REPUBLIC OF ALBANIA
NATIONAL AUTHORITY FOR ELECTRONIC CERTIFICATION AND CYBER SECURITY
DIRECTORATE OF CYBER SECURITY ANALYSIS

Technical analysis for REMCOS RAT malicious file

Version: 1.0
Date: 09/04/2024

Table of Contents:

Executive Summary 4
Technical Information 5
Analysis of iAFV.exe file 5
Dynamic Analysis of Tyrone.dll 13
Static Analysis of Remcos RAT 15
Dynamic Analysis of Remcos RAT 18
Indicators of Compromise 21
MITRE Techniques 21
Recommendations 22

Table of Figures

Figure 1: Infection chain 5
Figure 2: Information about the iAFV.exe file 6
Figure 3: Windows Form Application 6
Figure 4: CCZ byte array 7
Figure 5: Calling the variable czz and executing 8
Figure 6: CCVC() function 8
Figure 7: .exe file detected 9
Figure 8: Gamma(RecationDiffusionLib 9
Figure 9: Calling the variable zyWL. 10
Figure 10: Base64 encoded value zyWL. 10
Figure 11: image.bpm decoded zyWL 10
Figure 12: Packers Analysis 11
Figure 13: Unpacking phase 12
Figure 14: Tyron.dll 12
Figure 15: Code Obfuscation 13
Figure 16: XML on Privileges 13
Figure 17: Importing the DLL 14
Figure 18: Persistence in the system 14
Figure 19: Task Scheduler - Scheduled System Tasks 15
Figure 20: Creation of the file iAFV.exe 15
Figure 21: REMCOS RAT 15
Figure 22: Function SendInput (winuser.h), Keylogger 16
Figure 23: Functions called towards the library 17
Figure 24: Downloading files 17
Figure 25: Executing commands in cmd 18
Figure 26: Analysis in debugger 18
Figure 27: logs.dat 19
Figure 28: The criticality of IP 194[,]187[,]251[.]115 according to VirusTotal 19
Figure 29: Categorization from Threat Intel Platform 19
Figure 30: URL of Command and Control (C2) 20
Figure 31: Command and Control 20
Figure 32: Data Exfiltration 20
Figure 33: URL in Json 21

This report is written to document and analyze attempted cyber attacks on critical infrastructures in the Republic of Albania.The content of this report is based on the information available up to the date of completion of the analysis.

The forwarding of this report aims to inform and raise awareness of the interested parties on the documented cyber incident. The report should not be treated as final until its final update.

This report has limitations and should be interpreted with caution!

Some of these restrictions include:

The first stage:

Sources of information: The report is based on information available at the time of its preparation. Meanwhile, some aspects may be different from current developments.

The second stage:

Analysis Details: Due to resource limitations, some aspects of the incident may not have been analyzed in depth. Any additional unknown information may reflect changes in the report.

The third stage:

Limited Analysis: Due to the complex nature of the cyber attack, the analysis may be limited in some aspects. The interpretation of the event is subjective and may be affected by the absence of some key data.

The fourth stage:

Information Security: To protect confidential resources and information, some details may be redacted or not included in the report. This decision was made to maintain the integrity and security of the data used.

AKCESK reserves the right to change, update, or change any part of this report without prior notice.

This report is not a final document (extraction of the malicious actors' input details will be made available to you at a later time).

The findings of the report are based on the information available at the time of the investigation and analysis. There are no guarantees regarding possible changes or updates to the information reported during the following period. The authors of the report assume no responsibility for the misuse or consequences of any decision-making based on this report.

Executive Summary

The authority conducted a detailed technical analysis of the malicious file Remcos Remote Access Trojan (RAT), which targeted a critical infrastructure within the Republic of Albania. This report summarizes the findings from both static and dynamic analysis of the malicious file, highlighting key indicators of compromise, techniques employed by the malicious file based on the MITRE ATT&CK framework, and provides recommendations to mitigate the threat.

Key Findings:

The malicious file was identified through the analysis of suspicious files associated with a Phishing campaign in Albania. The analysis confirmed that the files belong to the Remcos RAT family, a type of virus that enables remote operations by malicious actors, including keylogging, audio and video collection, and exfiltration of browser history and credentials. Detailed examinations were performed on various components of the malicious file, including iAFV.exe, Tyrone.dll, and other related files, revealing their properties and the sophisticated methods used to evade detection by defensive systems (antivirus) and detailed analysis.

Indicators of compromise were identified, including hash values for various files and network indicators, providing vital data for cybersecurity defenses.

The report underscores the need for vigilance and proactive measures against sophisticated cyber threats, highlighting the importance of regular updates and the implementation of recommended security practices to protect critical infrastructure.

Based on the analysis performed and the artifacts found, referencing the techniques, tactics, and procedures used, it is believed that the group behind the Phishing campaign could potentially be the Iranian APT33 group.

This is attributed to the fact that this group utilizes TTPs (Tactics, Techniques, and Procedures) as presented in the analysis as follows:

- Use of PowerShell, a legitimate Windows application for executing commands.
- Use of various techniques to bypass defensive systems and antivirus software.
- Use of legitimate applications to conceal malicious code.
- Use of junk files to confuse code analysis and their storage in the Temp directory.
- Use of tools to store passwords and other elements of the victim.
- Use of base64 encoding to facilitate Command and Control (C2) communication.
- Use of mass campaigns targeting infrastructures, organizations, and state entities.
- Creation of persistence in affected systems by hiding in legitimate services and programs.
- Use of malicious files to add other necessary files during the attack.
- Use of the C# language with the corresponding .NET library.
- Use of scheduled tasks of the Windows operating system as legitimate to create

persistence.

- Use of malicious files to store sensitive data from the victim's computer.
- Use of XOR and other complex algorithms to hide malicious code.

Technical Information

Referring to recent reports of a phishing attack campaign in Albania, several suspected malicious files were downloaded for analysis. During the static and dynamic analysis of these files, it was determined that one of the files belongs to the Trojan family, specifically the Remcos Remote Access Trojan (RAT). This RAT enables various remote operations by malicious actors. The analysis also revealed that this virus is capable of executing keylogging, taking screenshots, collecting audio and video, and leaking browser history and credentials. Additionally, indicators of compromise and Command and Control (C2) servers were identified during the investigation.

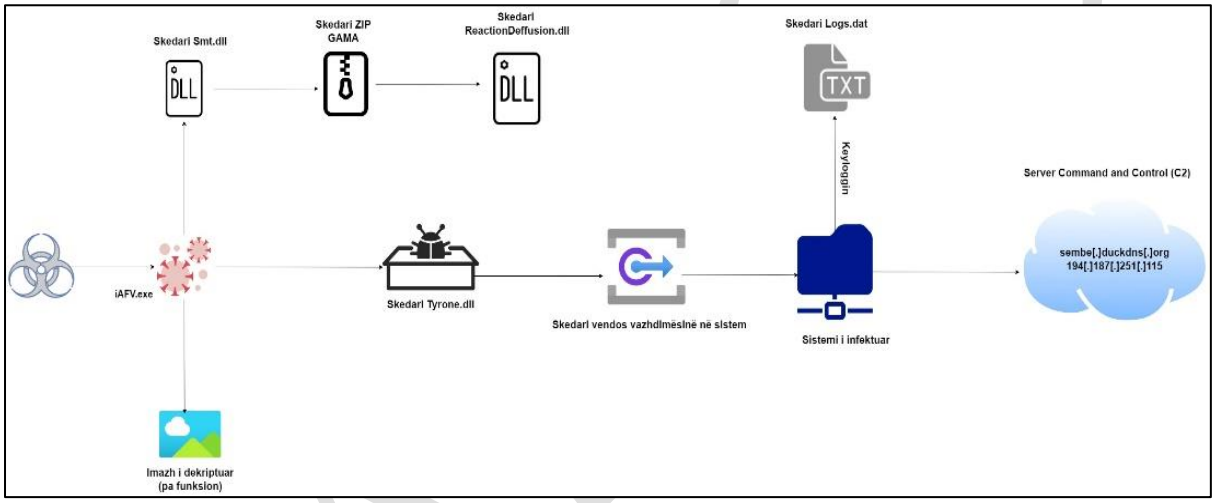


Figure 1: Infection chain

Analysis of iAFV.exe file

The executable file **iAFV.exe** is a .NET library file written in the C# programming language.
Sha256: f4eaa74eb268a58cff6f5d37607758bd49cc00af060da799857ae10cfd59efb2

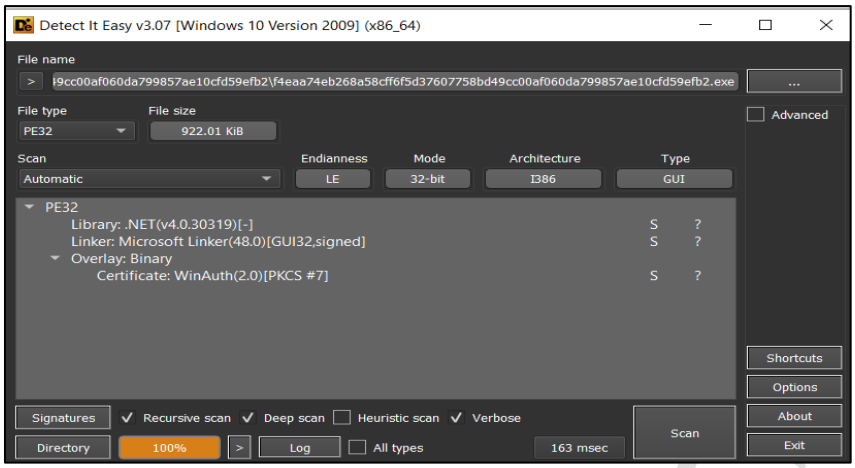


Figure 2: Information about the iAFV.exe file

During the analysis of the decompiled code, it was observed that the exported project appears to be intended for storing appointments and features UI buttons in Polish. When the code segment activating the CCZ object is commented out, the program functions as a Windows Form application and appears legitimate as shown in Figure 2, without presenting any malicious code.

