J Comput Virol Hack Tech (2013) 9:125–135
DOI 10.1007/s11416-013-0180-9

ORIGINAL PAPER

The Office Demon: Minos

Jonathan Dechaux

Received: 15 October 2012 / Accepted: 20 January 2013 / Published online: 17 February 2013

Contents

Abstract Office documents (*Microsoft Office* and *LibreOffice*) has become a standard for transmitting information. They are used daily by many users. It should however be remembered that this type of documents are much more than inert files. They may contain an executable part who is called *macro*. *Macros* are present since the creation of these Office suites to automate some actions. It is possible to divert the initial use of *macros* to make it a true infection vector of systems. Since 2007 and the case of the attack on the German chancellery, the number of attacks via this type of documents continues to grow. The ability to access high-level programming languages and interact with the target system, greatly increases the risk of attacks. Changing the security of these Office suites is easy, leaving the door open for malicious attacks without the user noticing. In this paper, we present one tool that is a proof of concept. It is intended for the prevention of the user. It aims to give demonstrations of risks associated with Office documents. It is able to change the security of Office suites and infect documents directly with a macro. It also includes a USB mode of infection, to retrieve all documents from a USB stick and then infect all Office documents. It is possible with *Minos*, to control and modify the security of versions 2003, 2007 and 2010 of *Microsoft Office* and versions 3.4 and 3.5 of *LibreOffice*. Similarly it is possible to infect documents *Word, Excel, Powerpoint* of *Microsoft Office* and their counterparts in *LibreOffice*. If a file already contains *macros*, you can either delete the *macro* and replace it by your *macro* or include your *macro* next to the other *macros*. The data presented in this report are technical and operational. We have worked in environments with restricted rights showing that it is possible to make powerful attacks by infecting Office documents.

J. Dechaux (✉)
CVO Lab., Laval, France
e-mail: jonathan.dechaux@esiea-ouest.fr

1 Introduction

Since the case of the German chancelery in August 2007, the number of attacks with Office documents is truly meaningful. The risk of automated actions generated by pieces of code is higher, sophisticatd attacks can be performed with a simple spreadsheet. Most of people think that those type of documents are inert and people who know their existence are using them by necessity [11,12].

Microsoft Office and *LibreOffice (ex OpenOffice)* are the two main actors in the office suites development. Since 2003, both have been very innovative and inventive in communication market. The result is two programs with a strong concurrence and a fantastic explosion of functionalities.

The prevention of the user need to be more efficient. With Office features who is growing everyday and the inefficiency of the AV community, we need to protect the users by showing them the risks associated with this type of documents. Minos was designed to alert the user, not to attack a target. By a simple utilisation, every charge of computer security will be able to make some demos, in front of all collaborators [2].

All of our tests and this proof-of-concept have been tested both for Microsoft Office 2003, 2007 and 2010 and LibreOffice 3.4, 3.5, under the Windows 7 operating system (user without privileges, no UAC enabled) with AV active [3].

The paper is organized as follows. Section 2 presents the security of macros and focuses on the features that make the execution of macro automatic without raising any alert by the application. Section 3 deals with the functions who are used to modify this security. Section 4 presents the application and all the features around it, how to manipulate it. Section 5 deals with the future of the application, security that have not been treated, the application on other operating systems.

2 Security of Office suites

In this section, we are going to present the different security issues in the two office suites. We will suppose that the reader know what Office document are and what is the macro concept. Macros are scripts written in different languages (VBA for Microsoft Office; LibreOffice Basic, Python, Beanshell and Javascript for LibreOffice) in order to automatize actions [10].

Every macro has different security that determines its execution:

- Macro Security Level (Both suites)
- Trusted Locations (Both suites)
- Templates/Add-ins (Both suites)
- Trusted Documents (Microsoft Office)
- Macro Application (LibreOffice)

2.1 Microsoft Office

By default, in Office security policy the macros are not allowed to be run unless they are in a trusted location. However it is possible to modify this default security setting according to the need and wish of every user. We are going to work with the *Macro Security Level* and the *Trusted Locations* for *Microsoft Office*. We will explain where we can find those variables and how many different values can be setting up [5,7].

