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

AgentTesla Malware,
Technical Analysis

Version: 1.0
Date: 22/04/2024

TABLE OF CONTENTS

TABLE OF FIGURES

This report has been prepared to document and analyze cyber attack attempts against critical infrastructures within the Republic of Albania. The content of this report is based on information available up to the date the analysis was completed.

The distribution of this report aims to inform and raise awareness among stakeholders about the documented cyber incident. The report should not be treated as final until it is ultimately updated.

Some of these limitations include:

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

Phase Two:
Analysis Details: Due to resource limitations, some aspects of the malicious file may not have been thoroughly analyzed. Any additional unknown information could reflect changes in the report.

Phase Three:
Information Security: To protect sources and confidential information, some details may be mitigated or not included in the report. This decision was taken to maintain the integrity and security of the data used.

The National Authority for Electronic Certification and Cyber Security reserves the right to change, update, or modify any part of this report without prior notice.

The findings of the report are based on information available during the investigation and analysis period. There is no guarantee regarding possible changes or updates to the reported information over time. The report authors are not responsible for any misuse or consequences of decisions based on this report.

Executive Summary

The National Authority for Electronic Certification and Cyber Security conducted a detailed technical analysis of the **Agent Tesla Remote Access Trojan (RAT) v4** malicious file, which targeted a critical infrastructure within the Republic of Albania. This report summarizes findings from both static and dynamic analysis of the malicious file, highlighting key indicators of compromise, techniques used 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 by the monitoring team in the form of a phishing email targeting one of the critical infrastructures monitored by the Authority. The analysis confirmed that the files belong to the **AgentTesla RAT** family, a type of virus that allows malicious actors to spy on compromised systems and steal credentials. Detailed examinations were conducted on various components of the malicious file, including **kugR.exe, Tyrone.dll,** and other related files, revealing their properties and sophisticated methods used to evade detection by antivirus systems and detailed analysis.

Indicators of compromise were identified, including hash values for different files and network indicators.

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

Technical Information

Referring to monitoring team reports on a **phishing email** targeting one of the critical infrastructures in Albania, several suspected malicious files were downloaded for further analysis. Static and dynamic analysis of the files revealed that one of the files belongs to the **Trojan** family, specifically **Agent Tesla RAT v4**, with the main objective being credential theft and system surveillance (spyware). The analysis revealed that this virus obtains stored credentials of SMTP, various browsers (Mozilla, Chrome, etc.), Outlook, Discord, NordVPN, etc.
Through source code analysis, credentials were also found which are used to obtain data via SmtpClient. For more discreet communication, malicious actors use compromised emails.

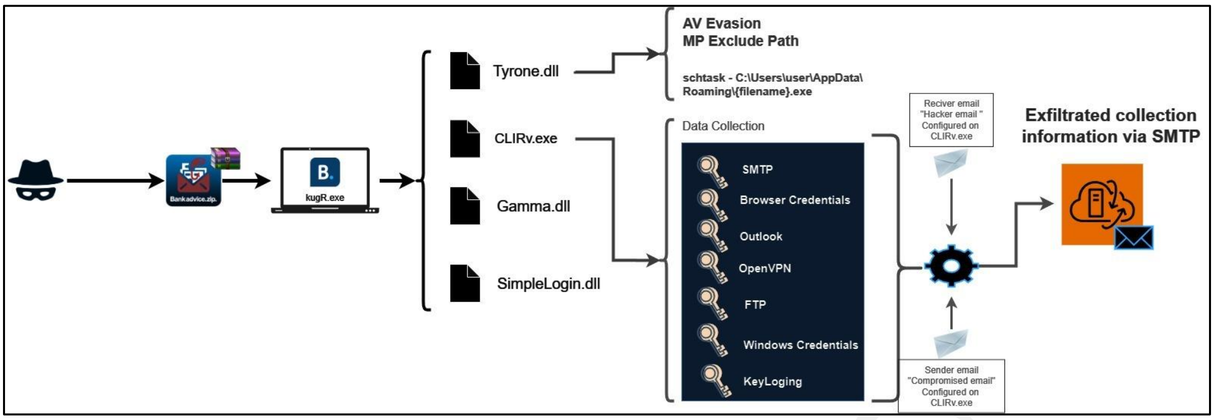

Figure 1: Distribution Scheme of the Malicious File

Analysis of kugR.exe File

The **kugR.exe** executable is a .NET library file written in the C# programming language.

SHA256:
EF171F71804FE96BF375379C691E1F93B3FE38A3535B24F8F19D104E5EECF7AA

Figure 2: kugR.exe

At first glance, the above figure appears to show a legitimate application involved in room booking (Booking). However, a static analysis of the code reveals that in addition to legitimate implementations, there is **obfuscated** and **packed code** (a technique used by software developers to make their code unreadable to others).

