Malware Analysis - AsyncRat

By Bar Magnezi

Published: 2025-04-23 · Archived: 2026-04-05 14:39:19 UTC

Sample:

```
17a59db354f270147d5da27aa7978a3c
```

BackgroundPermalink

AsyncRAT is a Remote Access Tool (RAT) designed to remotely monitor and control other computers through a secure encrypted connection. It provides functionality such as keylogger, remote desktop control, and many other functions. In addition, AsyncRAT can be delivered via various methods such as spear-phishing, malvertising, exploit kit and other techniques.

Static AnalysisPermalink

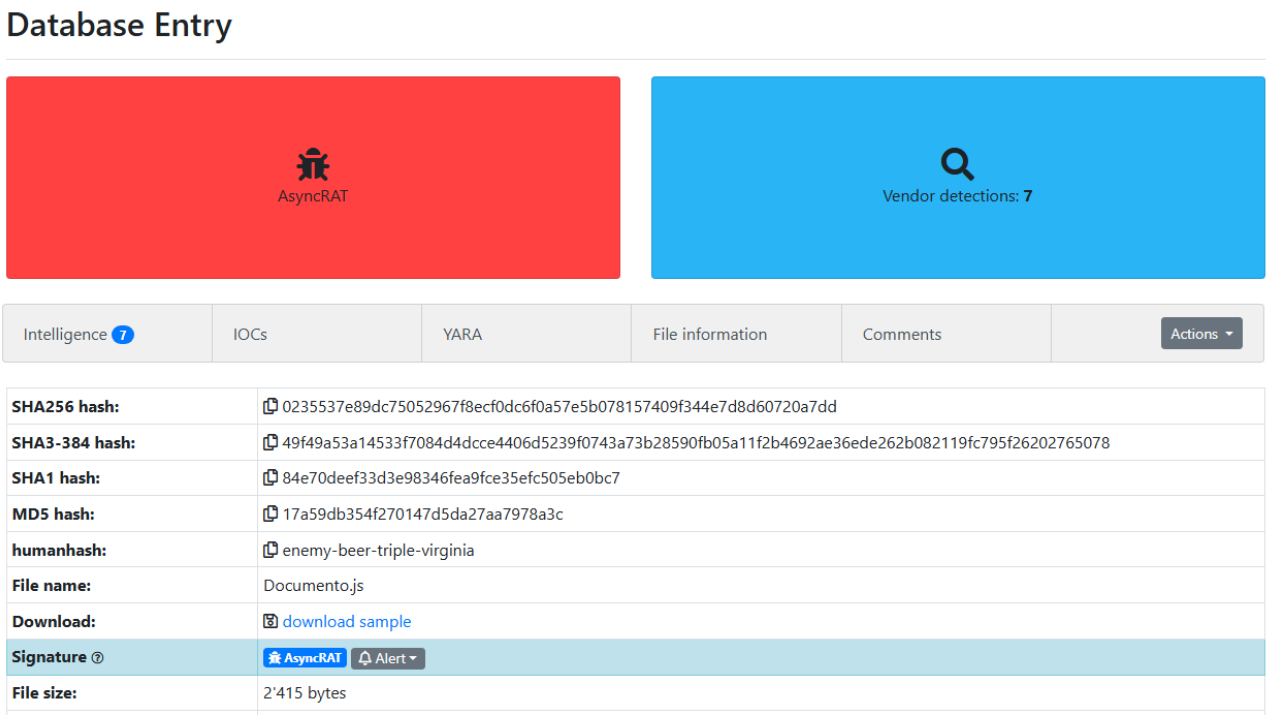

Figure 1: Malware Bazaar Entry

This sample was initially uploaded from the Netherlands and has since spread, with notable activity observed in Israel and the United States.

Figure 2: First Stage Code

The first stage uses a relatively simple split and join technique to construct a new string. To disable the original functionality and observe variable values, I modified the code to use console.log, as shown in Figure 3.

Figure 3: Disarmed JS Code

In addition, I used a neat trick: I opened the browser's developer tools and ran the code directly there to observe the output, as demonstrated in Figure 4.

Figure 4: Dev Tool Code

These results uncovered a Pastebin URL used to retrieve the second stage of the malware.