2.1.1 Macro Security Level

Since the beginning, *Microsoft Office* can set up four different level, except for *Access 2003*, who has only three different level. The four security level for macros are:

- **Level 4** - Disable all macro without notification (alert).
- **Level 3** - Disable all macro with notification (alert).
- **Level 2** - Disable all macro except digitally signed macros.
- **Level 1** - Enable all macro (non recommended; malicious code can be executed).

Level 3 is set up by default at the *Microsoft Office* installation.

Macro Security Level is handled in the Windows registry base. The registry key involved is located in the *HKEY_CURRENT_USER* section. The path with respect a given *Office* application application is:

```
Software\Microsoft\Office\<version>
\<application>\Security
```

where $< version >$ is the number for the version (11.0 (2003), 12.0 (2007), 14.0 (2010)) and $< application >$ is one element in {Word, Excel, Powerpoint, Access}.

The key for our purpose is the *Security* key. By default, at the application setup, this key does not contain any value and the level 3 is enabled. Whenever the user modifies the macro security level, a field appeared. This value is different with the version. For the 2003 version of *Microsoft Office*, the key is named *Level*, and for the 2007 and 2010 version, it's named *VBAWarnings*.

This REG_DWORD field can have four different values according to the security level chosen and the version of *Office*. For the 2003 version, the value is the number of the level:

- 4 (0x00000004) (level 4)
- 3 (0x00000002) (level 3)
- 2 (0x00000003) (level 2)
- 1 (0x00000004) (level 1)

For the 2007 and 2010 version, the value for the level 2 and 3 are reversed:

- 4 (0x00000004) (level 4)
- 2 (0x00000002) (level 3)
- 3 (0x00000003) (level 2)
- 1 (0x00000004) (level 1)

Since the 2003 version, *Outlook* is the only application who has conserved the original name *Level* and the original configuration for its *Macro Security Level*.

2.1.2 Trusted Locations

The *Trusted Locations* security appears with the 2007 version. It's only concern four applications:

- Word
- Excel
- Powerpoint
- Access

When installing the *Microsoft Office* suite, various trusted locations are set up. As defined in the *Office* security policy, any macro in those trusted locations will be executed by default, whatever may be the level of security chosen by the user. Part of these trusted locations are user-specific while the other are common to all users. In addition some of these trusted locations offer the possibility that the subfolders are also considered and managed as trusted ones. In some path, we can find the number of version installed.

Access By default, Access set up a single lobal (common) trusted location.

```
Location2:
C:\Program Files\Microsoft Office\
OfficeX\ACCWIZ\

where "OfficeX" is "Office12" for 2007
and "Office14" for 2010
```

Excel By default, six trusted locations are set up. Two are user-specific while the four other ones are common to all users.

```
Location0:
C:\Program Files\Microsoft Office
\OfficeX\XLSTART\
Location1 :
%APPDATA%\Microsoft\Excel\XLSTART
Location2 :
%APPDATA%\Microsoft\Templates
Location3 :
C:\Program Files\Microsoft Office\
Templates\
Location4 :
C:\Program Files\Microsoft Office\
OfficeX\STARTUP\
Location5 :
C:\Program Files\Microsoft Office\
OfficeX\Library\

where "OfficeX" is "Office12" for 2007
  and "Office14" for 2010
```

Locations 0, 3, 4 and 5 have the *AllowSubFolders* value set to 1 to make any of its subfolders trusted locations as well.

Powerpoint By default, four locations are set up. Two are user-specific and two are common to all users.

```
Location0 :
%APPDATA%\Microsoft\Templates
Location1 :
C:\Program Files\Microsoft Office\
Templates\
Location2 :
%APPDATA%\Microsoft\Addins
Location3 :
C:\Program Files\Microsoft Office\
Document Themes X\

where "Document Themes X" is "Document
Themes 12" for 2007 and
"Document Themes 14" for 2010
```

Locations 0, 1 and 3 have the *AllowSubFolders* value set to 1 to make any of its subfolders trusted locations as well.