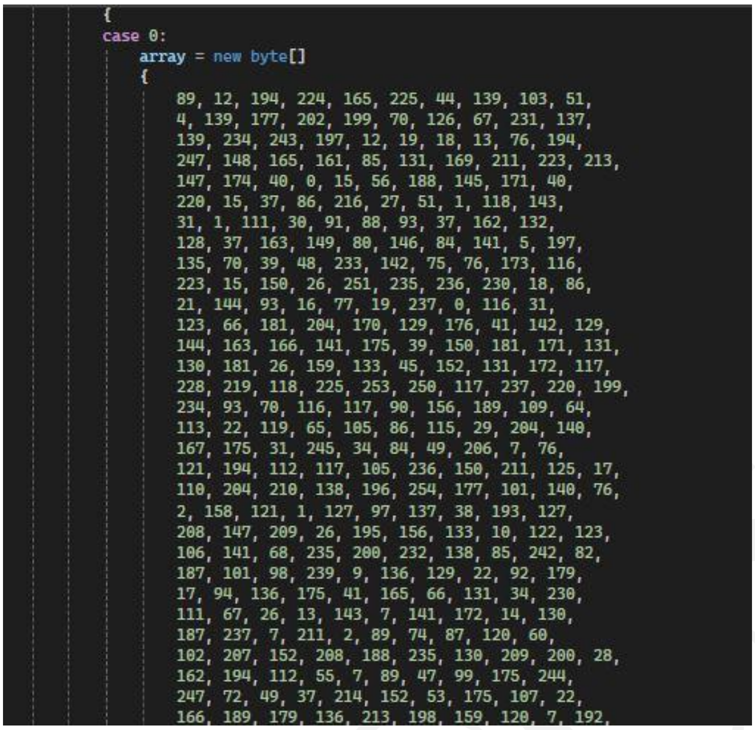

Figure 3: Vector of Bytes

Figure 4: Character Vector

In such cases, threat actors use complex algorithms that, during the execution of the main file, restore these code parts to executable formats (.exe), which are translated into hexadecimal format beginning with 4D 5A. In .NET, **Reflection** is used to invoke methods from **dll** formats or executable files. This technique is used because, during the execution of the main file, it might not always be detectable by antivirus software. Given such a high level of concealment, we perform a check for **code packers**.

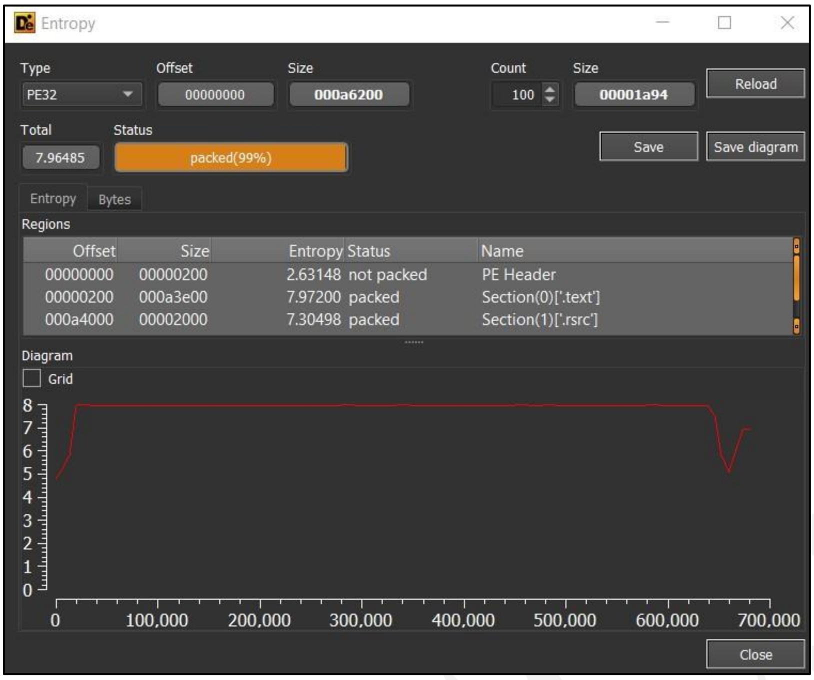

Figure 5: Entropy of the file kugR.exe

It is evident that we have parts of the file with entropy over the value of 5 (five), an indicator that we are dealing with packed code. Therefore, we proceed with the analysis part by attempting to revert the file to a more readable format.

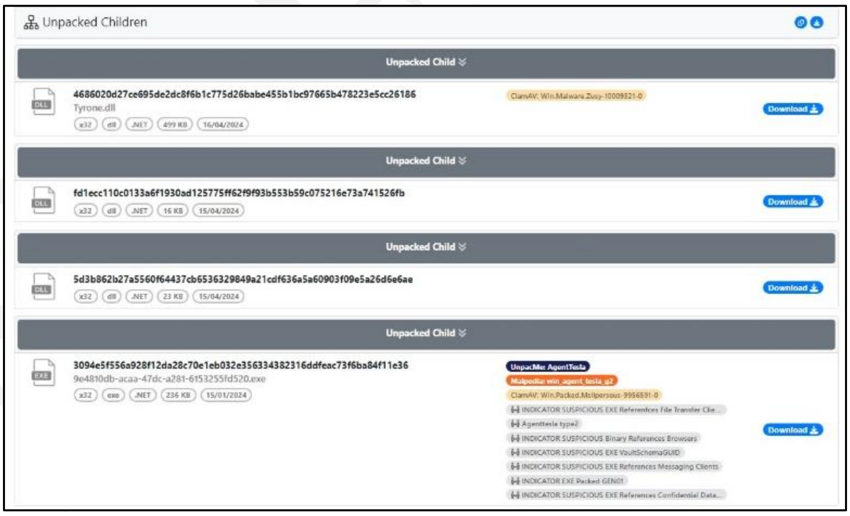

Figure 6: Unpacked Files

From the parent file emerge four other files written in C# using the .NET library. When we import these files, it is evident that they are different projects:

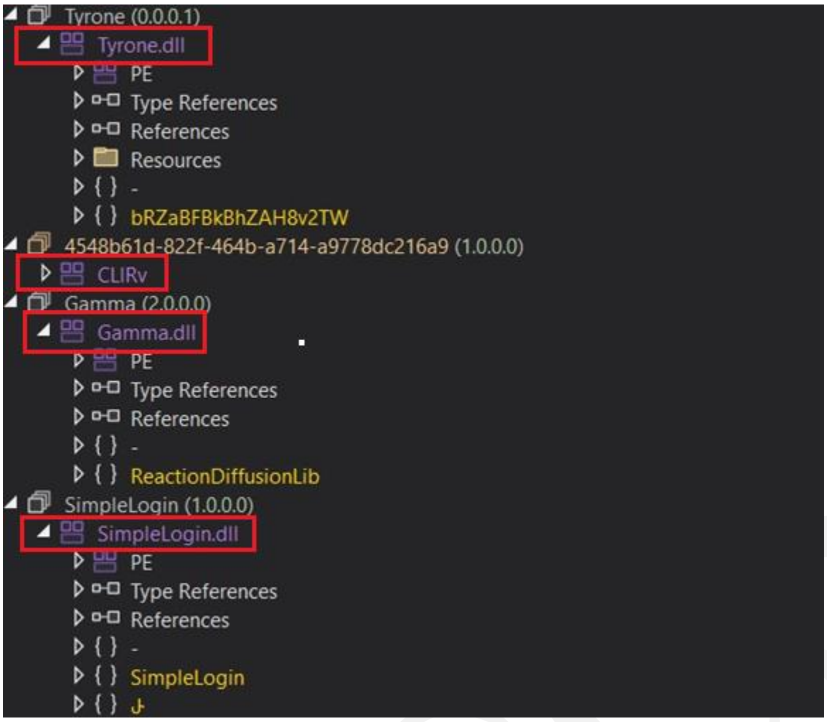

Figure 7: Unpacked Projects

The **Tyrone.dll** file is a .dll (dynamic-link-library) file written in C#. This file contains several implemented namespaces where a very high level of code concealment is evident. Some of the string values discovered are:

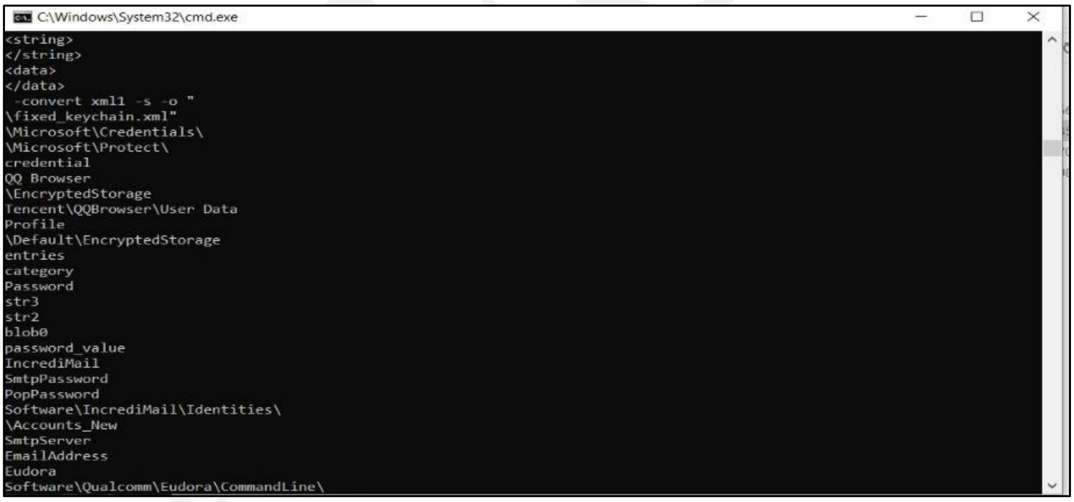

Figure 8: Discovered Strings

To understand the function of this file, we create a **Console** type project and generate an executable file **(.exe)** and load this **dll** by **invoking** its implemented methods. This is done because only during runtime can we obtain the values of each function variable. We choose the path of the dll file. We load the file and attempt to call one of the implemented methods. We set a breakpoint in the dll file and observe the displayed value. During execution, it is evident that in the path *C:\Users\User_1\AppData\Roaming*, an executable file is created, which is the same as the main file but randomly named. Also, during execution, the **schtasks.exe** file is executed. This file is used to create a task named **UPDATE**. This is done so that malicious actors can create persistence.

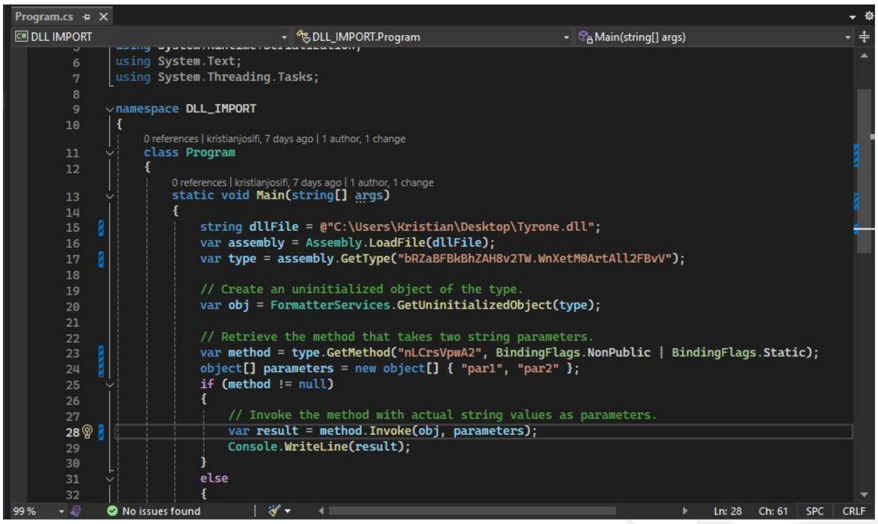

Figure 9: Invocation of the Tyrone.dll File

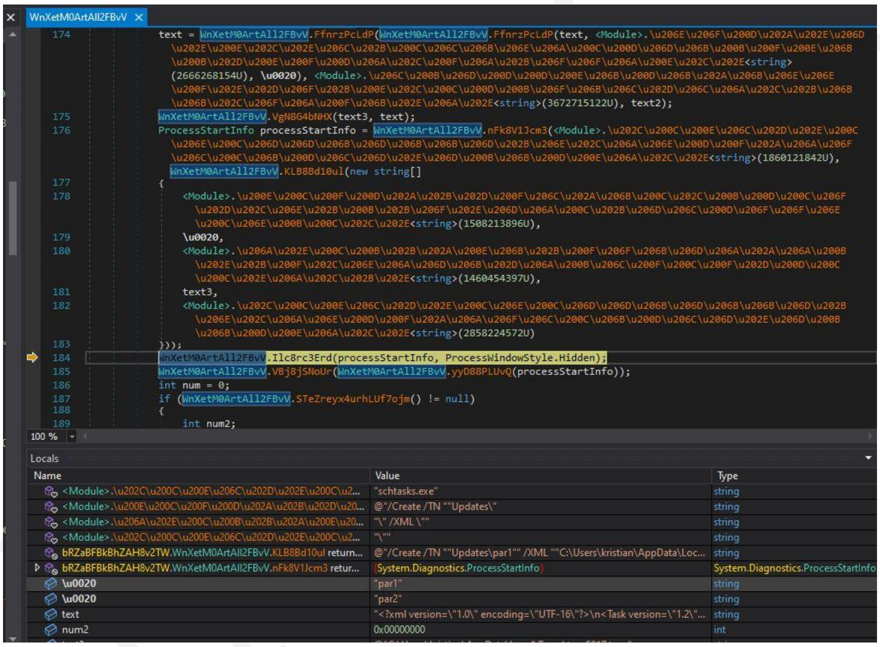

Figure 10: schtasks.exe

From the source code of the created file, it appears that an attempt is made to modify rights, as an encoded string with base64 during decoding translates into a file in **XML** format.