Second StagePermalink

The Second stage features heavily obfuscated code, containing around 7,500 lines, intended to obstruct analysis and evade detection. As shown in Figure 5, this snippet represents a small segment of the heavily obfuscated code.

Figure 5: Obfuscated Code

The majority of the obfuscation techniques involved injecting junk code to mask the malware's original functionality. After removing the junk code, what remained was a simple function that modifies a string and a long string. In the final step, the code is executed using WScript as shown in Figure 6.

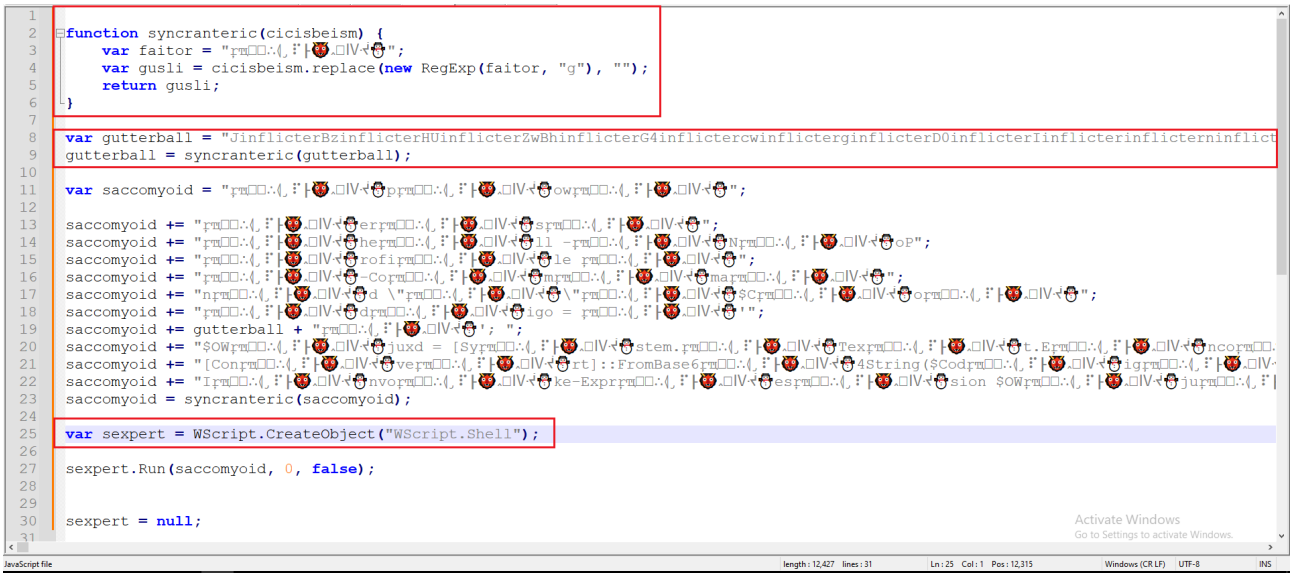

Figure 6: Clearing The Code

Using CyberChef, I was able to replicate the functionality of the previously observed string manipulation. This revealed the type of manipulation applied to the long string: it replaces a specific word with the letter 'A' and then decodes the result from Base64.