Word By default, three locations are set up. Two are user-specific.

```
Location0 :
%APPDATA%\Microsoft\Templates
Location1 :
C:\Program Files\Microsoft Office\
Templates\
Location2 :
%APPDATA%\Microsoft\Word\Startup
```

Only *Location 1* has the *AllowSubFolders* value set to 1.

A path beginning with %APPDATA% refers, on a Windows 7 machine, to the absolute path:

```
C:\Users\<username>\AppData\Roaming\
```

As for the *Macro Security Level*, *Trusted Locations* are managed at the registry base as well, in the *HKEY_CURRENT_USER* registry section. The path with respect a given Office application application is:

```
Software\Microsoft\Office\<version>\
<application>\Security\Trusted
Locations
```

In the *Trusted Locations* area, we can find the different keys under the name *LocationX*. For example, we can find the *Location0*, *Location1* and *Location2* field for Word. For each *LocationX* key, there are two or three values. Two values, *Description* and *Path* are always present. The third one, *AllowSubFolders* is optional.

The two interesting values are *Path* and *AllowSubFolders*, when it's defined. The *Path* value describes as its name suggests it the path we intend to define as trusted. The value

AllowSubFolders, it is used to declare that subfolders in the path must be trusted as well. It is equal to 1 (0x00000001) most of the time.

Microsoft deny to put any drive letter as C: or D: to be a trusted location but if you try to put C:\, the application will allow you to put an entire drive as a trusted location.

2.2 LibreOffice

Since *OpenOffice* became *LibreOffice*, the name and the location of the configuration file and some values have changed. It's always an XML file, and we can find in it the variables for the *Macro Security Level*, the *Trusted Locations* and the *Macro Application*. The name of this file is *registrymodifications.xcu* and the path is:

```
C:\Users\<username>\AppData\Roaming\
LibreOffice\3\user
```

Since the beginning of the suite, each application, Text, Calc, Presentation, ..., has the same *Macro Security Level*, the same *Trusted Locations* and the same *Macro Application*. If you change, for example, the *Macro Security Level*, you change it for every application [1,9].

2.2.1 Macro Security Level

LibreOffice can set up four different level. The four security level for macros are:

- **Level 4** - Very High security level.
- **Level 3** - High security level.
- **Level 2** - Medium security level.
- **Level 1** - Low security level (not recommended).

Level 3 is set up by default at the *LibreOffice* installation.

By default, at the installation setup, the security level is not specified in this file. Whenever the user modifies the macro security level, a XML block appeared. The key is named *MacroSecurityLevel* and between the two tags $<value>$ and $</value>$, we found the value of this level.

```
<item oor:path="/org.openoffice.Office.
 Common/Security/Scripting">
 <prop oor:op="fuse" oor:name=
 "MacroSecurityLevel">
   <value>0</value>
 </prop>
</item>
```

This field can have four different values according to the security level:

- 3 (level 4)
- 2 (level 3)
- 1 (level 2)
- 0 (level 1)

2.2.2 Trusted Locations

By default, at the installation setup, the trusted locations are not specified in this file. Whenever the user modifies the trusted locations, a XML block appeared. The key is named *SecureURL* and between the two tags $<value>$ and $</value>$, we found all the trusted locations. Each location is separated by two tags $<it>$ and $</it>$

```
<item oor:path="/org.openoffice.Office.
 Common/Security/Scripting">
 <prop oor:op="fuse" oor:name=
 "SecureURL">
   <value>
     <it>file:///C:/</it>
   </value>
 </prop>
</item>
```

Contrary to Microsoft Office, subfolders are enabled by default.

2.2.3 Macro Application

By default, at the installation setup, one *Macro Application* is defined by *LibreOffice*. The module name is *Module1* and the name of the macro is *Main*. This macro is not linked to an event and it's an empty macro.