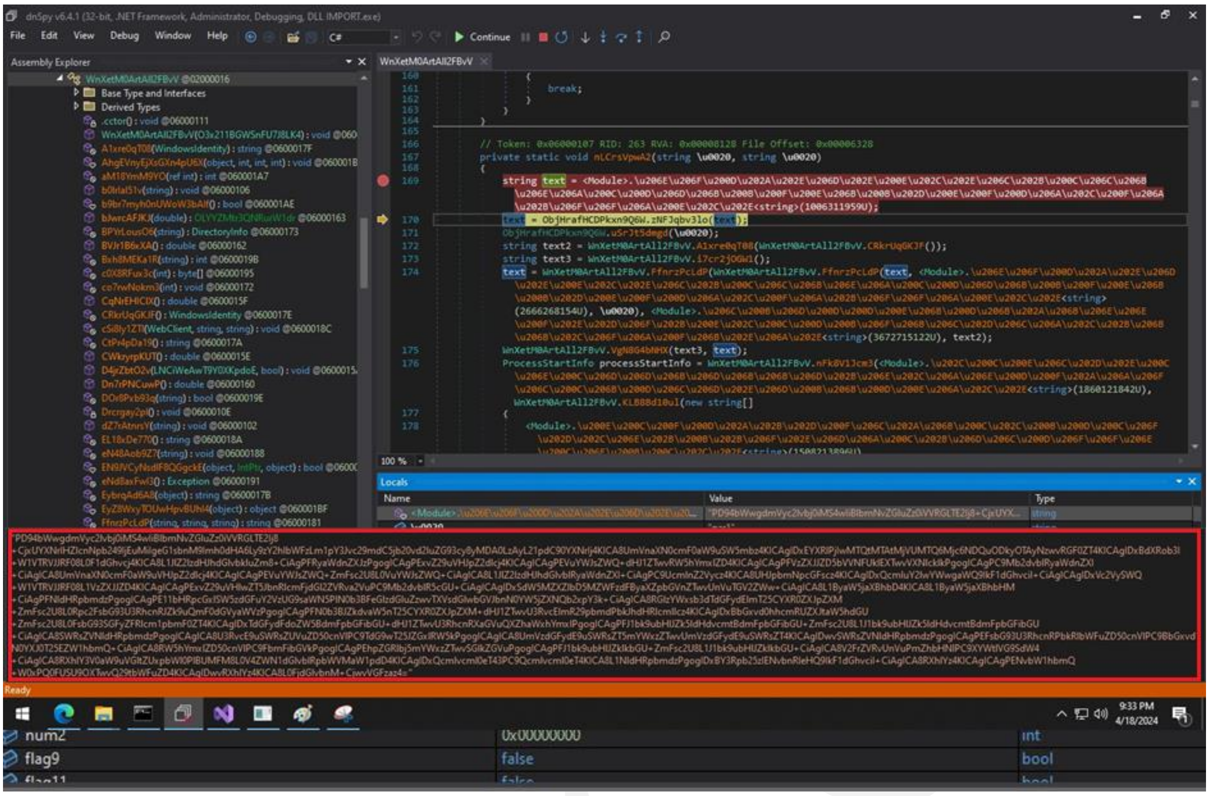

Figure 11: Base64 Encoding

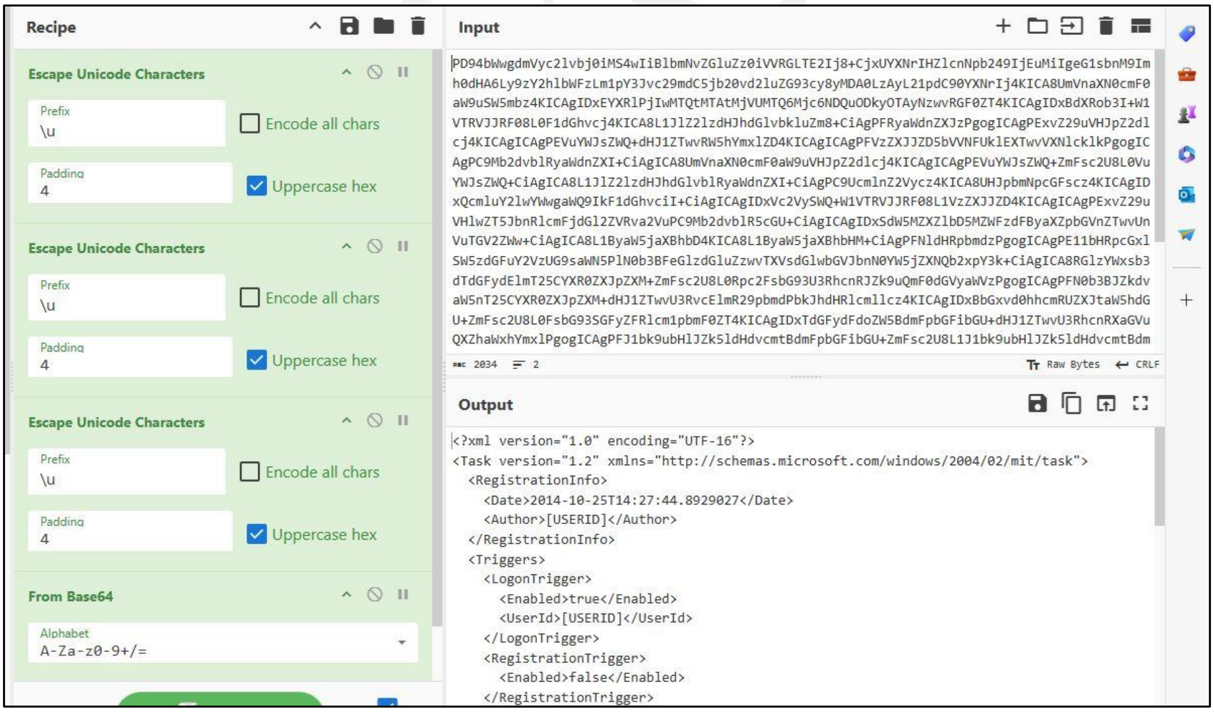

Figure 12: The rights in XML

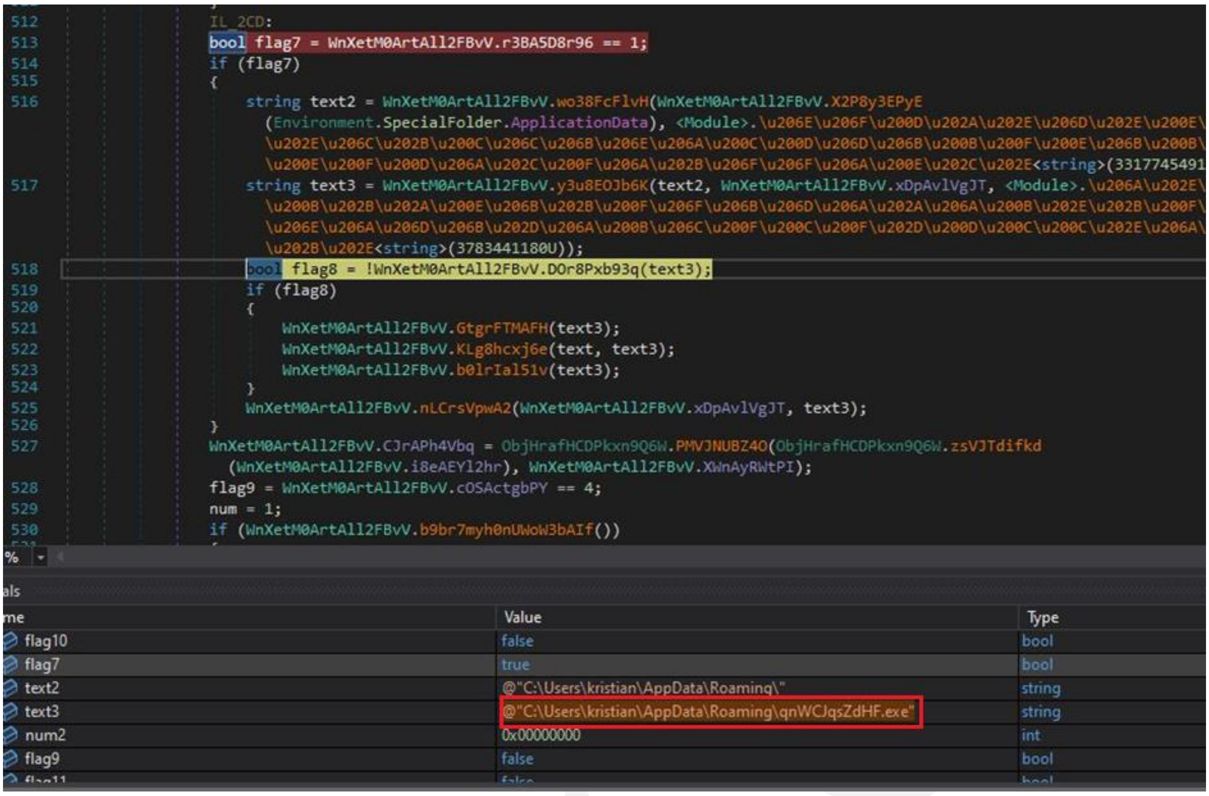

Figure 13: APPDATA

It is also evident a command executed in PowerShell, which serves to bypass the antivirus.