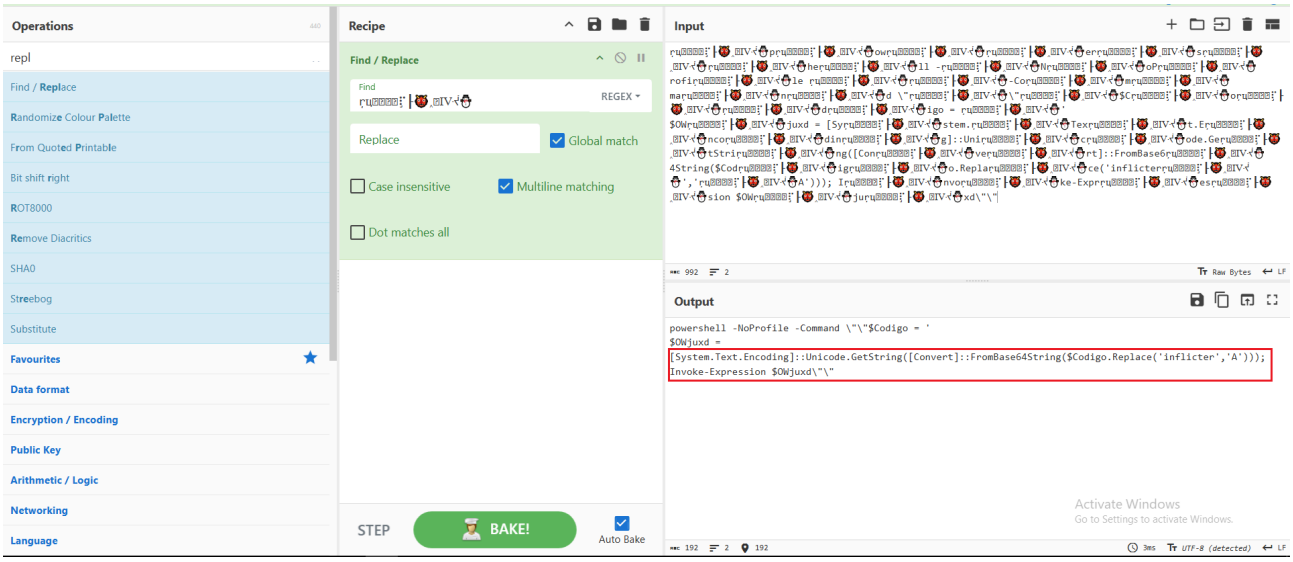

Figure 7: CyberChef Extraction

Next, I took the long string and applied the manipulation uncovered earlier, which revealed what appears to be a reversed URL, as shown in Figure 8.

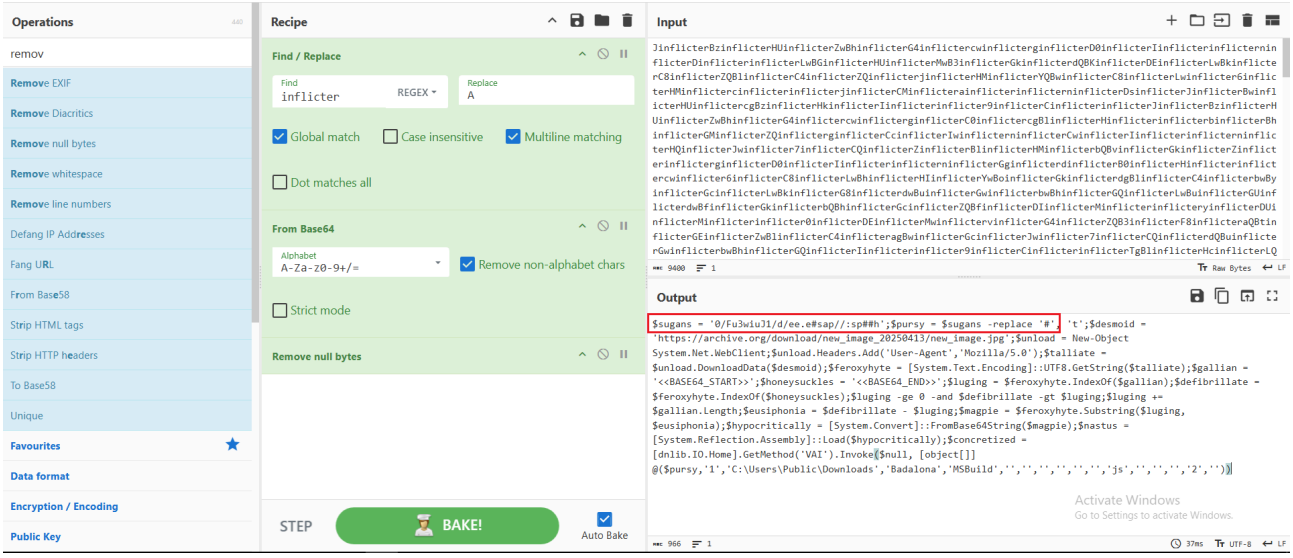

Figure 8: CyberChef to decode the string

Downloading the content hosted at that URL revealed a long, reversed Base64-encoded string. After decoding it, a new executable file was uncovered, as shown in Figure 9.