The folder where the Macros Application (for Basic language) are located is:

```
C:\Users\<username>\AppData\Roaming\
LibreOffice\3\user\Basic\Standard
```

In this folder you will find the *Macros Application* in eah file with an *.xba* extension.

The events linked the *Macros Application* are stored in the configuration file. The key is named *BindingURL* and between the two tags $<value>$ and $</value>$, we found a *Macro Application*. To see the event associated to the *Macro Application*, take a look at the property *oor:name* in the node named *node*. For example this *Macro Application* is linked to the event *At the opening of the file*:

```
<item oor:path="/org.openoffice.Office.
 Events/ApplicationEvents/Bindings">
 <node oor:name="OnLoad" oor:op=
  "replace">
   <prop oor:name="BindingURL" oor:op=
  "fuse">
     <value>vnd.sun.star.script:Standard.
      Module1.Main?language=
```

```
      Basic&location=application</value>
    </prop>
  </node>
</item>
```

3 Modification of Office security

Microsoft Office security is managed in the *Registry* and *LibreOffice* security is in an user file. We will use different C/C++ functions to find and modify this security [6].

3.1 Microsoft Office

As mentioned before, macro management is performed at the operating system level (registry base). The following functions from the Windows API enable to access the registry:

- RegOpenKeyEx
- RegCreateKeyEx
- RegCloseKey
- RegSetValueEx
- RegEnumKeyEx
- RegEnumValue
- RegQueryValueEx

RegOpenKeyEx This function enables to open a registry key.

```
LONG WINAPI RegOpenKeyEx(
   __in HKEY hKey,              /* Reg key handle */
   __in_opt LPCTSTR lpSubKey,   /* Subkey name */
   __reserved DWORD ulOptions, /* Reserved */
   __in REGSAM samDesired,      /* Desired rights */
   __out PHKEY phkResult        /* Result pointer */
);
```

RegCreateKeyEx This function enables to create a new registry key.

```
LONG WINAPI RegCreateKeyEx(
   __in HKEY hKey,            /* Key handle */
   __in LPCTSTR lpSubKey,     /* Subkey name to create */
   __reserved DWORD Reserved, /* Reserved field */
   __in_opt LPTSTR lpClass,   /* Key type (user defined) */
   __in DWORD dwOptions,      /* Key options */
   __in REGSAM samDesired,    /* Rights attached to the key */
   __in_opt LPSECURITY_ATTRIBUTES lpSecurityAttributes,
                              /* Process security level */
   __out PHKEY phkResult,     /* Pointer to the result */
   __out_opt LPDWORD lpdwDisposition
);
```

RegCloseKey This function just enables to close a registry key in a clean way.

```
LONG WINAPI RegCloseKey(__in HKEY hKey);
```

RegSetValueEx This function is very useful to modify values of a registry key.

```
LONG WINAPI RegSetValueEx(
   __in HKEY hKey,              /* Registry key handle */
   __in_opt LPCTSTR lpValueName, /* Name to modify */
   __reserved DWORD Reserved,    /* Reserved field */
   __in DWORD dwType,            /* Value type */
   __in_opt const BYTE *lpData,  /* Data to substitute */
   __in DWORD cbData             /* Data size */
);
```

RegEnumKeyEx This function is very useful to enumerates the subkeys of the specified open registry key.

```
LONG WINAPI RegEnumKeyEx(
   __in         HKEY hKey,
   __in         DWORD dwIndex,
   __out        LPTSTR lpName,
   __inout      LPDWORD lpcName,
   __reserved   LPDWORD lpReserved,
   __inout      LPTSTR lpClass,
   __inout_opt  LPDWORD lpcClass,
   __out_opt    PFILETIME lpftLastWriteTime
);
```

RegEnumValue This function is very useful to enumerates the values for the specified open registry key.

```
LONG WINAPI RegEnumValue(
  __in          HKEY hKey,
  __in          DWORD dwIndex,
  __out         LPTSTR lpValueName,
  __inout       LPDWORD lpcchValueName,
  __reserved    LPDWORD lpReserved,
  __out_opt     LPDWORD lpType,
  __out_opt     LPBYTE lpData,
  __inout_opt   LPDWORD lpcbData
);
```