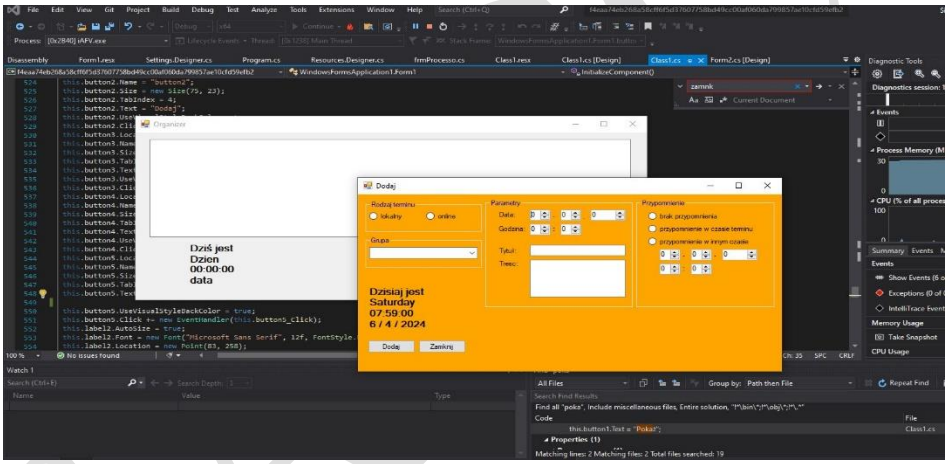


Figure 3: Windows Form Application

At first glance, the code appears legitimate and non-malicious, but within the frmProcesso.resx file, encoded in XML format, there is a variable named "ccz" of type byte array encoded.

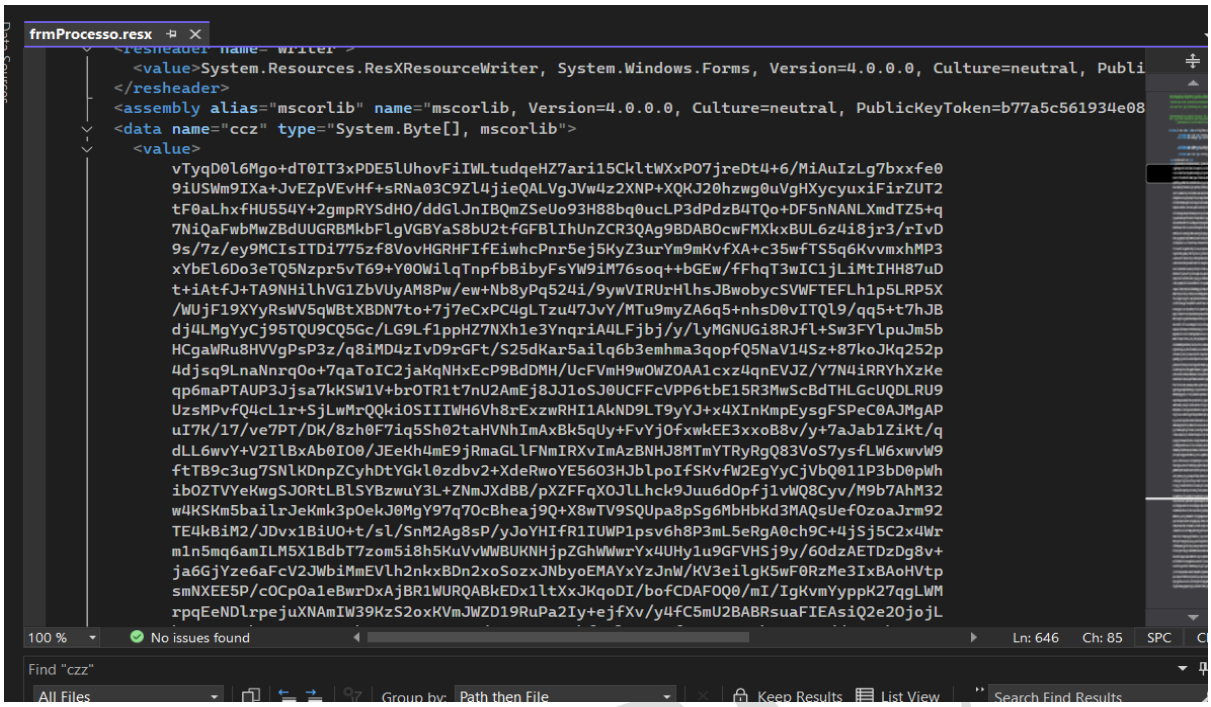


Figure 4: CCZ byte array

In the project file **Class1.cs**, a call to this variable is evident, which fills a byte array named **numArray1** using the **GetObject("ccz")** function of the **ComponentResourceManager** class. Additionally, the presence of a second byte array named **numArray2,** which is filled by calling the function **CCVC()** located in the **Form1.cs** file, is noted.

Subsequently, through a for loop, the byte array **numArray1** is modified using an algorithm that also utilizes byte array n**umArray2**. The calculation within the for loop involves **XOR**, indicating an encryption mechanism.

There is also the use of the **Activator** class, which passes **numArray1** as a parameter to the **CreateInstance** function and calls the **InvokeMember** function. The way in which the loading occurs is unconventional as it joins two strings **l.toUpper()** which is the letter **L + oad**, thus forming **"Load"**. This concatenation of characters is a common technique used to bypass antivirus defense systems. Additionally, it uses the first two characters of a string **array "7A79574C*6E6573"** passed as a parameter, further indicating sophisticated manipulation to evade detection.

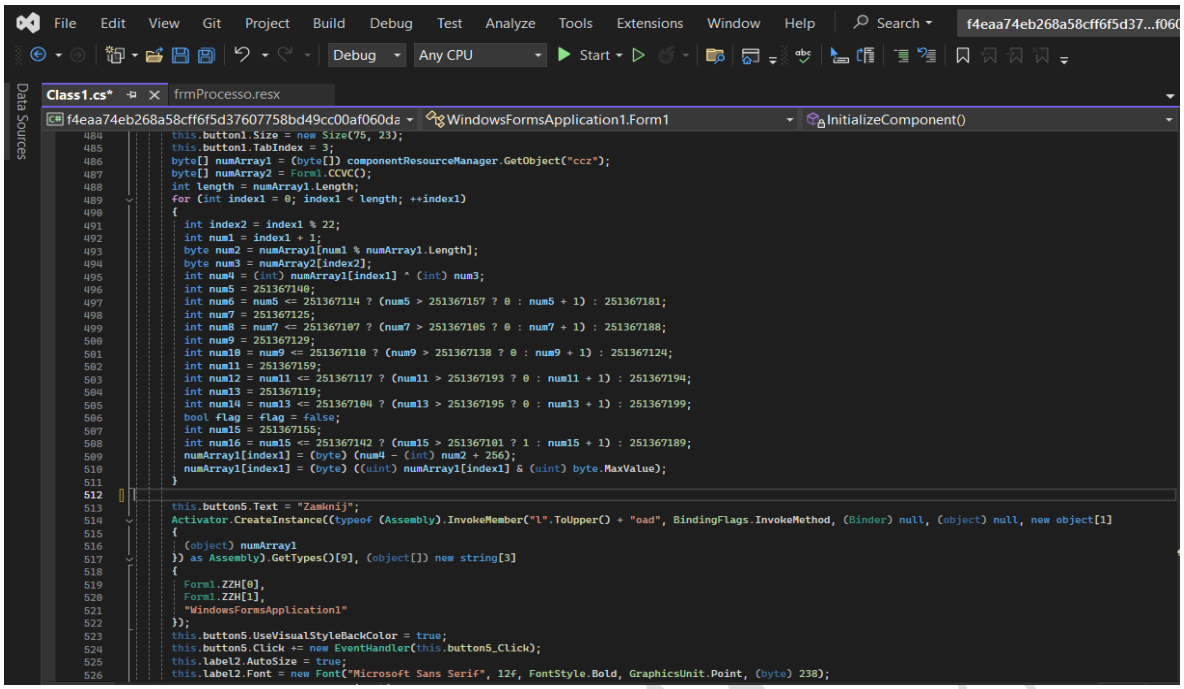


Figure 5: Calling the variable czz and executing

```
1 reference
private static byte[] CCVC()
{
  return ((IEnumerable<byte>) new byte[13]
  {
    (byte) 52,
    (byte) 56,
    (byte) 53,
    (byte) 70,
    (byte) 52,
    (byte) 72,
    (byte) 56,
    (byte) 52,
    (byte) 71,
    (byte) 72,
    (byte) 53,
    (byte) 71,
    (byte) 52
  }).Concat<byte>((IEnumerable<byte>) new byte[9]
  {
    (byte) 50,
    (byte) 67,
    (byte) 66,
    (byte) 83,
    (byte) 55,
    (byte) 72,
    (byte) 53,
    (byte) 57,
    (byte) 52
  }).ToArray<byte>();
}
}
```

