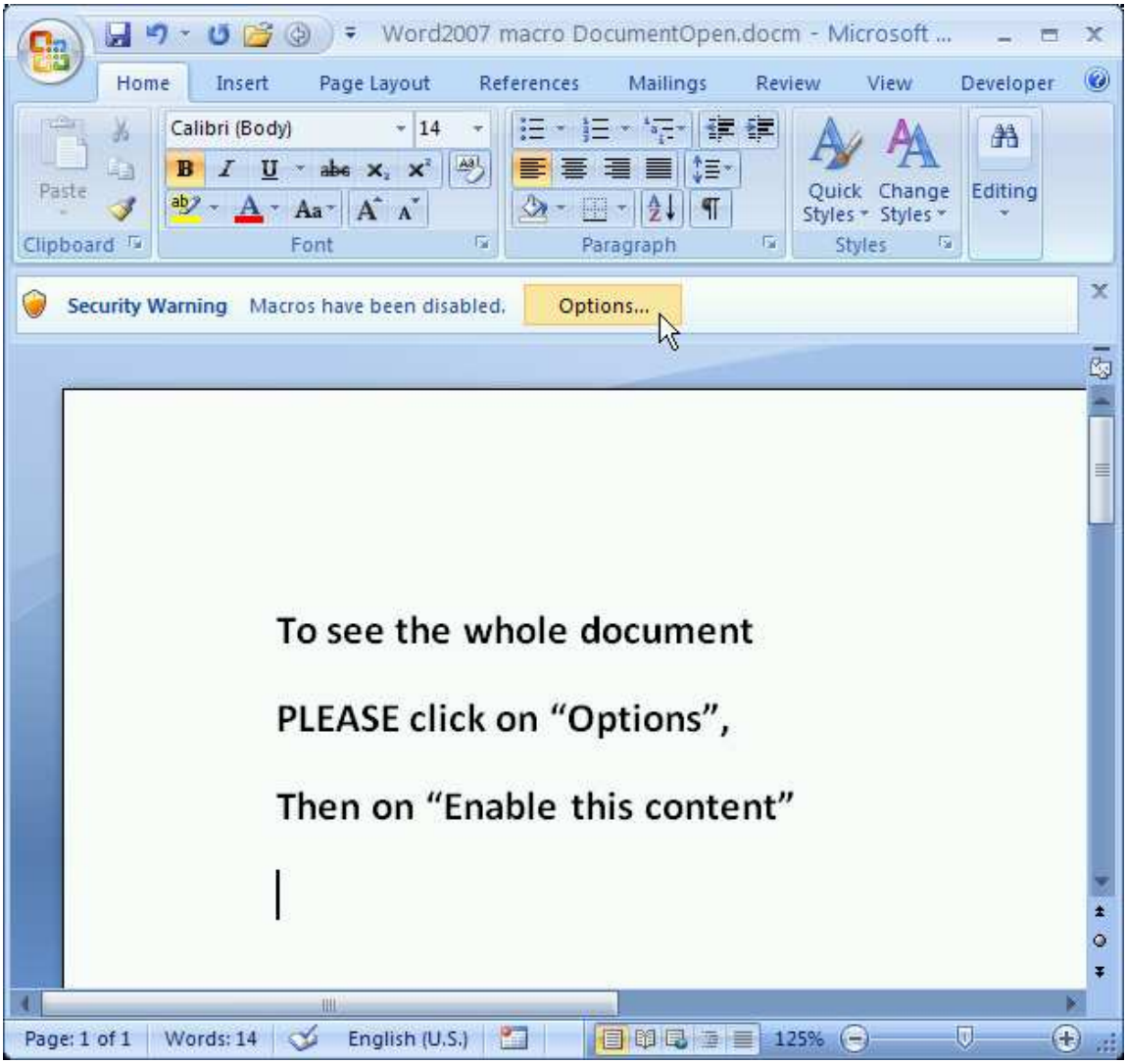
Tous les paramètres de sécurité sont maintenant regroupés dans une seule fenêtre appelée « Trust center » pour faciliter la configuration. On y trouve 4 niveaux de sécurité pour les macros :
- désactiver toutes les macros sans notification
- désactiver toutes les macros avec notification (par défaut)
- désactiver toutes les macros sauf les macros signées
- autoriser toutes les macros

Le Trust center propose aussi des niveaux de sécurité pour les ActiveX.