RegQueryValueEx This function is very useful to retrieves the type and data for the specified value name associated with an open registry key.

```
LONG WINAPI RegQueryValueEx(
  __in          HKEY hKey,
  __in_opt      LPCTSTR lpValueName,
  __reserved    LPDWORD lpReserved,
  __out_opt     LPDWORD lpType,
  __out_opt     LPBYTE lpData,
  __inout_opt   LPDWORD lpcbData
);
```

For example to modify the *Macro Security Level* for *Word* 2010 to the lowest level:

```
HKEY    hKey;
TCHAR   path[] = TEXT("Software\\Microsoft\\
         Office\\14.0\\Word\\Security");
TCHAR   warning[] = TEXT("VBAWarnings");
DWORD   number = 0;
```

```
if( RegOpenKeyEx( HKEY_CURRENT_USER, path, 0,
   KEY_ALL_ACCESS,&hKey ) == ERROR_SUCCESS )
{
RegSetValueEx( hKey, warning, 0, REG_DWORD,
      (const BYTE *)&number, sizeof(number) );
   RegCloseKey( hKey );
}
```

3.2 LibreOffice

We will use the Qt Xml Parser, defined by *QDomDocument* and *QDomElement*. *QDomDocument* is used to define a XML document and we will use the function:

- documentElement
- createElement
- createTextNode

QDomElement is used to define a node in the XML document and we will use different functions:

- firstChildElement
- nextSiblingElement
- isNull
- tagName
- attribute
- setAttribute
- text
- insertAfter
- appendChild
- removeChild

documentElement This function returns the root element of the document.

```
QDomElement QDomDocument::documentElement
   () const
```

createElement This function creates a new element called tagName that can be inserted into the DOM tree.

```
QDomElement QDomDocument::createElement
   ( const QString & tagName )
```

createTextNode This function creates a text node for the string value that can be inserted into the document tree.

```
QDomText QDomDocument::createTextNode
 ( const QString & value )
```

firstChildElement This function returns the first child element with tag name tagName if tagName is non-empty; otherwise returns the first child element. Returns a null element if no such child exists.

```
QDomElement QDomNode::firstChildElement
(const QString & tagName =  QString() )
 const
```

nextSiblingElement This function returns the next sibling element with tag name tagName if tagName is non-empty; otherwise returns any next sibling element. Returns a null element if no such sibling exists.

```
QDomElement QDomNode::nextSiblingElement
 ( const QString & tagName =
  QString() ) const
```

isNull This function returns true if this node is null (i.e. if it has no type or contents); otherwise returns false. document tree.

```
bool QDomNode::isNull () const
```

tagName This function returns the tag name of this element. For an XML element like this:

```
QString QDomElement::tagName () const
```

attribute This function returns the attribute called name. If the attribute does not exist defValue is returned.

```
QString QDomElement::attribute ( const
 QString & name, const
 QString & defValue = QString()) const
```

setAttribute This function adds an attribute called name with value value. If an attribute with the same name exists, its value is replaced by value.

```
void QDomElement::setAttribute ( const
 QString & name,const QString & value )
```

text This function returns the element's text or an empty string.

```
QString QDomElement::text () const
```

insertAfter This function inserts the node newChild after the child node refChild. refChild must be a direct child of this node. If refChild is null then newChild is appended as this node's last child.

```
QDomNode QDomNode::insertAfter ( const
  QDomNode & newChild, const
  QDomNode & refChild )
```

appendChild This function appends newChild as the node's last child.

```
QDomNode QDomNode::appendChild ( const
 QDomNode & newChild )
```

removeChild This function removes oldChild from the list of children. oldChild must be a direct child of this node.

```
QDomNode QDomNode::removeChild ( const
 QDomNode & oldChild )
```