Figure 6: CCVC() function

When a line of code is added after the **for loop** in the project code with **Console.WriteLine(numArray1)** to understand the behavior with the **byte array**, it becomes apparent from the output that we have the **HEX** values **4A 5D.** These values suggest that we

are dealing with an executable file. Therefore, these values are saved to a file, and subsequent analysis confirms that we are indeed dealing with a file written in **ASP.NET** using **C#.**

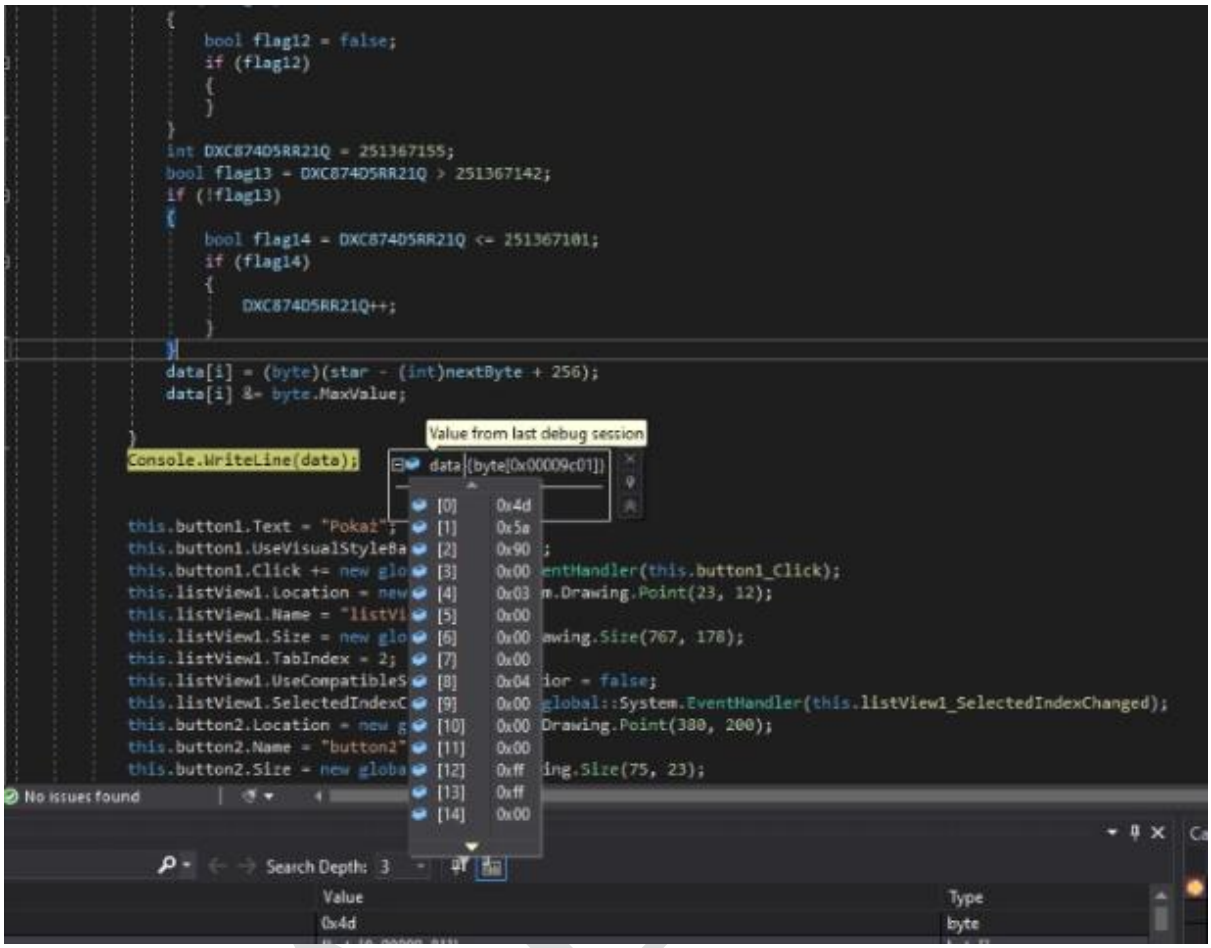


Figure 7: .exe file detected

We export the code again as a project and notice that the project is named Smt.csproj. In this project, no significant details are observed except for .resx files containing encoded values, which in this case are encoded in base64. After decoding the base64 value, a gzip category file is revealed, and we download it to examine its contents. Upon importing, the project is named Gamma. From the extraction of this file, we find another project, but in this case, the file is a DLL named ReactionDiffusionLib. Code analysis of this project reveals that this DLL is nothing but a decoy to confuse the analysis.

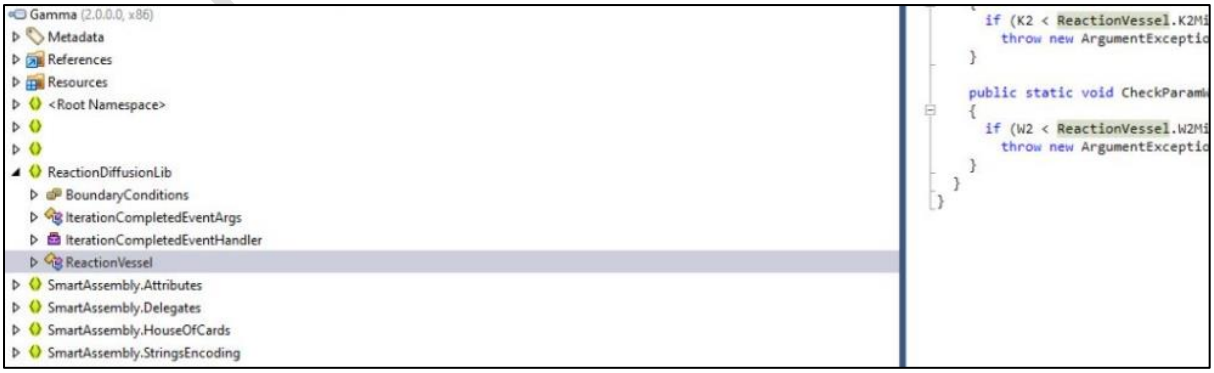


Figure 8: Gamma(RecationDiffusionLib

In addition to the variable '**ccz**', the main project code contains another encoded value named '**zyWL**'. From the code, it is understood that this pertains to a Bitmap file. Upon retrieving the value of this variable, we attempt to decode it and obtain an image.

```
1 reference
internal static Bitmap zyWL
{
    get => (Bitmap) WindowsFormsApplication1.Properties.Resources.ResourceManager.GetObject(nameof (zyWL), WindowsFormsApplication1.Properties.Resources.resourceCul
}
}
```

Figure 9: Calling the variable zyWL.

```
<data name="zyWL" mimetype="application/x-microsoft.net.object.binary.base64">
  <value>AAEAAAD/////AQAAAAAAAAAMAgAAAFFTeXN0ZW0uRHJhd2luZywgVmVyc2lvbj00LjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPWIwM2Y1ZjdmMTFkNTBhM2EFAQAAABVTeXN0
</data>
</root>
```

Figure 10: Base64 encoded value zyWL.

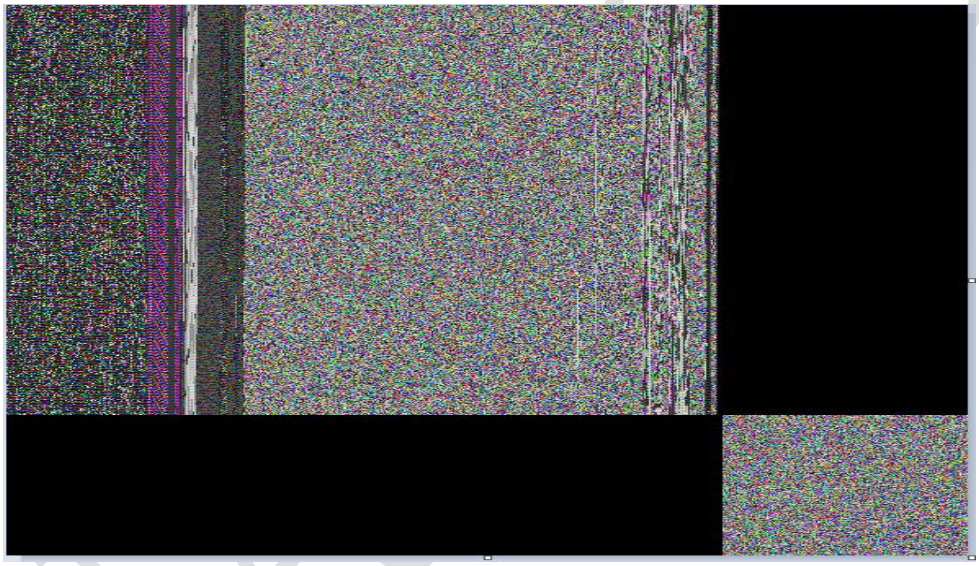


Figure 11: image.bpm decoded zyWL

It is evident that there are numerous encoded values within the code, and it appears that various packers and obfuscators are being used.