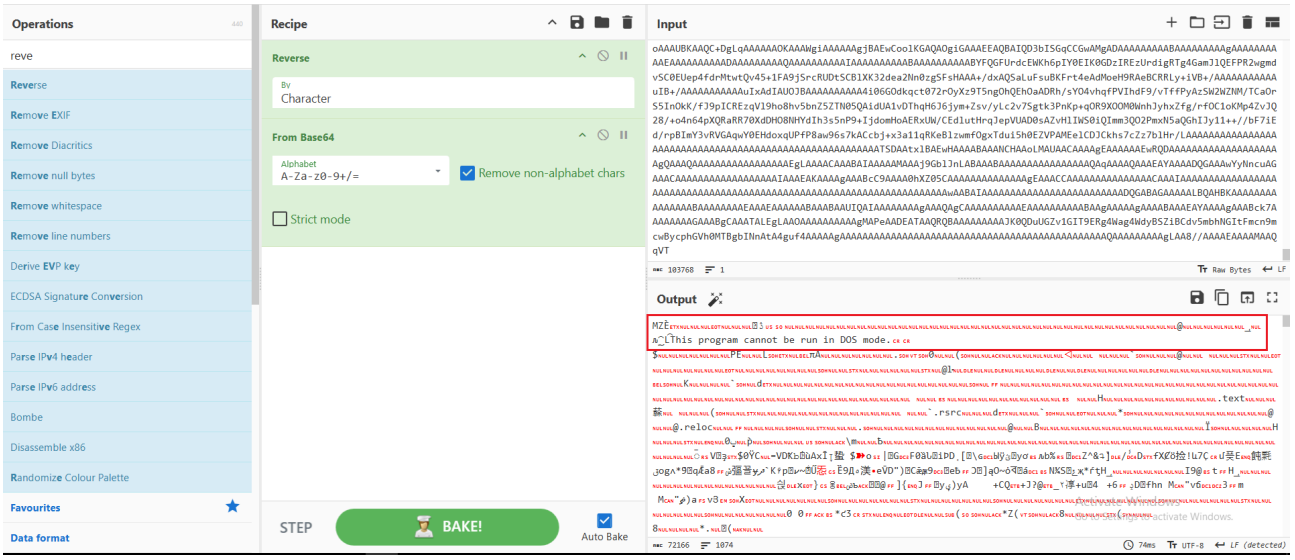

Figure 9: Outputs New EXE

Third Stage[Permalink]

The third stage of the malware is written in .NET and is most likely the unpacked version of the final payload.