Stockage des macros : Les macros sont stockées dans un fichier vbaProject.bin, dont l'emplacement à l'intérieur de l'archive varie selon les applications :
- Word : word/vbaProject.bin
- Excel : xl/vbaProject.bin
- Powerpoint : ppt/vbaProject.bin

Ce fichier est au format binaire OLE2, et il n'est pas décrit dans les spécifications Open XML actuelles [13].

Si la macro a un nom spécifique, comme par exemple « Document_Open » pour Word, elle peut se déclencher automatiquement à l'ouverture du document.

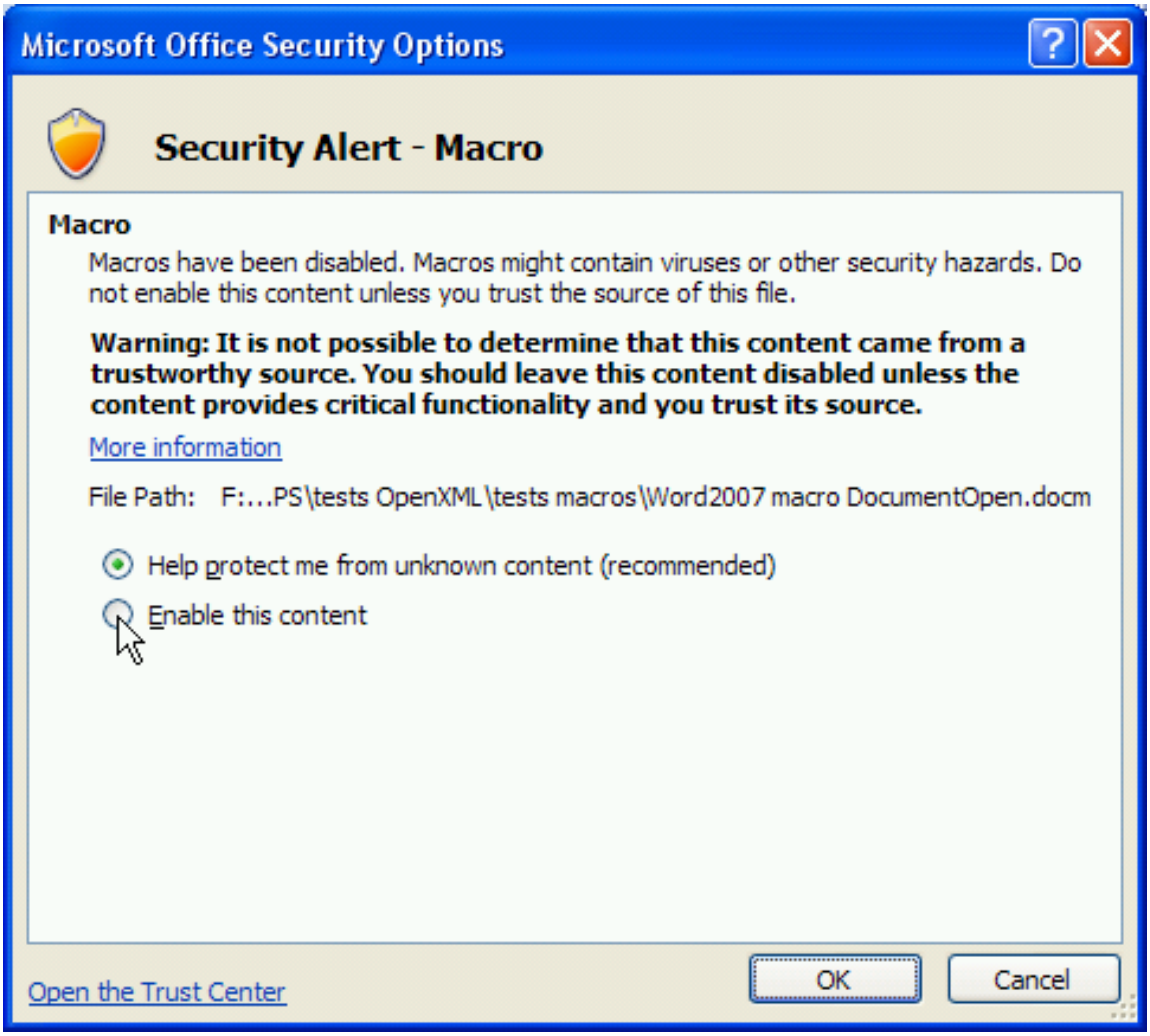


4.4 Objets OLE

Comme OpenDocument et les versions précédentes d'Office, il est possible d'inclure des objets OLE dans des documents Open XML, et on retrouve les mêmes problèmes de sécurité.

Ces objets sont généralement stockés dans leur format d'origine, à des emplacements différents suivant l'application : par exemple « word/embeddings » pour Word. Un objet OLE Package est stocké au format OLE2.

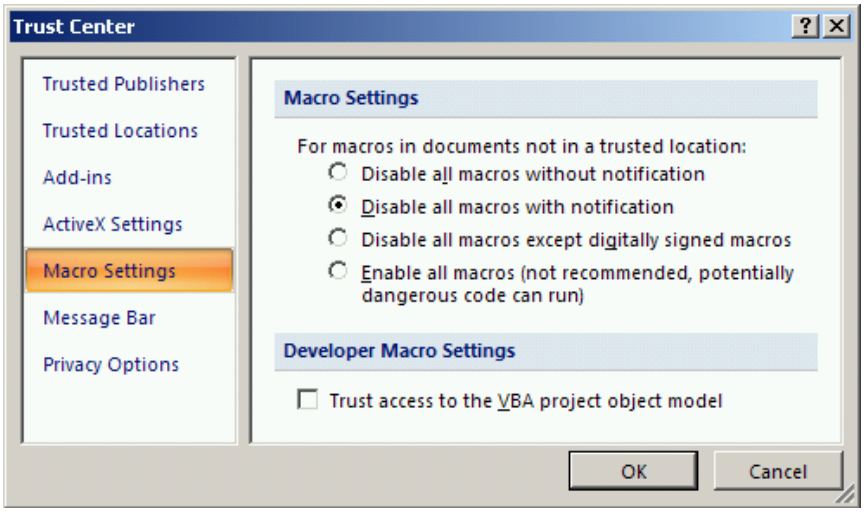
Comme pour les macros, le stockage des objets OLE n'est pas décrit dans la version actuelle des spécifications Open XML.



Certains cas de figure peuvent produire des résultats étonnants : par exemple, il est possible d'inclure un classeur Excel avec macros (.xlsm) en tant qu'objet dans un document Word sans macro (.docx). Il n'y a pas d'avertissement à l'ouverture du document Word, par contre il y en a un à l'ouverture de l'objet Excel, qui propose directement d'activer ou non les macros. Et ceci même si le niveau de sécurité est réglé sur « désactiver toutes les macros sans notification »...

4.5 Classeurs binaires Excel 2007

Excel 2007 permet de sauvegarder un classeur dans un format Open XML hybride, appelé « classeur binaire », avec une extension « .xlsb ». Ce format reprend certaines bases d'Open XML, mais remplace une partie des données XML par des fichiers binaires pour améliorer les performances,



dans un format ressemblant à BIFF8, non documenté. Les classeurs binaires peuvent contenir des macros.