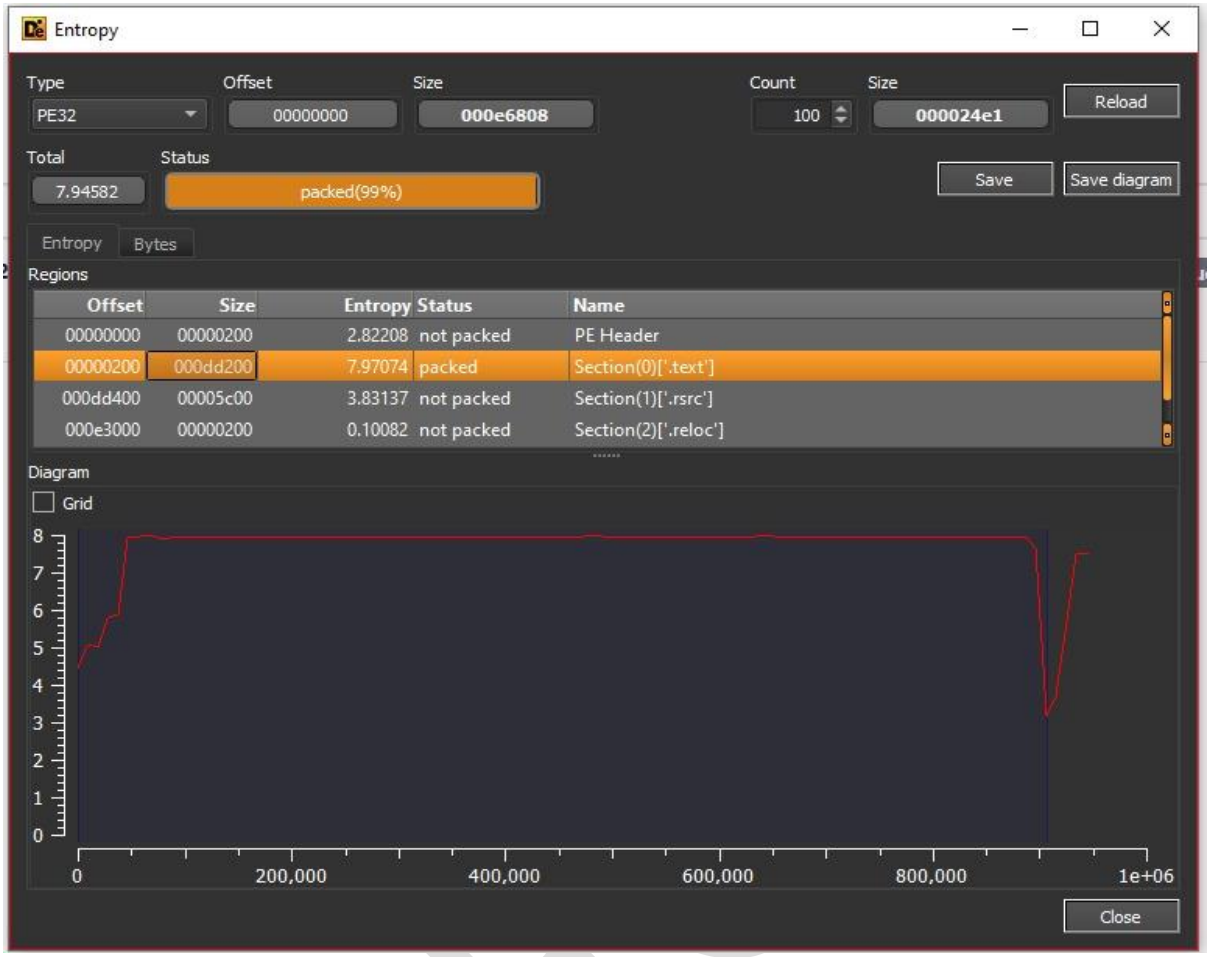


Figure 12: Packers Analysis

We employ an unpacker and reveal that from the main file, three child files emerge. The file with the hash:
sha256 **23f10d177ec53b6c4589adc03621906d7c65b9ae8ec4ff402ebd287014dbbcae** is the **Tyrons.dll** file.

The file with the hash sha256:
71dab87ac5b7b80468ef8ccb16b74b39cc862b7fb9a6e430e4cd7e375dbe6c27 is the Smt.dll file identified in the analysis above.

The most interesting file is the last one, which carries an **icon**. This **icon** is recognizable as it is used in the **REMCOS RAT** malicious file. It is observed that we have an entropy (level of concealment) above 5, indicating that we are dealing with packed (hidden) code.

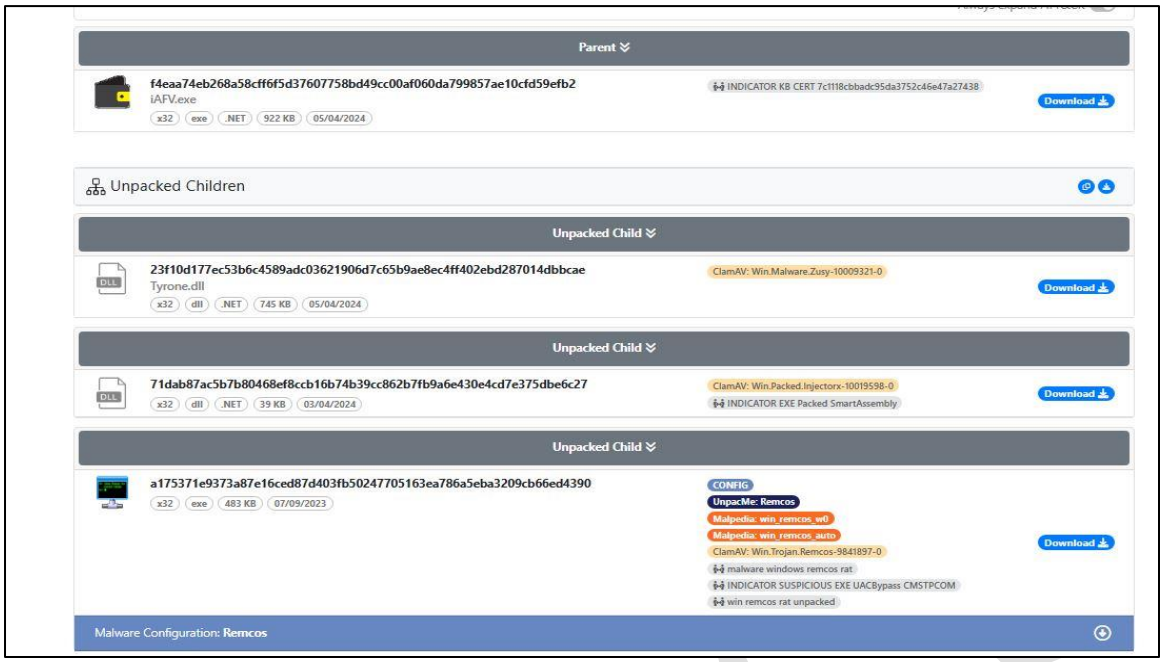


Figure 13: Unpacking phase

We import the file Tyrone.dll as a project and observe that it is written in ASP.NET, but it has a very high level of obfuscation, making it difficult to understand its purpose. The only viable approach remains through DEBUG in dynamic analysis.