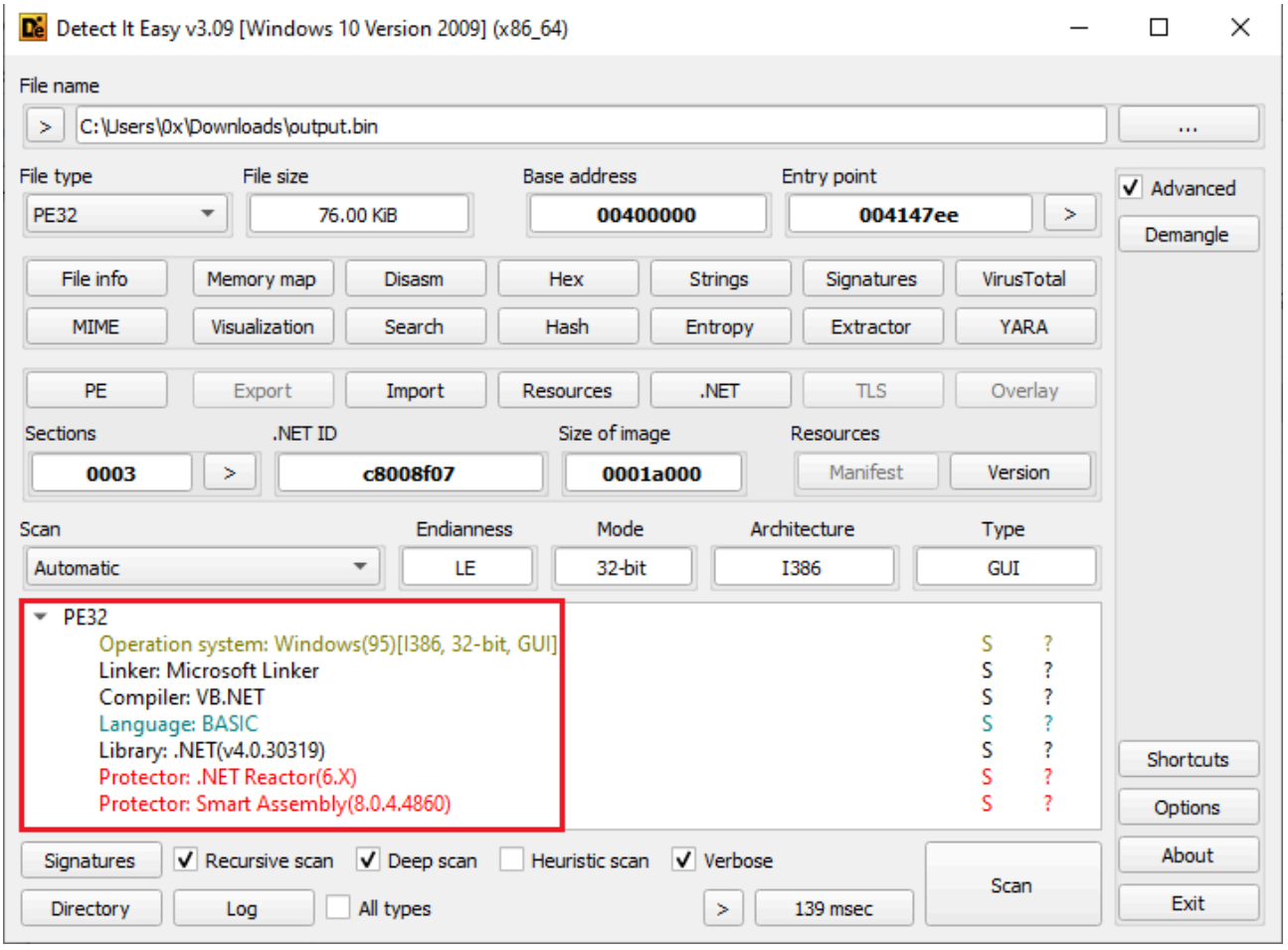

Figure 10: Using Detect it Easy

Figure 11 highlights the capabilities of the RAT, including C2 communication, code execution, debugging features, and more.

MBC Objective	MBC Behavior
ANTI-BEHAVIORAL ANALYSIS	Debugger Detection::CheckRemoteDebuggerPresent [B0001.002] Debugger Detection::WudfIsAnyDebuggerPresent [B0001.031] Sandbox Detection [B0007] Virtual Machine Detection [B0009]
ANTI-STATIC ANALYSIS	Executable Code Obfuscation [B0032]
COMMAND AND CONTROL	C2 Communication::Receive Data [B0030.002]
COMMUNICATION	DNS Communication::Resolve [C0011.001] HTTP Communication::Get Response [C0002.017] Socket Communication::Create TCP Socket [C0001.011] Socket Communication::Create UDP Socket [C0001.010]
CRYPTOGRAPHY	Cryptographic Hash::MD5 [C0029.001] Cryptographic Hash::SHA256 [C0029.003] Encrypt Data::AES [C0027.001] Generate Pseudo-random Sequence::Use API [C0021.003]
DATA	Decode Data::Base64 [C0053.001] Encode Data::Base64 [C0026.001]
DEFENSE EVASION	Obfuscated Files or Information::Encoding-Standard Algorithm [E1027.m02] Obfuscated Files or Information::Encryption-Standard Algorithm [E1027.m05]
DISCOVERY	Application Window Discovery [E1010] File and Directory Discovery [E1083] System Information Discovery [E1082]
FILE SYSTEM	Create Directory [C0046]
OPERATING SYSTEM	Registry::Query Registry Value [C0036.006] Registry::Set Registry Key [C0036.001]
PROCESS	Create Mutex [C0042] Create Thread [C0038] Suspend Thread [C0055] Terminate Process [C0018]

Figure 11: Capabilities Of The RAT

Given that the malware was written in .NET, I used dnSpy to decompile and analyze the code, which allowed me to extract the full configuration, as presented in Figure 12.