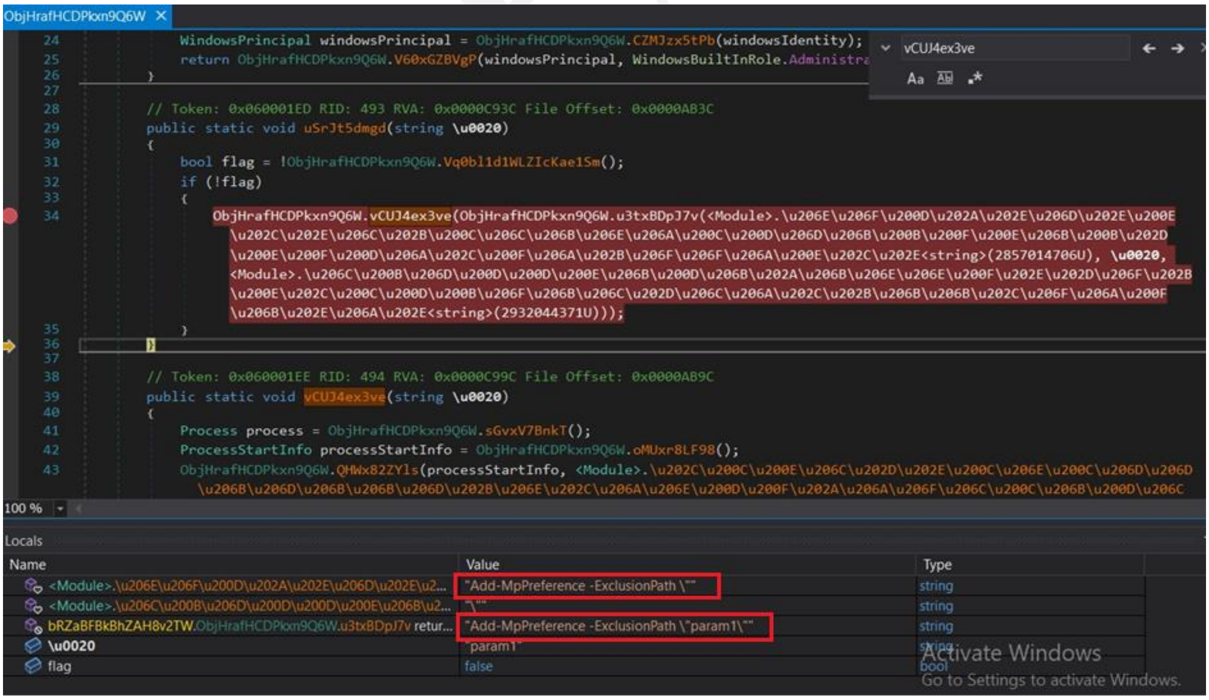

Figure 14: PowerShell Command for Antivirus Bypass

During the analysis of the **CLIRv.exe** file, it is evident that this file is a malicious **Agent Tesla** file.
SHA256:
1403E7C01BF67C9AC15E1D9068FAABDD21C05132CCE0C517C69425DB766FF140