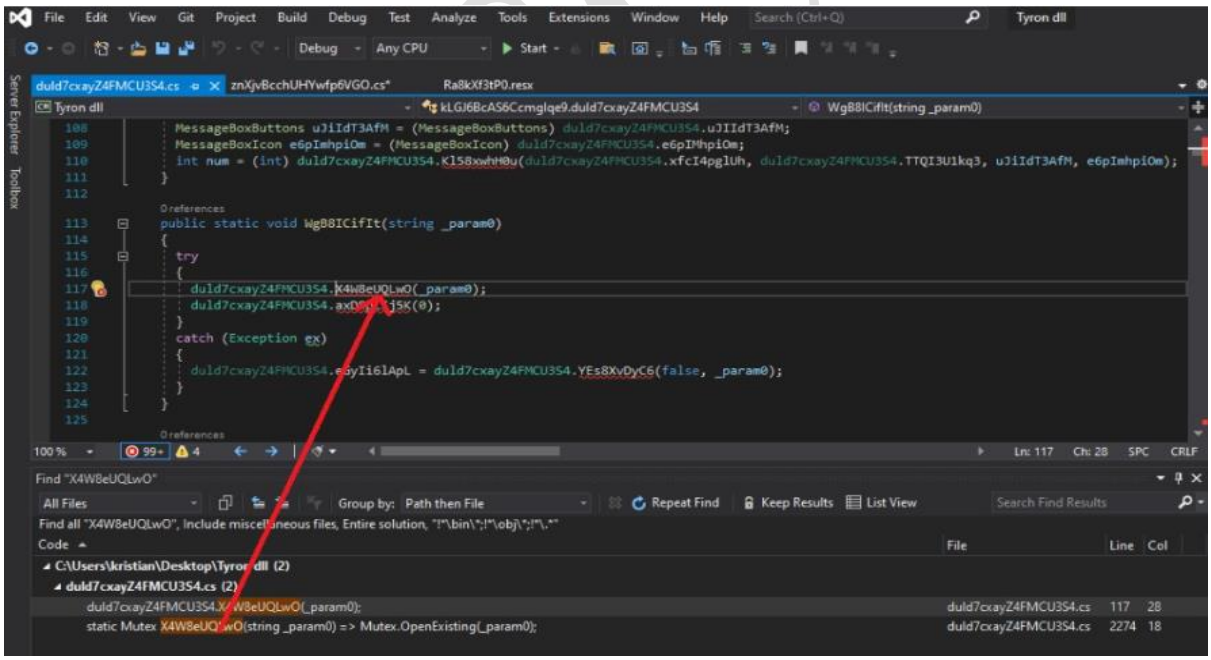


Figure 14: Tyron.dll

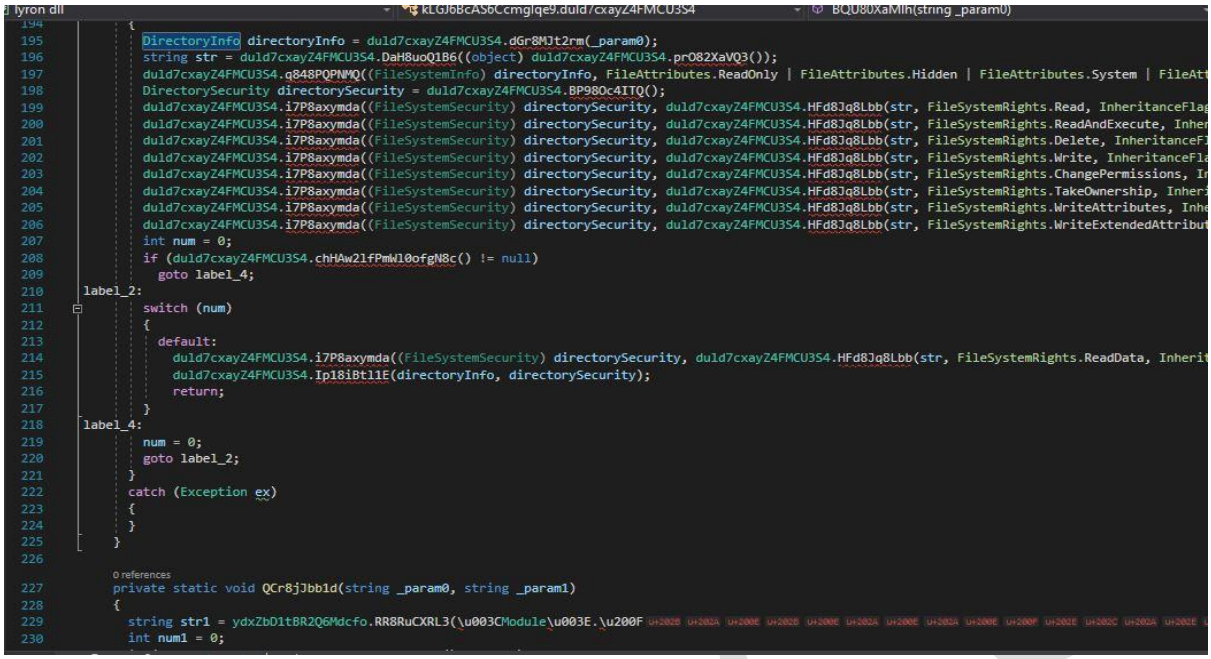


Figure 15: Code Obfuscation

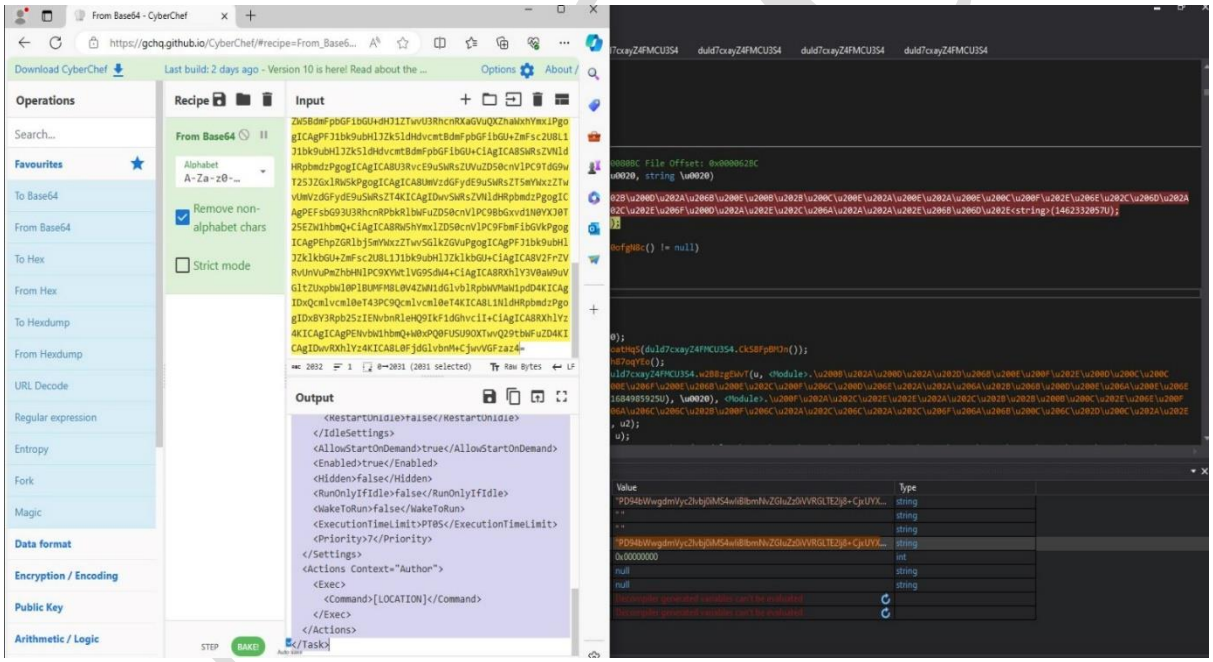


Figure 16: XML on Privileges

Dynamic Analysis of Tyrone.dll

To determine the function of this DLL, we need to import it into a self-created Console App project and follow every function via breakpoints.

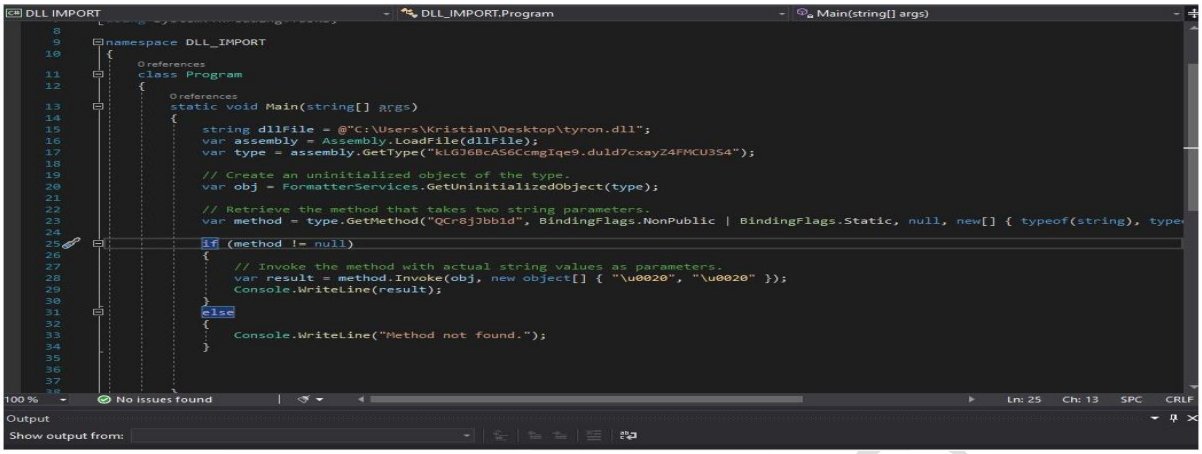


Figure 17: Importing the DLL

A simulation of the code was performed during the analysis of the **DLL** stored on the Desktop, and we **invoke** the hidden functions. In the case of the figure, we have the function **QcrB8Jbobld().** We place a breakpoint in the DLL code at this function and follow it step-by-step to see the output values.