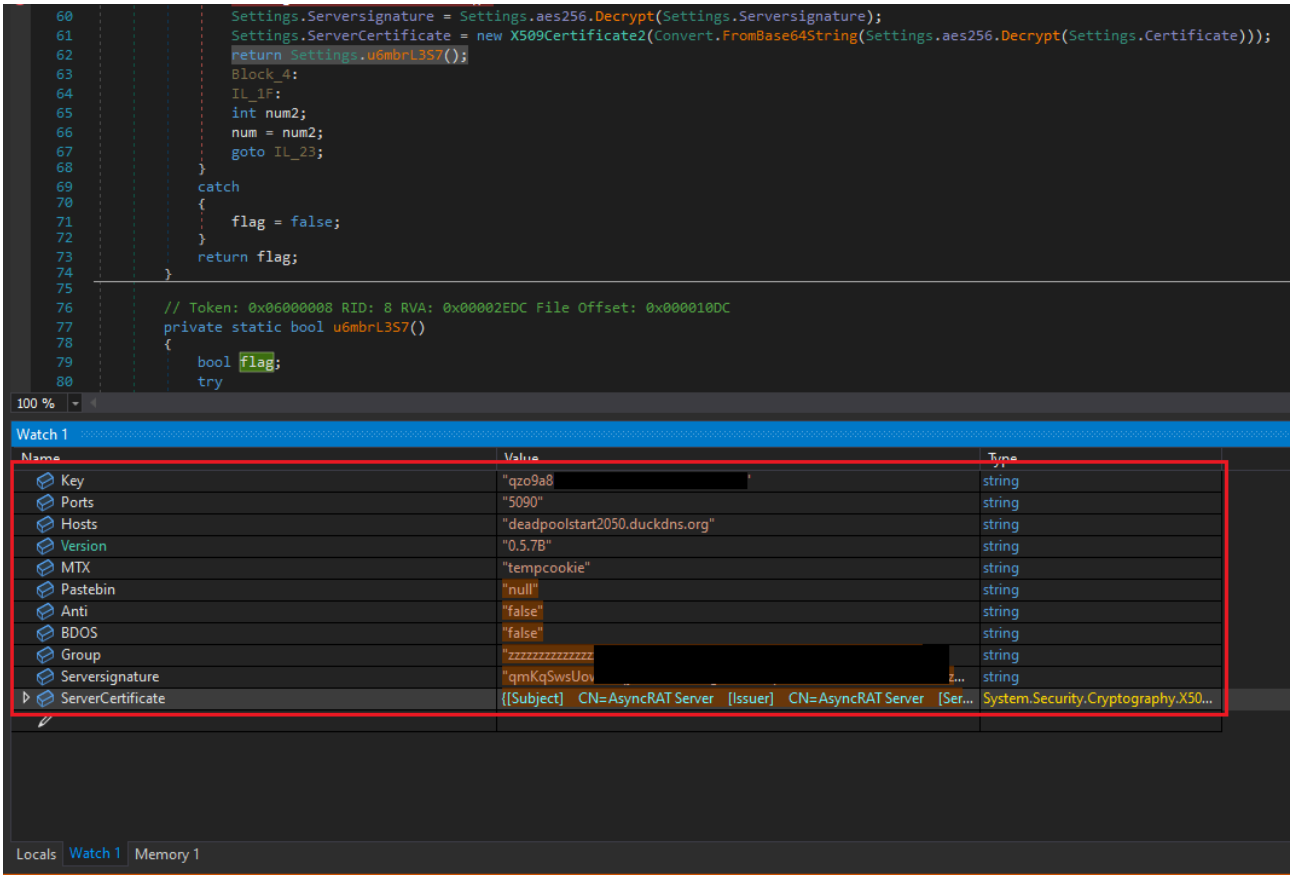

Figure 12: Settings Configuration Of AsyncRAT

As expected from a RAT, it also collects various environment details such as the hostname, user ID, and more.