4.6 Fuite d'informations

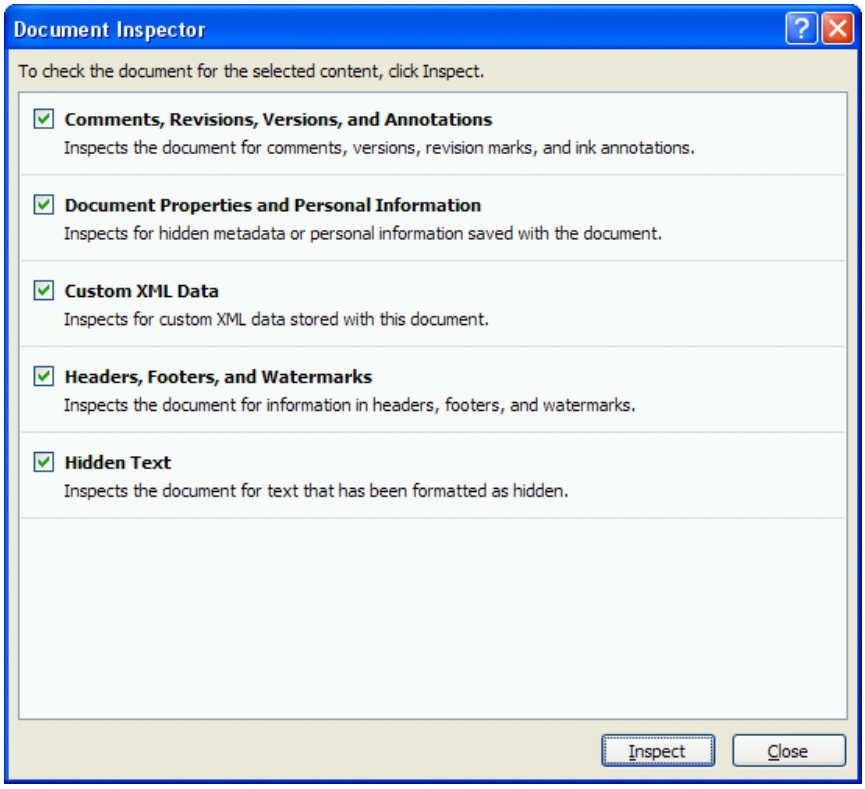
Office 2007 fournit un nouvel outil « Document Inspector » pour détecter et supprimer les différents types d'informations cachées qui peuvent se trouver dans un document. Il s'agit d'une version améliorée de l'outil « RHDTool » qui pouvait être installé en option avec Office 2003. C'est bien sûr une fonction très utile, cependant comme pour OpenOffice, les objets OLE ne sont pas considérés comme des données cachées.

4.7 Conclusion sur la sécurité d'Open XML et MS Office 2007

Au vu de cette étude, le nouveau format Open XML pose autant de problèmes de sécurité que les anciens formats de MS Office, pour ce qui concerne le code malveillant ou la fuite d'informations. Et malgré les promesses, la sécurité par défaut de MS Office 2007 n'est pas fondamentalement plus « sûre » qu'avant. Pour certains points comme le niveau de sécurité par défaut pour les macros on pourrait même considérer qu'il s'agit d'une légère régression.

Même si le format Open XML est basé sur des spécifications ouvertes, il peut contenir des parties en format propriétaire non documenté (MS OLE2 ou BIFF par exemple). De plus certaines fonctionnalités importantes comme les macros ne sont pas décrites dans le standard, et le format ne peut donc pas être considéré comme 100% ouvert.

Comme OpenDocument, Open XML est plus facile à analyser et à filtrer que les formats non ouverts. Cependant la structure d'Open XML est plus complexe et plus riche, ce qui complique toute analyse.



5 Comment se protéger

Pour se protéger contre l'entrée de code malveillant et la fuite d'information par les documents bureautiques, il existe principalement deux solutions techniques, qui sont complémentaires :

- Sécuriser le paramétrage des suites bureautiques
- Filtrer les documents sur une passerelle d'interconnexion ou sur les supports amovibles

Nous ne parlerons pas ici des solutions organisationnelles qui consistent par exemple à mieux informer l'utilisateur sur les dangers des codes malveillants et de la fuite d'information cachée.