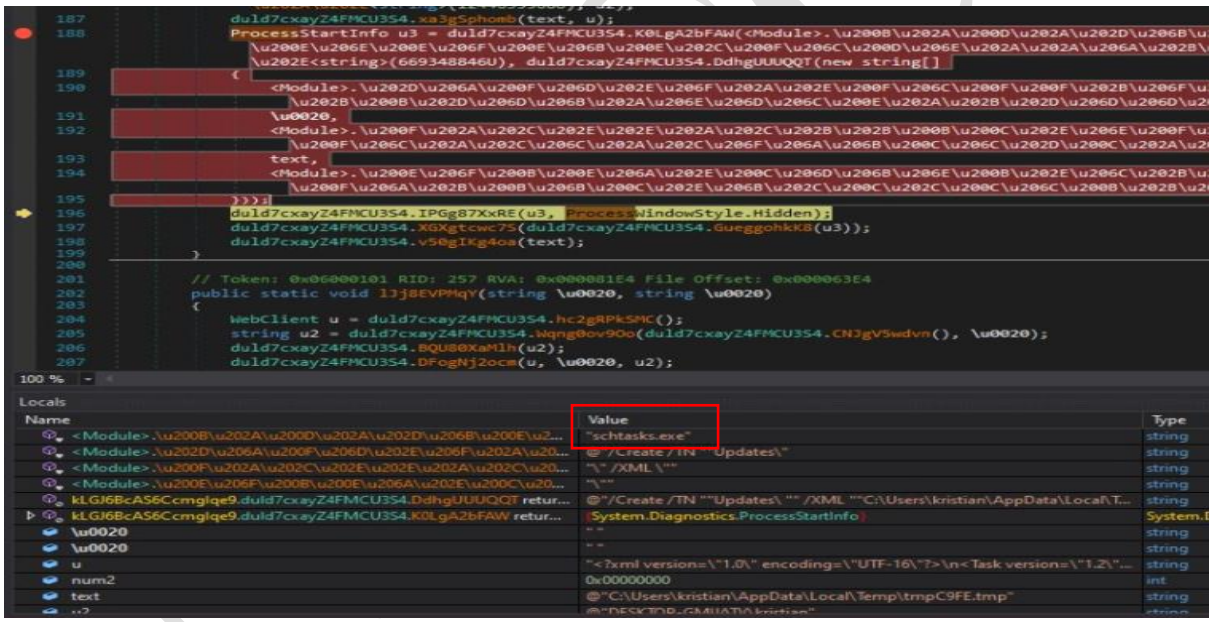


Figure 18: Persistence in the system

From the values in the debugger, we understand that there is a process execution aimed at creating persistence since a task scheduler is created in the file **C:\Users\PC\AppData\Local\temp\tempC9FE.tmp**

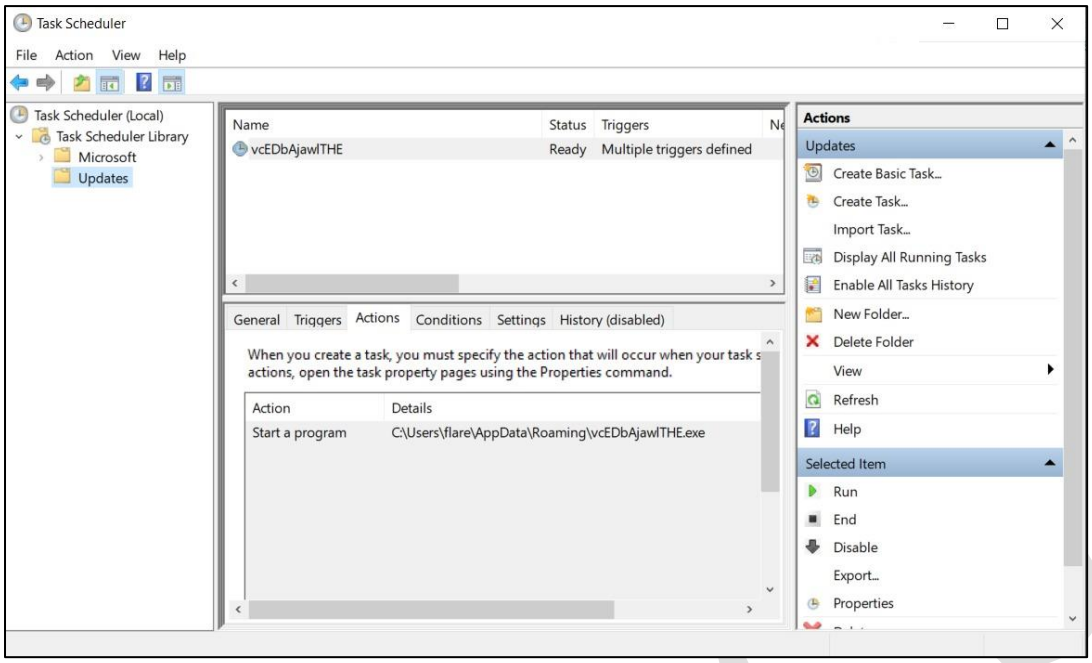


Figure 19: Task Scheduler - Scheduled System Tasks

The file that is created and executed is shown in the figure below.

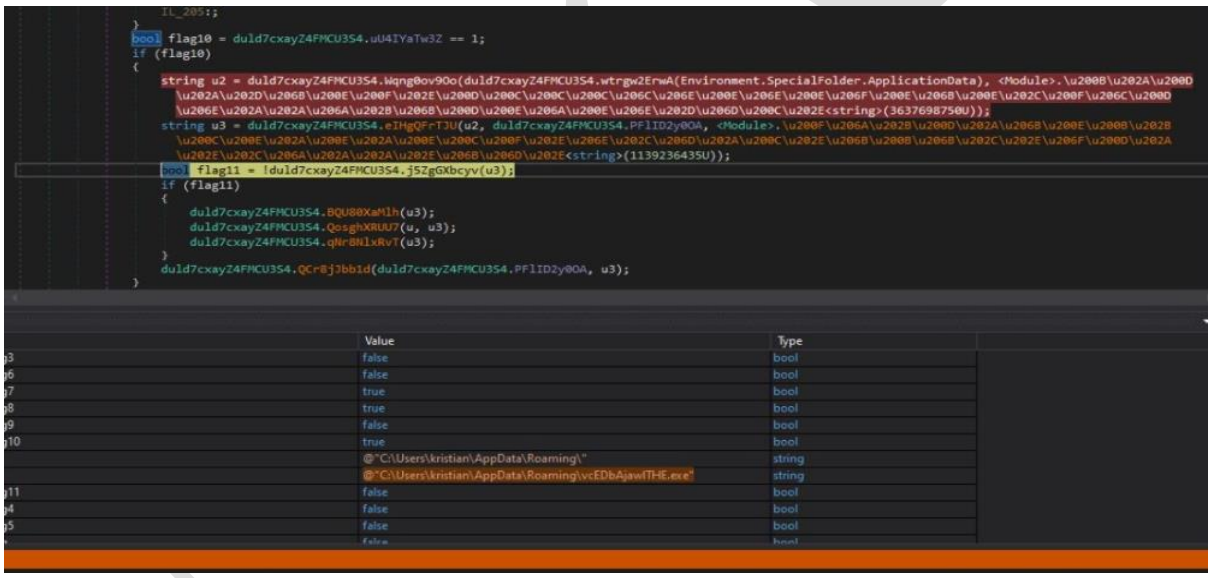


Figure 20: Creation of the file iAFV.exe

The file created in the Roaming folder is the same file as iAFV.exe with hash sha256 **f4eaa74eb268a58cff6f5d37607758bd49cc00af060da799857ae10cfd59efb2.**

Static Analysis of Remcos RAT

Figure 21: REMCOS RAT

From the extraction of the iAFV.exe file, the file with hash was found:
sha256 - **a175371e9373a87e16ced87d403fb50247705163ea786a5eba3209cb66ed4390**. When searching for character strings in this application, we see the string "!**This program cannot be run in DOS mode"** indicating that we are dealing with an executable file, therefore we rename it by adding the .exe suffix. At this moment, the file acquires the icon as in the figure above and it becomes clear that we are dealing with the malicious **REMCOS RAT** file.

During the **Reverse Engineering** phase of this file, it was revealed that we are dealing with a keylogger that records every keystroke, audio, video, and all other actions performed on the infected system. This process is carried out by the function SendInput from the **Windows library – winuser.h**.

```
void __fastcall FUN_004198e1(undefined4 param_1,char param_2,char par

{
  tagINPUT local_1c;

  local_1c.type = 1;
  if (param_2 == '\x01') {
    local_1c.field1_0x4.mi.dy = 0;
    local_1c.field1_0x4.ki.wVk = 0x10;
    SendInput(1,&local_1c,0x1c);
  }
  if (param_3 == '\x01') {
    local_1c.field1_0x4.mi.dy = 0;
    local_1c.field1_0x4.ki.wVk = 0x11;
    SendInput(1,&local_1c,0x1c);
  }
  if (param_4 == '\x01') {
    local_1c.field1_0x4.mi.dy = 0;
    local_1c.field1_0x4.ki.wVk = 0x12;
    SendInput(1,&local_1c,0x1c);
  }
  local_1c.field1_0x4.mi.dy = 0;
  local_1c.field1_0x4.ki.wVk = (WORD)param_1;
  SendInput(1,&local_1c,0x1c);
  local_1c.field1_0x4.mi.dy = 2;
  local_1c.field1_0x4.ki.wVk = (WORD)param_1;
  SendInput(1,&local_1c,0x1c);
  if (param_2 == '\x01') {
    local_1c.field1_0x4.ki.wVk = 0x10;
    local_1c.field1_0x4.mi.dy = 2;
    SendInput(1,&local_1c,0x1c);
```