For example to modify the *Macro Security Level* for *LibreOffice* to the lowest level:

```
int    number = 0;
QFile  xml_doc("C:\\Users\\<username>\\AppData\\Roaming\\
               LibreOffice\\3\\user\\
               registrymodifications.xcu");

if(xml_doc.open(QIODevice::ReadOnly))
{
  QDomDocument doc;
  if(doc.setContent(&xml_doc, false))
  {
    QDomElement root = doc.documentElement();
    QDomElement racine = root.firstChildElement();

    QDomElement item = doc.createElement("item");
    item.setAttribute("oor:path",
     "/org.openoffice.Office.Common/Security/Scripting");

    QDomElement prop = doc.createElement("prop");
    prop.setAttribute("oor:name", "MacroSecurityLevel");
    prop.setAttribute("oor:op", "fuse");

    QDomElement value = doc.createElement("value");
    QDomText data = doc.createTextNode(TextNumber);

    root.appendChild(item);
    item.appendChild(prop);
    prop.appendChild(value);
    value.appendChild(data);

    xml_doc.close();

    if(xml_doc.open(QIODevice::WriteOnly))
      xml_doc.write(doc.toString(4).toUtf8());
    xml_doc.close();
  }
}
```

4 The Office Demon: Minos

Minos was designed to be efficient and easy to use. He was made with an interface developped with the Qt environment and different functions mixing Qt programmation language and C/C++ functions.

4.1 Interface

Minos was developped with the *Qt developpment SDK*. We use a 32-bit version of *Qt*. For the appearance of *Minos*, we use a *CSS* sheet who is working very well with *Qt*. For the programmation of every functions, we use the *C* language and the *Qt* language.

4.1.1 Tabs

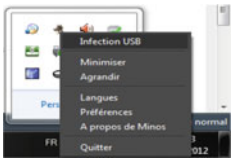
Minos has two general tabs, *Security* and *Documents*. The tab *Security* has all features for managing security for both suites. It has two sub-tabs, *Microsoft Office* and *LibreOffice*. Each sub-tab manage the security for the suite with the *Macro Security Level*, the *Trusted Locations* and *Macro Application* for *LibreOffice*.

The tab *Documents* allows you to infect a document, a folder or a USB device. You can infect *Word*, *Excel*, *Text* and *Calc* documents. You can inject your macro in the document next to other macros or replace all macros by your macro. The *Destination* field is require because all the files infected will be dumped before the infection.

4.1.2 Icon and Windows Menu

Minos has an icon in the windows sidebar. When you try to close *Minos*, a pop-up will tell you that *Minos* is still active. If you take a look at the right bottom, you can find the icon of *Minos*.

If you double-click on the icon, *Minos* will show up. If you right-click on the icon, a menu will appear. You can maximize, minimize the application, you can change the language, change the preferences, you can put on the infection for USB device (only if you have done this before) and you can quit the application.



4.1.3 Languages

We developped *Minos* in two languages, *English* and *French*. By clicking on the *Options* tab and selecting *Langues (Languages)*, a messagebox ask you what is your language. When you click on English or French, the entire application go to this language. All buttons, messagebox, questions, ... are turned into this language.

To do this we used the tool *Qt Linguistic*. This tool allow you to associate a french sentence in the chosen language. When you will activate the language that you want to use, the translator will change. We use two fonction to load and unload a translator:

- installTranslator
- removeTranslator

If in the future we want to make a *Spanish* or *German* version for example, we just need to create a translator and translate all sentences in that language. It's an easy way to have a multi-languages application.

4.1.4 Windows startup

An other point, developped in *Minos*, is the possibility to start with the windows session. *Minos* will not start for every user but just for the user that used *Minos*. By clicking on the *Options* tab and selecting *PrÃl'fÃl'rences (Preferences)*, a pop-up will appear with a checkbox.

If you choose to run *Minos* on every session start, Minos will create two registry values:

- `logon in HKEY_CURRENT_USER\Software\Microsoft\Minos`
- `minos in HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`

The value *logon* is defined for setting the configuration for an other run of *Minos*. When you first check the box for starting *Minos* with Windows, the next start *Minos* will be aware that the starting with Windows is in place, and the box will be checked. This value *logon* is set to 1 for a starting with Windows and delete to not start with Windows.