From the static analysis of the code, this file is written in C# on .NET. The source code shows that we are dealing with a credential stealer. The malicious file has implemented some credentials which belong to an smtp server for sending emails.

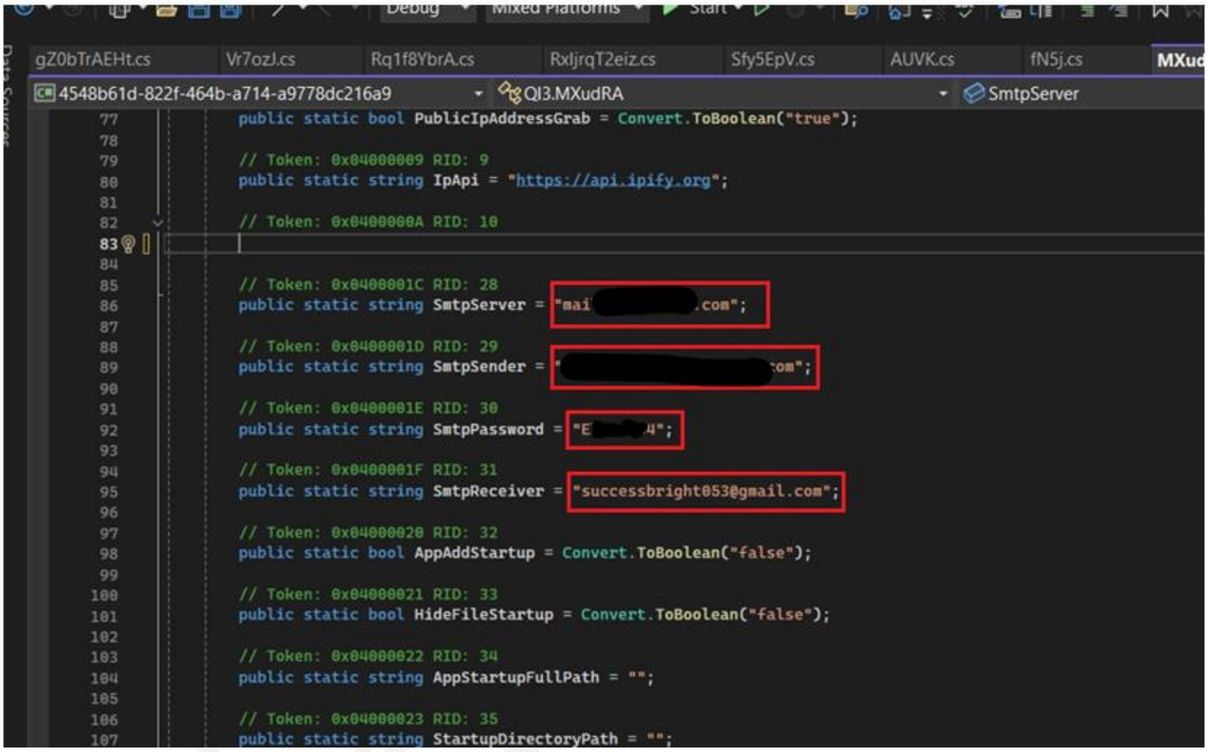

Figure 15: SMTP Credentials

Also, in this file, a class named **8WQvgbiWI1.Cs** is identified. This class serves to create instances of the class and fills them with data, namely the host, user, password, application. We have implemented an interface whose function is the **Grab()** function. This interface has several implementations within the application, where each implementation of the function serves to receive credentials from different applications.