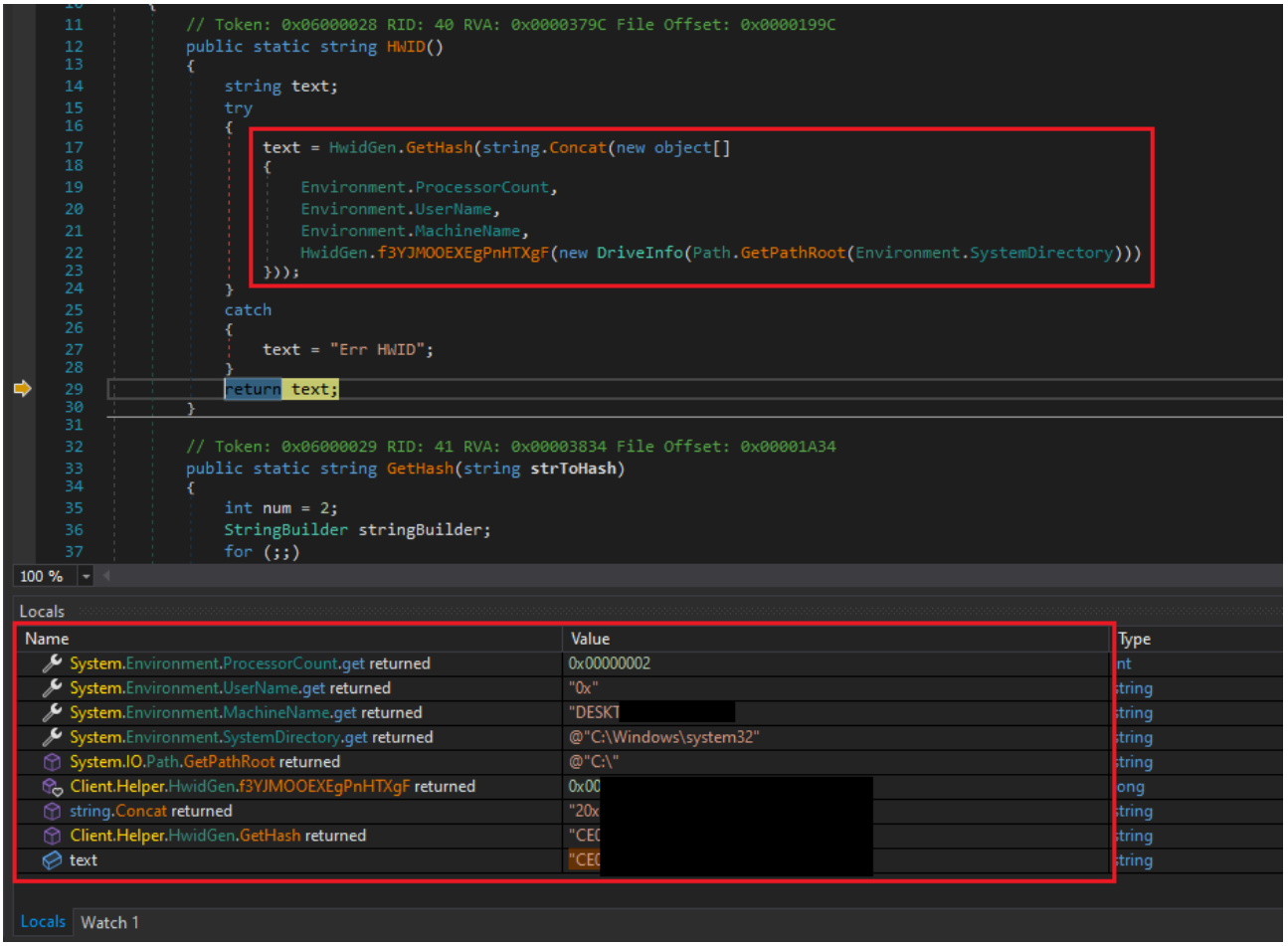

Figure 13: System Information

Dynamic Analysis[Permalink]

Once executed, the RAT attempts to establish a connection on port 5090 at regular intervals, as observed in TCPView and shown in Figure 14.