The value *minos* is the path of *Minos*, defined in the right place in the registry to start at the opening of the session. We add an argument to this path to make a difference between starting with Windows or starting by the user. When the application start and find an argument, she starts with no apparent window but the application is present in the icon menu of Windows.

4.2 Modification of security

4.2.1 Macro Security Level

The principle est exactly the same between *Microsoft Office* and *LibreOffice*, only functions used are different. For *Microsoft Office* you can choose one application or all of them and for *LibreOffice* it's the same *Macro Security Level* for all applications. So you select your application and you click on the button *Analyze*. The color and the number of the level will be displayed.

After that, by scrolling down in the list of your applications, you can choose the level and change it. Simply select a new level and click on the button *Modify*. The color and the number will change. To reset all colors, numbers, etc, click on the button *Reset*.

4.2.2 Trusted Locations

The principle est exactly the same between *Microsoft Office* and *LibreOffice*, only functions used are different. For *Microsoft Office* you choose one application and for *LibreOffice* it's the same *Trusted Locations* for all applications.

To display how many *Trusted Locations* are present, click on the button *Analyze*. Every *Trusted Locations* for the application will be displayed. If you want to add a new *Trusted Location* for the application, you have to select the path of a folder with the button *Browse*. If you want to modify a path, simply check the path and select the path with the button Browse. When you have selected a new path, click on the button *Modify*. To delete a *Trusted Location* select it and click on the button *Delete*.

4.2.3 Macro Application

This security is bigger and more complex to manage. You have the macro and the macro event to manage. When you click on the button *Analyze*, all macros application and all events will be displayed. If you select a *Macro Application* and click again on *Analyze*, the text of the macro will be displayed on the top.

To add a new macro, you can select a *txt* file with the button *Browse* or write directly the text of your macro in the top area. When you will click on the button *Add*, a pop-up will ask you the name of the macro module. To add a new event with a macro, select the macro on the left side and select an event in the list, click on the button *Add*.

To modify a macro, select the macro, analyze it, modify the text and click on the button *Modify*. To modify a macro event, select the event to modify, select a new event in the list and click on the button *Modify*. To modify the macro associated to an event, simply delete the event, and recreate the event with the right macro. To delete a macro or an event, select it and click on the button *Delete*.

4.3 Document infection

4.3.1 Static infection

Minos is based on *USB Dumper* for the documents infection. The principle is to call the documents API to manage the macro part of the document. USB Dumper was limited in the injection type. *USB Dumper* allows you to inject a macro in a *Word* and an *Excel* document, and if a document contains a macro, the result will be approximate, may be the macro will be inserted or not.

Minos can infect the *Word* and *Excel* format and the counterparts in *LibreOffice*, *Text* and *Calc*. You can replace all the macro or inject your macro next to the others macro.

You have to fill in the *Destination* folder, for the dumping part. Click on the button *Browse* and select your folder. Do the same for the *Target* folder. After that you have to select your macro for the format you want to infect. If you want to infect the *Word* format, click on the button *Browse* and select a *txt* file who contains your macro. Do the same for the *Excel* field if you want ton infect the *Excel* format too.

If you want ton infect the *Text* format, you have to select your macro, with the button *Browse*, and select the event for your macro, with the list. Do the same thing for the *Calc* format, if you want to infect it. After that you have to choose your infection type between *Substitution* and *Injection*. The better infection is the injection mode. The macro will be implanted next to other macros but its action will be perform first.

4.3.2 Dynamic infection

The principle for the *Dynamic infection* is the same as the *Static infection*. The difference is only in the *Target* field. When you will start the USB infection, the field target need to be empty and will not be considered after. When a USB device will be plugged, all documents will be copied automatically to the destination, and for each format choosen, the infection will be made.