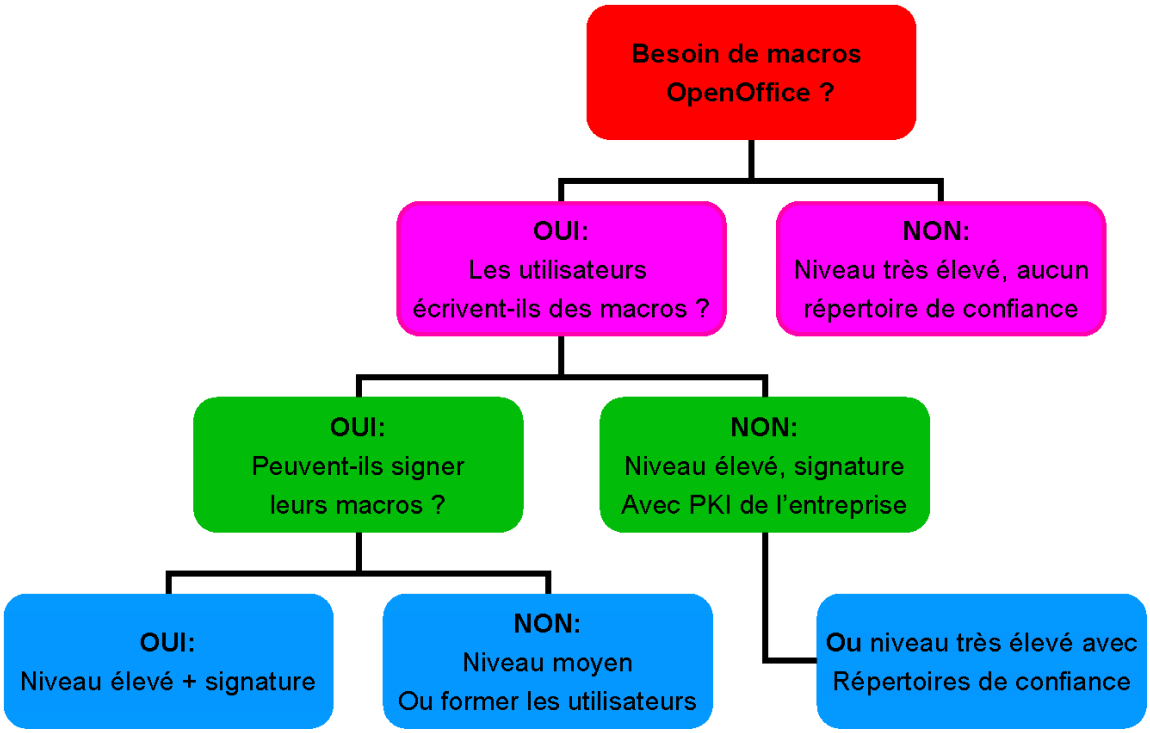
5.1 Paramétrage sécurisé d'OpenOffice et MS Office 2007

Voici quelques principes généraux qui permettent d'améliorer la sécurité des suites bureautiques en fonction des besoins :

- Bien sûr, commencer par appliquer systématiquement les correctifs de sécurité et les mises à jour.
- Durcir le niveau de sécurité des macros, en fonction des besoins *réels* des utilisateurs.
- Utiliser la signature numérique des macros si besoin, si possible en s'appuyant sur une PKI. A défaut, employer les répertoires de confiance (en les configurant finement).
- Protéger les paramètres de sécurité contre les modifications de l'utilisateur.

– Déployer les paramètres de sécurité sur un parc de machines pour simplifier l'administration, par exemple à l'aide de GPO.
– Se protéger contre les objets OLE Package, qui sont très rarement employés :
  – interdire l'exécution de « C :\Windows\System32\Packager.exe ».

Voici un exemple d'arbre de décision qui permet de choisir un niveau de sécurité adapté pour les macros OpenOffice :



5.2 Filtrage des documents

Les documents peuvent être filtrés sur une passerelle (messagerie, web, FTP, ...) ou sur un « sas de dépollution » pour supports amovibles. Il peut s'agir d'une simple analyse antivirus, ou bien d'un filtrage plus évolué, par exemple pour retirer tout contenu actif des documents (macros, scripts, objets OLE, ...) ou les informations cachées.

Comme les 2 formats OpenDocument et Open XML sont basés sur des technologies standards comme ZIP et XML, on peut imaginer qu'il est aisé d'analyser et filtrer ces formats. En effet on peut facilement localiser tous les éléments actifs comme les macros, les objets OLE, les scripts, etc.