Figure 22: Function SendInput (winuser.h), Keylogger

Apart from functioning as a **keylogger**, this malicious file serves as **Command and Control** (C2), this is evident in the use of the **WS2_32.DLL** library.

```
  char *pcVar8;
  undefined in_stack_fffffc0;
  undefined local_20 [28];

  uVar1 = connect(*(SOCKET *)(param_1 + 4),*(sockaddr **)(DAT_00472adc + 0x18),
                  *(int *)(DAT_00472adc + 0x10));
  if (uVar1 == 0) {
    pvVar4 = (HANDLE)0x0;
    if (*(char *)(param_1 + 1) == '\0') {
LAB_00404a17:
      return CONCAT31((int3)((uint)pvVar4 >> 8),1);
    }
    if (*(char *)(param_1 + 0x31) != '\0') {
      pvVar5 = (void *)(param_1 + 0x34);
      uVar6 = 0xf;
      FUN_0040531e(&stack0xffffffc0,"TLS Handshake...       | ",pvVar5);
      uVar7 = SUB41(pvVar5,0);
      FUN_00402093(&stack0xfffffa8,"i");
      FUN_0041b43d(in_stack_fffffa8,in_stack_fffffac,in_stack_fffffb0,in_stac
                   uVar7,in_stack_fffffc0);
    }
    ppiVar2 = FUN_00420bae();
    *(int ***)(param_1 + 0x4c) = ppiVar2;
    if (ppiVar2 != (int **)0x0) {
      iVar3 = FUN_00420ddd((int)ppiVar2,*(undefined4 *)(param_1 + 4));
```

Figure 23: Functions called towards the library

The file also has other functions such as downloading files located on the compromised computer, executing commands in cmd.

```
pvVar8 = FUN_0041bc0c(auStack_208,auStack_1f0);
uVar24 = CONCAT44(pvVar8,0x407f8a);
FUN_004052fd(&stack0xfffffdb0,"Downloading file: ",pvVar8);
FUN_00402093(&stack0xfffffd98,"i");
FUN_0041b43d(uVar16,uVar17,uVar20,uVar22,(char)uVar24,(char)((ulonglong)uVar24
             in_stack_fffffdb0);
FUN_00401fd8(auStack_208);
FUN_00401f09(auStack_1f0);
```

Figure 24: Downloading files

Figure 25: Executing commands in cmd

Dynamic Analysis of Remcos RAT

During the static analysis, no strings or any further useful information were found, therefore, we need to execute the file in the Debugger. During the execution in the debugger, it was revealed that:

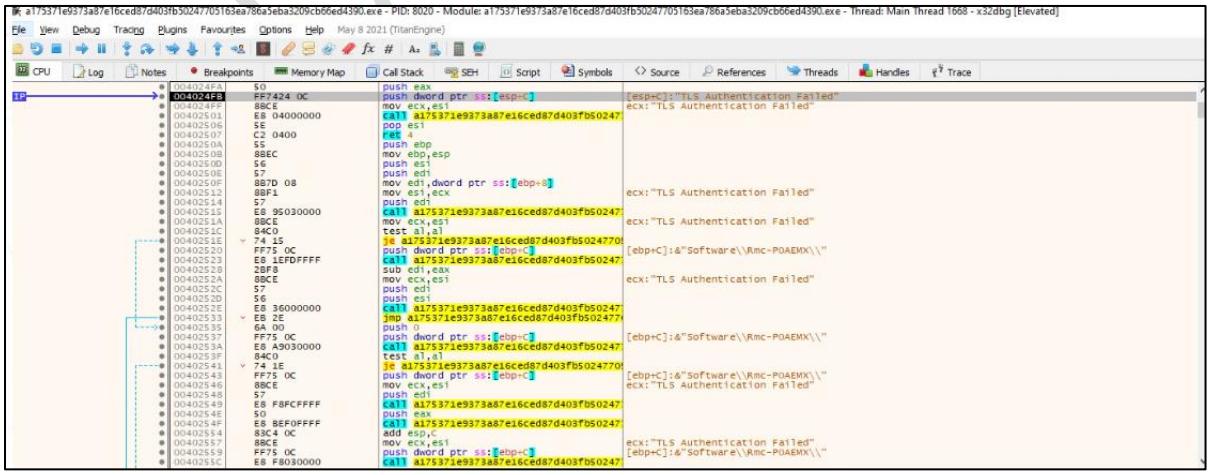


Figure 26: Analysis in debugger

Rmc-P0AEMX generates a **Mutex** (ensures that multiple instances are not being executed at the same time). It must generate a file where the information of pressed keys will be stored.

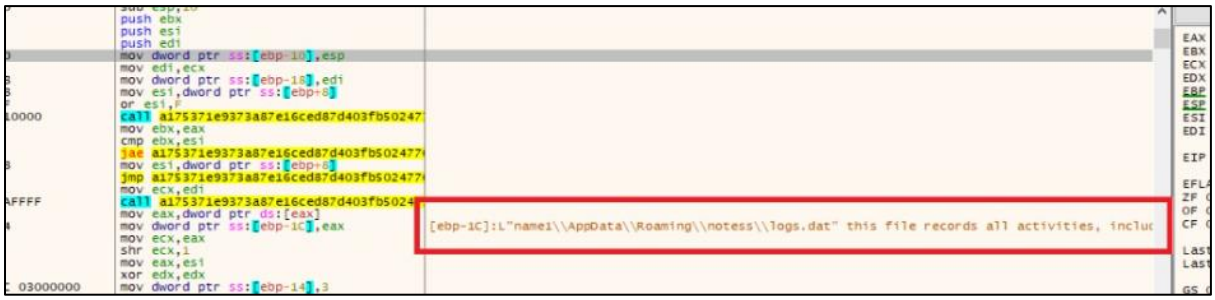


Figure 27: logs.dat

During the code execution:

A path **notess** is created.
In the path **C:\Users\Username1\AppData\Roaming\notess** a file **logs.dat** is created which allows the storage of all the user's activity.
During the execution, **a URL** was also revealed which serves as command and control (C2) which is
sembe[.]duckdns[.]org:14645 and belongs to IP: 194[.]187[.]251[.]115.
The IP belongs to **M247 Europe SRL – Brussels, Belgium** (AS 9009) and is a Virtual Private Server (VPS).
According to many national cybersecurity companies, this IP is considered high risk and a potential cyber attacker.

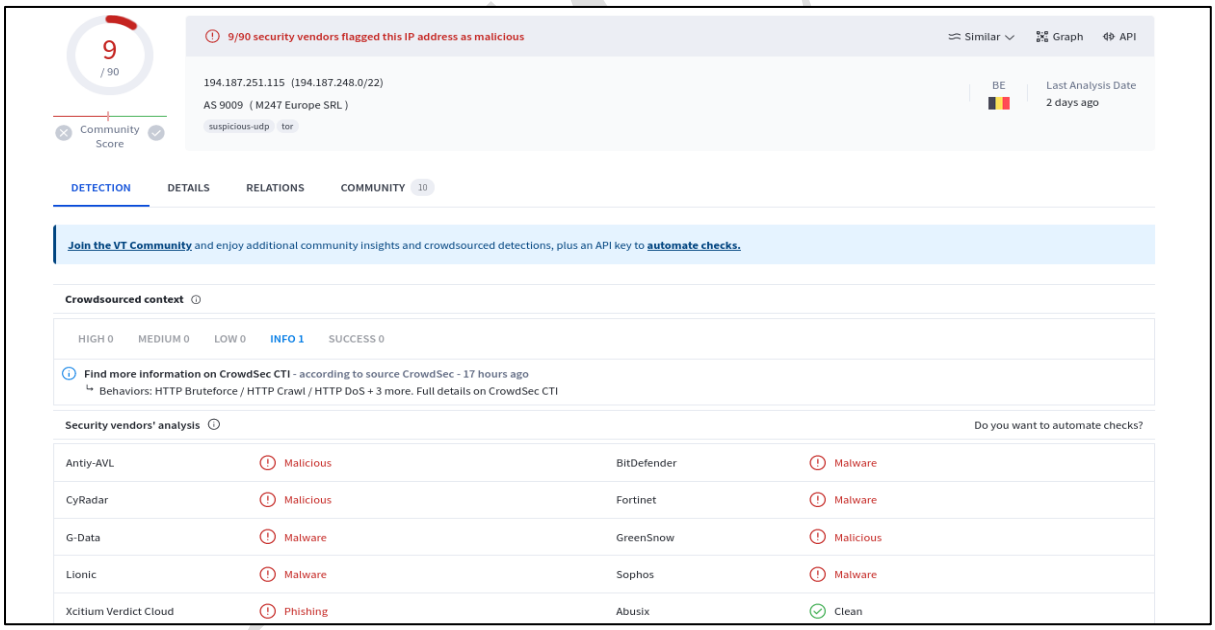
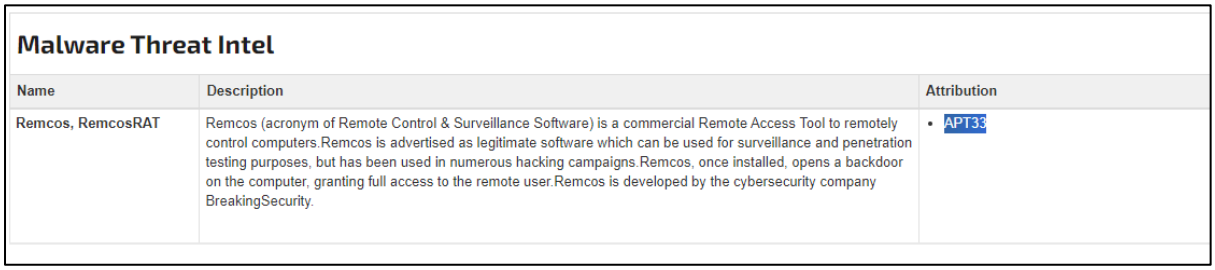