Minos doesn't make the infection directly on the USB device. Each file will be copied to the *TMP* folder of the user, renamed with a temporary name, and infected. If the infection works, the file will be copied on the USB device. We made this process to prevent the detection by the user. The only black point is if the user works directly on the USB device. We think to implement an error management for the files copying.

5 Future developments

In the future, two new features will be implemented:

- Infection of Template/Add-Ins
- Infection of Trusted Documents

This two points are the two last security that Minos did not modified. For the *Template/Add-Ins*, the first move will be to modified the *Word Template*, who is already active in each new *Word* document. After that we will enlarge this modification to all *Template/Add-Ins* of *Microsoft Office* and *LibreOffice*.

For the *Trusted Documents*, the idea is to search some *Trusted Documents* in the registry and modify the macro of those files. This point is almost done with the previous work of infecting documents, we just need to manage the interface to add a new tab and make the search algorithm.

We will put a new tab to embed some executables in an *Office* documents. The user will choose what macro he will play in the document and what executable the document needs to embed for a future extraction. This point is almost done with the Zip/Unzip part of *Minos*.

An other work is to create a *Minos* for *Linux* and *Mac*. *Qt* is designed for crossing-platorm so normally we will use the same interface. We will need to find the right path for *Microsoft Office* and *LibreOffice* and their security on those operating systems [8].

6 Conclusion

Minos includes a lot of tools to show the danger of macros. It's not finished and it will be not because of the rise of *Office* applications. The detection by antivirus software fails most of the time. The main reason of this is that antiviruses think legitimate to use some actions produces by legitimate users applications.

Office documents represents a very high risk considered as inerts by most of people. This misperception increasing the risks of attacks on sensitive systems. The user needs to be an entire part of the security [4].

Minos is only here to prevent but the security no longer relies on the applications. Most of applications' security is defined and managed at the operating system level. *Minos* is here to work with the user not for the user. Each person

who is working with some documents needs to understand the risk and the importance of actions.

References

1. Dechaux, J., Filiol, E.: Microsoft office vs libreoffice: Security comparison regarding viral attacks (2011). http://conference.libreoffice.org/
2. Dechaux, J., Filiol, E., Fizaine, J.-P.: Office documents: new weapons of cyberwarfare (2010). http://archive.hack.lu/2010/Filiol-Office-Documents-New-Weapons-of-Cyberwarfare-paper.pdf
3. Dechaux, J., Filiol, E., Fizaine, J.-P.: Perverting emails; a new dimension in internet (in) security (2011). http://academic-conferences.org/eciw/eciw-home.htm
4. Dechaux, J., Fizaine, J.-P.: Returning the trust against the user (2010). http://www.esiea-recherche.eu/data/iawacs2010/slides/dechaux_fizaine_iawacs2010.pdf
5. Desnos, A.: Implementation of k-ary viruses in python. In: Hack.lu 2009 (2009). http://2010.hack.lu/archive/2009/kaires.pdf
6. ESIEA: Pwn2kill challenge. iAWACS 2010 (2010). http://www.esiea-recherche.eu/iawacs_2010_en.html
7. Filiol, E.: Formalisation and Implementation Aspects of K-ary (malicious) Codes. Springer, France (2007)
8. Filiol, E.: Les virus informatiques : théorie, pratique et applications, 2nd edn. Springer, France, ISBN: 978-2-287-98199-9 (2009)
9. Filiol, E., Fizaine, J.-P..: Openoffice v3.x security design weaknesses. In: Black Hat Europe 2009 (2009). http://www.blackhat.com/html/bh-europe-09/bh-eu-09-archives.html#Filiol
10. Mansfield, R.: Mastering VBA for Microsoft Office 2007. Wiley, New York, ISBN: 978-0470279595 (2008)
11. Nourdine: Cyber-attaque contre la france : les détails (2010). http://www.lemondenumerique.com/article-27046-cyber-attaque-contre-la-france-les-details.html
12. Spiegel Online: Merkel's china visit marred by hacking allegations (2007). http://www.spiegel.de/international/world/0,1518,502169,00.html