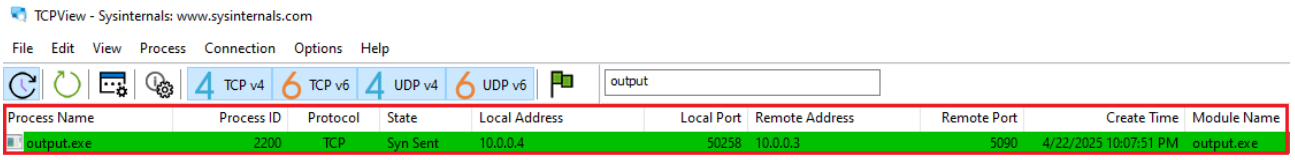

Figure 14: TCPView Output

The network communication can also be observed using Wireshark, providing further insight into the RAT's connection attempts.

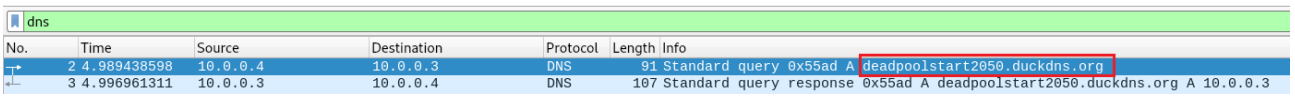

Figure 15: Using WireShark

IOCs[Permalink]

- Hash:

```
17a59db354f270147d5da27aa7978a3c
40fb01ac9879cf7ea9e9a375bd525a66
```

- URL

```
hxxps://paste[.]ee/d/1Juiw3uF/0
hxxps://paste[.]ee/d/m6drh6pM/0
deadpoolstart2050[.]duckdns[.]org
```

Curious about what AsyncRAT looks like from the attacker's perspective? The following images provide a glimpse into the control panel of the AsyncRAT server, highlighting some of its core functionalities.

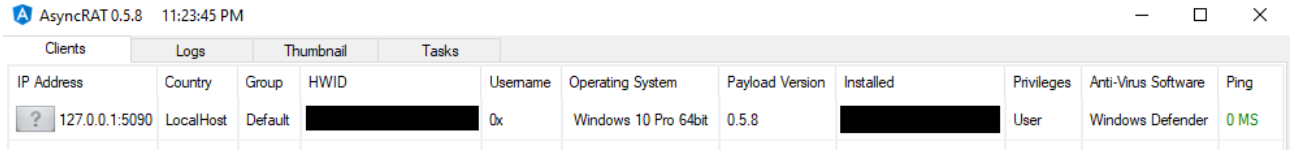

Figure 16: RAT GUI

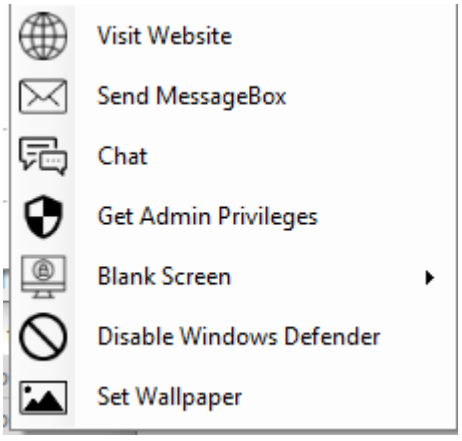

Figure 17: RAT Options

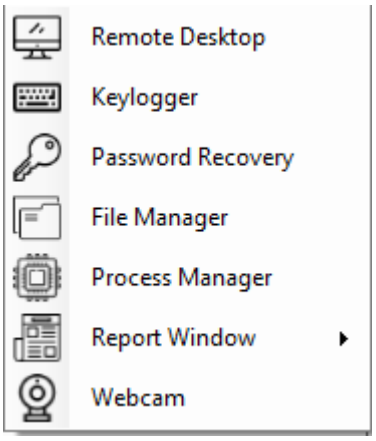

Figure 18: RAT Options

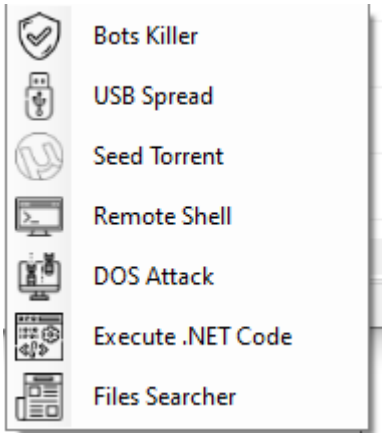

Figure 19: RAT Options

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