Exemple de filtre pour OpenDocument : Pour retirer tout contenu actif :

- Macros : retirer tout fichier dans les répertoires Basic et Scripts.
- Objets OLE : retirer tout fichier commençant par « Object ».
- Dans le fichier content.xml :
  - supprimer les objets OLE : ¡draw :object-ole¿
  - supprimer les scripts : ¡text :script¿
  - supprimer les applets : ¡draw :applet¿
  - mettre à jour tout tag lié aux macros, par exemple : ¡office :event-listeners¿

Exemple de filtre pour Open XML : Pour retirer tout contenu actif :
- Macros : supprimer tout fichier « vbaProject.bin » et « vbaData.xml »
- Objets OLE : supprimer tout fichier « *.bin »

Exemple de filtre pour OpenDocument et Open XML en Python : Voici un exemple très simple de filtre en langage Python, qui se contente de supprimer les fichiers « à risque » dans un document :

```
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)
```

5.3 Contournement des filtres et antivirus

Il est tentant de filtrer ces formats ouverts avec des techniques simples, en employant par exemple un outil ou une bibliothèque permettant de manipuler les archives ZIP, ou encore une recherche de texte dans les fichiers XML.

Un attaquant peut cependant camoufler les contenus malveillants de diverses façons, afin de contourner une passerelle de filtrage ou un antivirus. Pour cela il suffit d'exploiter les fonctionnalités des suites bureautiques, d'XML et ZIP. Voici quelques exemples de camouflages possibles :

Renommage de documents Open XML avec macros : Tout d'abord il n'est pas suffisant de se fier aux extensions de fichiers : notamment même si un « .docx » ne peut pas contenir de macros, il est toujours possible de renommer un « .docm » en « .doc » pour faire exécuter des macros.

Renommage de macros OpenDocument : Il est possible de renommer le fichier d'une macro OpenDocument à l'intérieur de l'archive ZIP avec n'importe quelle extension à la place de .xml, .bsh, .js ou .py. Il suffit pour cela de modifier manifest.xml et content.xml pour pointer vers la macro renommée. L'extension des fichiers n'est donc pas un indicateur fiable pour détecter les macros. Heureusement, les répertoires où sont stockées ces macros sont aujourd'hui des valeurs fixes (« Basic » et « Scripts »), on peut donc s'y fier pour détecter et supprimer les macros OpenDocument.

Renommage de macros VBA dans Open XML : A cause de la structure modulaire d'Open XML (OPC), il est possible de renommer le fichier de stockage des macros « vbaProject.bin » avec un nom quelconque. Par exemple dans un document Word :

- renommer « vbaProject.bin » en « pasdemacrosici.txt »
- mettre à jour les relations dans « word/_rels/document.xml.rels »
- dans « [Content_Types].xml », remplacer « bin » par « txt »

Cette simple manipulation permet de contourner le filtre Python précédent, en conservant les macros actives. Il n'est donc pas possible de se fier aux noms des fichiers pour détecter la présence de macros dans Open XML. Une solution plus sûre est d'employer un véritable parseur XML pour détecter les objets de type « vbaProject » et « vbaData » (ou « oleObject » pour les objets OLE). Une autre solution est d'analyser le contenu des fichiers à la recherche de formats binaires OLE2, avec de possibles faux-positifs.

Open XML - Encodage US-ASCII et « bit de camouflage » : Comme Internet Explorer (cf. [18]), Office 2007 prend en charge l'encodage « US-ASCII » d'une façon bien étrange : Tous les caractères dont le code ASCII est supérieur à 127 voient leur 8ème bit simplement retiré avant que le code XML soit analysé. Ce comportement permet donc un camouflage très simple pour contourner des filtres qui ne feraient pas cette vérification. Voici un exemple où les balises ¡HIDDENTAG¿ sont camouflées sur ce principe :