Figure 28: The criticality of IP 194[,]187[,]251[.]115 according to VirusTotal



Malware Threat Intel

Name	Description	Attribution
Remcos, RemcosRAT	Remcos (acronym of Remote Control & Surveillance Software) is a commercial Remote Access Tool to remotely control computers.Remcos is advertised as legitimate software which can be used for surveillance and penetration testing purposes, but has been used in numerous hacking campaigns.Remcos, once installed, opens a backdoor on the computer, granting full access to the remote user.Remcos is developed by the cybersecurity company BreakingSecurity.	• APT33

Figure 29: Categorization from Threat Intel Platform

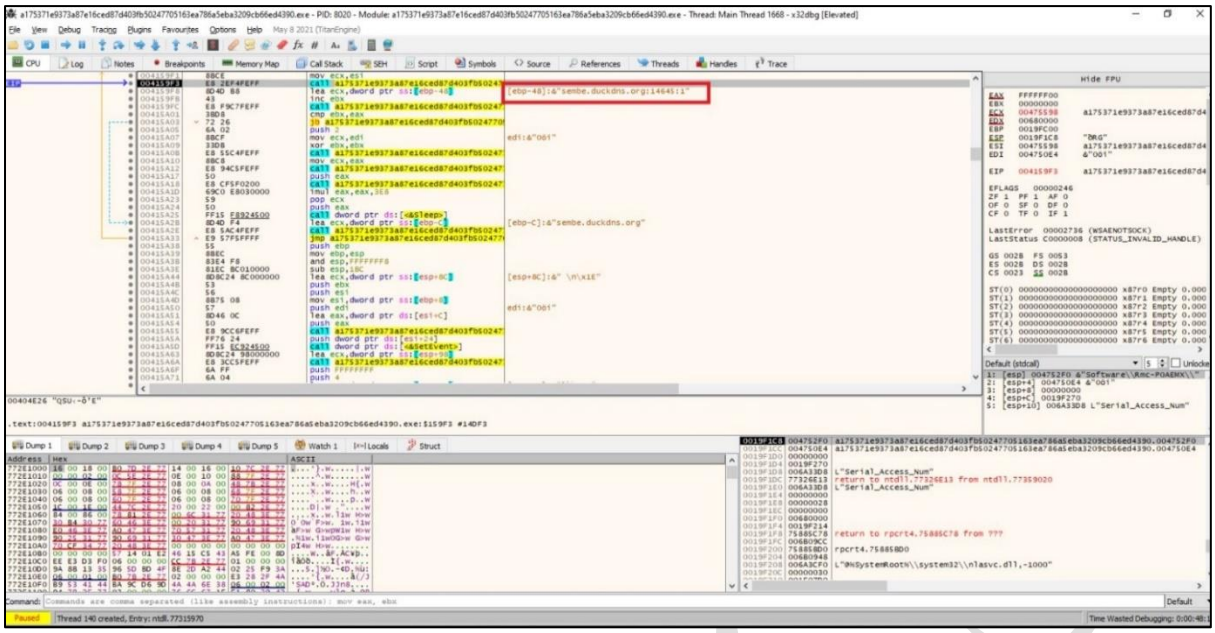


Figure 30: URL of Command and Control (C2)

This is also noticeable in the background of processes as below:

Figure 31: Command and Control

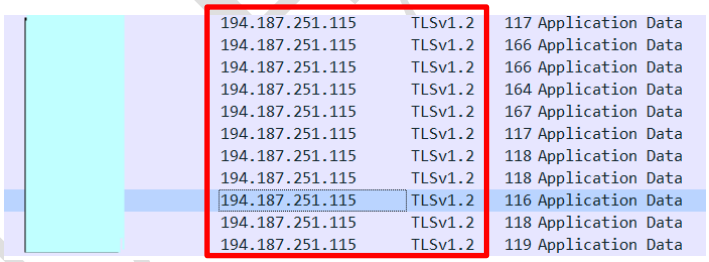


Figure 32: Data Exfiltration

During the execution, the **URL** was also revealed:
hxxp[://]geoplugin[.]net/json[.]gp which if opened gives us information about the IP from where the request is made, the location, the currency exchange rate of this country **in json format.**

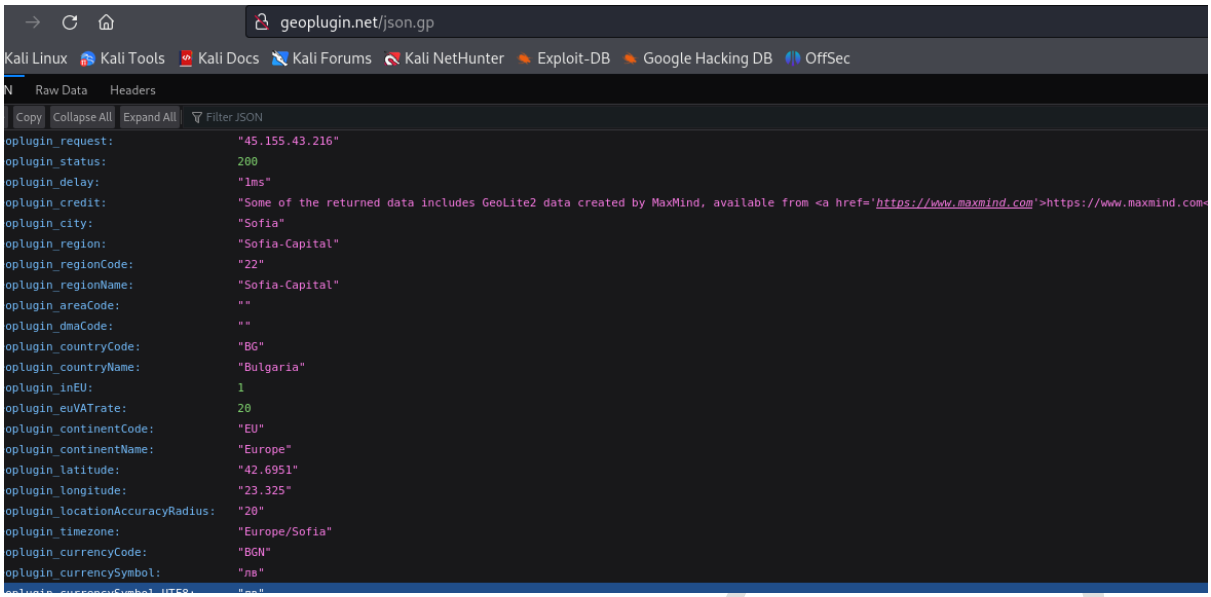


Figure 33: URL in Json

Indicators of Compromise

HASHES:

- f4eaa74eb268a58cff6f5d37607758bd49cc00af060da799857ae10cfd59efb2 - iAFV.exe
- 23f10d177ec53b6c4589adc03621906d7c65b9ae8ec4ff402ebd287014dbbcae - Tyrone.dll
- 71dab87ac5b7b80468ef8ccb16b74b39cc862b7fb9a6e430e4cd7e375dbe6c27 - Smt.dll
- a175371e9373a87e16ced87d403fb50247705163ea786a5eba3209cb66ed4390 - REMCOS RAT.exe

Domain:

- sembe[.]duckdns[.]org

IP:

- 194[.]187[.]251[.]115

MITRE Techniques

No.	Tactics	Techniques
1	Initial Access (TA0001)	T1566: Phishing
		T1566.001: Spear phishing Attachment
2	Execution (TA0002)	T1204: User Execution
		T1059.001: PowerShell
		T1059.005: Visual Basic
3	Persistence (TA0003)	T1547.001: Registry Run Keys/ Startup Folder

4	Defense Evasion (TA0005)	T1211: Exploitation for Defense Evasion
		T564.003: Hidden Window
		T1055: Process Injection
		T1027: Obfuscated Files or Information
5	Discovery (TA0007)	T1057: Process Discovery
		T1082: System Information Discovery
		T1614: System Location Discovery
		T1217: Browser Information Discovery
6	Collection (TA0009)	T1115: Clipboard Data
		T1056.001: Keylogging
		T1113: Screen Capture
		T1005: Data from Local System
7	Exfiltration (TA0010)	T1041 – Exfiltration Over Command-and-Control Channel
8	Command and Control (TA0011)	T1001.0012: Steganography
		T1071: Application Layer Protocol

Recommendations

- Immediate blocking of the Indicators of Compromise, mentioned above, on your defensive devices.
- Ongoing analysis of logs from SIEM (Security Information and Event Management).
- Training non-technical staff about "Phishing" attacks and how to avoid getting infected by them.
- Installation of network perimeter devices that perform deep traffic analysis relying not only on access list rules but also on its behavior (NextGen Firewalls).
- Segment identified systems into different VLANs, applying "Access control list for the entire network perimeter", webservers should be separated from their Database, Active Directory should be in a separate VLAN.
- Application and use of the LAPS technique for Microsoft systems for managing Local Administrators' passwords.
- Traffic filters should be applied in the case of remote access to hosts (employees/third parties/clients).
- Implement solutions that perform filtering, monitoring, and blocking of malicious traffic between Web applications and the internet using Web Application Firewall (WAF).
- Perform traffic analysis at the behavioral level for endpoint devices, applying EDR, XDR solutions. This brings the analysis of malicious files not only at the signature level but also at the behavioral level.
- Design a solution for managing user access "Identity Access Management" to control the identity and privileges of users in real-time according to the "zero-trust" principle.