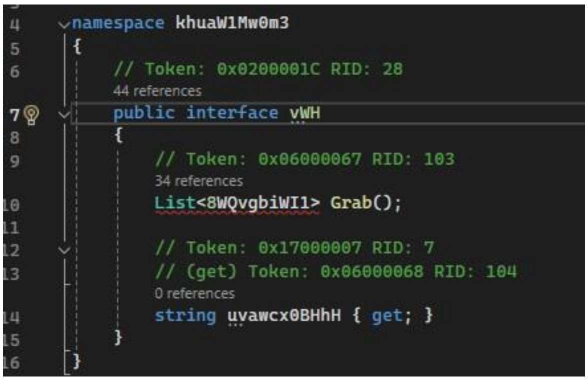

Figure 16: Grab() Method

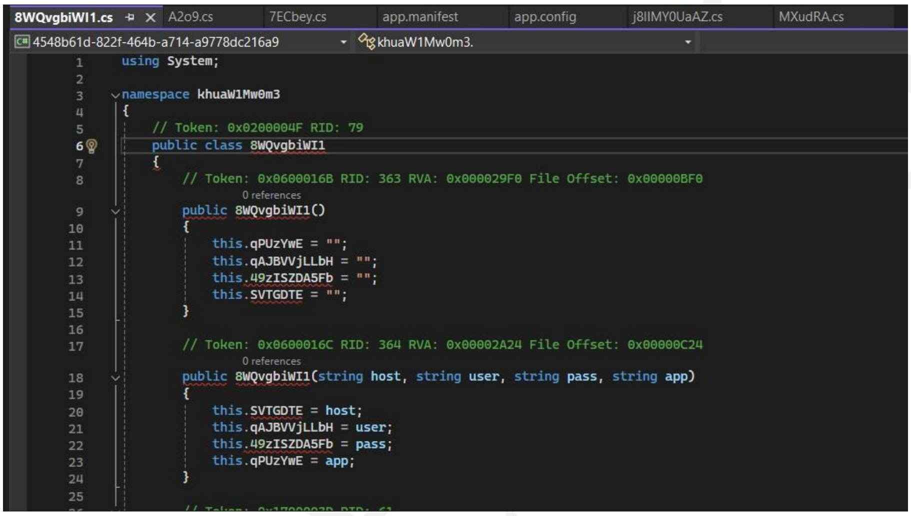

Figure 17: Data Retrieval Class

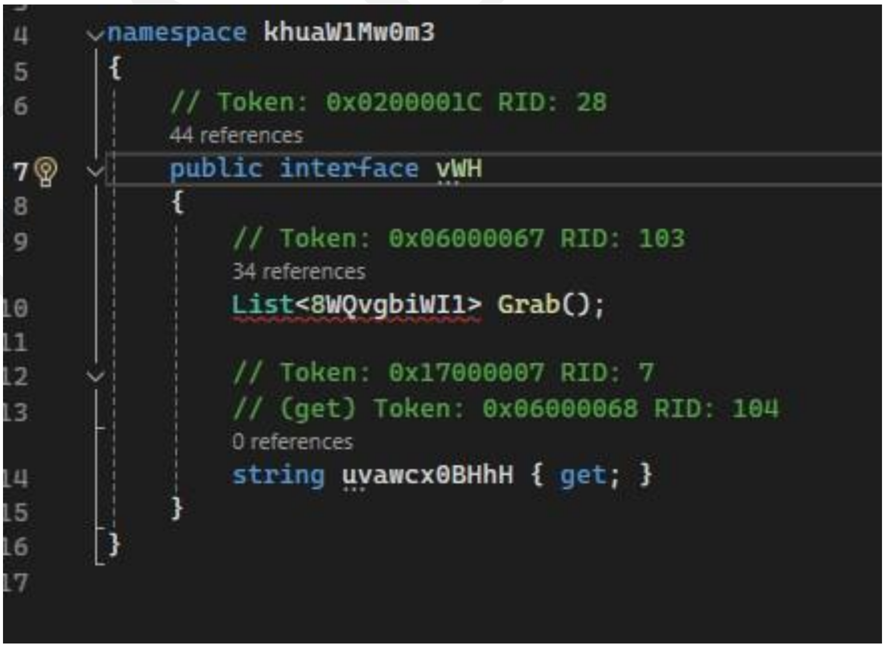

Figure 18: Grab() Function

When examining one of the implementations of the **Grab() function,** for example in the case of Outlook:

- The implementation saves a list of type **8WQvgbiWI1.**
- It creates an array of **Registry key** objects and begins the enumeration process to search for information on default registries where data about various applications is stored.
- It creates an instance of **8WQvgbiWI1** and fills the variables with data such as the username, password, and host.
- Each instance is added to the list and then returned to the function, returning this list. In the source code, the file also has a keylogger implemented that records keystrokes made by the user. Through several integer numbers, it checks the status of the keylogger to enable it.

Figure 19: Keylogger

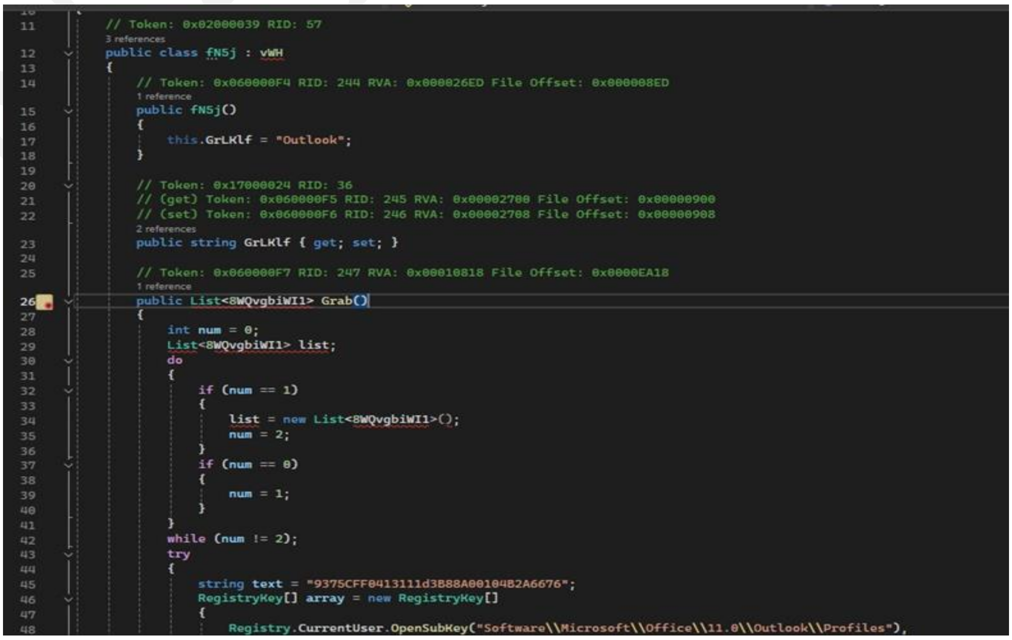

Figure 20: Data Retrieval from Outlook

The file also shows the implementation of a function that serves to send an email. Also seen in the source code is a string, **IpApi,** used to obtain the user's IP address. Information gathered from the infected computer is sent via email by the user at *electronics@xxxxx[.]com* (compromised email) to the user at *successbright053@gmail[.]com* (email of the malicious actor).

Dynamic Analysis of Agent Tesla

Dynamic analysis involves executing the malicious file to see how it behaves in a closed sandbox environment. During execution, it was observed that the email attempted to be sent to the user is successful. The following figure shows the data from the infected computer along with its IP address, sent via **smtpclient** to the malicious actor.