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

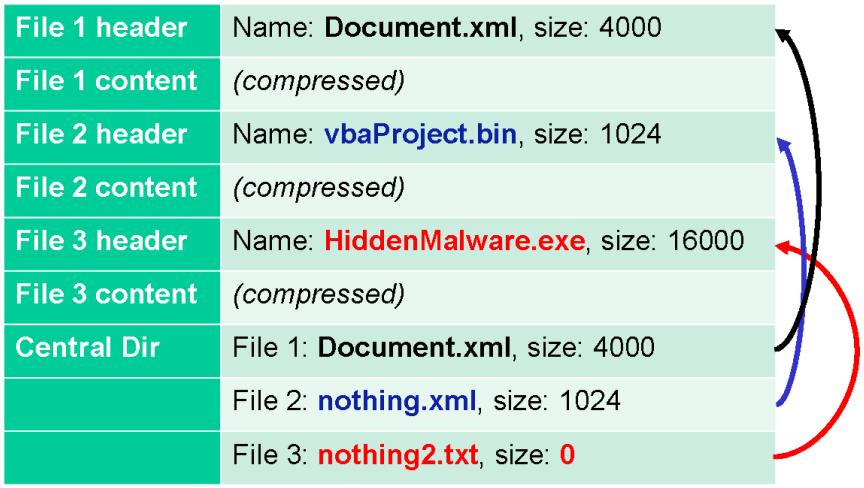
Open XML - Encodage UTF-7 : Sur le même principe, il est possible de camoufler des balises en employant un encodage UTF-7, et en utilisant des formes alternatives de caractères. Ceci est normalement interdit, mais Office 2007 l'autorise (tout comme Internet Explorer). Exemple :

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

Pourtant les spécifications Open XML [13] précisent bien que seuls les encodages UTF-8 et UTF-16 devraient être autorisés dans les fichiers XML...

On peut remarquer que le parseur XML d'OpenOffice est bien plus strict et n'autorise pas ce type de camouflage.

Archives ZIP mal formées - noms de fichiers dupliqués : Dans une archive ZIP standard, les noms des fichiers sont dupliqués à 2 emplacements, dans le répertoire central à la fin de l'archive et dans l'entête de chaque fichier. C'est aussi le cas de la taille du fichier et d'autres informations. Il est possible de créer des archives ZIP mal formées en modifiant le nom d'un fichier dans un de ces emplacements uniquement. Beaucoup d'applications ne vérifient pas la cohérence de ces 2 noms, et



certaines ne se fient qu'à l'un ou l'autre. Voici un exemple de fichier ZIP mal formé : OpenOffice ne considère que le répertoire central. Si un filtre ou un antivirus ne lit que l'entête des fichiers, il est donc possible de le leurrer avec cette technique.

De son côté, MS Office 2007 vérifie bien la cohérence entre les 2 noms. Cependant s'il détecte une incohérence quelconque, il propose à l'utilisateur de réparer le fichier. Ce qui est plus étonnant encore, la correction se fait toujours de façon à rendre les macros exécutables, que l'on ait modifié le répertoire central ou les entêtes ! Cette technique permet donc de contourner tout filtre ou antivirus qui ne s'appuierait que sur les entêtes ou le répertoire central pour son analyse.

Compression Zip64 : Des améliorations du format ZIP ont été proposées ces dernières années, afin d'augmenter la taille maximale des archives ou le taux de compression. Les spécifications Open XML prévoient explicitement la possibilité d'employer le format Zip64. Il est donc nécessaire de prendre en charge ce nouveau format pour pouvoir analyser tous les documents.

Pour créer un filtre ou un antivirus robuste : L'analyse et le filtrage d'OpenDocument et d'Open XML ne doivent donc pas être pris à la légère, sous prétexte qu'il s'agit de formats ouverts basés sur ZIP et XML.

Voici quelques points importants à vérifier pour une analyse sûre :

- Utiliser une bibliothèque ZIP robuste, capable de détecter les archives mal formées.
- Dans les fichiers ZIP, rejeter toute incohérence entre le répertoire central et les entêtes de fichiers.
- Rejeter toute archive ZIP avec un format non supporté par la bibliothèque utilisée, ou non prévu par les spécifications : Zip64, nouvel algorithme de compression, chiffrement, ...
- Utiliser systématiquement un parseur XML complet et robuste. Ne jamais se contenter d'une simple recherche de texte ou d'expressions régulières.
  - En particulier l'analyse de documents Open XML est plus complexe car elle requiert une interprétation plus poussée des données XML, en suivant le principe des OPC (Open Packaging Conventions, voir [13]).