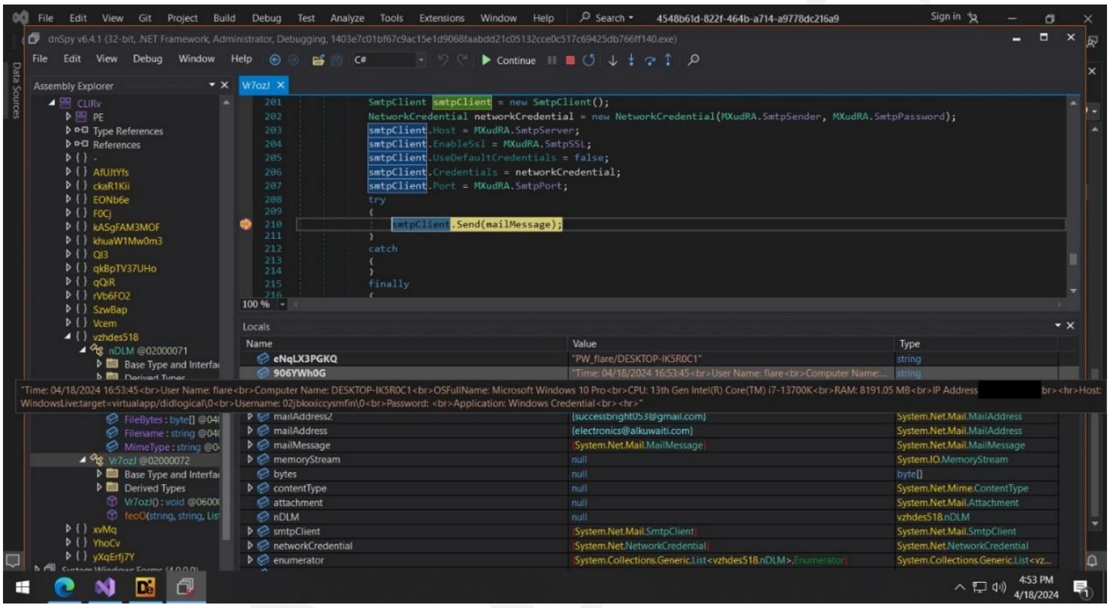

Figure 21: Email Sending

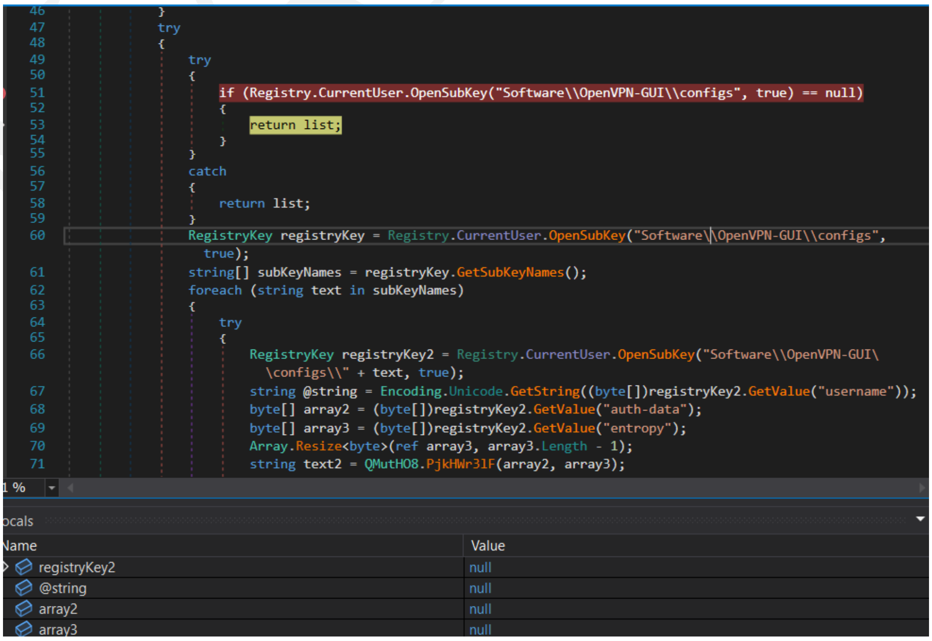

Figure 22: Execution of the Grab() Function

Indicators of Compromise

HASHES :
- *kugR.exe*

ef171f71804fe96bf375379c691e1f93b3fe38a3535b24f8f19d104e5eecf7aa
- *Tyrone.dll*

ead31b8d3cd588c72271e6671c16b7fd310099dbbccb61fe6f272cbc24b77ee8
- *Bank Advice.dll*

22a7e79314c5904ce3a5b0ef9f3ab7dfca2f487acbbb049414f1df7f8f95a3bf
- *CLIRv.exe*

1403E7C01BF67C9AC15E1D9068FAABDD21C05132CCE0C517C69425DB766FF140

Email:
successbright053@gmail[.]com

MITRE ATT&CK Techniques

Nr.	Tactics	Technique
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
6	Collection (TA0009)	T1560: Archive Collect Data

		T1217: Browser Information Discovery
		T1115: Clipboard Data
		T1005: Data from Local System
7	Exfiltration (TA0010)	T1048.003 – Exfiltration Over Unencrypted NON Command-and-Control Protocol
8	Command and Control (TA0011)	T1071.003: Mail Protocols

Recommendations

AKCESK recommends that infrastructures implement the following best practices to reduce the risk of attacks by these malicious actors:

- Immediate blocking of the Indicators of Compromise mentioned above on your defensive devices.
- Continuous analysis of logs coming from SIEM (Security Information and Event Management).
- Training non-technical staff about "Phishing" attacks and ways to avoid infection from 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).
- Segmentation of identified systems into different VLANs, applying "Access control list for the entire network perimeter", web services 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 the management of Local Administrators' passwords.
- Applying traffic filters in the case of remote access to hosts (employees/third parties/clients).
- Implementation of solutions that perform filtering, monitoring, and blocking of malicious traffic between Web applications and the internet, Web Application Firewall (WAF).
- Conducting traffic analysis at the "behavior" level for endpoint devices, implementing EDR, XDR solutions. This brings the analysis of malicious files not only at the signature level but also at the behavior level.
- Designing a solution for user access management "Identity Access Management" to control the identity and privileges of users in real-time according to the "zero-trust" principle.