- Vérifier l'encodage des données XML : rejeter tout encodage non prévu dans les spécifications, et décoder les données de façon stricte pour rejeter les caractères anormaux.
- Si possible mettre à profit les schémas fournis par les éditeurs.
- Rejeter toute incohérence entre la structure interne et le nom du fichier (par exemple Open XML nommé « .doc »)

6 Conclusion

Les deux nouveaux formats bureautiques OpenDocument et Open XML sont très séduisants, et leur caractère ouvert est a priori très bénéfique en terme de sécurité. La détection de contenus actifs et d'informations cachées est notamment beaucoup plus facile qu'avant.

Cependant ils posent les mêmes problèmes de sécurité que les formats propriétaires utilisés jusqu'ici, et il n'y a pas lieu de se sentir plus à l'abri des contenus malveillants ou des fuites d'information qu'auparavant.

Certaines nouveautés de ces formats ou des suites bureautiques ajoutent même de nouveaux problèmes qu'il faudra prendre en compte, notamment quelques possibilités de camouflage dues aux formats ZIP et XML.

On peut penser qu'il faudra un certain temps avant que les antivirus et les logiciels d'analyse de contenu prennent en charge OpenDocument et Open XML de façon satisfaisante. Cet article apporte notamment quelques pistes pour améliorer le filtrage de ces nouveaux formats.

Références

1. Chambet, P., Detoisien, E., Filiol, E. : La fuite d'informations dans les documents propriétaires. In OSSIR 6/10/2003, `http://www.ossir.org/windows/supports/2003/2003-10-06/OSSIR-Fuite\%20infos.pdf`
2. De Drézigué, D., Fizaine, J.-P., Hansma, N. : In-depth Analysis of the Viral Threats with OpenOffice.org Documents. *Journal in Computer Virology*, `http://www.springerlink.com/content/1772-9904/?k=openoffice` (2006).
3. Filiol, E. : Analyse du risque viral sous OpenOffice.org 2.0.x. In rump sessions SSTIC06, `http://actes.sstic.org/SSTIC06/Rump_sessions/SSTIC06-rump-Filiol-Risque_viral_sous_OpenOffice.pdf`
4. Filiol, E., Fizaine, J.-P. : Le risque viral sous OpenOffice 2.0.x. *MISC Le Journal de la sécurité informatique*, 27 (2006).
5. Lagadec, P. : Formats de fichiers et code malveillant. In Actes de SSTIC03, `http://actes.sstic.org/SSTIC03/Formats_de_fichiers/` (2003).
6. Lagadec, P. : OpenOffice/OpenDocument and MS Open XML security. In PacSec 2006 conference, `http://pacsec.jp/psj06archive.html` (2006).
7. « Microsoft Office » : Common Vulnerabilities and Exposures, `http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=microsoft+office`
8. Microsoft XML Paper Specification - XPS, `http://www.microsoft.com/whdc/xps/default.mspx`
9. OOo scripting framework and Python, `http://udk.openoffice.org/python/scriptingframework/index.html`
10. OOoPy, module Python pour modifier un document OpenOffice, `http://ooopy.sourceforge.net`
11. 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`

12. 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
13. Office Open XML File Formats - Standard ECMA-376, http://www.ecma-international.org/publications/standards/Ecma-376.htm
14. « OpenOffice » : Common Vulnerabilities and Exposures, http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=openoffice
15. OpenOffice.org URL Handling Security Vulnerability (Linux/Solaris), urlhttp ://www.openoffice.org/security/CVE-2007-0239.html
16. Rautiainen, S. : OpenOffice security. In VB2003 conference, http://www.f-secure.com/weblog/archives/openoffice_security.pdf (2003).
17. Secunia advisory for MS06-065, http://secunia.com/advisories/20717
18. http://www.securityfocus.com/archive/1/437948